TCG Attestation
PTS Protocol: Binding to TNC IF-M

Specification Version 1.0
Revision 28
August 24, 2011
Published

Contact:

admin@trustedcomputinggroup.org

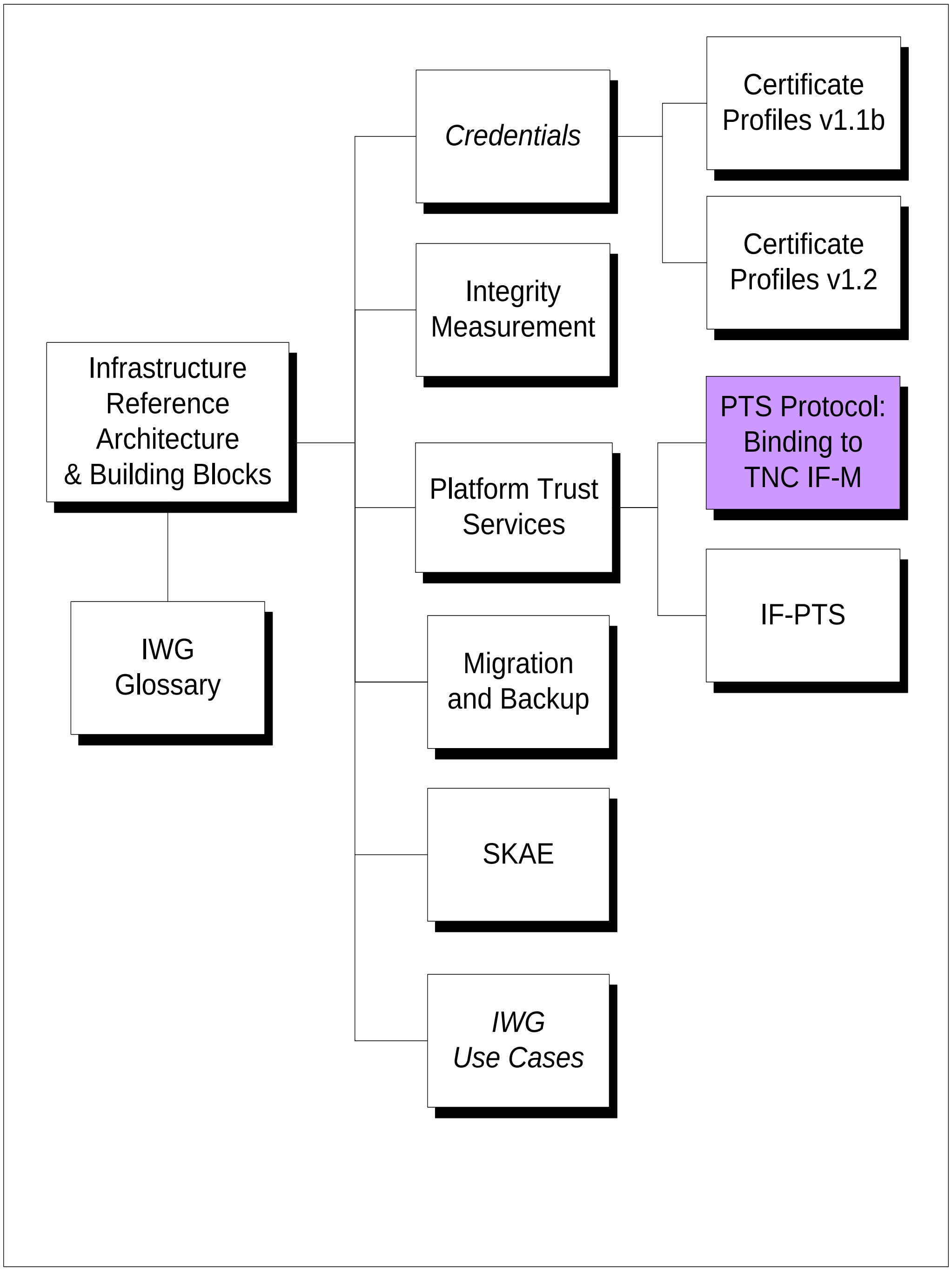
TCG PUBLISHED

TCG

Disclaimer

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

IWG Document Roadmap


Infrastructure Reference Architecture & Building Blocks
IWG Glossary
Credentials
Integrity Measurement
Platform Trust Services
Migration and Backup
SKAE
IWG Use Cases
Certificate Profiles v1.1b
Certificate Profiles v1.2
PTS Protocol: Binding to TNC IF-M
IF-PTS

Acknowledgement

The TCG wishes to thank all those who contributed to this specification. This document builds on considerable work done in the various working groups in the TCG.

Special thanks to the members of the TNC contributing to this document:

Name	Member Company
Rene Bourquin	General Dynamics
Mike Boyle	United States Government
Carlin Covey	Freescale Semiconductor
Steve Hanna	Juniper
Wyllys Ingersoll (IWG Co-chair)	Oracle
Greg Kazmierczak	Wave Systems
Scott Kelly	Hyperthought
Carolin Latze	89Grad
Gilles Peskine	Gemalto
Vincent Prunet	Trusted Logic
Paul Sangster (Editor and IWG Co-chair)	Symantec
Gloria Serrao	United States Government
Adrian Stanger	United States Government
Ned Smith	Intel
Lee Wilson	IBM

Special thanks to Carolin Latze who provided much of the text involving the registry attributes and to Steve Hanna, Carlin Covey, Wyllys Ingersoll, Scott Kelly, Carolin Latze and Gloria Serrao who provided review comments that significantly improved the specification.

Paul Sangster
Specification Editor

Table of Contents

1 Scope and Audience

This specification was created by the Infrastructure Work Group (IWG) within the Trusted Computing Group. The IWG is responsible for the definition of common infrastructure protocols, interfaces and services (both on end systems and the network) necessary to support the use of Trusted Platforms operating on a variety of types of platforms (e.g. PC Client, Servers, Mobile Devices).

