

SACM
Internet-Draft
Intended status: Standards Track
Expires: September 8, 2016

C. Coffin
D. Haynes
C. Schmidt
The MITRE Corporation
J. Fitzgerald-McKay
Department of Defense
March 7, 2016

SWID Message and Attributes for PA-TNC
draft-coffin-sacm-nea-swid-patnc-00

Abstract

This document specifies the SWID Message and Attributes for PA-TNC. It extends the PA-TNC specification [[RFC5792](#)] by providing specific attributes and message exchanges to allow endpoints to report their software inventory information to a NEA server (as described in [[RFC5209](#)]) using SWID tags [[SWID](#)].

Status of This Memo

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

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

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

This Internet-Draft will expire on September 8, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	4
1.1.	Scope and Audience	4
1.2.	Keywords	7
1.3.	Definitions	7
2.	Background	8
2.1.	Role of SWID Message and Attributes for PA-TNC	8
2.2.	Supported Use Cases	9
2.2.1.	Use Software Inventory as a Factor in Determining Endpoint Access	9
2.2.2.	Maintain a Central Repository Reflecting an Endpoint's Software Inventory	10
2.2.3.	PA-TNC Use Cases	10
2.3.	Non-supported Use Cases	11
2.4.	Specification Requirements	12
2.5.	Non-Requirements	13
2.6.	Assumptions	13
2.7.	Non-Assumptions	14
2.8.	Caveat Regarding Persistent Connections	14
2.9.	SWID Message and Attributes for PA-TNC Diagram Conventions	15
3.	SWID Message and Attributes for PA-TNC System Requirements	15
3.1.	SWID Tags as Inventory Evidence	16
3.2.	Basic SWID Tag Inventory Exchange	16
3.3.	SWID Tag Identifiers	17
3.3.1.	Tag Identifier Data	18
3.3.2.	Tag Identifier Instances	19
3.3.3.	Comparing Tag Identifiers and Tag Identifier Instances	20
3.3.3.1.	Matching Tag Identifiers Indicate the Same Software Product	21
3.3.3.2.	Matching Tag Identifiers DO NOT Necessarily Indicate Identical Tag Files	21
3.3.3.3.	Matching Tag Identifier Instances MIGHT Indicate Identical Tag Files	22
3.3.3.4.	Differing Tag Identifiers DO NOT Necessarily Indicate Different Software Products	22
3.3.4.	Using Tag Identifiers in SWID Attributes	23
3.4.	Targeted Requests	23
3.5.	Monitoring Changes in an Endpoint's SWID Tag Collection	24
3.6.	Reporting Change Events	26
3.6.1.	Change Event Records	27
3.6.2.	Updating Inventory Knowledge Based on Events	28

3.6.3.	Using Event Records in SWID Attributes	29
3.6.4.	Partial and Complete Lists of Event Records in SWID Attributes	29
3.6.5.	Synchronizing Event Identifiers and Epochs	31
3.7.	Supporting Multiple Instances of a Single Tag	33
3.7.1.	Inventory Reporting in the Presence of Multiply- Instantiated Tags	33
3.7.2.	Event Reporting in the Presence of Multiply Instantiated Tags	33
3.8.	Subscriptions	34
3.8.1.	Establishing Subscriptions	34
3.8.2.	Managing Subscriptions	35
3.8.3.	Terminating Subscriptions	35
3.8.4.	Subscription Status	36
3.8.5.	Fulfilling Subscriptions	37
3.8.5.1.	Subscriptions Reporting Inventories	39
3.8.5.2.	Subscriptions Reporting Events	39
3.8.5.3.	Targeted Subscriptions	40
3.8.5.4.	No Subscription Consolidation	41
3.8.5.5.	Delayed Subscription Fulfillment	41
3.9.	Multiple Sources of SWID Tags	42
3.10.	Error Handling	43
4.	SWID Message and Attributes for PA-TNC Protocol	45
4.1.	PA Subtype (AKA PA-TNC Component Type)	45
4.2.	PB-TNC and PA-TNC Messages	46
4.3.	PA-TNC Attribute Header	46
4.4.	SWID Attribute Overview	47
4.5.	SWID Attribute Exchanges	49
4.6.	SWID Message and Attributes for PA-TNC Attribute Enumeration	51
4.7.	Normalization of Text Encoding	52
4.8.	Request IDs	53
4.9.	SWID Request	54
4.10.	SWID Tag Identifier Inventory	58
4.11.	SWID Tag Identifier Events	61
4.12.	SWID Tag Inventory	66
4.13.	SWID Tag Events	68
4.14.	Subscription Status Request	72
4.15.	Subscription Status Response	72
4.16.	PA-TNC Error as Used by SWID Message and Attributes for PA-TNC	74
4.16.1.	SWID_ERROR, SWID_SUBSCRIPTION_DENIED_ERROR and SWID_SUBSCRIPTION_ID_REUSE_ERROR Information	76
4.16.2.	SWID_RESPONSE_TOO_LARGE_ERROR Information	77
4.16.3.	SWID_SUBSCRIPTION_FULFILLMENT_ERROR Information	79
5.	Security Considerations	80
5.1.	Evidentiary Value of SWID Tags	81

5.2.	Integrity of the SWID Tag Collection	81
5.3.	Sensitivity of Collected Tags	82
5.4.	Integrity of Endpoint Records	83
5.5.	SWID-PC Access Permissions	84
5.6.	Sanitization of Tag Fields	84
5.7.	Tag Library Poisoning	85
5.8.	PA-TNC Security Threats	85
6.	Privacy Considerations	85
7.	Relationship to Other Specifications	86
8.	IANA Considerations	86
8.1.	Registry for PA-TNC Attribute Types	87
8.2.	Registry for PA-TNC Error Codes	87
8.3.	Registry for PA Subtypes	88
9.	References	88
9.1.	Normative References	89
9.2.	Informative References	89
Appendix A.	Examples	90
A.1.	A Simple SWID Tag	90
A.2.	SWID Request Attributes	92
A.2.1.	Simple Request	92
A.2.2.	Subscription Request for Events	93
A.2.3.	Targeted Request	93
A.3.	SWID Response Attributes	95
A.3.1.	SWID Tag Identifier Events Attribute	95
A.3.2.	SWID Tag Inventory Attribute	97
	Authors' Addresses	98

[1.](#) Introduction

[1.1.](#) Scope and Audience

The Network Endpoint Assessment (NEA) Working Group defines 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, monitor endpoint health, and for many other activities. Information about the software present on an endpoint is an important consideration for such activities. Software Identification tags (SWID tags) [\[SWID\]](#) are formatted records (usually XML documents) that identify a specific software product. In this case, a "software product" can be a distinct release of some piece of software, such as an operating system, web browser, etc.; a patch or plug-in for such an application; or a suite of such applications. The SWID specification describes the format of these documents as well as rules governing their use on computer systems. In particular, software that supports SWID tags is expected to deposit an identifying tag on the endpoint when the software is installed, modify or replace the tag as the software is updated, and delete the tag when the software is

uninstalled. SWID tags can also be created and managed by third-party tools or by local enterprises, allowing for tags to indicate the presence of software even when that software's manufacturer has not included SWID support. As such, by collecting a list of tags on an endpoint, one receives evidence as to the software present on that endpoint. The attributes defined in this document are used to communicate software inventory evidence, in the form of SWID tags, between the posture collector and posture validator using the PA-TNC interface, as shown in Figure 1 below.

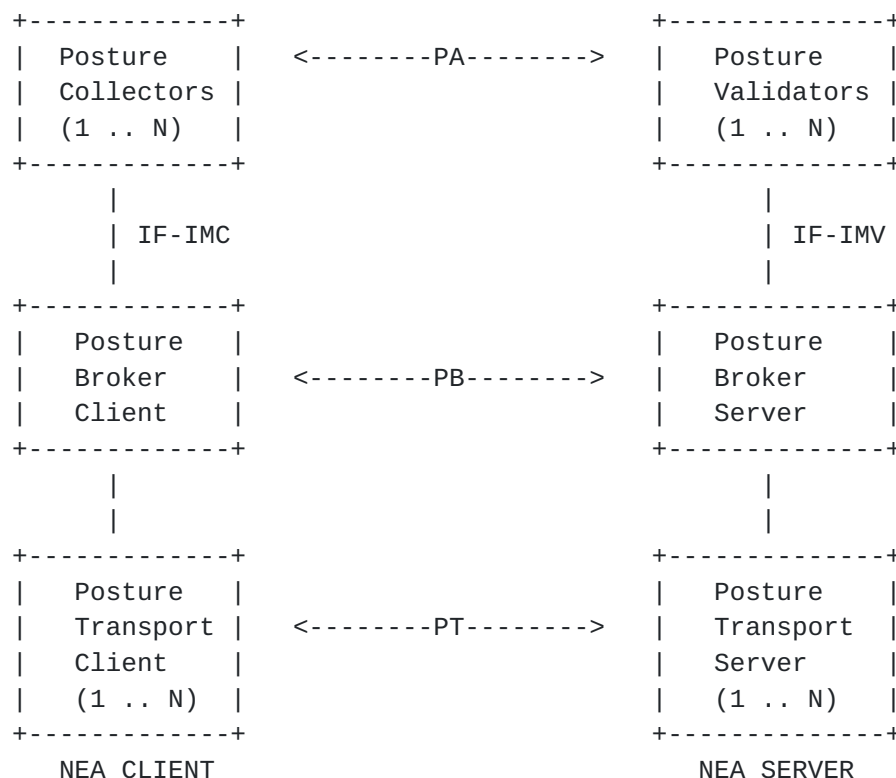


Figure 1: NEA Reference Model

The use of standard protocols and formats for conveying evidence about endpoint state (in this case, endpoint inventory information) has a number of benefits. The use of standard protocols and formats facilitates interoperability between products developed by different vendors. This allows consumers to select the product that has features that best fit with the needs of their environment, with the expectation that it will be able to interoperate with other parts of their infrastructure (at least with regard to the aforementioned protocols and formats). In addition, because a standard is expected to be implemented by multiple independent parties, this means that the standard protocols and formats receive more review than might be

expected in a proprietary solution. When the standard is managed by a group that is responsive to feedback from such implementers, as is the case with the IETF, this can lead to improvements in efficiency and security of those protocols and formats. For these reasons, a standard means of conveying endpoint inventory information such as the one described in this document provides significant value to users. Vendors benefit from utilizing SWIDs to serve as evidence of software inventory because it reduces their need to develop remote software inventory tools for the increasing variety of endpoint platforms. If those endpoints support SWID Message and Attributes for PA-TNC, vendors can use these protocols to gather software inventory information remotely.

This specification defines a new set of PA-TNC attributes, carried over PA-TNC messages, which are used to communicate requests for SWID tags and events surrounding those tags, and for conveying that information back to a NEA Server.

Possession of a list of an endpoint's SWID tags is very useful in understanding and maintaining the security state of an enterprise. For example, if an enterprise policy requires the presence of certain pieces of software and/or prohibits the presence of other software, SWID tags can be used to indicate compliance or non-compliance with these requirements. SWID tags indicating software presence and the patch level of that software can be compared to vulnerability or threat alerts to determine an endpoint's exposure to attack. SWID tags provide a great deal of information about unfamiliar software products, including the software author and potentially including where the software is installed on the endpoint and what files on the endpoint are associated with this installed software. All of these uses make an understanding of an endpoint's SWID tag collection highly useful to NEA Servers and other enterprise security applications.

Before reading this specification any further, the reader should review and understand the NEA reference architecture as described in the Network Endpoint Assessment (NEA): Overview and Requirements [[RFC5209](#)]. The reader should also understand the capabilities and requirements common to PA-TNC interfaces as defined in [RFC 5792](#) [[RFC5792](#)].

In addition, this document relies upon features in the Trusted Computing Group's Trusted Network Communications (TNC) IF-IMC 1.3 [[IF-IMC](#)] and IF-IMV 1.4 [[IF-IMV](#)] specifications. These specifications describe standard interfaces and requirements for the interoperation between TNC IMCs and TNC Clients, and between TNC IMVs and TNC Servers. The TNC and NEA architectures are interoperable and the following components are equivalent:

+-----+-----+	
TNC Component	NEA Component
+-----+-----+	
Integrity Measurement Verifier (IMV)	Posture Validator
Integrity Measurement Collector (IMC)	Posture Collector
TNC Server (TNCS)	Posture Broker Server
TNC Client (TNCC)	Posture Broker Client
+-----+-----+	

Table 1: Equivalent components in TNC and NEA architectures

As a result, the IF-IMC specification effectively standardizes an interface between a Posture Collector and the Posture Broker Client, while the IF-IMV specification effectively standardize an interface between a Posture Validator and a Posture Broker Server. However, as there is currently no NEA equivalent for IF-IMC or IF-IMV, the names for these specifications as well as the names of functions and data structures that appear in those specifications uses TNC rather than NEA terminology. Readers should be aware of the functional equivalence of the TNC and NEA terms.

If the reader is building a posture collector that supports PA-TNC, the reader is encouraged to read the TNC IF-IMC Specification prior to reading this specification. If the reader is building a posture validator that supports IF-IMV, the reader is encouraged to read the TNC IF-IMV Specification prior to reading this specification.

1.2. Keywords

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in [RFC 2119](#) [RFC2119]. This specification does not distinguish blocks of informative comments and normative requirements. Therefore, for the sake of clarity, note that lower case instances of must, should, etc. do not indicate normative requirements.

1.3. Definitions

This section defines terms with special meaning within this document.

SWID-PC - A Posture Collector (PC) that conforms to this specification. Note that such a posture collector might also support other PA-TNC exchanges beyond SWID Message and Attributes for PA-TNC.

SWID-PV - A Posture Validator (PV) that conforms to this specification. Note that such a posture verifier might also support other PA-TNC exchanges beyond SWID Message and Attributes for PA-TNC.

Endpoint's SWID Tag Collection - The set of SWID tags installed and managed on an endpoint for software installed on that endpoint. An endpoint's SWID tag collection might include SWID tags from multiple sources, including but not limited to SWID tag files deposited on the file system during software installation, SWID tags generated to report output from software discovery tools, and SWID tags dynamically generated by a software or package management system on an endpoint.

2. Background

2.1. Role of SWID Message and Attributes for PA-TNC

The International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC) published the specification governing SWID tag construction and use in 2009. [\[SWID\]](#) Since that time, a growing number of vendors have integrated SWID tags into their software products. Doing so significantly simplifies the task of identifying these 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, a readily available list of SWID tags provides simple and immediate evidence as to the presence of the given piece of software.

SWID tags can also be useful even when a piece of software does not supply the tags itself. Discovery processes are permitted to express their findings using SWID tags, place these in the endpoint's SWID tag collection, and maintain them like vendor-created SWID tags. This means that an endpoint's SWID tag collection is not necessarily limited to containing SWID tags for software whose authors have taken the time to integrate SWID maintenance into their installation and update processes. Similarly, software and package managers on an endpoint (such as RPM and YUM) keep records of installed software, and these records can be exported as a series of SWID tags, allowing these managers to expose their information about software inventories in a standards-based manner. Finally, for organizations that centrally manage the distribution of software, in-house-developed SWID tags can be added to any software product that does not natively support SWID tags allowing the organization to accurately identify any software it has distributed.

The NEA Server needs access to this inventory evidence if it is to use this information to assess endpoint compliance with policy. The SWID Message and Attributes for PA-TNC specification has been created for this purpose. Specifically, the attributes defined in this specification allow a Posture Validator to request evidence of an endpoint's inventory in the form of SWID tags and allow the Posture Collector to respond with the appropriate information.

It is not necessary to understand the details of SWID tag construction and maintenance to understand the behaviors described in the SWID Message and Attributes for PA-TNC specification, and it is beyond the scope of this specification to discuss the details of the SWID standard. Implementers, however, will likely need to be familiar with the SWID tag format and how to locate tags on an endpoint. The SWID specification is available from ISO/IEC at http://www.iso.org/iso/catalogue_detail.htm?csnumber=53670. The XML schema for a SWID tag file is available from ISO: <http://standards.iso.org/iso/19770/-2/2009/schema.xsd>. The most current working and production versions of the XML schema for SWID tags can be found in the directory listing at <http://standards.iso.org/iso/19770/-2/>. The US National Institute of Standards and Technology (NIST) also has published guidelines for SWID tag creation, which provide further guidance for those interested in the use and best practices surrounding SWID tags. [\[NIST-SWID\]](#)

2.2. Supported Use Cases

This section describes the SWID Message and Attributes for PA-TNC use cases supported by this specification. The primary use of exchanging SWID tag information over the PA-TNC interface is to enable a challenger (e.g. NEA Server) to obtain inventory evidence about some system in a way that conforms to NEA procedures while taking advantage of the simplicity and precision of SWID tags. Collected SWID tags 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.

2.2.1. Use Software Inventory as a Factor in Determining Endpoint Access

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 by endpoints that have certain prohibited pieces of software installed, such as applications that pose known security risks. To support such policies, the NEA Server needs to collect evidence indicating the software inventory of an endpoint that is seeking to initiate or continue connectivity to the enterprise.

SWID Message and Attributes for PA-TNC facilitates policy decisions that consider an endpoint's software inventory by providing the NEA Server with a list of the SWID tags in the endpoint's SWID tag collection. The tags in this collection serve as evidence as to the endpoint's installed software. The SWID-PC can provide a complete or

partial list of tags to the SWID-PV as required to determine policy compliance. The SWID-PV can then use this as evidence of compliance or non-compliance with enterprise policy.

2.2.2. Maintain a Central Repository Reflecting an Endpoint's Software Inventory

Many tools can use information about an endpoint's software inventory to monitor and enforce the security of an enterprise. For example, a software patching service can use an endpoint's software inventory to determine whether certain endpoints have software that requires patching. A vulnerability management tool might identify endpoints with known vulnerabilities (patched or otherwise) and use this to gauge enterprise exposure to attack. A license management tool might verify that all copies of a particular piece of software are accounted for within the enterprise. The presence of a central repository representing a real-time understanding of each endpoint's software inventory facilitates all such activities. Using a central repository that can ensure the freshness of its collected information is generally more efficient than having each tool collect the same inventory information from each endpoint individually and leads to a more consistent understanding of enterprise state.

SWID Message and Attributes for PA-TNC supports this activity through a number of mechanisms. As noted above, it allows a SWID-PC to provide a complete list of the tags present in an endpoint's SWID tag collection to the SWID-PV, which can then pass this information on to a central repository such as a Configuration Management Database (CMDB) or similar application. In addition, SWID-PCs are required to be able to monitor for changes to an endpoint's SWID tag collection in near real-time and push reports of changes to the SWID-PV as soon as those changes are detected. Thus any central repository fed by a SWID-PV receiving such information can be updated soon after the change occurs. Keeping such a central repository synchronized with the state of each endpoint's SWID tag collection allows tools that use this information for their own security activities to make decisions in a consistent, efficient manner.

2.2.3. PA-TNC Use Cases

SWID Message and Attributes for PA-TNC are intended to operate over the PA-TNC interface and, as such, are intended to meet the use cases set out in the PA-TNC specification.

2.3. Non-supported Use Cases

Some use cases not covered by this version of SWID Message and Attributes for PA-TNC include:

- o This specification does not address how the endpoint's SWID tag collection is populated. In particular, NEA components are not expected to perform software discovery activities beyond compiling the tags in an endpoint's SWID tag collection. This collection might potentially come from multiple sources on the endpoint (e.g., SWID tags generated dynamically by package management tools or discovery tools, as well as SWID tag files discovered on the file system). While an enterprise might make use of software discovery procedures to identify installed software, especially software that does not install or manage its own SWID tag, such procedures are outside the scope of this specification.
- o This specification does not address converting inventory information expressed in a proprietary format into the SWID tag format or converting a SWID tag into a proprietary format. Instead, it focuses exclusively on defining interfaces for the transportation of SWID tags in the expectation that this is the format around which reporting tools will converge.
- o This specification provides no mechanisms for a posture validator to request a specific list of tags based on arbitrary tag properties from the endpoint. For example, requesting only tags representing software from a particular vendor is not supported. After the endpoint's SWID tag 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 SWID tags, but this specification does not address using such queries to constrain the initial collection of this information from the endpoint.
- o This specification does not address utilization of certain SWID tag fields designed to facilitate local tests (i.e., on the endpoint) of endpoint state. For example, the optional `package_footprint` field of a 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. Future versions of this specification might add support for such testing.

2.4. Specification Requirements

Below are the requirements that the SWID Message and Attributes for PA-TNC specification is required to meet in order to successfully play its role in the NEA architecture.

- o 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, the SWID Message and Attributes for PA-TNC ought to minimize overhead delays and make PA-TNC communications as rapid and efficient as possible.

Efficiency is also important when one considers that some network endpoints are small and low powered, some networks are low bandwidth and/or high latency, and some transport protocols (such as PT-EAP, Posture Transport (PT) Protocol for Extensible Authentication Protocol (EAP) Tunnel Methods [[RFC7171](#)]) or their underlying carrier protocol might allow only one packet in flight at a time or only one roundtrip. However, when the underlying PT protocol imposes fewer constraints on communications, this protocol ought to be capable of taking advantage of more robust communication channels (e.g. using larger messages or multiple roundtrips).