This specification builds upon the existing IWG work defining Platform Trust Service (PTS) that is capable of creating attestation evidence describing the security state of the system in response to a challenge by a challenger. The current PTS interface specification does not specify the network protocols or the grammar for describing what the challenger would like to include as attestation evidence reported by the PTS. This specification defines the protocol and grammar used by the remote challenger to request particular attestation evidence from a system and the responses. In order to offer a solution that is consistent with the XML encoded Integrity Report [INT-REPORT] approach used by the PTS and the type-length-value (TLV) binary encoding used by the TNC architecture both types of messages will be defined.

Architects, designers, developers and technologists who wish to implement, use, or understand the PTS protocol described in this specification should read this document carefully. Before reading this document any further, the reader should review and understand the: TNC architecture [IF-TNC-ARCH], IF-M protocol [IF-M], IWG Integrity Management Model [IF-IWG-ARCH], IF-PTS interface specification [IF-PTS] and the Integrity Report specification [INT-REPORT].

2 Background

2.1 Terminology

Attestation – The process of vouching for the accuracy of information. External entities can attest to shielded locations, protected capabilities, and Roots of Trust. A platform can attest to its description of platform characteristics that affect the integrity (trustworthiness) of a platform. Both forms (TLV or XML-based) of attestation require reliable evidence of the attesting entity.

Attestation by the TPM - An operation that provides proof of data known to (stored in) the TPM. This is done by digitally signing specific internal TPM data using an AIK. The acceptance and validity of both the integrity measurements and the AIK itself are determined by the challenger. The AIK certificate is obtained by the platform using either the Privacy (AKA Attestation) CA or DAA protocol.