- o Loosely Coupled to the SWID Specification

Because the SWID specification is managed by ISO/IEC, the IETF has no direct influence over this specification or any revisions made to it. For this reason, the SWID Message and Attributes for PA-TNC specification ought to minimize its requirements and assumptions with regard to the structure and content of the SWID tags. While some level of visibility into tag contents is required for certain features of this specification, minimization of such dependencies is necessary to improve compatibility with future revisions of the SWID specification.

- o Scalable

SWID Message and Attributes for PA-TNC needs to be usable in enterprises that contain tens of thousands of endpoints or more. As such, it needs to allow a 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.

- o Interoperable

This specification defines the protocol for how PCs and PVs can exchange and use SWID tags to provide a NEA Server with information about an endpoint's software inventory. Therefore a key goal for this specification is ensuring that all SWID PCs and PVs, regardless of the vendor who created them, are able to interoperate in their performance of these duties.

- o Support precise and complete historical reporting

This specification is expected to outline capabilities that support real-time understanding of the state of endpoint in a network in a way that can be used by other tools. One means of facilitating such an outcome is for a Configuration Management Database (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 PC be able to report any changes to its SWID tag 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 SWID tag collection throughout the entire interval when it was not connected.

2.5. Non-Requirements

There are certain requirements that the SWID Message and Attributes for PA-TNC specification explicitly is not required to meet. This list is not exhaustive.

- o End to End Confidentiality

SWID tags have no inherent mechanism for confidentiality, nor is this property automatically provided by PA-TNC interface use.

Confidentiality is generally provided by the underlying transport protocols, such as the PT Binding to TLS [[RFC6876](#)] or PT-EAP Posture Transport for Tunnelled EAP Methods [[RFC7171](#)] - see [Section 7](#) for more information on related standards. Should users wish confidentiality protection of assessment instructions or results, this needs to be provided by parts of the NEA architecture other than this specification.

2.6. Assumptions

Here are the assumptions that SWID Message and Attributes for PA-TNC makes about other components in the NEA architecture.

- o Reliable Message Delivery

The Posture Broker Client and Posture Broker Server are assumed to provide reliable delivery for PA-TNC messages and therefore the SWID Attributes sent between the SWID PCs and the PVs. In the event that reliable delivery cannot be provided, the Posture Collector or Posture Validator is expected to terminate the connection.

2.7. Non-Assumptions

The SWID Message and Attributes for PA-TNC specification explicitly does not assume:

- o Authenticity and Accuracy of SWID tags with Regard to Endpoint Inventory

This specification makes no assumption as to whether the SWID tags that it reports are authentic tags (rather than maliciously generated) or that these tags correctly reflect software state on 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 existing SWID tags to the NEA Server. Similarly, this specification makes no assumption with regard to the completeness of the SWID tag 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 tag in an endpoints SWID tag collection. See [Section 5](#) for more on this security consideration.

2.8. Caveat Regarding Persistent Connections

One of the features defined in this specification describes the ability for SWID-PCs to monitor the state of their endpoint's SWID tag collection and, when changes are detected, to push updates to the SWID-PV without the SWID-PV sending a request. (This feature is described in more detail in [Section 3.8](#).) This capability is tied to the SWID-PC's ability to initiate an Integrity Check Handshake (as described in [section 2.10.1](#) of the IF-IMC 1.3 specification [IF-IMC]), which in turn is only possible if there is an active connection with a valid connection ID (as described in [section 2.10.2](#) of the IF-IMC 1.3 specification).

The other specifications of the NEA architecture do not require endpoints to maintain active connections outside of an Integrity Check Handshake. While implementers are allowed to maintain connections outside of an Integrity Check Handshake (and there are advantages to doing so), maintaining open connections also consumes resources on both the endpoint and the NEA Server, so some implementers may choose to close connections as soon as a handshake completes. Moreover, for devices with intermittent connectivity

(such as mobile phones), maintaining an active, ongoing connection may be impractical.

This specification requires that all SWID-PCs and SWID-PVs support subscriptions. However, for the reasons noted above, subscriptions may not be practical in all environments where SWID-PCs and SWID-PVs are deployed. Parties deploying SWID-PCs and SWID-PVs as part of their enterprise protection strategy are encouraged to understand the limits of specific devices and their NEA architecture as a whole. In some cases, while other features of the SWID Message and Attributes for PA-TNC specification may be employed, the use of subscriptions may be limited or impractical without additional infrastructure changes.

2.9. SWID Message and Attributes for PA-TNC Diagram Conventions

This specification defines the syntax of the SWID Message and Attributes for PA-TNC using diagrams. Each diagram depicts the format and size of each field in bits. Implementations **MUST** send the bits in each diagram as they are shown from left to right for each 32-bit quantity traversing the diagram from top to bottom. Multi-octet fields representing numeric values **MUST** be sent in network (big endian) byte order.

Descriptions of bit fields (e.g. flags) 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).

3. SWID Message and Attributes for PA-TNC System Requirements

The SWID Message and Attributes for PA-TNC specification facilitates the exchange of SWID tag inventories and event information. Specifically, each application supporting SWID Message and Attributes for PA-TNC includes a component known as the SWID-PC that receives messages sent with the SWID Attributes component type. The SWID-PC is also responsible for sending appropriate SWID Attributes back to the SWID-PV in response. Similarly, the SWID-PV exists on a NEA Server and is responsible for interpreting responses, forwarding information to a CMDB if desired, and making policy decisions based upon the received information. This section outlines what a SWID tag inventory is, important features of tags used by this specification, and the requirements on SWID-PCs and SWID-PVs in order to support the stated use cases of this specification.

3.1. SWID Tags as Inventory Evidence

As noted in [Section 2.1](#), SWID tags are intended to be open, easily accessible evidence indicating the presence of a particular piece of software on an endpoint. A SWID tag contains multiple fields intended to uniquely identify a single software product. Ideally, a SWID tag is managed by the software that installs, modifies, replaces, amends (e.g. patches, updates), and/or uninstalls the product. Discovery processes, software package managers, and other tools can also create and manage tags as a way to represent software that they discover or manage on the endpoint.

It is important to note that, even in the ideal cases where a product manages its own SWID tag, a tag is inherently distinct from the product that it identifies. For this reason, a SWID tag needs to be treated as evidence of software presence, but cannot be treated as proof of software presence. That noted, a standardized representation of evidence indicative of an endpoint's software inventory is a powerful tool in managing software within an enterprise.

3.2. Basic SWID Tag Inventory Exchange

In the most basic exchange supported by this specification, a SWID-PV sends a request to the SWID-PC requesting a copy of all the SWID tags in the endpoint's SWID tag collection. This simple exchange is shown in Figure 2.

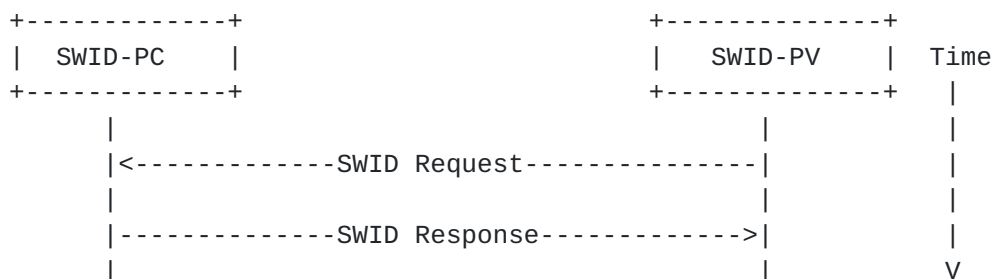


Figure 2: Basic SWID Message Exchange

Upon receiving such a SWID Request from the SWID-PV, the SWID-PC is expected to locate the endpoint's SWID tags and then create copies of all identified SWID tags and place them within its response attribute.

SWID-PVs MUST discard without error any SWID Response attributes that they receive for which they do not know the SWID Request parameters

that led to this SWID Response. This is due to the fact that the SWID Request includes parameters that control the nature of the response (as will be described in the following sections) and without knowing those parameters the SWID Response cannot be reliably interpreted. Most often receiving an unsolicited SWID Response attribute happens when a NEA Server has multiple SWID-PVs; one SWID-PV sends a SWID Request but, unless exclusive delivery is used by the SWID-PC in sending the response, both SWID-PVs receive copies of the resulting SWID Response. In this case, the SWID-PV that didn't send the SWID Request would lack the context necessary to correctly interpret the SWID Response it received and would simply discard it. Note, however, that proprietary measures might allow a SWID-PV to discover the SWID Request parameters for a SWID Response even if that SWID-PV did not send the given SWID Request. As such, there is no blanket requirement for a SWID-PV to discard all SWID Responses to SWID Request the SWID-PV did not generate itself, only that SWID-PVs are required to discard SWID Responses for which they cannot get the necessary context to interpret.

In the case that it is possible to do so, the SWID-PC MAY send its SWID Response attribute to the SWID-PV that requested it using exclusive delivery as described in [section 3.3.2.2](#) of the IF-IMV r1.4 specification. Exclusive delivery ensures that only the sender of the SWID Request receives the resulting SWID Response. However, exclusive delivery is not always possible (it requires the use of IF-IMC 1.3 or later and IF-IMV 1.3 or later, and not all products currently do this) or necessarily desirable in all cases. As such, SWID Message and Attributes for PA-TNC does not require support for exclusive delivery of attributes.

[3.3.](#) SWID Tag Identifiers

SWID tags can contain a great deal of information about a software product. In addition to identifying name, manufacturer, and version of a software product, SWID tags might contain references to related products, associated files and libraries, dependencies on other software, and many other details. Moreover, SWID tags might be customized on the endpoint to indicate when the SWID tag was last checked for accuracy relative to the endpoint's installed software and other information about how the software was received. (This document refers to this customized information as "installationspecific" tag information.) For this reason, actual possession of a SWID tag can be useful for reasoning about details of an endpoint's software inventory. However, a SWID tag file that contains all optional fields might be tens of KB in size. This means that an endpoint's full SWID inventory, encompassing hundreds of applications, can be quite large.

If bandwidth is a concern within an enterprise, there is a way to identify a SWID tag without needing the complete tag. All tags contain specific fields that can be used to distinguish a tag for a particular piece of software from tags for different pieces of software. The Tag Creator RegID is a string that uniquely identifies the creator of this SWID tag (who might or might not be the same as the entity who created the described software) while the Unique ID is a string that uniquely identifies the described piece of software according to that tag creator. These two pieces of information together create a "tag identifier".

3.3.1. Tag Identifier Data

Some attributes defined in this specification contain fields to hold tag identifiers rather than whole tags. When populating these fields, both the Tag Creator RegID and the Unique ID values MUST be copies of the values of fields within the SWID tag that is being identified. The specific fields of a SWID tag that correspond to the Tag Creator RegID and Unique ID values vary between the different releases of the ISO SWID tag specification. It is important to note that, in all other parts of this specification, the terms Tag Creator RegID and Unique ID refer to the general field values defined above rather to any term used in any specific release of the ISO SWID tag specification.

To identify the SWID tag field corresponding to the Tag Creator RegID, identify the field containing the regid value of the entity that created the given SWID tag. Note that this might not be the same entity that created the software. The Tag Creator RegID MUST be the regid, rather than any prose name that might be associated with the tag creator. The specific structure of a regid is defined in the ISO/IEC SWID specification.

To identify the SWID tag field corresponding to the Unique ID, find the field that contains a string that uniquely identifies a specific product, version, edition, revision, etc. of a piece of software. Note that this is a single field within the SWID tag, rather than a concatenation of multiple fields. In particular, SWID tags often contain designated fields for just the product name, product version, product edition, etc., but these fields are not used to populate the Unique ID. Instead, look for a single field that is designed to uniquely identify a specific software product, version, etc. (and thus uniquely identifies a specific tag, at least according to the tag's creator).

Consult the ISO/IEC specification for the specific fields that correspond to the requirements of the Tag Creator RegID and Unique ID, as defined above. For example, in the 2009 version of the ISO/

IEC SWID specification [[SWID](#)], the Tag Creator RegID corresponds to the value of the software_identification_tag/software_id/tag_creator_regid field in a SWID tag. The Unique ID corresponds to the value of the software_identification_tag/software_id/unique_id field in a SWID tag. In subsequent releases of the ISO/IEC SWID specification, different fields might be used to convey the same information.

3.3.2. Tag Identifier Instances

A tag identifier (i.e., the combination of the Tag Creator RegID and the Unique ID fields) uniquely identifies a particular SWID tag, which corresponds to a single software product. Assuming that this product manages its own SWID tag (i.e., creates the tag on installation and deletes the tag when the product is uninstalled) then every system with an instance of this software product installed would also have a copy of this same SWID tag file with the same tag identifier field values. (Presence of SWID tags managed by other tools, such as discovery tools, would also depend on the presence of those tools on the device.) In fact, if multiple instances of the same software product are installed on a single device (i.e., it has been installed twice in different locations) that device would have two instances of the same SWID tag, one for each installation. Both instances of the SWID tag would have the same tag identifier field values. This is true even though the tags themselves might differ with regard to their installation-specific tag fields. In many cases it is important to distinguish between instances of a particular tag on a particular endpoint. For example, if one is alerted to the deletion of a particular SWID tag and there are multiple instances of that SWID tag on the endpoint, one will likely wish to know which instance was deleted.

Individual instances of SWID tags are distinguished by providing an "Instance ID" value along with the tag identifier. An Instance ID is a string that is uniquely associated with a particular instance of a SWID tag on a particular endpoint. The exact nature of the Instance ID depends on the source of the SWID tag. If the SWID tag is represented as a file on disk, the Instance ID might be the full path of the SWID tag file, including the name of the SWID tag file itself. (Note that the SWID tag filename **MUST** be included in the tag file path because it is possible for two SWID tags, each for different instances of the same software product, to co-exist in the same directory under different file names.) In the case that the SWID tag is dynamically generated upon request by some source, such as an RPM or YUM package manager, the generation process **MUST** create an Instance ID to distinguish instances of a particular tag. Inclusion of this Instance ID ensures each tag is uniquely identified on a given endpoint.

To the extent that it is possible, the generation of Instance IDs SHOULD be repeatable for a single installation of a single SWID tag. In the case where a product is installed once, and then SWID tags are generated upon request, each time the SWID tag is generated the tag identifier instance SHOULD all have the same Instance ID value. For example, if a package manager generates a SWID tag in response to a request based on some record it possesses, and then later generates the SWID tag again based on the same record of package installation, then the same Instance ID value SHOULD be created both times. This is necessary to allow remote parties to understand whether a reported SWID tag instance is for the same product installation they saw reported earlier or if it represents a new installation of the same product. Note, however, that some exceptional situations might result in the changing of a product's Instance ID. For example, it is not explicitly prohibited by the SWID specification for tags to move after installation, and thus have their tag file path change. If the file path was used as the tag's Instance ID, subsequent tag identifier instances for that same product might appear to be different. Implementers and users need to be aware of this possibility.

The combination of a tag identifier with an Instance ID is referred to as a "tag identifier instance". A tag identifier instance uniquely identifies a particular instance of a particular tag on a given endpoint. Note that two endpoints might produce identical tag identifier instances, but these do not mean that the tag files on the two endpoints are identical - the tags in question indicate the same software product on both endpoints (since the respective tag identifiers are identical), but the tags might still differ in their installation-specific fields. Therefore, it is important to remember that tag identifier instances are only comparable in the scope of a single endpoint; when comparing across different endpoints, only the tag identifier fields (Tag Create RegID and Unique ID) can be meaningfully compared - any Instance ID value will need to be excluded from comparison.

3.3.3. Comparing Tag Identifiers and Tag Identifier Instances

Comparison of tag identifiers can be used to determine whether a particular SWID tag is present in an endpoint's SWID tag collection. A pair of SWID tag identifiers is said to "match" if their Tag Creator RegID and Unique ID fields are identical. Similarly, a pair of tag identifier instances is said to match if their Tag Creator RegID, Unique ID, and Instance ID fields are identical. Fields in SWID Message and Attributes for PA-TNC attributes that contain tag identifiers or tag identifier instances MUST always be normalized to Network Unicode, so comparison between values transported in attributes can be a simple string comparison. When comparing tag

identifiers and tag identifier instances from attributes with the corresponding values from other sources (such as when comparing them to a full SWID tag file or similar record), the relevant fields from the latter need to undergo normalization prior to comparison. See [Section 4.7](#) for more on normalization of the encoding for these fields. Comparisons are case-sensitive.

Matching tag identifiers and tag identifier instances indicate very specific things about the respective tags. The following sections describe what one can and cannot deduce based on matching tag identifiers.

[3.3.3.1](#). Matching Tag Identifiers Indicate the Same Software Product

The Unique ID value of a tag identifier represents a value that the given tag creator will only use to indicate a particular software product (e.g., a particular release of a particular application). The ISO SWID specification prohibits the tag creator from associating a Unique ID value with multiple, different software products. At the same time, the Tag Creator RegID element value uniquely identifies a given tag creator. As such, even if two different tag creators were to assign the same Unique ID value to two different software products, the Tag Creator RegID values will be different, and therefore the tag identifiers will be different. For these reasons, the expectation is that if one sees two tags with the same tag identifiers, these tags are both associated with the same software product (assuming the tag's fields are correctly populated).

[3.3.3.2](#). Matching Tag Identifiers DO NOT Necessarily Indicate Identical Tag Files

Some optional fields in SWID tags can reflect installation-specific information. As such, the SWID tags for a piece of software residing on two different endpoints (or installed twice on a single endpoint) will have the same tag identifier value (same tag creator with the same Unique ID for the same software product) but might contain different information in their installation-specific fields. For this reason, one cannot assume that just because two endpoints provide the same tag identifier value for their software inventories, that the tags on those endpoints are identical in all their fields (although one can deduce that the same software product was present on both endpoints).

Informative note: Initial drafts of the revised ISO SWID specification indicate that modification of SWID tags might no longer be permitted by parties other than the original tag creator (usually the vendor of the software identified by the tag). If this becomes part of the revised SWID specification, then for SWID tags that

conform to this revised specification, this will mean that matching tag identifiers do imply identical tag files.

3.3.3.3. Matching Tag Identifier Instances MIGHT Indicate Identical Tag Files

For a single endpoint, matching tag identifier instance values might indicate identical tag files, at least within a narrow time window. Tag identifier instance values are unique to a specific SWID tag record on that particular endpoint at a particular point in time. The Instance ID in the tag identifier instance information ought to be unique relative to any other instances of the same SWID tag currently also on that endpoint. However, tag identifier instances are still not guaranteed to be unique to a single SWID tag file over a long period of time. Consider a piece of software that is installed (adding a SWID tag), uninstalled (removing the SWID tag), and then reinstalled (adding that SWID tag back but with a different installation-specific field values). It is possible that the two SWID tag files, present at different points in time, might have identical tag identifier instance values even though the tag files themselves were different.

As noted above, SWID tag identifier instances are only comparable within the context of a single endpoint. When SWID tag identifier instances are collected from multiple endpoints and then compared, the Instance ID MUST be ignored in any comparison of tag identifiers from different endpoints.

3.3.3.4. Differing Tag Identifiers DO NOT Necessarily Indicate Different Software Products

While a tag identifier uniquely identifies a software product (i.e., that tag identifier cannot be associated with a different software product), a single product might have more than one tag identifier. This is because it is possible for more than one tag creator to create a SWID tag for the same software product. Multiple tags for the same software product but created by different tag creators will have different Tag Creator RegID values and will also likely differ in their Unique ID value. Thus, these two tags will have different tag identifiers even though they were associated with the same software product. In fact, in some circumstances, two parties might create two different SWID tag records for a single instance of the same software product. For example, when a product is installed, it creates a SWID tag file on the file system, and a software discovery tool also notes the installation of the product and generates its own SWID tag record for the same installation. In this case, that single installation is associated with two SWID tags with different SWID tag identifiers. In short, identical tag identifiers always indicate the

same software product, but different tag identifiers do not necessarily indicate different software products.

3.3.4. Using Tag Identifiers in SWID Attributes

A SWID attribute reporting an endpoint's SWID tag collection can contain SWID tag identifier instances instead of copies of SWID tag files. The message exchange is identical to the diagram shown in Figure 2, but the contents of the SWID Response are SWID tag identifier instances instead of tags. The SWID Request attribute indicates whether the response is required to use full tags or tag identifier instances. Using tag identifier instances can reduce the attribute size of the response by multiple orders of magnitude when compared to sending the same inventory using full tags. A SWID-PC responds to a SWID Request attribute requesting SWID tag identifier instances the same way it responds to a request for full SWID tags, except that instead of copying each SWID tag entirely into the attribute body of the response, it provides the specific values that comprise a SWID tag identifier instance for each tag.

3.4. Targeted Requests

Sometimes a SWID-PV does not require information about every tag on an endpoint but only needs to know about certain tags. For example, an endpoint might be required to have a particular patch installed. In determining compliance with this policy, the SWID-PV is only interested in the specific SWID tag associated with this patch. Instead of requesting a complete inventory just to see if the patch's SWID tag is present, the SWID-PV can make a "targeted request" for the tag in question.

Targeted requests follow the same message exchange described in Figure 2. The SWID-PV targets its request by providing one or more SWID tag identifiers in its SWID Request attribute. The SWID-PC MUST then limit its response to contain only tags that match the indicated tag 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 SWID-PC's response might consist of full tags or of tag identifier instances, depending on the parameters of the SWID Request.

Targeted requests cannot target specific SWID tag instances; the SWID Request does not include fields for Instance IDs. As a result, when responding to a targeted request, a SWID-PC MUST return applicable results for every instance of the identified tags.