Attestation Evidence – Data about one or more components that can be cryptographically verified by the challenger and determined to be associated with a particular TPM/Trusted Platform. This could include posture information about a component that can be verified directly or indirectly using information extended into a PCR that was included in a TPM_Quote operation (thus signed by an AIK private key). For this specification, attestation evidence can be data in several different possible formats/encodings including TLV or XML.

Attestation of the Platform - An operation that provides proof of a set of the platform's integrity measurements. This is done by digitally signing a set of PCRs using an AIK in the TPM.

Challenger - Party responsible for querying and verifying the integrity measurement information from the system requesting access to a protected resource or asset (e.g. the network as per the TNC paradigm). The challenger might be a part of the relying party offering the protected resource or could be working on its behalf.

Requestor - Party requesting access to a protected resource or asset. This party may also be known in this specification less formally as the ‘attested system’ since it’s the system that needs to provide the attestation evidence in order to prove its trustworthiness to the challenger. The requestor may choose to attest the challenger which would result in them swapping roles.

Template Reference Manifest - Template Reference Integrity Measurement Manifest is used by the attested system to format the Integrity Report structure of the components requested by the challenger. Because an attestation of a component may cause a transitive trust chain and/or dependency graph of sub-components to be returned, the PTS will use the Template Reference (Integrity Measurement) Manifest as a basis for filling out the structure of the Integrity Report. Negotiating a particular template between the parties allows for agreement on the expected structure and sub-component ordering used in the Integrity Report, thus making verification much easier. Rather than create a new XML document this specification re-uses the existing Reference Manifest as a Template. Therefore, when a pair of systems are performing an attestation using the Integrity Report, the systems will negotiate a Reference Manifest they have in common and use this for formatting the resulting Integrity Report. The remote challenger's logic becomes simpler, since it can compare the Integrity Report directly with the expected values in its parallel Reference Manifest.

Verifier - Another term with same meaning as challenger described above. Verifier was used in the IWG architecture but challenger will be used in this document for greater clarity and consistency with other documents.

2.2 Purpose of PTS Protocols

The current IWG integrity management model described in the IWG Architecture specification establishes an information framework enabling a remote challenger to evaluate integrity information (including TPM measurements) received from a remote system. In order for the challenger to request and obtain specific information about a remote attested system, the challenger must be able to speak an attestation protocol and express what aspects of the remote system it would like to evaluate. The IWG has defined the local interfaces to a service that could be operating on the remote system known as the Platform Trust Service (PTS). The PTS is capable of performing a number of trusted platform services including the creation of reports (e.g. Integrity Report) containing the attestation evidence backed by TPM-based cryptographic proof of the authenticity of the report.

This specification defines the protocol syntax and semantics used by the remote challenger to request attestation evidence for specific aspects of the requestor's system. The binding of the PTS attestation protocol described in this specification leverages the TNC Architecture and specifically the IF-M application layer protocol. The TNC Architecture is composed of three hierarchically layered protocols (transport, session and messaging) enabling network-based assessments of an endpoint both prior to admission to the network (before endpoint possesses an IP address) and after the endpoint is present on the network using TCP/IP communications. For example, TNC supports assessment of endpoints using EAP over LAN (EAPoL) assessments prior to being given TCP/IP access on the network. In situations where the endpoint does not have IP-level access to the network, the endpoint will have more limited ability to communicate (fewer roundtrips or limits on data size). Therefore, the TNC architecture defines a concise type-length-value (TLV) oriented binary set of protocols to maximize efficiency and minimize processing requirements of the endpoint. The PTS protocol described in this specification will reside within the TLV-based IF-M application layer protocol of TNC Architecture. In order to be consistent with the TNC Architecture, this protocol will itself use TLV encodings while also allowing for the transport of XML-based reports like the Integrity Report.

2.3 PTS Protocol within TNC Architecture

This section discusses how the PTS protocol binding to IF-M integrates with the TNC architecture. The following diagram shows the TNC Architecture and highlights where the PTS protocol would reside.