Note that targeted requests identify the SWID tags relevant to the request only through SWID tag identifiers for those tags. This

specification does not support arbitrary, parameterized querying of tags. For example, one cannot request all tags from a certain software publisher, or all tags created by a particular tag creator. Targeted requests only allow a requestor to request specific tags (as identified by their tag identifiers) and receive a response that is limited to the named tags. There is also no assumption that a SWID-PC will recognize "synonymous tags" - that is, tags by different tag creators for the same software product. The SWID-PC returns only tags that match the tag identifiers in the SWID Request, even if there might be other SWID tags in the endpoint's SWID tag collection for the same software product.

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

3.5. Monitoring Changes in an Endpoint's SWID Tag Collection

The SWID collection on an endpoint is not static. As software is installed, uninstalled, patched, or updated, the SWID tag collection is expected to change to reflect the new software state on the endpoint. For tags managed by an application's installer, tag changes usually occur at the time of installation or update. For tags added by discovery tools, software and package managers, and other sources, changes to the endpoint's SWID tag collection occur when some process discovers the new or altered software product, which typically lags behind the actual installation or update time.

All SWID-PCs MUST be able to be able to detect changes to the SWID tag repositories on their endpoint. Specifically, SWID-PCs MUST be able to detect:

- o The creation of tags
- o The deletion of tags
- o The alteration of tags

An "alteration" is anything that modifies the contents of a SWID tag file (or would modify it, if the tag file is dynamically generated on demand) in any way, regardless of whether the change is functionally meaningful. Changes MUST be monitored for all utilized sources of SWID tags. This includes, but is not limited to, monitoring sources that dynamically generate SWID tags.

SWID-PCs MUST detect such changes to the endpoint's SWID tag collection in close to real-time (i.e., within seconds) when the Posture Collector is operating. In addition, in the case where there

is a period during which the SWID-PC is not operating, the SWID-PC MUST be able to determine the net change to the endpoint's SWID tag collection over the period it was not operational. Specifically, the "net change" represents the difference between the state of the endpoint's SWID tag collection when the SWID-PC was last operational and monitoring its state, and the state of the endpoint's SWID tag collection when the SWID-PC resumed operation. Note that a net change might not reflect the total number of change events over this interval. For example, if a SWID tag file was altered three times during a period when the SWID-PC was unable to monitor for changes, the net change of this interval might only note that there was an alteration to the file, but not how many individual alteration events occurred. It is sufficient for a SWID-PC's determination of a net change to note that there was a difference between the earlier and current state rather than enumerating all the individual events that allowed the current state to be reached.

The SWID-PC MUST assign a time to each detected change in the endpoint's SWID tag collection. These timestamps correspond to the SWID-PC's best understanding as to when the detected change occurred. These timestamps MUST be as accurate as possible. For changes to the endpoint's SWID tag collection that occur while the SWID-PC is operating, the SWID-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 SWID tag collection that occur while the SWID-PC is not operational, upon becoming operational the SWID-PC needs to make a best guess as to the time of the relevant events (possibly by looking at timestamps on the files), but these values might be off. In the case of dynamically generated SWID tags, the time of change is the time at which the data from which the SWID tags are generate changes, not the time at which a changed SWID tag is generated. For example, if SWID tags are dynamically generated based on data in an RPM database, the time of change would be when the RPM record was changed.

With regard to deletions of SWID tags, the SWID-PC needs to detect the deletion and MUST retain a copy of the full deleted tag so that the tag itself can be provided to the SWID-PV upon request. This copy of the SWID tag MUST be retained for a reasonable amount of time. Vendors and administrators determine what "reasonable" means, but a copy of the tag SHOULD be retained for as long as the event recording the deletion of the tag remains in the SWID-PC's records. This is recommended because, as long as the event is in the SWID-PC's records, the SWID-PC might send an event attribute (described in [Section 3.6](#)) that references this tag, and a copy of the tag is needed if the SWID-PV wanted a full copy of the relevant tags.

With regard to alterations to a SWID tag file, SWID-PCs MUST detect any alterations to the contents of a tag file. Alterations need to be detected even if they have no functional impact on the tag file. For example, the addition of whitespace between XML attributes does not have any impact on the meaning of the SWID tag file, but still needs to be detected as a tag file alteration by a SWID-PC. A good guideline is that any alteration to a file that might change the value of a hash taken on the file's contents needs to be detected by the SWID-PC. A SWID-PC might be unable to distinguish modifications to the content of a tag file from modifications to the metadata the file system associates with the tag file. For example, a SWID-PC might use the "last modification" timestamp as an indication of alteration to a given tag file, but a file's last modification time can change for reasons other than modifications to the file contents. A SWID-PC is still considered compliant with this specification if it also reports metadata change events that do not change the SWID tag file itself as alterations to the SWID tag file. In other words, while SWID-PC authors are encouraged to exclude modifications that do not affect the bytes within the tag file when detecting alterations to a SWID tag 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 SWID-PC always cover all modifications to the contents of tag files, the SWID-PC is considered compliant even if it also registers alterations that do not modify the contents of a tag file as well. When recording an alteration to a tag file, the SWID-PC is only required to note that an alteration occurred. The SWID-PC is not required to note or record how the tag file altered, nor is it possible to include such details in SWID Attributes reporting the change to a SWID-PV.

3.6. Reporting Change Events

As noted in the preceding section, SWID-PCs MUST be able to detect changes to the SWID tag repositories (tag creation, tag removal, and tag alteration) in near real-time while the SWID-PC is operational, and MUST be able to account for any net change to the endpoint's SWID tag collection that occurs when the SWID-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 SWID Message and Attributes for PA-TNC, this is facilitated by reporting change events. All SWID-PCs MUST be capable of receiving requests for change events and sending change event attributes. All SWID-PVs MUST be capable of requesting and receiving change event attributes.

3.6.1. Change Event Records

A change event record consists of either a complete SWID tag or SWID tag identifier instance along with the following pieces of information:

- o The nature of the change (i.e., tag creation, tag deletion, or tag alteration)
- o An Event Identifier (EID) value
- o An EID Epoch value

An EID is a 4-byte unsigned integer that the SWID-PC assigns sequentially to each observed event (whether detected in real-time or deduced by looking for net changes over a period of SWID-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 SWID-PC and any SWID-PVs with which it communicates. EID Epoch values SHOULD be generated randomly and in such a way that it is unlikely that the same EID Epoch is generated twice, even if the SWID-PC reverted to an earlier state (e.g., resetting it to factory defaults). In the case where a SWID-PC needs to reset its EID counter, either because it has exhausted all available EID values or because the SWID-PC's event log becomes corrupted, then 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 SWID-PC might not otherwise be able to determine an ordering for events. In these cases, the SWID-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 SWID Request attribute indicates that a SWID-PV wants an inventory response rather than an event response, while an EID of 0 in a SWID Response is used to indicate the initial state of the endpoint's SWID tag collection prior to the observation of any events. Thus the very first recorded event in a SWID-PC's records within an EID Epoch MUST be assigned a value of 1 or greater. Note that EID and EID Epoch values are assigned by the SWID-PC without regard to whether events are being reported to one or more SWID-PVs. The SWID-PC records events and assigns EIDs during its operation. Any and all SWID-PVs that request

event information from the SWID-PC will have those requests served from the same records and thus will see the same EIDs and EID Epochs for the same events.

The SWID-PC MUST ensure there is no coverage gap (i.e., change events that are not recorded in the SWID-PC's records) in its records of change events. This is necessary because a coverage gap might give a SWID-PV a false impression of the endpoint's state. For example, if a SWID-PV saw an event indicating that a particular SWID tag had been installed, and saw no subsequent events indicating that tag had been deleted, it might reasonably assume that this tag was still installed (assuming the Epoch has not changed). If there is a coverage gap in the SWID-PC's records, however, this assumption is false. For this reason, the SWID-PC's event records MUST NOT contain gaps. In the case where there are periods where it is possible that changes occurred without the SWID-PC detecting or recording them, the SWID-PC MUST either compute a net change and update its records appropriately, or 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 SWID-PC's Epoch changes, all of these associations between EIDs and events are cancelled, and EID values once again become free for assignment.

3.6.2. 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 earlier state.

Every inventory sent by a SWID-PC to a SWID-PV (as described in [Section 3.2](#) through [Section 3.4](#)) includes the EID Epoch and EID of the last event recorded prior to that inventory being compiled. This allows the SWID-PV to place all subsequently received event records in context relative to this inventory (since the EIDs represent a total ordering of all changes to the endpoint's SWID tag collection). Specifically, a SWID-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 in that state. From that point on, if change events are observed, the

attribute describing these events indicates the nature of the change, the affected SWID tags, and the order in which these events occurred (as indicated by the sequential EIDs). Using this information, any remote record of the endpoint's SWID tag collection can be updated appropriately.

3.6.3. Using Event Records in SWID Attributes

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

SWID-PVs can target requests for event records by including one or more tag identifiers, as described in [Section 3.4](#), in the SWID Request that requests an event record list. A targeted request for event records is used to indicate that only events affecting SWID tags that match the provided SWID tag identifiers are to be returned. Specifically, in response to a targeted request for event records, the SWID-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 SWID tag does not match one of the provided SWID tag 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 SWID-PCs and SWID-PVs MUST support targeted request for event records.

3.6.4. Partial and Complete Lists of Event Records in SWID Attributes

Over time, a SWID-PC might record a large number of change events. If a SWID-PV requests all change events covering a large period of time, the resulting SWID Response attribute might be extremely large, especially if the SWID-PV is requesting the use of full SWID tags instead of the use of SWID Identifier instances (as described in [Section 3.3.4](#)). In the case that the resulting attribute is too large to send (either because it exceeds the 4GB attribute size limit imposed by the PA-TNC specification, or because it exceeds some smaller size limit imposed on the SWID-PC) the SWID-PC MAY send a partial list of events back to the SWID-PV.

Generation of a partial list of events in a SWID Response attribute requires the SWID-PC to identify a "consulted range" of EIDs. A consulted range is the set of event records that are examined for inclusion in the SWID Response attribute and that are included in that attribute if applicable. Recall that, if a SWID Request is targeted, only event records that involve the indicated SWID tags would be applicable. (See [Section 3.4](#) for more on Targeted Request.) If a request is not targeted, all event records in the considered range are applicable and included in the SWID Response attribute.

The lower bound of the consulted range MUST be the EID provided in the SWID Request. (Recall that a SWID Request indicates a request for event records by providing a non-0 EID value in the SWID Request. See [Section 3.6.3](#).) 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 SWID Response attribute if it is applicable to the request. The EID of this last event record is called the "Last Consulted EID". The SWID-PC chooses this Last Consulted EID based on the size of the event record list it is willing to provide to the SWID-PV.

A partial result list MUST include all applicable event records within the consulted range. This means that for any applicable event record whose EID is greater than or equal to the EID provided in the SWID Request and whose EID is less than or equal to the Last Consulted EID, that event record MUST be included in the SWID Response conveying this partial list of event records. This ensures that every partial list of event records is always complete within its indicated range.

All SWID Response attributes that convey event records (either using full SWID tags or using SWID tag identifier instances) include an Epoch, Last EID, and Last Consulted EID field. The Last EID contains the EID of the last event record known to the SWID-PC at the time that the SWID Response attribute was generated. The Last EID might or might not be part of the consulted range. As noted above, the Last Consulted EID field contains the EID of the last event record in the consulted range. The Epoch field contains the EID Epoch associated with the Last EID and Last Consulted EID fields as well as all the EIDs in event records contained within the SWID Response attribute. Note that, if responding to a targeted SWID Request, the SWID Response attribute might not contain the event record whose EID matches the Last Consulted EID value. For example, the last consulted EID record might have been deemed inapplicable because it did not match the specified list of SWID tag identifiers in the SWID Request.

If a SWID-PV receives a SWID Response attribute where the Last EID and Last Consulted EID fields are identical, the SWID-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 and Last Consulted EID values differ, the SWID-PV has received a partial result list. In the latter case, if the SWID-PV wishes to try to collect the rest of the partially delivered result list it then sends a new SWID Request whose EID is one greater than the Last Consulted EID in the preceding response. Doing this causes the SWID-PC to generate another SWID 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 SWID Response where the Last EID and Last Consulted EID are equal, the SWID-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 SWID Request might specify an EID that is greater than the EID of the last event recorded by a SWID-PC. In accordance with the behaviors described in [Section 3.6.3](#), a SWID-PC MUST respond to such a request with a SWID Response attribute of the appropriate type (using SWID tags or SWID tag identifier instances as specified in the SWID Request) that contains zero event records. This is because the SWID-PC has recorded no event records with EIDs greater than or equal to the EID in the SWID Request. In such a case, the Last Consulted EID field MUST be set to the same value as the Last EID field in this SWID response attribute. This case is called out because consulted range on a SWID-PC in such a situation is a negative range, where the "first" EID in the range (provided in the SWID Request) is greater than the "last" EID in the range (this being the EID of the last recorded event on the SWID-PC). Implementers need to ensure that SWID-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.

[3.6.5](#). Synchronizing Event Identifiers and Epochs

Since EIDs are sequential within an Epoch, if a SWID-PV's list of event records contains gaps in the EID values within a single Epoch, the SWID-PV knows that there are events that have not been accounted for. The SWID-PV can either request a new event list to collect the missing events or request a full inventory to re-sync its understanding of the state of the SWID tags on the endpoint. In either case, after the SWID-PV's record of the endpoint's SWID tag

collection has been updated, the SWID-PV records the new latest EID value and tracks events normally from that point on.

If the SWID-PV receives any attribute from a SWID-PC where the EID Epoch differs from the EID Epoch that was used previously, then SWID-PV or any entity using this information to track the endpoint's SWID tag collection knows that there is a discontinuity in their understanding of the endpoint's state. To move past this discontinuity and reestablish a current understanding of the state of the endpoint's SWID tag collection, the SWID-PV needs to receive a full inventory from the endpoint. This is because it is not possible to account for all events on the SWID-PC over the interval since the previous Epoch was used, because there is no way to query for EIDs from a previous Epoch. Once the SWID-PV has received a full inventory for the new Epoch, the SWID-PV records the latest EID reported in this new Epoch and can track further events normally.

A SWID-PC MUST NOT report events with EIDs from any Epoch other than the current EID Epoch. The SWID-PC MAY choose to purge all event records from a previous Epoch from memory after an Epoch change. Alternately, the SWID-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 SWID-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 SWID-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 SWID-PC.) A SWID-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 SWID-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 SWID-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 SWID-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 SWID-PC has suffered an error of some kind, possibly indicative of at least

partial corruption of its event log. In this case, the SWID-PV SHOULD treat the situation as if there was a change in Epoch and treat any local copy of the endpoint's SWID tag collection as out-of-sync until a full inventory can be reported by the SWID-PC. In this case, the SWID-PV SHOULD flag the event so it can be examined to ensure it is now operating properly.

3.7. Supporting Multiple Instances of a Single Tag

One important consideration is that it is possible for multiple instances of a SWID tag to be present on an endpoint. (I.e., multiple SWID tag files whose tag identifiers are the same.) This can happen if there are multiple instances of the indicated software product installed on the endpoint. In order to account for the possibility, all SWID-PCs MUST follow specific rules, outlined below.

3.7.1. Inventory Reporting in the Presence of Multiply-Instantiated Tags

When sending an inventory, either full or based on a targeted request, the SWID-PC MUST include one entry for each instance of a relevant tag. (All tags are relevant in a full inventory. In a targeted request for an inventory, only tags that match the tag identifiers provided by the SWID-PV are considered relevant.) For example, if a particular piece of software is installed twice on an endpoint, and thus there are two instances of its SWID tag present in the endpoint's SWID tag collection, an inventory for which this tag is relevant will contain at least two records for this piece of software, one for each tag instance. (It might contain more if multiple tag creators each created tags for the same piece of installed software.) In the case where the SWID-PC's response is expressed using full tags, the response MUST contain one copy of each instance of the given tag. In other words, the SWID-PC MUST send one copy of each tag instance, rather than send multiple copies of one tag instance. In the case where the SWID-PC's response is expressed using tag identifiers, the response MUST include the tag identifier instance for each instance of the given tag.

3.7.2. Event Reporting in the Presence of Multiply Instantiated Tags

When reporting events, the specific tags that were added, deleted, or changed MUST be indicated. For example, in the case where tags A and B are two instances of the same SWID tag, each for separate installations of the same software product, and tag A changes in the endpoint's SWID tag collection, the SWID-PC MUST report the event using tag A (rather than reporting it using B). This means that the report MUST contain the tag file or the tag identifier instance for the affected tag.

3.8. Subscriptions

Thus far, all message exchanges discussed assume that a SWID-PV sent an SWID Request attribute and the SWID-PC is providing a direct response to that request. The SWID Message and Attributes for PA-TNC specification also supports the ability for a SWID-PC to send a message with a SWID Attribute to the SWID-PV in response to observed changes in the endpoint's SWID tag collection, instead of in direct response to a SWID Request. An agreement by a SWID-PC to send content when certain changes are detected to the endpoint's SWID tag collection is referred to in this specification as a "subscription", and the SWID-PV that receives this content is said to be "subscribed to" the given SWID-PC. All SWID-PCs and SWID-PVs MUST support the use of subscriptions.

3.8.1. Establishing Subscriptions

A SWID-PV establishes a subscription on a particular SWID-PC by sending a SWID Request attribute with the Subscription flag set. The SWID Request attribute is otherwise identical to the SWID Requests discussed in previous sections. Specifically, such a SWID Request might request full tags or tag identifier instances, might be targeted, and might request change event records or endpoint inventory. Assuming no error is encountered, a SWID-PC MUST send a SWID Response attribute in direct response to this SWID Request attribute, just as if the Subscription flag was not set. As such, the message exchange that establishes a new subscription in a SWID-PC has the same flow seen in the previous message exchanges, as depicted in Figure 2. If the SWID-PV does not receive a PA-TNC Error attribute (as described in [Section 3.10](#) and [Section 4.16](#)) in response to their subscription request, the subscription has been successfully established on the SWID-PC. The SWID 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 4.8](#).)

A SWID-PC MUST have the ability to record and support multiple simultaneous subscriptions from a single party and subscriptions from multiple parties. A SWID-PV MUST have the ability to record and support multiple simultaneous subscriptions to a single party and subscriptions to multiple parties.

3.8.2. Managing Subscriptions

The SWID-PC MUST record each accepted subscription along with the identity of the party to whom attributes are to be pushed in compliance with the subscription. If the attribute is received by the SWID-PC using the TNC_IMC_ReceiveMessage ([section 3.8.4](#) of IF-IMC 1.3 [[IF-IMC](#)]) function, then only the connection ID of the SWID-PV's NEA Server is available to the SWID-PC. If the attribute is received by the SWID-PC using the TNC_IMC_ReceiveMessageLong function ([section 3.8.6](#) of IF-IMC) then the SWID-PC has both the NEA Server's connection ID and the Posture Validator ID of the sending SWID-PV. SWID-PCs SHOULD support TNC_IMC_ReceiveMessageLong function calls. SWID-PCs MUST record the connection ID of the SWID-PV's NEA Server for each accepted subscription. SWID-PCs SHOULD record the SWID-PV's Posture Validator ID for each accepted subscription if this information is available.

Likewise, SWID-PVs MUST record each accepted subscription for which they are the subscribing party along with its Subscription ID and the identity of the SWID-PC that will be fulfilling the subscription. The SWID-PV needs to retain this information in order to correctly interpret pushed SWID Response attributes sent in fulfillment of the subscription. As with the SWID-PC, the SWID-PV might only have access to the connection ID of the SWID-PC's endpoint, or might have both this connection ID and the SWID-PC's Posture Collector ID depending on the supported IF-IMV functions [[IF-IMV](#)]. SWID-PVs SHOULD support TNC_IMV_ReceiveMessageLong function calls so as to be capable of receiving the SWID-PC's Posture Collector ID. SWID-PVs MUST record the connection ID of the SWID-PC's endpoint for each accepted subscription. The SWID-PV SHOULD record the SWID-PC's Posture Collector ID for each accepted subscription if this information is available.

3.8.3. Terminating Subscriptions

Subscriptions MAY be terminated at any time by the subscribing SWID-PV by setting the Clear Subscriptions flag in a SWID Request. (See [Section 4.9](#) for more on using this flag.) In the case that the SWID-PC receives both the connection ID and the Posture Validator ID of the SWID-PV requesting that subscriptions be cleared (i.e., the clear subscription request is received via a TNC_IMC_ReceiveMessageLong function) and the SWID-PC has been recording PV IDs associated with subscriptions when available, the SWID-PC MUST only clear subscriptions that match both the connection ID and the PV ID, and MUST clear all such subscriptions. In the case that the SWID-PC only has the connection ID of the party requesting that subscriptions be cleared or the SWID-PC has not been recording Posture Validator IDs associated with subscriptions even when available, it MUST only clear

subscriptions that match the connection ID and that have no associated Posture Validator ID, and MUST clear all such subscriptions.

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

This specification does not give the SWID-PC the ability to unilaterally terminate a subscription. However, if the SWID-PC experiences a fatal error fulfilling a subscription, resulting in sending a PA-TNC Error attribute of type `SWID_SUBSCRIPTION_FULFILLMENT_ERROR`, then the subscription whose fulfillment led to the error MUST be treated as terminated by both the SWID-PC and the SWID-PV. Only the subscription experiencing the error is cancelled and other subscriptions are unaffected. See [Section 3.10](#) for more on this error condition.

Finally, a subscription is terminated if the connection between the SWID-PC and SWID-PV is deleted, as indicated by the connection state changing to DELETE. (Described in [section 3.5.2.2](#) of IF-IMC [[IF-IMC](#)] and [section 3.5.2.2](#) of IF-IMV [[IF-IMV](#)].) Doing this renders the SWID-PC incapable of pushing additional SWID Response attributes to the subscribing party. Both the SWID-PC and SWID-PV MUST delete all subscriptions for which the connection has been deleted. SWID-PCs MUST support the `TNC_IMC_NotifyConnectionChange` function, as defined in IF-IMC 1.3 [section 3.8.2](#), so that the SWID-PC can be informed when a connection's state changes to DELETE. Likewise, a SWID-PV MUST support the `TNC_IMV_NotifyConnectionChange` function, as defined in IF-IMV 1.4 [section 3.8.2](#), for the same reason.

[3.8.4. Subscription Status](#)

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

A SWID-PV requests a list of its subscriptions on a given SWID-PC by sending that SWID-PC a Subscription Status Request. The SWID-PC MUST then respond with a Subscription Status Response (or a PA-TNC Error if an error condition is experienced). The Subscription Status

Response contains one subscription record for each of the active subscriptions for which the SWID-PV is the subscribing party. Specifically, in the case that the Subscription Status Request arrives with both a connection ID and a Posture Validator ID and the SWID-PC has been recording Posture Validator IDs associated with subscriptions when available, the SWID-PC MUST include only subscription records associated with both the given connection ID and Posture Validator ID, and MUST include all such records. In the case that the Subscription Status Request arrives with only a connection ID or the SWID-PC has not been recording Posture Validator IDs associated with subscriptions even when available, the SWID-PC MUST include only subscription records associated with the given connection ID and that have no associated Posture Validator ID, and MUST include all such records.

3.8.5. Fulfilling Subscriptions

As noted in [Section 3.5](#) SWID-PCs MUST have the ability to automatically detect changes to an endpoint's SWID tag collection in near real-time. For every active subscription, the SWID-PC MUST send an attribute to the subscribed SWID-PV whenever a change is detected to relevant tags within the endpoint's SWID tag collection. The SWID-PC MAY choose to exclusively deliver this attribute in the case that the SWID-PV's Posture Validator ID is known. (See [section 3.3.2.2](#) of IF-IMC 1.3 [[IF-IMC](#)] or [section 3.3.2.2](#) of IF-IMV 1.4 [[IF-IMV](#)] for more on exclusive delivery.) Such an attribute is said to be sent "in fulfillment of" the given subscription and any such attribute include that subscription's Subscription ID. If the establishing request for that subscription was a targeted request, then only tags that match the SWID tag identifiers provided in that establishing request are considered relevant. Otherwise, (i.e., for non-targeted requests) any tag is considered relevant for this purpose. Figure 3 shows a sample message exchange where a subscription is established and then later messages are sent from the SWID-PC in fulfillment of the established subscription.

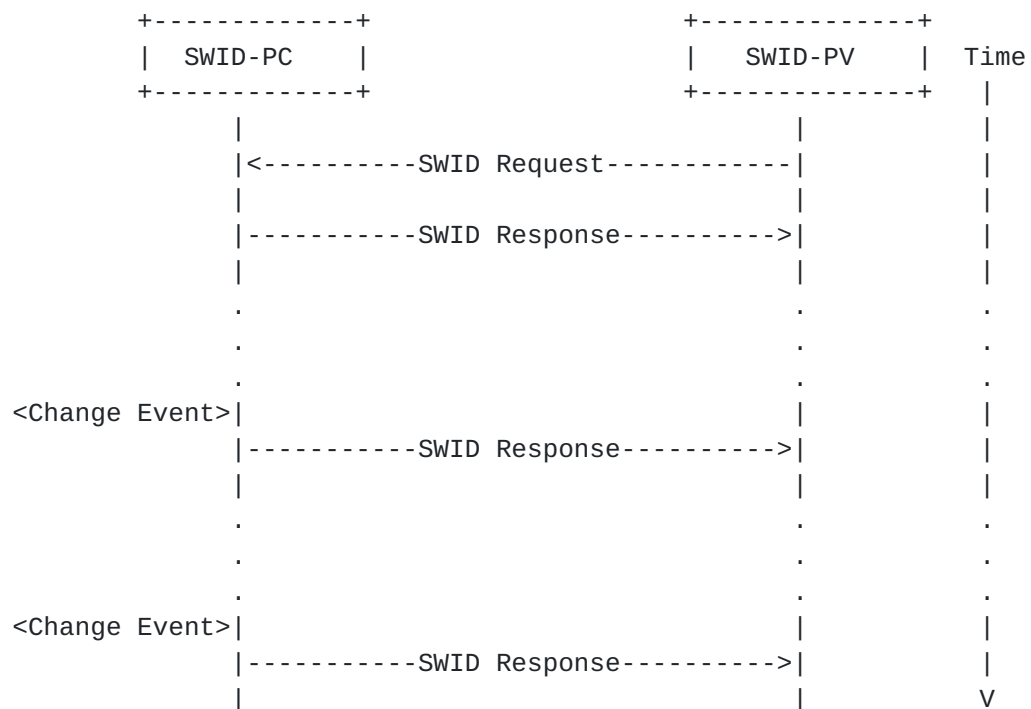


Figure 3: Subscription Establishment and Fulfillment

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 contents of an attribute sent in fulfillment of a subscription have the same format as would a direct response to the establishing request. For example, if the establishing request stipulated a response that contained an event record list wherein affected SWID tags were indicated using SWID tag identifier instances, all attributes sent in fulfillment of this subscription will also consist of event record lists expressed using SWID tag identifier instances. As such, all SWID Responses displayed in the exchange depicted in Figure 3 have the same format. A SWID Response generated in fulfillment of an active subscription MUST be a valid SWID 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 SWID-PV that had the same request parameters and which arrived immediately after the given change event. There are a few special rules that expand on this guideline:

3.8.5.1. Subscriptions Reporting Inventories

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

3.8.5.2. Subscriptions Reporting Events

The way in which a SWID-PV indicates it wishes to establish a subscription requesting event records is by providing a non-zero EID in the SWID Request establishing the subscription (see [Section 3.6.1](#)). However, when the SWID-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 indicated 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 SWID-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 SWID-PC MUST only report events with later EIDs. In the case that the EID Epoch of the SWID-PC changes, the SWID-PC MUST treat EID values in the new Epoch as being after all EIDs assigned in the previous Epoch regardless of the relative numeric values of these EIDs.

Note that while a subscription is active, the subscribing SWID-PV MAY make other requests for event records that overlap with events that are reported due to a subscription. Such requests are unaffected 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 SWID-PV.

A SWID-PV needs to pay attention to the EID Epoch in these messages, as changes in the Epoch might create discontinuities in the SWID-PV's understanding of the endpoint's SWID tag collection state, as discussed in [Section 3.6.5](#). In particular, once the EID Epoch changes, a SWID-PV is unable have confidence that it has a correct understanding of the state of an endpoint's SWID tag collection until after the SWID-PV collects a complete inventory.

SWID-PCs MAY send partial lists of event records in fulfillment of a subscription. (See [Section 3.6.4](#) for more on partial list of event records.) In the case that a SWID-PC sends a partial list of event records, 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 SWID-PC sends a partial list it does not wait for another change event to send another SWID Response, but continues sending SWID Responses until it has sent all event records that would have been included in a complete fulfillment of the subscription.

[3.8.5.3](#). Targeted Subscriptions

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

In addition, a SWID-PC MUST send an attribute in fulfillment of a targeted subscription only when changes to the endpoint's SWID tag collection impact one or more tags matching the subscription's establishing request's tag identifiers. A SWID-PC MUST NOT send any attribute in fulfillment of a targeted subscription based on detected change to the endpoint's SWID tag collection that did not involve any of the tags targeted by that subscription.

3.8.5.4. No Subscription Consolidation

A SWID-PV MAY establish multiple subscriptions to a given SWID-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 SWID-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, SWID-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 SWID-PV.

3.8.5.5. Delayed Subscription Fulfillment

A SWID-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 SWID-PV in a single SWID Response attribute. This can help reduce network bandwidth consumption between the SWID-PC and the SWID-PV. For example, consider a situation where 10 changes occur a tenth of a second apart. If the SWID-PC does not delay in assembling and sending SWID Response attributes, the SWID-PV will receive 10 separate SWID Response attributes over a period of 1 second. However, if the SWID-PC waits half a second after the initial event before assembling a SWID Response, the SWID-PV only receives two SWID 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, SWID-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 SWID tag is deleted and then re-added), the SWID-PC MUST still report each change event rather than just reporting the net effect of changes over the delay period. In other words, delayed fulfillment can decrease the number of attributes sent by the SWID-PC, but it does not reduce the total number of change events reported.

SWID-PCs are not required to support delayed fulfillment of subscriptions. However, in the case that the SWID-PC does support delayed subscription fulfillment, it MUST be possible to configure the SWID-PC to disable delayed fulfillment. In other words, parties

deploying SWID-PCs need to be allowed to disable delayed subscription fulfillment in their SWID-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. Multiple Sources of SWID Tags

As noted in [Section 2.1](#), the SWID tags in an endpoint's SWID tag collection might potentially come from multiple sources. For example, SWID tags might be deposited on the file system and collected therefrom. SWID tags might also be dynamically generated by tools such as software and package managers (e.g., RPM or YUM) or might be dynamically translated from software discovery reports expressed in some non-SWID format.

A SWID-PC is not required to identify every possible source of SWID tags on its endpoint. Some SWID-PCs might be explicitly tied only to one or a handful of SWID tag sources. SWID-PCs are not required to be aware of SWID tags that come from sources other than those that they specifically support. In particular, if an endpoint has 3 sources of SWID tags, and a SWID-PC supports collecting SWID tags from two of those sources, not only is that SWID-PC only responsible for reporting tags that come from its two supported sources, but it is also only responsible for monitoring for change events from those two sources. This noted, for all of the SWID tag sources that a particular SWID-PC supports, it MUST completely support all requirements of this specification with regard to its supported sources. In other words, for supported sources, the SWID-PC is required to be capable of providing complete inventories of SWID tags; monitoring for changes in the SWID collections reported by those sources, correctly providing responses for both full and targeted requests, and providing either complete SWID tag files or SWID identifier instances as appropriate. The SWID-PC MUST NOT provide any inventory or event information from SWID tag sources for which it cannot provide this full support.

The SWID Response attributes provide no way of distinguishing as to which SWID tags, identifier instances, or event records are associated with specific sources. The SWID-PC MUST include the complete set of relevant data from all supported sources of SWID tags in every SWID Response. In other words, a full inventory is required to contain all the SWID tags from all supported sources, a targeted inventory is required to contain all relevant tags from all sources, and event tracking is required to cover all events from both sources. With regard to events, a SWID-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 SWID-PC is required to capture five events with consecutive EID values reflecting the order in which the events occur.

Note that, if a SWID-PC collects data from multiple sources, it is possible that some software products might be "double counted". This can happen if both sources of SWID tags provide a SWID tag for a single instance of a software product. Moreover, each of these provided tags will probably have different SWID tag identifier instances, since Instance IDs are managed by the process that extracts the SWID tags from the individual sources, and such processes are under no obligation to coordinate with each other as to the Instance ID value. When a SWID-PC reports information or records events from multiple SWID tag sources, it MUST use the information those sources provide, rather than attempting to perform some form of reduction. In other words, if multiple sources report a particular SWID tag corresponding to a single installation of a software product, all such tags from each source are required to be part of the SWID-PC's processing even if this might lead to multiple reporting, and the SWID-PC is not to ignore some tags to avoid such multiple reporting. Similarly, in the case that multiple sources report an event, the SWID-PC MUST create separate event records with separate EIDs for each of these, even if there is the chance that they represent the two sources reporting the same action on the endpoint. Entities tracking SWID tags collected via SWID-PCs and SWID-PVs need to be aware that such double-reporting might occur. How (or if) such occurrences are detected and resolved is up to the implementers of those entities.

3.10. Error Handling

In the case where the SWID-PC detects an error in a SWID 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 as well as [Section 4.16](#) in this specification for error codes specific to SWID attributes.) In the cast that an error is detected in a SWID Request the SWID-PC MUST NOT take any action requested by this SWID Request, even if some requested action can be completed successfully despite the error in the attribute. In other words, a SWID Request that contains an error is ignored by the SWID-PC beyond sending a PA-TNC Error attribute, and possibly logging the error locally.

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

In the case that the SWID-PC detects an error that prevents it from properly or completely fulfilling its obligations under an active subscription, the SWID-PC MUST send a PA-TNC Error attribute of type SWID_SUBSCRIPTION_FULFILLMENT_ERROR to the SWID-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 as well as an error "sub-type". The error sub-type is used to indicate the type of error condition the SWID-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 SWID_ERROR error code SHOULD be used in the sub-type. In the case that a SWID_SUBSCRIPTION_FULFILLMENT_ERROR is sent, the associated subscription MUST be treated as cancelled by both the SWID-PC and SWID-PV.

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

Both SWID-PCs and SWID-PVs SHOULD log errors so that administrators can trace the causes of errors. Log messages SHOULD include the type of the error, the time it was detected, and additional descriptive

information to aid in understanding the nature and cause of the error.

4. SWID Message and Attributes for PA-TNC Protocol

This section describes the format and semantics of the SWID Message and Attributes for PA-TNC protocol leveraging the existing SWID tag format. SWID Message and Attributes for PA-TNC uses the standard PA-TNC message header format. See the PA-TNC specification [[RFC5792](#)] for information on this header format.

4.1. PA Subtype (AKA PA-TNC Component Type)

The NEA PB-TNC 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. In order for the NEA protocols to carry SWID tags, this specification adds the following enumeration element to the table in [section 7.2](#) of the PA-TNC specification [[RFC5792](#)] using the IETF Standard name space (SMI Private Enterprise Number 0x000000):

PEN		Integer	Name	Defining Specification
0	9		SWID Attributes	SWID Message and Attributes for PA-TNC

Table 2: PA Subtype

Each PA-TNC attribute described in this specification is intended to be sent between the SWID-PC and SWID-PV, so will be carried in a PB-TNC message indicating a PA Subtype of SWID Attributes. Note that although the PA-TNC Error attribute is defined in the PA-TNC specification, when it is used in a SWID Attribute exchange, it uses the SWID Attributes Component Definition Value, as described in [Section 4.2.8](#) of the PA-TNC specification [[RFC5792](#)]. PB-TNC messages MUST always include the SWID Attributes Subtype defined in this section when carrying SWID Attributes over PA-TNC.

4.2. PB-TNC and PA-TNC Messages

A PA-TNC message is wrapped within a PB-TNC message. A single PA-TNC message might contain one or more PA-TNC attributes. All of these attributes within a single PA-TNC message use the same PA Subtype value. As such, SWID Attributes are never sent with attributes defined in other PA-TNC binding specifications in a single PA-TNC message. Note, however, that a single PB-TNC batch might contain multiple PB-TNC and PA-TNC messages, and each of those messages might use different PA Subtypes.

For more information on PB-TNC and PA-TNC messages and message headers, see the PB-TNC [[RFC5793](#)] and PA-TNC [[RFC5792](#)] specifications, respectively.

4.3. PA-TNC Attribute Header

The SWID Message and Attributes for PA-TNC protocol described in this specification is an extension of the PA-TNC protocol described in the NEA Architecture. PA-TNC was designed to be very flexible in order to carry many types of PA-TNC attributes that pertain to an enumerated set of component types (e.g. Table 2). PA-TNC attributes might be carried from Posture Collector to Posture Validator or vice versa and might carry information about endpoint state or other information to be sent between a Posture Validator and a Posture Collector. Therefore the SWID Message and Attributes for PA-TNC specification defines a collection of PA-TNC attributes relevant to the collection and transmission of SWID tag inventories.

Figure 4, reproduced from the PA-TNC specification, shows the format of a PA-TNC attribute. Multiple PA-TNC attributes can be sent in a single PB-TNC message, each housed within an attribute structure as described below.

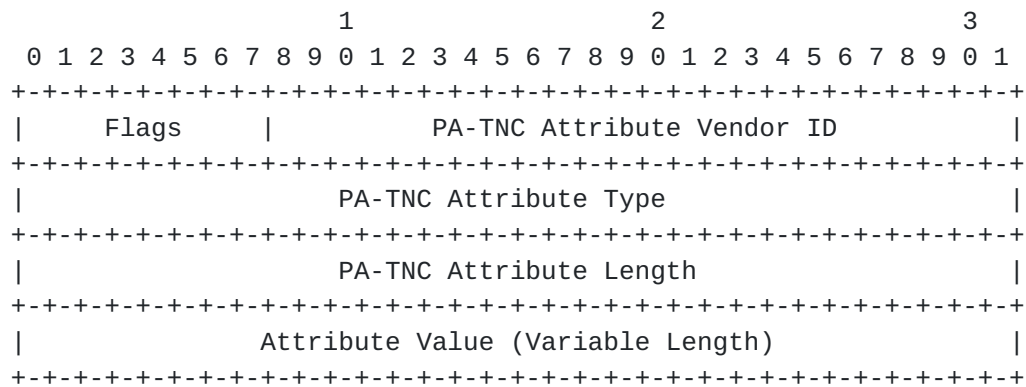


Figure 4: PA-TNC Header and Attribute Format

Field	Description
Flags	This field defines flags affecting the processing of the SWID Message and Attributes for PA-TNC. Permissible flags are given in the PA-TNC specification. [RFC5792]
Attribute Type	This field indicates the owner of the name space associated with the Attribute Type. Attributes defined in the SWID Message and Attributes for PA-TNC specification have a value corresponding to the IETF SMI Private Enterprise Number value (0x000000). The PA-TNC Error attribute is defined in the PA-TNC specification [RFC5792] and also uses the IETF SMI Private Enterprise Number Value (0x000000). See Table 4 for more information.
Attribute Vendor ID	This field defines the type of the Attribute. The values corresponding to SWID Attributes are given in Table 4.
Attribute Length	This field contains the length in octets of the entire Attribute, including the Attribute's header.
Attribute Value	This field contains the SWID Attribute.

Table 3: Fields of the PA-TNC Attribute Format

4.4. SWID Attribute Overview

The attributes defined in this specification appear below with a short summary of their purposes. Each attribute is described in greater detail in subsequent sections.

- o SWID Request - This attribute is used to request a SWID tag inventory or SWID event list from an endpoint. This attribute might also establish a subscription on the recipient SWID-PC. A SWID-PC MUST NOT send this attribute.
- o SWID Tag Identifier Inventory - This attribute is used to convey an inventory expressed using SWID tag identifier instances (instead of full tags). When a SWID-PC receives a SWID Request attribute requesting an inventory using SWID tag identifier instances, the SWID-PC MUST send a SWID Tag Identifier Inventory attribute (or a PA-TNC Error) in response. This attribute also MAY be sent by the SWID-PC in fulfillment of an active subscription. A SWID-PV MUST NOT send this attribute.
- o SWID Tag Identifier Events - This attribute is used to convey a list of events concerning changes to an endpoint's collection of SWID tags. Affected SWID tags are indicated using SWID tag identifier instances (instead of full tags). When a SWID-PC receives a SWID Request attribute requesting an event collection using with SWID tag identifier instances, the SWID-PC MUST send a SWID Tag Identifier Events attribute (or a PA-TNC Error) in response. This attribute also MAY be sent by the SWID-PC in fulfillment of an active subscription. A SWID-PV MUST NOT send this attribute.
- o SWID Tag Inventory - This attribute is used to convey an inventory expressed using full SWID tags (instead of SWID tag identifier instances). When a SWID-PC receives a SWID Request attribute requesting an inventory using full SWID tags, the SWID-PC MUST send a SWID Tag Inventory attribute (or a PA-TNC Error) in response. This attribute also MAY be sent by the SWID-PC in fulfillment of an active subscription. A SWID-PV MUST NOT send this attribute.
- o SWID Tag Events - This attribute is used to convey a list of events concerning changes to an endpoint's collection of SWID tags. Affected SWID tags are indicated using full SWID tags (instead of SWID tag identifier instances). When a SWID-PC receives a SWID Request attribute requesting an event collection using full SWID tags, the SWID-PC MUST send a SWID Tag Events attribute (or a PA-TNC Error) in response. This attribute also MAY be sent by the SWID-PC in fulfillment of an active subscription. A SWID-PV MUST NOT send this attribute.
- o Subscription Status Request - This attribute is used to request a SWID-PC send a summary of all the active subscriptions it has where the requesting party is the subscriber. The SWID-PC MUST

respond with a Subscription Status Response (or a PA-TNC Error). A SWID-PC MUST NOT send this attribute.