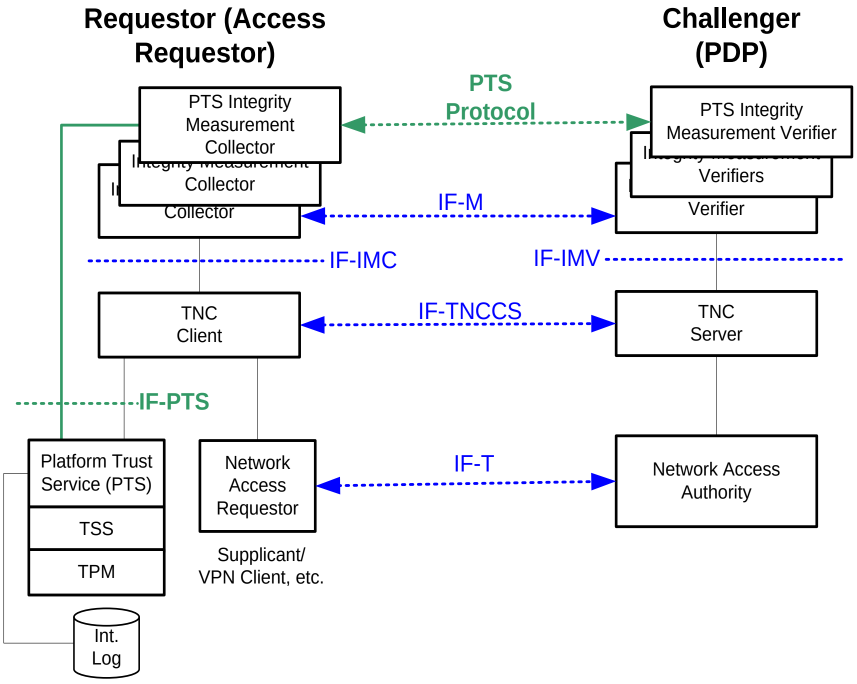

Requestor (Access Requestor)
Challenger (PDP)
PTS Protocol
PTS Integrity Measurement Collector
Integrity Measurement Collector
Integrity Measurement Collector
PTS Integrity Measurement Verifier
Integrity Measurement Verifiers
Integrity Measurement Verifier
IF-M
IF-IMC
IF-IMV
TNC Client
IF-TNCCS
TNC Server
IF-PTS
Platform Trust Service (PTS)
TSS
TPM
Int. Log
Network Access Requestor
Supplicant/ VPN Client, etc.
IF-T
Network Access Authority

Figure 1: PTS Protocol Integration with TNC Architecture

Notice that the TNC Architecture contains several layered protocols (IF-T, IF-TNCCS and IF-M). The PTS Protocol will be carried within the payloads of the IF-M protocol, so would layer hierarchically on top of the IF-M protocol. The PTS Protocol operates between the PTS-IMC and PTS-IMV to enable PTS-based attestation leveraging the underlying TPM. The PTS-IMC uses a local IPC channel to the PTS (discussed in the IF-PTS specification) to obtain the necessary attestation evidence. Use of the IF-PTS interface and the TSS middleware stack components are optional so implementations might leverage the PTS or TPM in other ways (e.g. the PTS could have other techniques for interacting with a TPM to obtain measurements).

At the bottom of the TNC architecture is the IF-T (transport) protocol that is capable of operating in a number of network connectivity situations. The two most common network connectivity situations for TNC deployments are: prior to the endpoint being placed on an IP network (given an IP address), and after the endpoint has some degree of TCP/IP access to the network. Endpoint assessment wishing to employ an attestation as part of the network admission process could be limited in number of round trips, bandwidth, latency or total assessment time, so more terse versions of the attestation exchange may be necessary. Currently TNC has defined IF-T bindings to EAP [IF-T-EAP] allowing use during an 802.1X exchange prior to admission to the network and TLS [IF-T-TLS] over a connected TLS session after the endpoint has network layer access.

2.4 Attestation Grammar

One of the challenges of performing an attestation is establishing the vocabulary used by the challenger to request for information about the attested system. Since each system is different in purpose and composition, it will be made up of different hardware and software components. The challenger could simply request a full report of everything running on the system, but this has scalability, network bandwidth and privacy and security issues. So this specification enables a more precise protocol for "asking functional questions" of the endpoint, but this requires that the challenger has at least a core set of nouns and verbs to form the questions and understand the responses.

Generally speaking the nouns need to refer to the functional components that might be in use on the system. The challenge is that the noun name space is very large, so we need to take a pragmatic path to defining an initial core set of component identifiers that future specification can evolve. For example, a noun in this context could be "TNC Client".

The verbs have a similar extensibility challenge, since the scope of the questions that the challenger may wish to ask about the noun can be broad. This specification will define a useful set of core questions while allowing for future extensibility. For example, a verb (or question) could be "What version of the identified component is present?"

Note that the IF-M protocol includes a simple naming scheme (discussed in section 3.1) that is used by the TNC Client and TNC Server to route messages to parties (frequently plug-ins) that expressed an interest in the particular component. The PTS Protocol described in this specification uses a more expressive naming scheme, since the PTS-IMC and PTS-IMV are identified by the IF-M messages.

2.4.1 Component Naming

It is envisioned that the challenger may need to assess a core set of components in order to determine the trustworthiness of the endpoint requesting service. These components typically will be comprised of other sub-components that potentially are shared with other components on the system. Such sharing is typical with components like libraries. Many components also have functional or security interdependencies on other components that are necessary in order to operate properly or to be considered trustworthy. These interrelationships between components form a dependency graph for which a challenger may ask the endpoint to provide attestation evidence in order to assess the components' trustworthiness. Therefore, it is envisioned that the attestation grammar will allow for the challenger to request attestation evidence for a particular component identified and to request the sub-component and trust dependency tree that exists below the component. The dependency tree could be requested implicitly thus removing the requirement for the challenger to know exact component identifiers for each sub-component of a requested component. For example, the challenger should be able to say "Give me attestation evidence for the Firewall kernel module and include all of its sub-components". Similarly, the challenger could request "Give me the attestation evidence for the Firewall kernel module and its underlying transitive trust chain".

2.5 Attestation Concepts

This section discusses some of the root trusted computing concepts that provide the basis for a trustworthy attestation protocol.

2.5.1 Transitive Trust Chain/Tree

At the time that a Trusted Platform is booted, a series of trusted loaders are run to instantiate the operational environment on the system. Each loader is responsible for measuring the loaded software image, recording the measurements and other meta-data about the software, optionally verifying the measurements against local policy and finally starting the software's execution. When viewed chronologically, this sequence of measuring results in a tree-like graph. Each branch in the tree is caused by a particular loader loading/starting software that functionally is capable of loading other software and therefore is responsible for doing contributing measurements. This graph from the bottom up is known as the transitive trust tree because there may be one or more branches in the path for a system that includes many loaders, such as a typical operating system. When viewed from the top (leaf node) down, the result is a transitive trust chain (since only one path leads down the trees to the root). The following diagram shows a simplified example of the transitive trust chain/tree graph for a simple system composed of basic kernel, operating system and application functions. Note that within each box a set of sub-components could exist such as libraries in the Application.