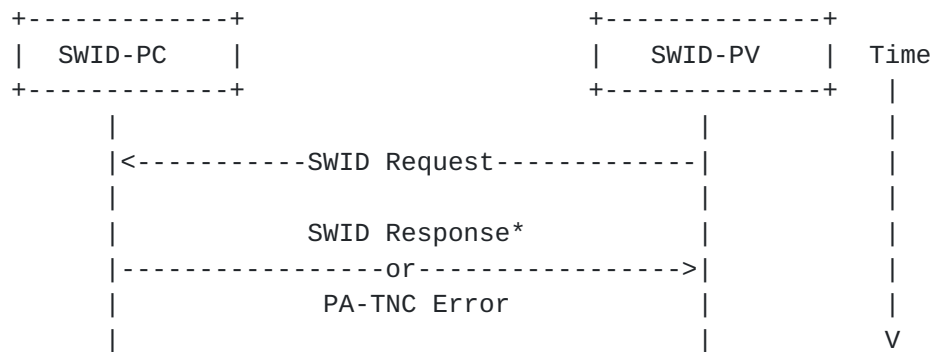
- o Subscription Status Response - This attribute is used to convey information about the active subscriptions that a SWID-PC has for a given subscriber. A SWID-PV MUST NOT send this attribute.
- o PA-TNC Error - This is the standard PA-TNC Error attribute as defined in PA-TNC [[RFC5792](#)] and is used to indicate that an error was encountered during a SWID Attribute exchange. It MUST be sent by a SWID-PC in response to a SWID Request in the case where the SWID-PC encounters a fatal error (i.e., an error that prevents further processing of an exchange) relating to the attribute exchange. A SWID-PV MUST NOT send this attribute. The SWID-PC MUST then ignore the erroneous attribute after a PA-TNC Error attribute is sent (i.e., do not attempt to act on an attribute that generated a PA-TNC Error beyond sending the PA-TNC Error). In the case where the SWID-PV experiences a fatal error, it MUST ignore the erroneous attribute without sending a PA-TNC Error attribute. It 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

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

All SWID-PVs MUST be capable of sending SWID Requests and be capable of receiving and processing all SWID Response attributes as well as PA-TNC Error attributes. All SWID-PCs MUST be capable of receiving and processing SWID Requests and be capable of sending all types of SWID Response attributes as well as PA-TNC Error attributes. In other words, both SWID-PVs and SWID-PCs are required to support their role in exchanges using any of the attribute types defined in this section. SWID-PVs MUST ignore any SWID Request attributes that they receive. SWID-PCs MUST ignore any SWID Response attributes or PA-TNC Error attributes that they receive.

[4.5.](#) SWID Attribute Exchanges

A SWID Attribute Exchange is used to provide the SWID-PV with a SWID tag inventory or event collection from the queried endpoint.

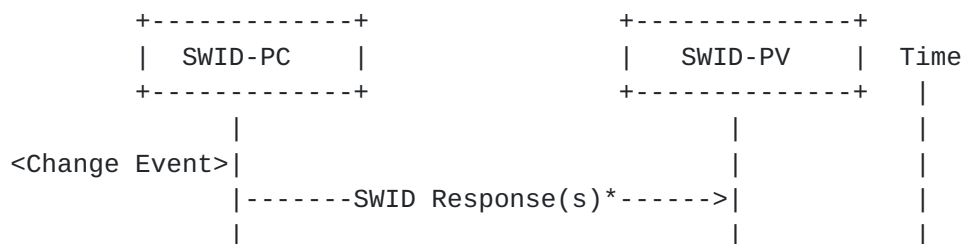


*SWID Response is one of the following: SWID Tag Identifier Inventory, SWID Tag Identifier Events, SWID Tag Inventory, or SWID Tag Events.

Figure 5: SWID Attribute Exchange (Direct Response to SWID Request)

In this exchange, the SWID-PV indicates to the SWID-PC, via a SWID 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 (full tags or tag identifier instances). The SWID-PC responds with the requested information using the appropriate attribute type. A single SWID Request **MUST** only lead to a single SWID Response or PA-TNC Error that is in direct response to that request.

In addition, if there is an active subscription on the endpoint, the SWID-PC sends a SWID Response to the SWID-PV following a change event on the endpoint in fulfillment of that subscription. Such an exchange is shown in Figure 6.



*SWID Response is one of the following: SWID Tag Identifier Inventory, SWID Tag Identifier Events, SWID Tag Inventory, or SWID Tag Events.

Figure 6: SWID Attribute Exchange (In Fulfillment of an Active Subscription)

Note that, unlike direct responses to a SWID Request, a single change event can precipitate multiple SWID Responses, but only if all but the last of those SWID Responses convey partial lists of event records, and the last of those SWID Responses conveys a complete list of event records. (That is, the initial responses are partial lists and the last response is the remainder of the relevant event records, completing the delivery of all relevant events at the time of the change event.) A single Change Event MUST NOT be followed by multiple SWID Response or PA-TNC Error attributes in any combination except as noted earlier in this paragraph.

All SWID-PVs and SWID-PCs MUST support both exchanges. In particular, SWID-PCs MUST be capable of pushing a SWID Response to a SWID-PV immediately upon detection of a change to the endpoint's SWID tag collection in fulfillment of established SWID-PV subscriptions, as described in [Section 3.8](#).

[4.6](#). SWID Message and Attributes for PA-TNC Attribute Enumeration

PA-TNC attribute types are identified in the PA-TNC Attribute Header (see [Section 4.2](#)) via the Attribute Type Vendor ID and Attribute Type fields. Table 4 identifies the appropriate values for these fields for each attribute type used within the SWID Message and Attributes for PA-TNC protocol.

Attribute Name	PEN	Integer	Description
SWID Request	0x000000	0x00000011	Request from a SWID-PV to a SWID-PC for the SWID-PC to provide a SWID tag inventory or event list
SWID Tag Identifier	0x000000	0x00000012	A collection of SWID tag identifier instances sent from a SWID-PC.
SWID Tag Inventory	0x000000	0x00000013	A collection of events impacting the endpoint's SWID tag collection, where impacted SWID tags are indicated using SWID tag identifier instances.
SWID Tag Events	0x000000	0x00000014	A collection of SWID tags sent from a SWID-PC.
SWID Tag Identifier	0x000000	0x00000015	A collection of events impacting the endpoint's SWID tag collection, where impacted SWID tags are indicated using full SWID tags.
Subscription Status	0x000000	0x00000016	A request for a list of a SWID-PV's active subscription.
Subscription Request	0x000000	0x00000017	A list of a SWID-PV's active subscriptions.
Subscription Status Response	0x000000	0x00000018	These attribute types are reserved for future use in revisions to SWID Message and Attributes for PA-TNC.
Reserved	0x000000	0x0000001F	
PA-TNC Error	0x000000	0x00000008	An error attribute as defined in the PA-TNC specification [RFC5792].

Table 4: SWID Attribute Enumeration

4.7. Normalization of Text Encoding

SWID tags do not have a required encoding format. The 2009 ISO SWID specification states that "For encoding purposes, the use of utf-8 is the suggested methodology for software identification tags...", but leaves implementers free to use different encodings if this makes

sense in their local environment. As such, implementers of the SWID Message and Attributes for PA-TNC specification cannot assume any specific encoding of SWID tag fields (although, in most current examples of SWID tags, SWID tag creators have followed the suggestion of using UTF-8 encodings). Similarly, sometimes the SWID Message and Attributes for PA-TNC specification requires the use of data taken from other sources, such a path from the endpoint's file system, and different platforms might use different encodings for this information. In order to ensure the ability to consistently and reliably compare information sent using the SWID Message and Attributes for PA-TNC exchange, certain field values (identified explicitly in the attribute definitions in the following sections) are required to undergo normalization prior to their inclusion in an attribute.

In order to ensure consistency of transmitted attributes, a field requiring normalization, as indicated in its description, and only such fields **MUST** be normalized to Network Unicode format as defined in [RFC 5198](#) [[RFC5198](#)]. Network Unicode format defines a refinement of UTF-8 that ensures a normalized expression of characters. SWID-PCs and SWID-PVs **MUST NOT** perform conversion and normalization on any field values except those specifically identified in the following sections.

[4.8.](#) Request IDs

All SWID 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 SWID-PV and the receiving SWID-PC. Specifically, the SWID-PV assigns each SWID Request attribute a Request ID value that is intended to be unique within the lifetime of a given network connection ID as assigned by the SWID-PV's Posture Broker Server (as described in [section 3.5.2.2](#) of the IF-IMV specification [[IF-IMV](#)]). 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 that are not being used as Subscription IDs. (See below in this section for an explanation of Subscription ID assignment.) In this case of Request ID reuse, Request IDs **SHOULD** be reused in the order of their original use. For example, if a Request ID of X was the first Request ID used within a particular network connection and if the Request IDs are exhausted, X will be the first reused Request ID. In other words, a SWID-PC **SHOULD NOT** use a given Request ID value more than once within a persistent connection between a given Posture Broker Client-Posture Broker Server pair, but, in the case where reuse is necessary due to exhaustion of possible ID values, the SWID-PC **SHOULD** structure the reuse to maximize the time between original and subsequent use. The

Request ID value is included in a response attribute directly responding to this SWID Request to indicate which SWID 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. SWID-PCs are not required to check for duplicate Request IDs.

In the case that a SWID Request requests the establishment of a subscription and the receiving SWID-PC agrees to that subscription, the Request ID of that SWID 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 SWID-PV MUST NOT reuse a Request ID value in communicating to a given SWID-PC while that Request ID is also serving as a Subscription ID for an active subscription with that SWID-PC. In the case where a SWID-PC receives a SWID Request from a given SWID-PV where that Request ID is also the Subscription ID of an active subscription with that SWID-PV, the SWID-PC MUST respond with a PA-TNC Error attribute with an error code of SWID_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.

4.9. SWID Request

A SWID-PV sends this attribute to a SWID-PC to request that the SWID-PC send SWID tag-based information to the SWID-PV. A SWID-PC MUST NOT send this attribute.

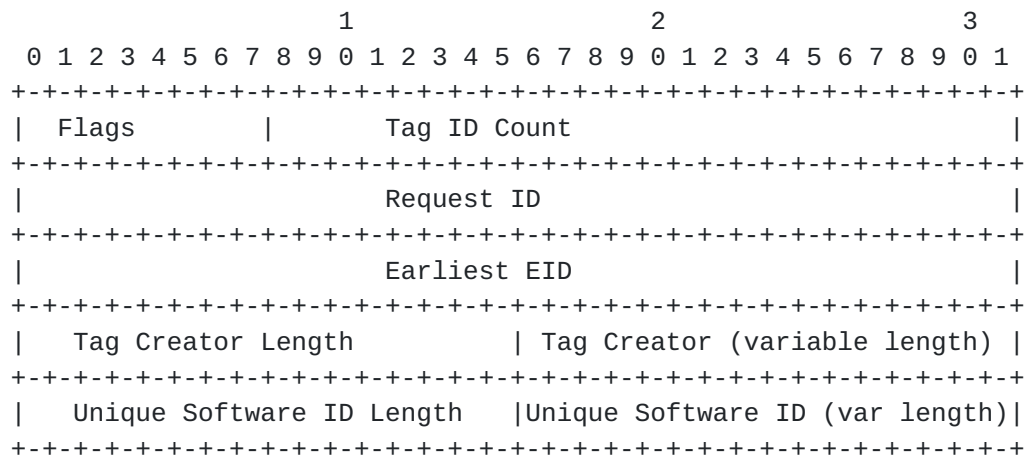


Figure 7: SWID Request Attribute

Field	Description
Flags: Bit 0	If set (1), the SWID-PC MUST delete all
- Clear	subscriptions established by the requesting SWID-
Subscriptions	PV (barring any errors).
Flags: Bit 1	If set (1), in addition to responding to the
- Subscribe	request as described, the SWID-PC MUST establish
	a subscription with parameters matching those in
	the request attribute (barring any errors).
Flags: Bit 2	If unset (0), the SWID-PC's response MUST consist
- Result Type	of complete SWID tags and thus the response MUST
	be a SWID Tag Inventory, a SWID Tag Events, or a
	PA-TNC Error attribute. If set (1), the response
	MUST consist of SWID tag identifier instances and
	thus the response MUST be a SWID Tag Identifier
	Inventory, a SWID Tag Identifier Events, or a PA-
	TNC Error attribute.
Flags: Bit	Reserved for future use. This field MUST be set
3-7 -	to zero on transmission and ignored upon
Reserved	reception.
Tag ID Count	A 3-byte unsigned integer indicating the number
	of tag identifiers that follow. If this value is
	non-zero, this is a targeted request, as
	described in Section 3.4 . This field is a 3-byte
	unsigned integer. The Tag Creator Length, Tag
	Creator, Unique Software ID Length, and Unique
	Software ID fields are repeated, in order, the
	number of times indicated in this field. In the
	case where tag identifiers are present, the SWID-
	PC MUST only respond with SWID tags or tag

	identifier instances that correspond to the
	identifiers the SWID-PV provided in this
	attribute (or with a PA-TNC Error attribute).
	This field value MAY be 0, in which case there
	are no instances of the Tag Creator Length, Tag
	Creator, Unique Software ID Length, and Unique
	Software ID fields. In this case, the SWID-PV is
	indicating an interest in all SWID tags on the
	endpoint (i.e., this is not a targeted request).
Request ID	A value that uniquely identifies this SWID
	Request from a particular SWID-PV.
Earliest EID	In the case where the SWID-PV is requesting SWID
	events, this field contains the EID value of the
	earliest event the SWID-PV wishes to have
	reported. (Note - the report will be inclusive of
	the event with this EID value.) In the case where
	the SWID-PV is requesting an inventory, then this
	field MUST be 0. (0x00000000) In the case where
	this field is non-zero, the SWID-PV is requesting
	events and the SWID-PC MUST respond using a SWID
	Tag Events, SWID Tag Identifier Events, or a PA-
	TNC Error attribute. In the case where this field
	is zero, the SWID-PV is requesting an inventory
	and the SWID-PC MUST respond using a SWID Tag
	Inventory, a SWID Tag Identifier Inventory, or a
	PA-TNC Error attribute.
Tag Creator	A 2-byte unsigned integer indicating the length
Length	in bytes of the Tag Creator field.
Tag Creator	A string containing the Tag Creator RegID value
	from within a SWID tag. This field value MUST be
	normalized to Network Unicode format, as
	described in Section 4.7 . This string MUST NOT
	be NULL terminated.
Unique	A 2-byte unsigned integer indicating the length
Software ID	in bytes of the Unique Software ID field.
Length	
Unique	A string containing the Unique ID value from
Software ID	within a SWID tag. This field value MUST be
	normalized to Network Unicode format, as
	described in Section 4.7 . This string MUST NOT be
	NULL terminated.

Table 5: SWID Request Attribute Fields

The SWID-PV sends the SWID Request attribute to a SWID-PC to request the indicated information. Note that between the Result Type flag and the Earliest EID field, the SWID-PC is constrained to a single

possible SWID Response attribute type (or a PA-TNC Error attribute) in its response to the request.

The Subscribe and Clear Subscription flags are used to manage subscriptions for the requesting SWID-PV on the receiving SWID-PC. Specifically, an attribute with the Subscribe flag set seeks to establish a new subscription by the requesting SWID-PV to the given SWID-PC, while an attribute with the Clear Subscription flag seeks to delete all existing subscriptions by the requesting SWID-PV on the given SWID-PC. Note that, in the latter case, only the subscriptions associated with the Connection ID and, if available, the Posture Validator ID of the requester are deleted as described in [Section 3.8.3](#). A newly established subscription has the parameters outlined in the 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 tag identifiers (if present) indicate if and how a subscription is targeted. In the case that the SWID-PC is unable or unwilling to comply with the SWID-PV's request to establish or clear subscriptions, the SWID-PC MUST respond with a PA-TNC Error attribute with the SWID_SUBSCRIPTION_DENIED_ERROR error code. (Note that if the SWID-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 SWID-PC in addition to setting up the subscription. A SWID-PC 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 Subscription flag - the SWID-PC MUST generate a response attribute without regard to the presence of this flag in addition to clearing its subscription list.

Both the Subscribe and Clear Subscription flags MAY be set in a single SWID Request attribute. In the case where this request is successful, the end result MUST be equivalent to the SWID-PC clearing its subscription list for the given SWID-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 SWID-PC MUST respond with a SWID Response attribute. (The specific type of SWID Response attribute depends on the Result Type and Earliest EID fields, as described above.) In the case where there is a failure that prevents some part this request from completing, the SWID-PC MUST NOT add a new subscription, MUST NOT clear the old subscriptions, and the SWID-PC MUST respond with a PA-TNC Error

attribute. In other words, the SWID-PC MUST NOT partially succeed at implementing such a request; either both actions succeed, or neither succeed.

The Earliest EID field is used to indicate whether the SWID-PV is requesting an inventory or event list from the SWID-PC. A value of 0 (0x00000000) represents a request for inventory information. Otherwise, the SWID-PV is requesting event information. For Earliest EID values other than 0, the SWID-PC's response MUST respond with event records, as described in [Section 3.6](#). Note that the request does not identify a particular EID Epoch, since responses can only include events in the SWID-PC's current EID Epoch.

The Tag ID Count indicates the number of tag identifiers in the attribute. This number might be any value between 0 and 16,777,216, inclusive. A single tag identifier is represented by four fields: Tag Creator Length, Tag Creator, Unique Software ID Length, and Unique Software ID. The two length fields are used to indicate the number of bytes allocated to their corresponding string field. The two string fields, Tag Creator and Unique Software ID, contain copies of the SWID tag's Tag Creator RegID and Unique ID values, respectively, converted and normalized to Network Unicode format, as described in [Section 4.7](#). Note that there is no field to indicate a particular Instance ID. Thus, targeted requests request all instances of the indicated SWID tags. The presence of one or more tag identifiers is used by the SWID-PV to indicate a targeted request, which seeks only inventories of or events affecting SWID tags corresponding to the given identifiers. The SWID-PC MUST only respond with tags that match the tag identifier structures provided in the SWID-PV's SWID Request attribute (as described in [Section 3.3.3](#)) and MUST include all instances of matching tags in its response.

[4.10](#). SWID Tag Identifier Inventory

A SWID-PC sends this attribute to a SWID-PV to convey a list of the endpoint's SWID tags expressed using SWID tag identifier instances. This list might represent a complete inventory or a targeted list of tags, depending on the parameters in the SWID-PV's request. A SWID-PV MUST NOT send this attribute. The SWID-PC either sends this attribute in fulfillment of an existing subscription where the establishing request has a Result Type of 1 and the Earliest EID is zero, or in direct response to a SWID Request attribute where the Result Type is 1 and the Earliest EID is zero.

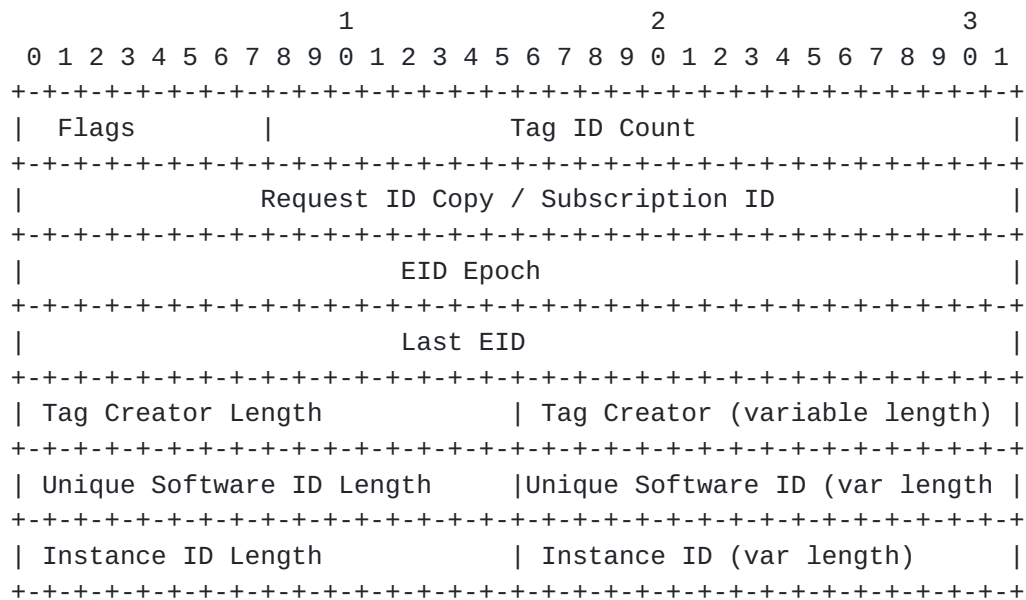


Figure 8: SWID Tag Identifier Inventory Attribute

Field	Description
Flags: Bit 0	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 SWID Request, this bit MUST be unset (0).
Flags: Bit 1-7 - Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Tag ID Count	The number of tag identifier instances that follow. This field is an unsigned integer. The Tag Creator Length, Tag Creator, Unique Software ID Length, Unique Software ID, Instance ID Length, and Instance ID 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 SWID Request attribute from a SWID-PV, this field MUST contain an exact copy of the Request ID field from that SWID 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 an unsigned 4-byte integer.
Last EID	The EID of the last event recorded by the SWID-PC, or 0 if the SWID-PC has no recorded events. This field is an unsigned 4-byte integer.
Tag Creator Length	A 2-byte unsigned integer indicating the length in bytes of the Tag Creator field.
Tag Creator	A string containing the Tag Creator RegID value from within a SWID tag. This field value MUST be normalized to Network Unicode format, as described in Section 4.7 . This string MUST NOT be NULL terminated.
Unique Software ID Length	A 2-byte unsigned integer indicating the length in bytes of the Unique Software ID field.
Unique Software ID	A string containing the Unique ID value from within a SWID tag. This field value MUST be normalized to Network Unicode format, as described in Section 4.7 . This string MUST NOT be NULL terminated.
Instance ID Length	A 2-byte unsigned integer indicating the length in bytes of the Instance ID field.
Instance ID	A string containing the Instance ID of a given tag instance. The exact value of this field depends on the party that provides this SWID tag. This field value MUST be normalized to Network Unicode format, as described in Section 4.7 . This string MUST NOT be NULL terminated.