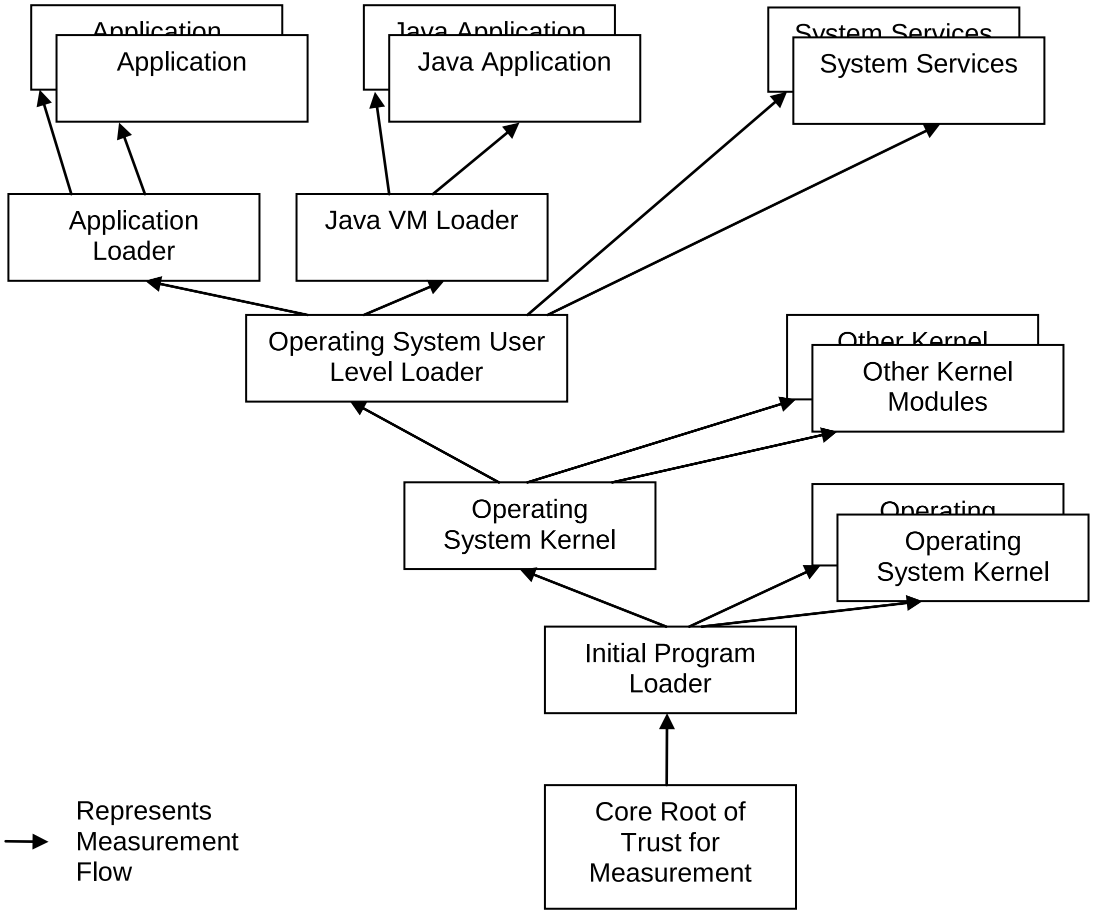


Figure 2: Example Transitive Trust Chain/Tree

In the simple example shown on figure 2, several transitive trust trees exist branching off of the Core Root of Trust for Measurement (CRTM) up to the user and Java applications. The CRTM is a fundamental trusted building block that is isolated from direct attack and remotely verifiable via a measurement or credential (e.g. Platform credential) issued by a trusted party. Just to highlight the potential complexity of this graph, it is possible that one of the Application components is a browser capable of loading ActiveX and Java applications, thus the browser component could have two sub-trees rising above it that the browser is responsible for measuring. Similarly, a library could be loaded multiple times on a platform if multiple applications are using it, so more complexity to the hierarchy is expected.

An example of a transitive trust chain could be traversing the measurement flow path from a System Service down to the Core Root of Trust for Measurement (System Service -> Operating System User Space Loader -> Operating System Kernel Loader -> Initial Program Loader -> Core Root of Trust for Measurement). In order for a remote challenger to establish the trustworthiness of the measurements provided for the System Service, the challenger might identify a particular System Service and request measurements for all the components down the transitive trust chain below. See figure 3 for an example of what a challenger could receive when requesting an attestation of a system service and its underlying transitive trust chain.