Table 6: SWID Tag 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 SWID Request, the Subscription Fulfillment bit MUST be unset (0). Note that the SWID response attribute sent in direct response to a SWID Request that establishes a subscription (i.e., a subscription's establishing request) MUST be treated as a direct response to that SWID Request (and thus the Subscription Fulfillment bit is unset). SWID 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 Tag ID Count field indicates the number of tag identifier instances present in this inventory. Each tag identifier instance is represented by a set of six fields: Tag Creator Length, Tag Creator, Unique Software ID Length, Unique Software ID, Instance ID Length,

and Instance ID. These six fields, collectively referred to as the "Tag ID Fields", will appear once for each reported tag instance. Note that an endpoint's SWID tag collection might contain multiple instances of a single tag (i.e., multiple tag files with the same tag identifier value). When this occurs, in the case where that tag is reported, then the response MUST contain a set of Tag ID Fields for each instance of that tag. (The tag might not be reported if the SWID-PV made a targeted request that does not match that tag's tag identifier.) For example, if an endpoint has three copies of tag X, and the SWID-PV requests a full inventory, then the response is required to include three sets of Tag ID Fields corresponding to the three instances of that tag. Only the Instance ID fields are different between these three instances.

When responding directly to a SWID Request attribute, the Request ID Copy / Subscription ID field MUST contain an exact copy of the Request ID field from that SWID Request. When this attribute is sent in fulfillment of an existing subscription on this Posture Collector, then 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 3.6](#) 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, this field MUST contain 0. These fields can be interpreted to indicate that the provided inventory (be it full or targeted) reflects the record of events on the endpoint after all changes up to and including this last event have been accounted for.

4.11. SWID Tag Identifier Events

A SWID-PC sends this attribute to a SWID-PV to convey events where the affected SWID tags are expressed using SWID tag identifier instances. A SWID-PV MUST NOT send this attribute. The SWID-PC either sends this attribute in fulfillment of an existing subscription where the establishing request has a Result Type is 1 and the Earliest EID is non-zero, or in direct response to a SWID Request attribute where the Result Type is 1 and the Earliest EID is non-zero.

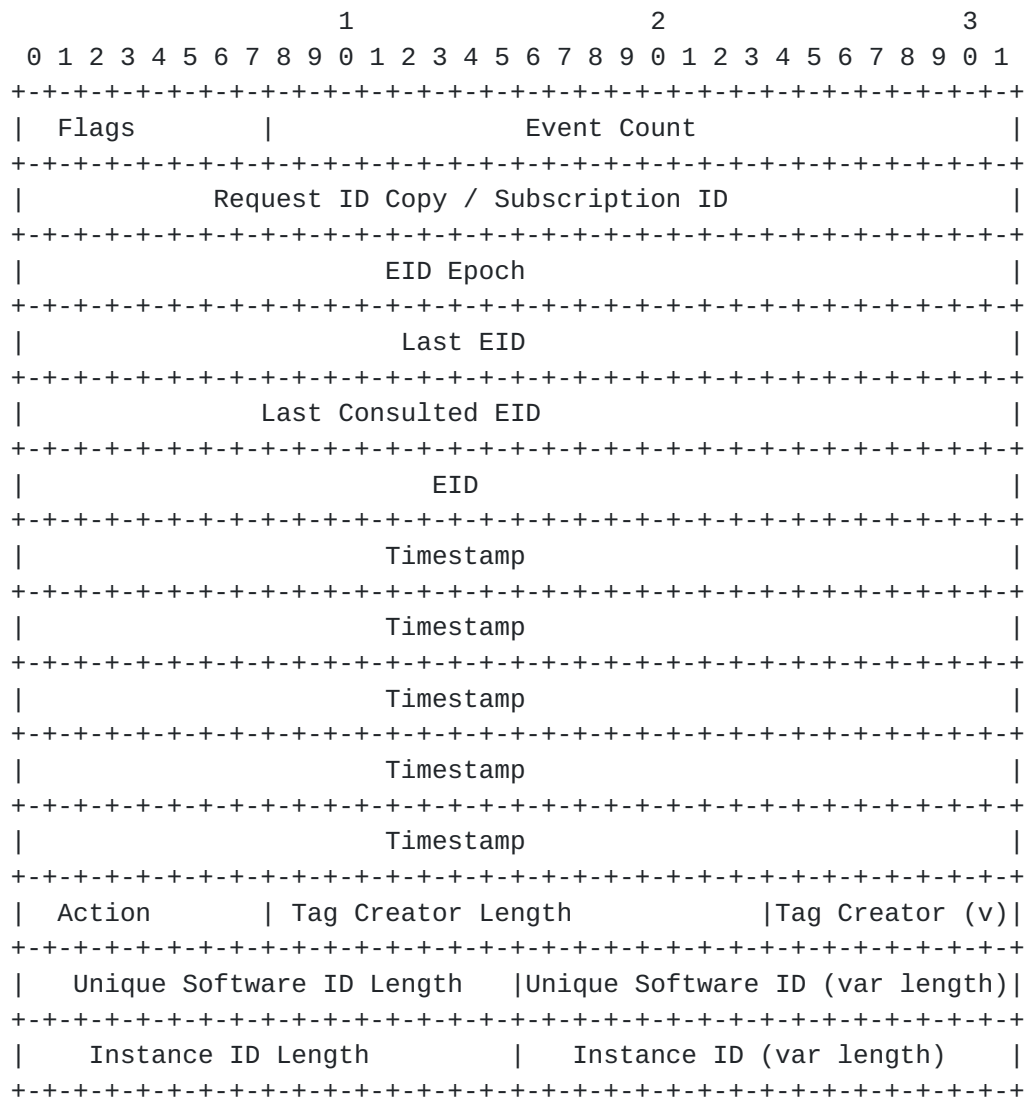


Figure 9: SWID Tag Identifier Events Attribute

Field	Description
Flags: Bit 0	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 SWID Request, this bit MUST be unset (0).
Flags: Bit 1-7	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, Action, Tag Creator Length, Tag Creator, Unique Software ID Length, Unique Software ID, Instance ID Length, and Instance ID fields are repeated, in order, the number of times indicated in this field. (An instance of these nine fields is referred to as an "event record" in this attribute. Thus the Event Count field indicates the number of event records.) 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 SWID Request attribute from a SWID-PV, this field MUST contain an exact copy of the Request ID field from that SWID 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 an unsigned 4-byte integer.
Last EID	The EID of the last event recorded by the SWID-PC, or 0 if the SWID-PC has no recorded events. This field contains the EID of the SWID-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 3.6.4 for more on partial list 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 SWID-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 RFC 3339 [5] compliant ASCII string in Coordinated Universal Time (UTC) time 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 [RFC3339] with the above restrictions. Leap seconds are permitted and SWID-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.
Action	The type of event that is recorded in this event record. Possible values are: 1 = CREATION - the addition of a tag to the endpoint's SWID tag collection; 2 = DELETION - the removal of a tag from the endpoint's SWID tag collection; 3 = ALTERATION - There was an alteration to a tag file within the endpoint's tag collection. All other values are reserved for future use and MUST NOT be used when sending attributes. In the case where a SWID-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.
Tag Creator Length	A 2-byte unsigned integer indicating the length in bytes of the Tag Creator field of the tag affected by the described event.
Tag Creator	A string containing the Tag Creator RegID value from within the SWID tag affected by the described event. This field value MUST be normalized to Network Unicode format, as described in Section 4.7. This string MUST NOT be NULL terminated.
Unique Software ID Length	A 2-byte unsigned integer indicating the length in bytes of the Unique Software ID field of the tag affected by the described event.
Unique Software ID	A string containing the Unique ID value from within the SWID tag affected by the described event. This field value MUST be normalized to Network Unicode format, as described in Section 4.7. This string MUST NOT be NULL terminated.
Instance ID Length	A 2-byte unsigned integer indicating the length in bytes of the Instance ID field.
Instance ID	A string containing the Instance ID of a given tag instance. The exact value of this field depends on the party that provides this SWID tag. This field value MUST be normalized to Network Unicode format, as described in Section 4.7 . This string MUST NOT be NULL terminated.

Table 7: SWID Tag Identifier Events Attribute Fields

The first few fields in the SWID Tag Identifier Events attribute mirror those in the SWID Tag Identifier Inventory attribute. The primary difference is that, instead of conveying an inventory using tag identifier instances, the attribute conveys zero or more event

records, including the EID, timestamp, action type, and tag identifier instance of the affected tag.

With regard to the Timestamp field, it is important to note that clock skew between the SWID-PC and SWID-PV as well as between different SWID-PCs within an enterprise might make correlation of timestamp values difficult. This specification does not attempt to resolve clock skew issues, although other mechanisms outside of this specification do exist to reduce the impact of clock skew and make the timestamp more useful for such correlation. Instead, SWID Message and Attributes for PA-TNC uses 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 SWID tag was removed and then subsequently re-added, one can get an indication as to how long the system was without the given tag (and, thus without the associated software). Since this will involve comparison of timestamp values all originating on the same system, clock skew between the SWID-PC and SWID-PV is not an issue. However, if the SWID-PC's clock was adjusted between two recorded events, it is possible for such a calculation to lead to incorrect understandings of 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 event records in a Tag Identifier Events Attribute are required to be part of the same EID Epoch. Specifically, all reported events **MUST** have an EID from the same EID Epoch as each other, and which is the same as the EID Epoch of the Last EID and Last Consulted EID values. The SWID-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 3.6.4](#)) this field value will differ from that of the Last EID field; 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 SWID Tag 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 tag identifier instance might appear multiple times in an attribute, such as if multiple events involving the associated tag were being reported.

4.12. SWID Tag Inventory

A SWID-PC sends this attribute to a SWID-PV to convey a list of SWID tags. A SWID-PV MUST NOT send this attribute. The SWID-PC either sends this attribute in fulfillment of an existing subscription where the establishing request had a Result Type of 0 and the Earliest EID is zero, or in direct response to a SWID Request attribute where the Result Type is 0 and the Earliest EID is zero.

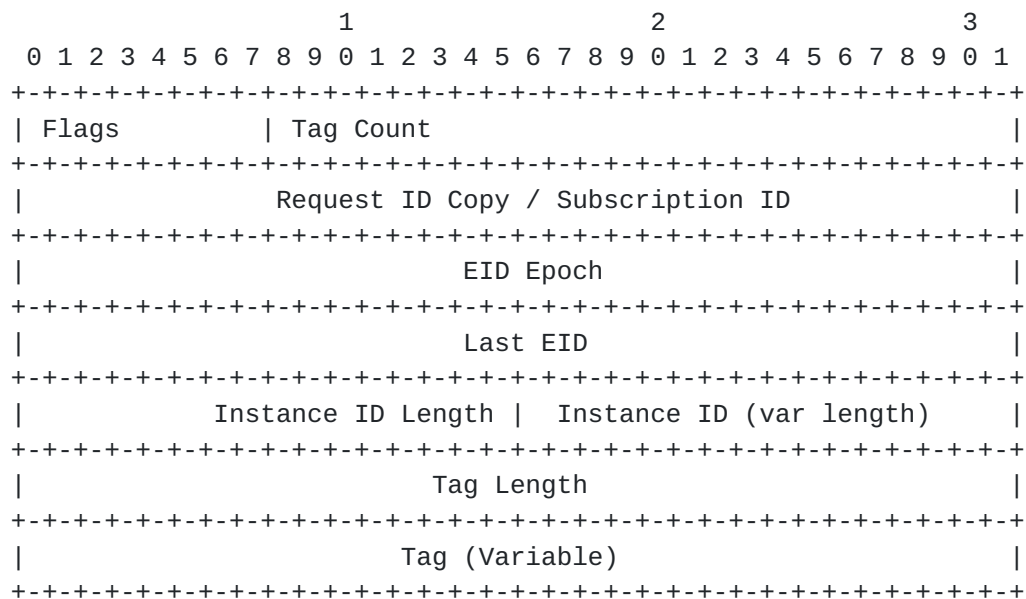


Figure 10: SWID Tag Inventory Attribute

Field	Description
Flags: Bit 0	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 SWID Request, this bit MUST be unset (0).
Flags: Bit 1-7	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Tag ID Count	The number of tags that follow. This field is a 3-byte unsigned integer. The Instance ID Length, Instance ID, Tag Length, and Tag 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	In the case where this attribute is in direct
Copy /	response to a SWID Request attribute from a SWID-
Subscription	PV, this field MUST contain an exact copy of the
ID	Request ID field from that SWID 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
	an unsigned 4-byte integer.
Last EID	The EID of the last event recorded by the SWID-PC,
	or 0 if the SWID-PC has no recorded events. This
	field is an unsigned 4-byte integer.
Instance ID	A 2-byte unsigned integer indicating the length in
Length	bytes of the Instance ID field.
Instance ID	A string containing the Instance ID of a given tag
	instance. The exact value of this field depends on
	the party that provides this SWID tag. This field
	value MUST be normalized to Network Unicode
	format, as described in Section 4.7 . This string
	MUST NOT be NULL terminated.
Tag Len	This is a 4-byte unsigned integer indicating the
	length of the following SWID tag in bytes.
Tag	A SWID tag as a string. In the case where the
	original SWID tag is not expressed using UTF-8
	encoding, it MUST be converted and normalized to
	Network Unicode format, as described in Section
	4.7. However, in the case where the original SWID
	tag is expressed using UTF-8 encoding, the SWID
	tag MUST be copied to this field without
	modification, even if the original SWID tag does
	not conform fully to Network Unicode format. This
	string MUST NOT be NULL terminated.

Table 8: SWID Tag Inventory Attribute Fields

The SWID Tag Inventory attribute contains some number of SWID tags. Given that the size of tags can vary considerably, the length of this attribute is highly variable and, if transmitting a complete inventory, can be extremely large. Enterprises might wish to constrain the use of SWID Tag Inventory attributes to targeted requests to avoid over-burdening the network unnecessarily.

Note that the Instance ID is included in this attribute along with the tag. This is because, unlike the Tag Creator RegID and Unique ID fields that make up the tag identifier, the Instance ID cannot always be extracted from fields within a SWID tag. As such, in order to be

able to associate a tag file with a given tag identifier instance, it is necessary to include the Instance ID value in the attribute.

When copying a SWID tag into the Tag field, conversion and normalization of the character encoding happens if and only if the source SWID tag does not use UTF-8 encoding. In the case where the source SWID tag is expressed using an encoding other than UTF-8, then that tag **MUST** be converted and normalized to use Network Unicode format prior to its inclusion in the tag field. However, in the case where the source SWID tag is expressed in UTF-8, the source tag **MUST** be copied to the Tag field without conversion or normalization. This is true even if the source SWID tag uses UTF-8 but is not fully conformant with Network Unicode format. This is done because any conversion or normalization of a full SWID tag is likely to break any cryptographic signatures included in the SWID tag. As such, conversion only happens to ensure a SWID tag is readable for the recipient (by ensuring it always uses UTF-8), but is otherwise avoided if possible. Recipients of this attribute can always be assured that the Tag field uses UTF-8 format, but cannot depend on full Network Unicode format compliance.

4.13. SWID Tag Events

A SWID-PC sends this attribute to a SWID-PV to convey a list of events where the affected SWID tags are expressed using full tags. A SWID-PV **MUST NOT** send this attribute. The SWID-PC either sends this attribute in fulfillment of an existing subscription where the establishing request has a Result Type of 0 and the Earliest EID is non-zero, or in direct response to a SWID Request attribute where the Result Type is 0 and the Earliest EID is non-zero.

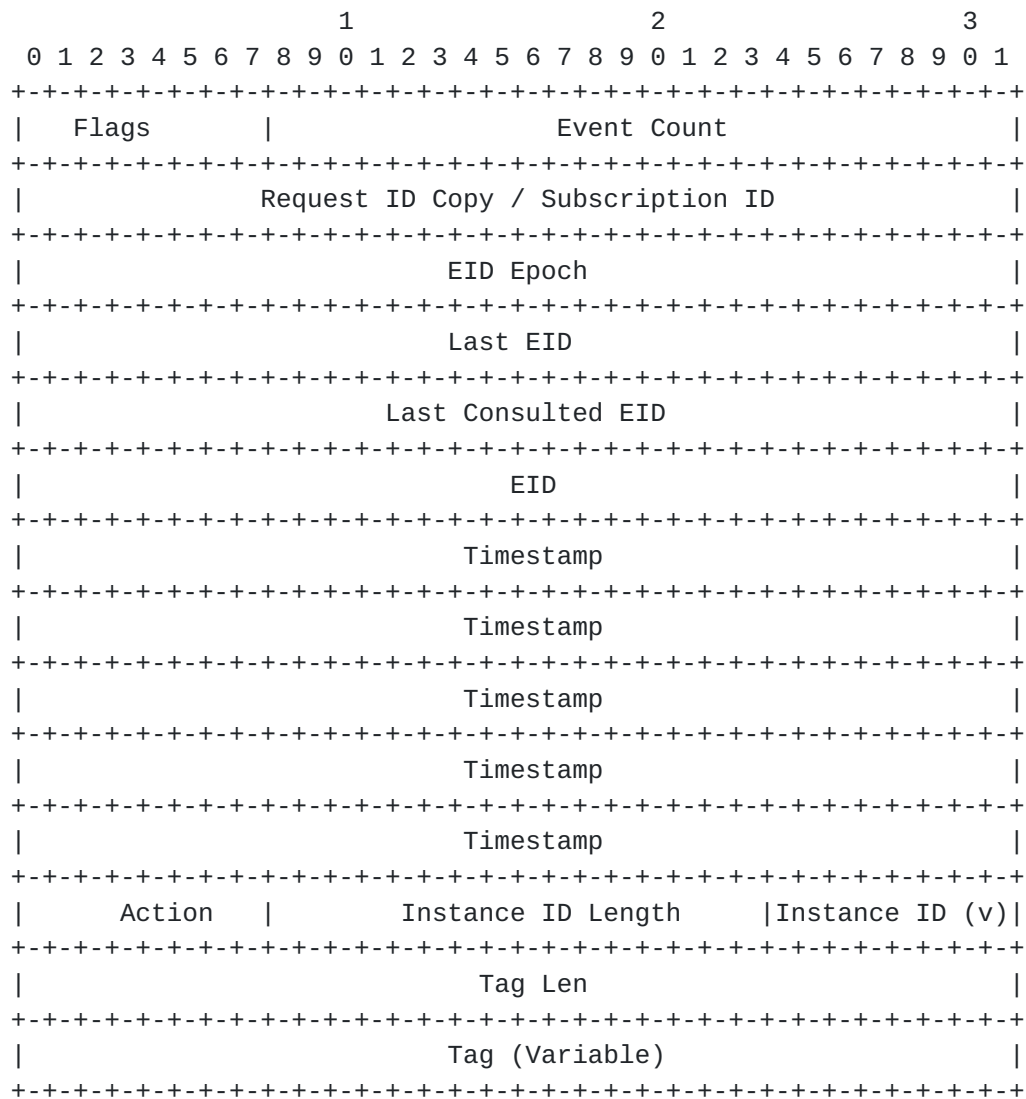


Figure 11: SWID Tag Events Attribute

Field	Description
Flags: Bit 0	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 SWID Request, this bit MUST be unset (0).
Flags: Bit 1-7	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, Action, Instance ID Length, Instance ID, Tag Length, and Tag fields are repeated, in order, the number of times indicated in this field. (An instance of these five fields is referred to as an "event record" in this attribute. Thus the Event Count field indicates the number of event records.) 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 SWID Request attribute from a SWID-PV, this field MUST contain an exact copy of the Request ID field from that SWID 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 an unsigned 4-byte integer.
Last EID	The EID of the last event recorded by the SWID-PC, or 0 if the SWID-PC has no recorded events. This field contains the EID of the SWID-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 3.6.4 for more on partial list 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 SWID-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 RFC 3339 [5] compliant ASCII string in Coordinated Universal Time (UTC) time 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 [RFC3339] with the above restrictions. Leap seconds are permitted and SWID-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.
Action	The type of event that is recorded in this event record. Possible values are: 1 = CREATION - the addition of a tag to the endpoint's SWID tag collection; 2 = DELETION - the removal of a tag from the endpoint's SWID tag collection; 3 = ALTERATION - There was an alteration to a tag file within the endpoint's tag collection. All other values are reserved for future use and MUST NOT be used when sending attributes. In the case where a SWID-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.
Instance ID Length	A 2-byte unsigned integer indicating the length in bytes of the Instance ID field.
Instance ID	A string containing the Instance ID of a given tag instance. The exact value of this field depends on the party that provides this SWID tag. This field value MUST be normalized to Network Unicode format, as described in Section 4.7 . This string MUST NOT be NULL terminated.
Tag Len	This is a 4-byte unsigned integer indicating the length of the following SWID tag in bytes.
Tag	A SWID tag as a string. In the case where the original SWID tag is not expressed using UTF-8 encoding, it MUST be converted and normalized to Network Unicode format, as described in Section 4.7. However, in the case where the original SWID tag is expressed using UTF-8 encoding, the SWID tag MUST be copied to this field without modification, even if the original SWID tag does not conform fully to Network Unicode format. This string MUST NOT be NULL terminated.

Table 9: SWID Tag 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 SWID Tag Inventory attribute, a SWID Tag 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 SWID Request attributes only request full tags be sent (Result Type set to 0) in a targeted request, thus constraining the response just to tags that match a given set of tag identifiers.

As with the SWID Tag Identifier Events Attribute, this attribute **MUST** only contain event records with EIDs coming from the current EID Epoch of the SWID-PC.

As with the SWID Tag Inventory Attribute, the SWID-PC **MUST** perform conversion and normalization of the SWID tag itself in the case where the source SWID tag is expressed using an encoding other than UTF-8, and **MUST NOT** perform conversion or normalization of the SWID tag itself in the case where the source SWID tag is expressed using UTF-8.

4.14. Subscription Status Request

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

This attribute has no fields.

A SWID-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).

4.15. Subscription Status Response

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

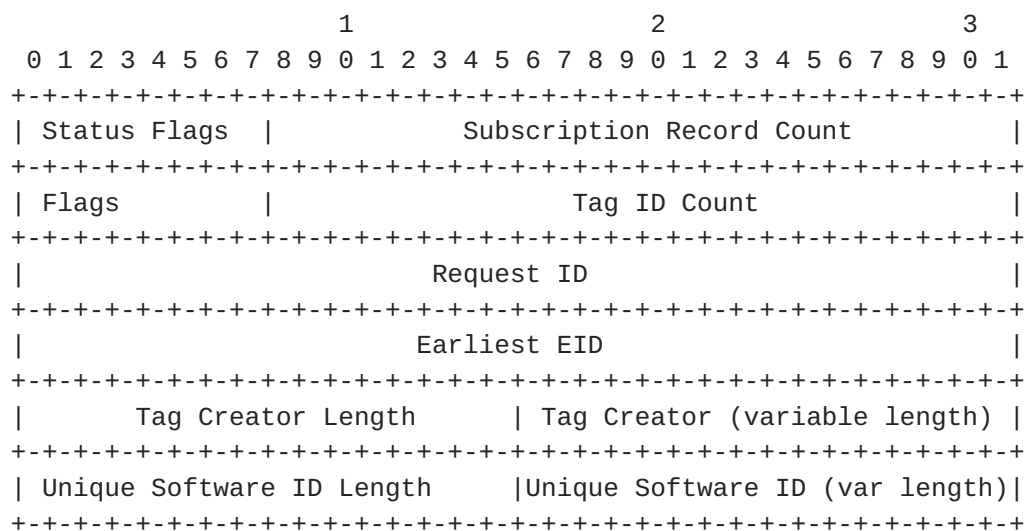


Figure 12: Subscription Status Response Attribute

Field	Description
Flags: Bit 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, Tag ID Count, Request ID, Earliest EID, Tag Creator Length, Tag Creator, Unique Software ID Length, and Unique Software ID 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.
Flags, Tag ID Count, Request ID, Earliest EID, Tag Creator Length, Tag Creator, Unique Software ID Length, and Unique Software ID	For each active subscription, these fields contain an exact copy of the fields with the same name as provided in the subscription's establishing request.

Table 10: 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 SWID-PC MUST use the requester's Connection ID and, if available, its Posture Validator ID to determine which subscriptions are associated with the requester.

A SWID-PC MUST send a Subscription Status Response attribute in response to a Subscription Status Request attribute. The only exception to this is if the SWID-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 of its Status Flags field, 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 a SWID 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 Tag ID Count field will be non-zero and, within that subscription record, the Tag Creator Length, Tag Creator, Unique Software ID Length, and Unique Software ID fields are repeated, in order, the number of times indicated in the Tag ID Count field. As such, each subscription record can be different sizes. Likewise, if the establishing request is not targeted (Tag ID Count field is 0), the subscription record has no Tag Creator Length, Tag Creator, Unique Software ID Length, or Unique Software ID fields.

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

4.16. PA-TNC Error as Used by SWID Message and Attributes for PA-TNC

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 SWID Message and Attributes for PA-TNC exchange. The latter case utilizes error codes defined below.

A PA-TNC Error attribute is sent instead of a SWID Response attribute due to factors that prevent the reliable creation of a SWID Response. As such, a SWID-PC MUST NOT send both a PA-TNC Error attribute and a SWID Response attribute in response to a single SWID Request attribute.

Table 11 lists the Error Code values for the PA-TNC Error attribute specific to the SWID Message and Attributes for PA-TNC exchange. In all of these cases, the Error Code Vendor ID field MUST be set to 0x000000, corresponding to the IETF SMI Private Enterprise Number. The Error Information structures for each error type are described in the following subsections.

Note that a message with a SWID Message and Attributes for PA-TNC attribute might also result in an error condition covered by the Standard PA-TNC Error Codes defined in PA-TNC. For example, the SWID Attribute might have an invalid parameter, leading to an error code of "Invalid Parameter". In this case, the SWID-PC MUST use the appropriate PA-TNC Error Code value as defined in [Section 4.2.8](#) of PA-TNC specification.

Error Code Value	Description
0x00000020	SWID_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 SWID Attributes. The Description field SHOULD contain additional diagnostic information.
0x00000021	SWID_SUBSCRIPTION_DENIED_ERROR. This indicates that the SWID-PC denied the SWID-PV's request to establish a subscription. The Description field SHOULD contain additional diagnostic information.
0x00000022	SWID_RESPONSE_TOO_LARGE_ERROR. This indicates that the SWID-PC's response to the SWID-PV's request was too large to be serviced. The error information structure indicates the largest possible size of a response supported by the SWID-PC (see Section 4.16.2). The Description field SHOULD contain additional diagnostic information.
0x00000023	SWID_SUBSCRIPTION_FULFILLMENT_ERROR. This indicates that the SWID-PC experienced an error 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 SWID-PC experienced. The SWID-PC and SWID-PV MUST treat the identified subscription as cancelled.
0x00000024	SWID_SUBSCRIPTION_ID_REUSE_ERROR. This indicates that the SWID-PC received a SWID Request from a given SWID-PV where the Request ID of that SWID Request is

	currently used as the Subscription ID of an active subscription with that SWID-PV. This error does not cancel the identified subscription.
0x00000025-0x0000002F	RESERVED. These Error Codes are reserved for use by future revisions of the SWID Message and Attributes for PA-TNC specification. Any PA-TNC Error attribute using one of these Error Codes MUST be treated as indicating a fatal error on the sender without further interpretation.

Table 11: PA-TNC Error Codes for SWID Message and Attributes for PA-TNC

The following subsections describe the structures present in the Error Information fields.

4.16.1. SWID_ERROR, SWID_SUBSCRIPTION_DENIED_ERROR and SWID SUBSCRIPTION ID REUSE ERROR Information

The SWID_ERROR error code indicates that the sender (the SWID-PC) has encountered an error related to the processing of a SWID Request attribute but which is not covered by more specific SWID error codes. The SWID_SUBSCRIPTION_DENIED_ERROR is used when the SWID-PV requests to establish a subscription or clear all subscriptions from the given SWID-PV, but the SWID-PC is unable or unwilling to comply with this request. The SWID_SUBSCRIPTION_ID_REUSE_ERROR is used when the SWID-PC receives a SWID 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.

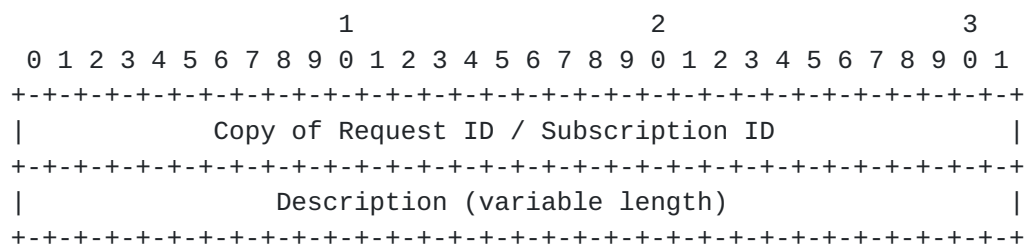


Figure 13: SWID Error, Subscription Error, and Subscription ID Reuse Information

Field	Description
Copy of Request ID / Subscription ID	In the case that this error condition is generated in direct response to a SWID Request attribute, this field MUST contain an exact copy of the Request ID field in the SWID 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 SWID_ERROR as the other errors are not generated by subscription fulfillment.) Note that, in this case, the indicated error appears as a sub-error for a SWID_SUBSCRIPTION_FULFILLMENT_ERROR, as described in Section 4.16.3 .
Description	A UTF-8 string describing the condition that caused this error. This field MAY be 0-length. However, senders SHOULD include some description in all PA-TNC Error attributes of these types. This field MUST NOT be NULL terminated.

Table 12: SWID Error, Subscription Error, and Subscription ID Reuse Information Fields

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

[4.16.2](#). SWID_RESPONSE_TOO_LARGE_ERROR Information

The SWID_RESPONSE_TOO_LARGE_ERROR error code indicates that a response generated by a SWID-PC in response to a SWID-PV's SWID Request attribute was too large to send.

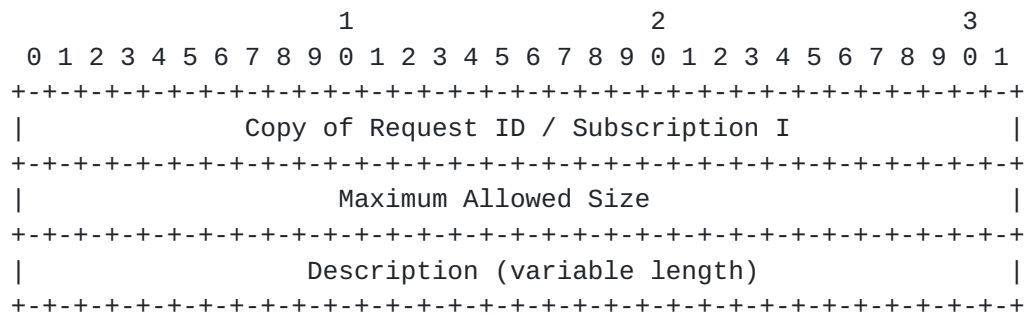


Figure 14: SWID Response Too Large Error Information

Field	Description
Copy of Request ID / Subscription ID	In the case that the attribute in question is generated in direct response to a SWID Request, this field MUST contain an exact copy of the Request ID field in the SWID 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 this case, the SWID_RESPONSE_TOO_LARGE_ERROR appears as a sub-error for a SWID_SUBSCRIPTION_FULFILLMENT_ERROR, as described in Section 4.16.3 .
Maximum Allowed Size	This field MUST contain an unsigned integer indicating the largest permissible size, in bytes, of SWID Attribute that the SWID-PC is currently willing to send in response to a SWID Request attribute.
Description	A UTF-8 string describing the condition that caused this error. This field MAY be 0-length. However, senders SHOULD include some description in all PA-TNC Error attributes of these types. This field MUST NOT be NULL terminated.

Table 13: SWID Response Too Large Error Information Fields

This error structure is used with the SWID_RESPONSE_TOO_LARGE_ERROR status code to identify the SWID Request attribute that precipitated the error condition and to describe the error. The Maximum Allowed Size field indicates the largest attribute the SWID-PC is willing to send in response to a SWID Request under the current circumstances.

Note that under other circumstances, the SWID-PC might be willing to return larger 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 SWID_ERROR and SWID_SUBSCRIPTION_DENIED_ERROR information structure.

4.16.3. SWID_SUBSCRIPTION_FULFILLMENT_ERROR Information

The SWID_SUBSCRIPTION_FULFILLMENT_ERROR error code indicates that the SWID-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) that is used to identify the nature of the encountered error.

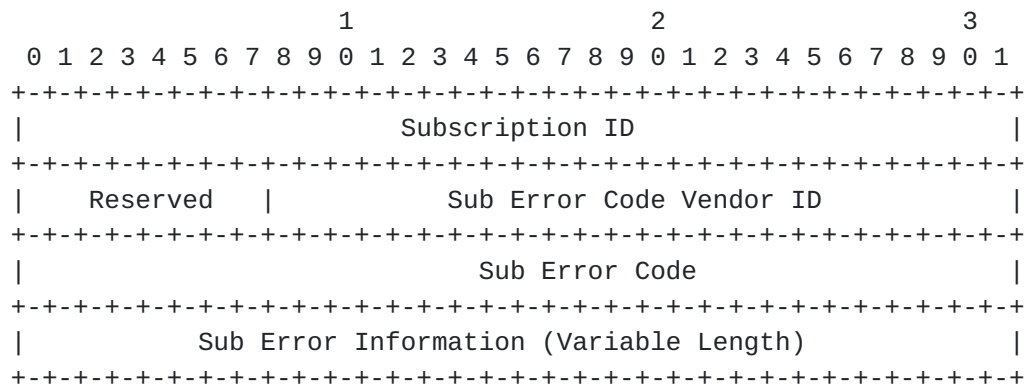


Figure 15: SWID Subscription Fulfillment Error Information

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 14: SWID Subscription Fulfillment Error Information Fields

This error structure is used with the SWID_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 type of error that the SWID-PC encountered while fulfilling the given subscription. The sub-error MUST NOT have an error code of SWID_SUBSCRIPTION_FULFILLMENT_ERROR.

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

5. Security Considerations

This section discusses some of the security threats facing Posture Collectors and Posture Validators that implement the SWID Message and Attributes for PA-TNC protocol. This section primarily notes potential issues for implementers to consider, although it does contain a handful of normative requirements to address certain

security issues. Implementers need to make the final decision as to how their implementations address the given issues. The issues identified below focus on capabilities specific to this document. Implementers are advised to consult other relevant NEA specifications for security issues that are applicable to such components in general.

Reading the Security Considerations section of any well-written specification can be discouraging, as a long list of possible threats is catalogued. Keep in mind that no security measure is absolute, but each one can be beneficial. By understanding the weaknesses of each security measure, we can put in place countermeasures to protect against exploitation of these weaknesses.

5.1. Evidentiary Value of SWID Tags

A SWID tag is only indirect evidence as to the installation of a piece of software on an endpoint. While the ideal is for the presence of a tag to correspond to the presence of the corresponding software, such a correlation hinges on software that accurately manages individual tags as software is added and removed. Utilization of the tests included in a tag's `package_footprint` and/or validation elements can provide more direct evidence of software presence, but this information might not be present in many tags and, because of its limited support, this version of the SWID Message and Attributes for PA-TNC specification does not support use of these flags. Tags can include cryptographic signatures of some or all of their fields, which can enable detection of field modification. This is extremely useful in ensuring that sensitive fields are not modified maliciously. However, the use of cryptographic signatures is not required in SWID tags, and even when utilized, not all fields will necessarily be protected by signatures. For these reasons, it is important to treat SWID tags as evidence rather than proof of software presence.

5.2. Integrity of the SWID Tag Collection

On a related note, while some systems might protect SWID tags against modification, on others there might be very few restrictions on who can add or delete tags on an endpoint. This can mean that a malicious party has relatively free rein to add and remove tags with the goal of obscuring the actual software inventory of the endpoint. As noted above, signatures on tag files can keep tag modification from going undetected, but an attacker can simply delete the signed tag and replace it with a modified tag that lacks the associated signature. In fact, even a signed tag can be added by an adversary and go undetected if the tag does not include fields to verify

software presence, such as the package_footprint or validation element

There is little a SWID-PC can do to prevent unauthorized modifications of the endpoint's SWID tag from occurring if the local system does not provide protections for tags. Instead, the SWID-PV and NEA Server need to operate with an awareness that this type of modification can occur. The use of mechanisms to corroborate software inventories can help detect malicious modification of an endpoint's SWID tag collection. Likewise, the SWID-PV can look for odd behavior such as the deletion and rapid re-installation of a particular tag, especially the replacement of a signed tag with an unsigned one. Parties that use SWID tags as evidence of compliance with security policies need to be aware of the possible risks of corruption of an endpoint's SWID tag collection.

5.3. Sensitivity of Collected Tags

Tags on an endpoint are generally not considered to be sensitive, although there can be exceptions to this generalization as noted in the section on Privacy Considerations. In general, an endpoint's SWID tag collection can be browsed and individual tags read by any party with access to the endpoint. This is generally not considered to be problematic, as those with access to the endpoint can usually learn of everything disclosed by that endpoint's tags simply by inspecting other parts of the endpoint.

The situation changes when an endpoint's SWID tags are collected and stored off of the endpoint itself, such as on a NEA Server or CMDB. Tags 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 tags might provide many details that are useful for mounting an attack. A list of the tags associated with an endpoint reveals a list of software installed on the endpoint. This list is very detailed, generally noting specific versions and even patch levels, which an adversary can use to identify vulnerable software and design efficacious attacks.

In addition, other information might also be gleaned from a collection of SWID tags:

- o A SWID tag 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.
- o A SWID tag 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.

- o Events affecting SWID tags 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 tag 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 necessary individuals and processes, and other basic security precautions.

5.4. Integrity of Endpoint Records

SWID-PCs maintain records of detected changes to the endpoint's SWID tag collection. These records are used to respond to a SWID-PV's request for change events. The SWID-PV might use a list of reported events to update its understanding of the endpoint's SWID tag collection without needing to receive a full inventory report from the SWID-PC. For this reason, preserving the integrity of the SWID-PC's record of events is extremely important. If an attacker modifies the SWID-PC's record of changes to the endpoint's SWID tag collection, this might cause the SWID-PV's understanding of the endpoint's SWID tag collection to differ from its actual state. Results might include leading the SWID-PV to believe that absent software was present, that present software was absent, that patches have been installed even if this is not the case, or to be unaware of other changes to SWID tags. As such, the SWID-PC MUST take steps to protect the integrity of its event record.

In addition, sometimes a SWID-PC captures metadata about existing tags or even creates copies of whole tags. Metadata might include hash values of tag files or records of the last time a particular tag file was modified, while whole tags might be preserved to record tags that were deleted from the endpoint's SWID tag collection. If an attacker is able to corrupt or modify this information, they might cause a SWID-PC to fail to detect certain change events, incorrectly report information, or otherwise fail to correctly fulfill SWID-PV requests. As such, this additional information about SWID tags, if collected, MUST be integrity protected.

Finally, records of established SWID-PV subscriptions also require protection against manipulation or corruption. If an attacker is able to modify or delete records of an established subscription by a SWID-PV, the SWID-PC might fail to correctly fulfill this subscription. The SWID-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 SWID-PV might collect a full inventory and realize from this that certain events had not been correctly reported in accordance with an established subscription. For this reason, the SWID-PC MUST protect the integrity of subscription records.

5.5. SWID-PC Access Permissions

A SWID-PC requires sufficient permissions to locate and read SWID tags on the endpoint that constitute the endpoint's SWID tag collection, and 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 file system. Depending on the operating environment and other activities undertaken by a SWID-PC (or software that incorporates a SWID-PC as one of its capabilities) additional permissions might be required by the SWID-PC software. The SWID-PC SHOULD NOT be granted permissions beyond what it needs in order to fulfill its duties.

5.6. Sanitization of Tag Fields

In most cases there is no constraint on an endpoint as to who can add tags. This open model allows applications that run in user space to register tags as easily as more privileged applications. However, this also means that any tool reading an endpoint's tags needs to treat these tags as un-vetted input and employ appropriate safeguards. In particular, tools that read SWID tags, including SWID-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 tag.

Fields of a SWID tag that change the SWID-PC's behavior, alter system state, or execute code need to be handled with special care. In particular, the validation element, which provides a command line that can nominally be executed to validate the tag's correctness, can be utilized by an attacker to point to a malicious executable. To defend against this, SWID-PCs MUST NOT execute an application indicated by a validation element unless the element is signed and the SWID-PC has determined that the signature is intact and trusted.

5.7. Tag Library Poisoning

It can be useful for a SWID-PV to have access to a library of tags. If the SWID-PV receives a list of tag identifier instances, it can consult this library and collect full tags corresponding to those identifiers. Assuming it does not need access to installation-specific information, it can perform calculations on these full tags as if it had received them from a SWID-PC. For example, it can use this library to derive software names, publishers, and version by finding the tag that corresponds to the given tag identifier value.

If the SWID-PV keeps a collection of full SWID tags for matching against tag identifiers, there might be a temptation to add any previously unknown tags that a SWID-PC might report to this library automatically. In fact, this behavior can pose a security risk. If the endpoint has been compromised and the tag manipulated on that endpoint, the tag that it provides to the SWID-PV might be misleading with regard to the software associated with this tag. If the SWID-PV automatically adds this corrupted tag to its library, not only will the computations with the compromised endpoint be affected, but computations with other endpoints that provide tag identifier instances that map to the corrupted tag will also be affected. Instead, if the SWID-PV does make use of a tag library, it is recommended to only populate that library with tags retrieved from a trusted source, or at least to segregate collections of reported tags by endpoint, so corrupted tags on one endpoint will not affect tag computations involving other endpoints. In general, tags retrieved from a trusted source and signed by a trusted authority are likely be safe for inclusion in a tag library.

5.8. PA-TNC Security Threats

In addition to the aforementioned considerations the SWID Message and Attributes for PA-TNC protocol is subject to the same security threats as other PA-TNC transactions, as noted in [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.

6. Privacy Considerations

As noted in [Section 5.3](#), 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 adequate safeguards are in place to prevent exposure of collected tags. For similar reasons, it

is advisable that an endpoint only send tags to a NEA Server that is authorized to receive this information and that can be trusted to safeguard this information after collection.

7. Relationship to Other Specifications

This specification makes frequent reference to and use of other specifications. This section describes these relationships.

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, the SWID-PC communicates with the endpoint's Posture Broker Client using IF-IMC [[IF-IMC](#)], while the SWID-PV communicates with the NEA Server's Posture Broker Server using IF-IMV [[IF-IMV](#)].

In addition, SWID 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 the SWID Message and Attributes for PA-TNC specification is able to convey a broader query, resulting in a more comprehensive set of evidence regarding an endpoint's installed software. Moreover, because this specification makes use of the well-defined structures in SWID tags, it is able to convey information that is more concise (by making use of specific identifier fields instead of sending the whole SWID tag) and/or more comprehensive (as the SWID structures contain many more fields than expressible in PA-TNC). As such, this specification provides important capabilities not present in the PA-TNC specification.

8. IANA Considerations

This section extends multiple existing IANA registries. Specifically, it extends the PA-TNC Attribute Types and PA-TNC Error Codes 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.

8.1. Registry for PA-TNC Attribute Types

[Section 4.6](#) of this specification defines several new PA-TNC attributes. The following values are added to the registry for PA-TNC Attribute Types defined in the PA-TNC specification. Note that Table 4 in [Section 4.6](#) lists these attributes but uses a hexadecimal value to identify their associated integer. The integer values given in that table are identical to those provided here. Note also that Table 4 includes an entry for PA-TNC Error attributes, but the IANA information associated with that attribute is already defined in the PA-TNC specification and is not reproduced here.

PEN	Integer	Name	Defining Specification
0	17	SWID Request	SWID Message and Attributes for PA-TNC
0	18	SWID Tag Identifier Inventory	SWID Message and Attributes for PA-TNC
0	19	SWID Tag Identifier Events	SWID Message and Attributes for PA-TNC
0	20	SWID Tag Inventory	SWID Message and Attributes for PA-TNC
0	21	SWID Tag Events	SWID Message and Attributes for PA-TNC
0	22	Subscription Status Request	SWID Message and Attributes for PA-TNC
0	23	Subscription Status Response	SWID Message and Attributes for PA-TNC
0	24	Subscription Status Response	SWID Message and Attributes for PA-TNC
0	25 - 31	Reserved for future use	SWID Message and Attributes for PA-TNC

8.2. Registry for PA-TNC Error Codes

[Section 4.16](#) of this specification defines several new PA-TNC Error Codes. The following values are added to the registry for PA-TNC Error Codes defined in the PA-TNC specification. Note that Table 11 in [Section 4.16](#) lists these attributes but uses a hexadecimal value to identify their associated integer. The integer values given in that table are identical to those provided here.

PE N	Integer	Name	Defining Specification
0	32	SWID_ERROR	SWID Message and Attributes for PA-TNC
0	33	SWID_SUBSCRIPTION_DENIED_ERROR	SWID Message and Attributes for PA-TNC
0	34	SWID_RESPONSE_TOO_LARGE_ERROR	SWID Message and Attributes for PA-TNC
0	35	SWID_SUBSCRIPTION_FULFILLMENT_ERROR	SWID Message and Attributes for PA-TNC
0	36	SWID_SUBSCRIPTION_ID_REUSE_ERROR	SWID Message and Attributes for PA-TNC
0	37-47	Reserved for future use	SWID Message and Attributes for PA-TNC

8.3. Registry for PA Subtypes

[Section 4.1](#) of this specification defines one new PA Subtype. The following value is added to the registry for PA Subtypes defined in the PB-TNC specification.

PEN	Integer	Name	Defining Specification
0	9	SWID Attributes	SWID Message and Attributes for PA-TNC

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", [RFC 3339](#), DOI 10.17487/RFC3339, July 2002, <<http://www.rfc-editor.org/info/rfc3339>>.
- [RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", [RFC 5198](#), DOI 10.17487/RFC5198, March 2008, <<http://www.rfc-editor.org/info/rfc5198>>.
- [RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", [RFC 5209](#), DOI 10.17487/RFC5209, June 2008, <<http://www.rfc-editor.org/info/rfc5209>>.
- [RFC5792] Sangster, P. and K. Narayan, "PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)", [RFC 5792](#), DOI 10.17487/RFC5792, March 2010, <<http://www.rfc-editor.org/info/rfc5792>>.
- [SWID] The International Organization for Standardization/ International Electrotechnical Commission, "Information Technology - Software Asset Management - Part 2: Software Identification Tag, ISO/IEC 19770-2", November 2009.

9.2. Informative References

- [IF-IMC] Trusted Computing Group, "TNC IF-IMC version 1.3", February 2013.
- [IF-IMV] Trusted Computing Group, "TNC IF-IMV version 1.4", August 2013.
- [NIST-SWID] The National Institute of Standards and Technology and The MITRE Corporation, "Guidelines for the Creation of Interoperable Software Identification (SWID) Tags", August 2013.
- [RFC5793] Sahita, R., Hanna, S., Hurst, R., and K. Narayan, "PB-TNC: A Posture Broker (PB) Protocol Compatible with Trusted Network Connect (TNC)", [RFC 5793](#), DOI 10.17487/RFC5793, March 2010, <<http://www.rfc-editor.org/info/rfc5793>>.

- [RFC6876] Sangster, P., Cam-Winget, N., and J. Salowey, "A Posture Transport Protocol over TLS (PT-TLS)", [RFC 6876](#), DOI 10.17487/RFC6876, February 2013, <<http://www.rfc-editor.org/info/rfc6876>>.
- [RFC7171] Cam-Winget, N. and P. Sangster, "PT-EAP: Posture Transport (PT) Protocol for Extensible Authentication Protocol (EAP) Tunnel Methods", [RFC 7171](#), DOI 10.17487/RFC7171, May 2014, <<http://www.rfc-editor.org/info/rfc7171>>.

[Appendix A](#). Examples

This appendix includes examples of a SWID tag file and SWID attributes. All examples represent fictional content. Examples are provided using the 2009 release of the ISO/IEC SWID specification.

[A.1](#). A Simple SWID Tag

Figure 16 shows an example SWID tag for a fictional software product called SomeApp created by Vendor Inc. This example includes only the required SWID tag fields. This tag is for version 2.3, build 12 of the product.


```
1]<?xml version="1.0" encoding="UTF-8"?>
2]<swid:software_identification_tag
3]  xmlns:swid="http://standards.iso.org/iso/19770/-2/2009/schema.xsd"
4]  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
5]    <swid:entitlement_required_indicator>
6]      true
7]    </swid:entitlement_required_indicator>
8]    <swid:product_title>SomeApp</swid:product_title>
9]    <swid:product_version>
10]      <swid:name>2.3 r12</swid:name>
11]      <swid:numeric>
12]        <swid:major>2</swid:major>
13]        <swid:minor>3</swid:minor>
14]        <swid:build>12</swid:build>
15]        <swid:review>0</swid:review>
16]      </swid:numeric>
17]    </swid:product_version>
18]    <swid:software_creator>
19]      <swid:name>Vendor Inc.</swid:name>
20]      <swid:regid>regid.2013-06.com.vendor</swid:regid>
21]    </swid:software_creator>
22]    <swid:software_licensor>
23]      <swid:name>Vendor Inc.</swid:name>
24]      <swid:regid>regid.2013-06.com.vendor</swid:regid>
25]    </swid:software_licensor>
26]    <swid:software_id>
27]      <swid:unique_id>
28]        someapp-21ec2020-3aea-1069-a2dd-08002b30309d
29]      </swid:unique_id>
30]      <swid:tag_creator_regid>
31]        regid.2013-06.com.vendor
32]      </swid:tag_creator_regid>
33]    </swid:software_id>
34]    <swid:tag_creator>
35]      <swid:name>Vendor Inc.</swid:name>
36]      <swid:regid>regid.2013-06.com.vendor</swid:regid>
37]    </swid:tag_creator>
38]</swid:software_identification_tag>
```

Figure 16: A Simple SWID Tag

The SWID tag described in Figure 16 is limited to only the information required by the SWID specification [[SWID](#)]. This information includes the following:

- o Lines 5-7: Entitlement requirement indicator. This indicates whether some sort of entitlement (e.g., a license) is required in order to install and/or use the software.
- o Line 8: Prose name of the product
- o Lines 9-17: Product version. This includes both a prose expression of the full product version and the version information broken down into distinct fields.
- o Lines 18-21: Software creator identification. This identifies the party that created the software. This includes both a prose name of the software creator and their "regid" value.
- o Lines 22-25: Software licensor identification. This identifies the party that holds the rights to license others to use the software.
- o Lines 26-33: Software unique identifier. This structure contains the regid for the party that created this tag and a value that party uses to uniquely identify the named software product. The SWID Message and Attributes for PA-TNC specification uses the values of these fields when constructing a SWID tag identifier, as described in [Section 3.3](#).
- o Lines 34-37: Tag creator identification. This identifies the party that created the tag.

Assume the SWID tag file is installed on the file system in the following location:

```
C:\ProgramData\Vendor\regid.2013-06.com.vendor_someapp-21ec2020-3aea-1069.swidtag
```

[A.2.](#) SWID Request Attributes

Below are hexadecimal dumps of example SWID Request attributes. SWID Request attributes are described in more detail in [Section 4.7](#).

[A.2.1.](#) Simple Request

This is a basic SWID request for inventory information - the request is not targeted nor does the request establish a subscription on the endpoint. The SWID Request dictates that the response be expressed using SWID tag identifier instances.


```

+-----+-----+
| 20 00 00 00 | Clear Subscriptions = 0 (don't clear      |
|              | subscriptions), Subscribe = 0 (don't establish a |
|              | new subscription), Result Type = 1 (respond using |
|              | SWID tag identifier instances), Tag ID Count = 0 |
|              | (non-targeted request)                          |
| ----- |
| 00 00 3a 76 | Request ID = 14966                             |
| ----- |
| 00 00 00 00 | Earliest EID = 0 (respond with inventory rather |
|              | than event records)                            |
+-----+-----+

```

Note that this attribute does not contain any Tag Creator Length, Tag Creator, Unique Software ID Length, or Unique Software ID fields because the Tag ID Count field is 0.

[A.2.2.](#) Subscription Request for Events

This attribute establishes a request for a new subscription that will report new SWID change events as they occur.

```

+-----+-----+
| 60 00 00 00 | Clear Subscriptions = 0 (don't clear      |
|              | subscriptions), Subscribe = 1 (establish a new |
|              | subscription), Result Type = 1 (respond using SWID |
|              | tag identifier instances), Tag ID Count = 0 (non- |
|              | targeted request)                              |
| ----- |
| 00 00 3a 76 | Request ID = 14967                             |
| ----- |
| 00 02 cc 3a | Earliest EID = 183354                         |
+-----+-----+

```

As before, this attribute does not contain any Tag Creator Length, Tag Creator, Unique Software ID Length, or Unique Software ID fields because the Tag ID Count field is 0. The immediate response to this message (assuming no errors are encountered) will be a list of events with EIDs that are greater than or equal to 183354. Thereafter, if any new events are recorded, those events (and only those events) will be sent back to the SWID-PV in fulfillment of this subscription. (See [Section 3.8.2](#) for more on subscription fulfillment.)

[A.2.3.](#) Targeted Request

This example shows a targeted request. Specifically, the request includes two SWID tag identifiers. The attribute requests that the corresponding full SWID tags be returned.


```

+-----+-----+
| 00 00 00 02 | Clear Subscriptions = 0 (don't clear
|              | subscriptions), Subscribe = 0 (don't establish
|              | a new subscription), Result Type = 0 (respond
|              | using full SWID tags), Tag ID Count = 2
|              | (targeted request identifying two SWID tags)
| -----
| 00 00 3a 76 | Request ID = 14968
| -----
| 00 00 00 00 | Earliest EID = 0 (respond with inventory rather
|              | than event records)
| =====
| 00 18       | START TAG IDENTIFIER 1
|              | Tag Creator Length = 24
| -----
| 72 65 67 69 64 | Tag Creator = regid.2013-06.com.vendor
| 2e 32 30 31 33 |
| 2d 30 36 2e 63 |
| 6f 6d 2e 76 65 |
| 6e 64 6f 72    |
| -----
| 00 2c       | Unique Software ID Length = 44
| -----
| 73 6f 6d 65 61 | Unique Software ID =
| 70 70 2d 32 31 | someapp-21ec2020-3aea-1069-a2dd-08002b30309d
| 65 63 32 30 32 |
| 30 2d 33 61 65 |
| 61 2d 31 30 36 |
| 39 2d 61 32 64 |
| 64 2d 30 38 30 |
| 30 32 62 33 30 |
| 33 30 39 64    |
| =====
| 00 18       | START TAG IDENTIFIER 2
|              | Tag Creator Length = 24
| -----
| 72 65 67 69 64 | Tag Creator = regid.2013-06.com.vendor
| 2e 32 30 31 33 |
| 2d 30 36 2e 63 |
| 6f 6d 2e 76 65 |
| 6e 64 6f 72    |
| -----
| 00 2f       | Unique Software ID Length = 47
| -----
| 61 6e 6f 74 68 | Unique Software ID =
| 65 72 61 70 70 | anotherapp-23a52020-3aea-1069-a2dd-0800884d4e21
| 2d 32 33 61 35 |
| 32 30 32 30 2d |
| 33 61 65 61 2d |
| 31 30 36 39 2d |

```



```

| 61 32 64 64 2d |
| 30 38 30 30 38 |
| 38 34 64 34 65 |
| 32 31          |
+-----+-----+

```

This message contains SWID tag identifiers for two SWID tags. The first of these tags is the example SWID tag described in [Appendix A.1](#). The second tag is created by the same tag creator, but indicates a different software product.

[A.3.](#) SWID Response Attributes

This section contains examples of SWID response attributes.

[A.3.1.](#) SWID Tag Identifier Events Attribute

This shows an example of a SWID Tag Identifier Events attribute. In this case, this attribute is sent in fulfillment of an established subscription rather than in direct response to a SWID Request attribute. (This is indicated by setting the Subscription Fulfillment flag.) The SWID Request attribute shown in [Appendix A.2.2](#) established this subscription (as indicated by the Subscription ID field).

This response contains two event records.

```

+-----+-----+
| 80 00 00 02 | Subscription Fulfillment = 1, Event Count = 2 |
| ----- |
| 00 00 3a 76 | Subscription ID = 14968 |
| ----- |
| 7e 82 1c aa | EID Epoch = 2122456234 |
| ----- |
| 00 02 cc 84 | Last EID = 183428 |
| ----- |
| 00 02 cc 84 | Last Consulted EID = 183428 (Same as Last EID |
| | so this is a complete result) |
| ===== | START EVENT RECORD 1 |
| 00 02 cc 83 | EID = 183427 |
| ----- |
| 32 30 31 33 2d 30 | Timestamp = 2013-07-21T04:32:16Z |
| 37 2d 32 31 54 30 | |
| 34 3a 33 32 3a 31 | |
| 36 5a | |
| ----- |
| 01 | Action = 1 (CREATION) |
| ----- |

```


00 18	Tag Creator Length = 24	

72 65 67 69 64 2e	Tag Creator = regid.2013-06.com.vendor	
32 30 31 33 2d 30		
36 2e 63 6f 6d 2e		
76 65 6e 64 6f 72		

00 2c	Unique Software ID Length = 44	

73 6f 6d 65 61 70	Unique Software ID =	
70 2d 32 31 65 63	someapp-21ec2020-3aea-1069-a2dd-08002b30309d	
32 30 32 30 2d 33		
61 65 61 2d 31 30		
36 39 2d 61 32 64		
64 2d 30 38 30 30		
32 62 33 30 33 30		
39 64		

00 51	Instance ID Length = 81	

43 3a 5c 50 72 6f	Instance ID =	
67 72 61 6d 44 61	C:\ProgramData\Vendor\regid.2013-06.	
74 61 5c 56 65 6e	com.vendor_someapp-21ec2020-3aea-1069.swidtag	
64 6f 72 5c 72 65		
67 69 64 2e 32 30		
31 33 2d 30 36 2e		
63 6f 6d 2e 76 65		
6e 64 6f 72 5f 73		
6f 6d 65 61 70 70		
2d 32 31 65 63 32		
30 32 30 2d 33 61		
65 61 2d 31 30 36		
39 2e 73 77 69 64		
74 61 67		
=====	START EVENT RECORD 2	
00 02 cc 84	EID = 183428	

32 30 31 33 2d 30	Timestamp = 2013-07-21T04:32:22Z	
37 2d 32 31 54 30		
34 3a 33 32 3a 32		
32 5a		

02	Action = 2 (DELETED)	

00 18	Tag Creator Length = 24	

72 65 67 69 64 2e	Tag Creator = regid.2009-08.com.company	
32 30 30 39 2d 30		


```

| 38 2e 63 6f 6d 2e |
| 63 6f 6d 70 61 6e |
| 79
| -----
| 00 24
| -----
| 32 34 38 35 34 39 | Unique Software ID Length = 36
| 37 35 2d 31 32 35 | Unique Software ID =
| 65 2d 65 65 33 65 | 24854975-125e-ee3e-98ac-45684248eefa
| 2d 39 38 61 63 2d |
| 34 35 36 38 34 32 |
| 34 38 65 65 66 61 |
| -----
| 00 47
| -----
| 43 3a 5c 50 72 6f | Instance ID = C:\Program
| 67 72 61 6d 20 46 | Files\Company\OurProduct\
| 69 6c 65 73 5c 43 | OurProduct_8749-84789200-02.swidtag
| 6f 6d 70 61 6e 79 |
| 5c 4f 75 72 50 72 |
| 6f 64 75 63 74 5c |
| 4f 75 72 50 72 6f |
| 64 75 63 74 5f 38 |
| 37 34 39 2d 38 34 |
| 37 38 39 32 30 30 |
| 2d 30 32 2e 73 77 |
| 69 64 74 61 67
+-----+-----+

```

This response contains two event records. Note that their timestamps indicate that they occurred a few seconds apart - some SWID-PCs might choose to wait a brief time before sending messages in fulfillment of subscriptions so as to send multiple event records in a single attribute. The first event record indicates the creation of the SWID tag shown in [Appendix A.1](#). The second event record indicates the deletion of a different SWID tag. Finally, note that since the Last EID field is equal to the EID of one of the reported event records, this indicates that the SWID-PC has no later recorded events.

[A.3.2.](#) SWID Tag Inventory Attribute

This shows an example of a SWID Tag Inventory attribute. In this case, this attribute is being sent in direct response to a SWID Request attribute, as indicated by the Subscription Fulfillment flag being unset. (Specifically, it is being sent in response to the SWID Request shown in [Appendix A.2.3](#), as can be shown by comparing the Request ID and Request ID Copy fields.) The result includes a single tag entry.


```

+-----+-----+
| 00 00 00 01 | Subscription Fulfillment = 0, Tag Count = |
|             | 1 |
| ----- |
| 00 00 3a 76 | Request ID Copy = 14968 |
| ----- |
| 7e 82 1c aa | EID Epoch = 2122456234 |
| ----- |
| 00 02 cc 84 | Last EID = 183428 |
| ===== | BEGIN TAG ENTRY |
| 00 51 | Instance ID Length = 81 |
| ----- |
| 43 3a 5c 50 72 6f 67 | Instance ID = |
| 72 61 6d 44 61 74 61 | C:\ProgramData\Vendor\regid.2013-06.com. |
| 5c 56 65 6e 64 6f 72 | vendor_someapp-21ec2020-3aea-1069.swidtag |
| 5c 72 65 67 69 64 2e | |
| 32 30 31 33 2d 30 36 | |
| 2e 63 6f 6d 2e 76 65 | |
| 6e 64 6f 72 5f 73 6f | |
| 6d 65 61 70 70 2d 32 | |
| 31 65 63 32 30 32 30 | |
| 2d 33 61 65 61 2d 31 | |
| 30 36 39 2e 73 77 69 | |
| 64 74 61 67 | |
| ----- |
| 05 ac | Tag Length = 1452 |
| ----- |
| 3c 3f 78 6d 6c 20 76 | The Tag field is equal to the SWID tag |
| ... 69 6f 6e 5f 74 | shown in Figure 16. Note that since the |
| 61 67 3e | original tag used UTF-8 encoding, the tag |
| | is copied without undergoing any |
| | conversion or normalization. |
+-----+-----+

```

This attribute contains a single SWID tag. As a response to the targeted SWID Request in [Appendix A.2.3](#), this indicates a single instance of the first requested SWID tag and no instances of the second requested tag were present in the endpoint's SWID tag collection. Moreover, if the same party receives both this attribute and the attribute in [Appendix A.3.1](#), one can tell that there have been no change events recorded since the preceding message, because the EID Epoch and Last EID values are unchanged.

Authors' Addresses

Chris Coffin
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
USA

Email: ccoffin@mitre.org

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

Email: dhaynes@mitre.org

Charles Schmidt
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
USA

Email: cmschmidt@mitre.org

Jessica Fitzgerald-McKay
Department of Defense
9800 Savage Road
Ft. Meade, Maryland
USA

Email: jmfitz2@nsa.gov