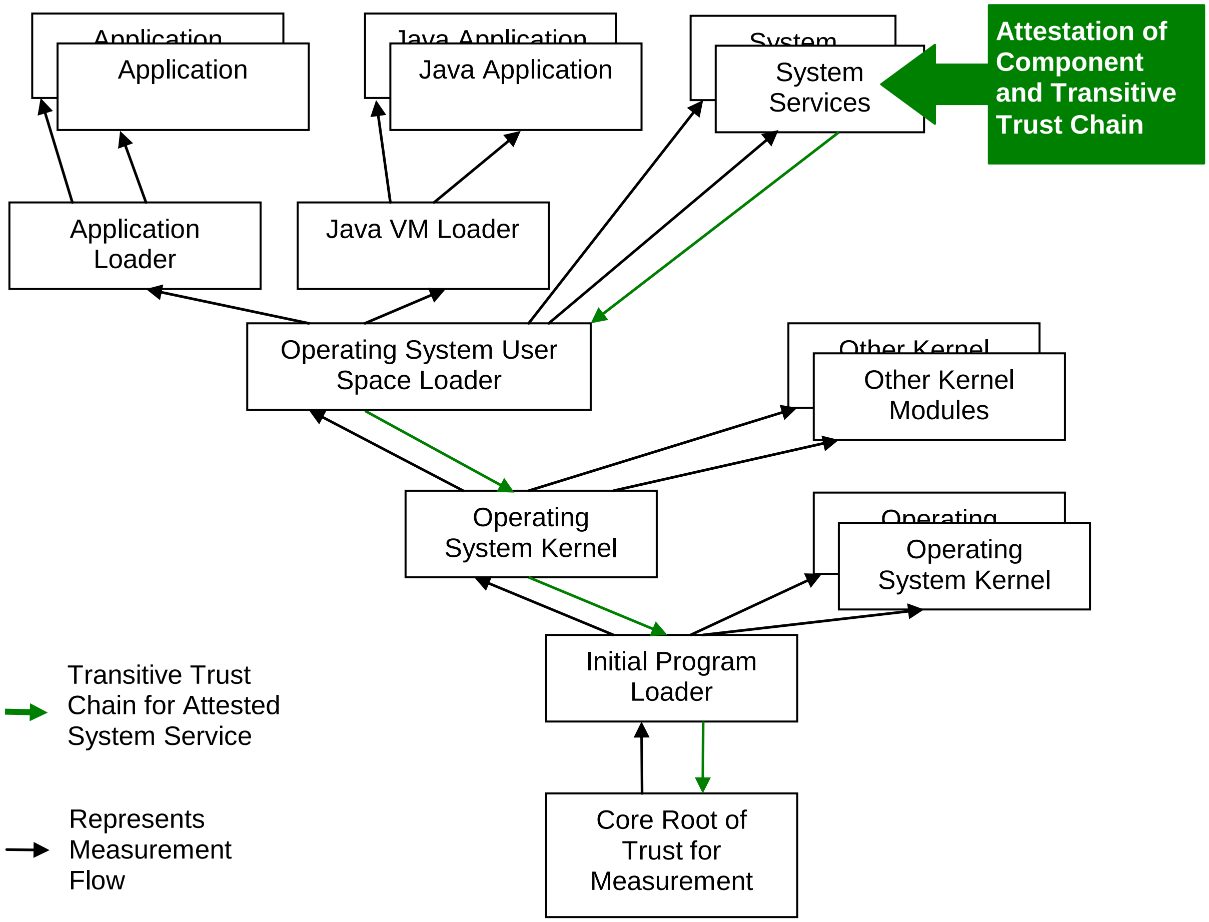


Figure 3: Transitive Trust Chain Below System Service

Note that one implementation model for a system is to include a System Service known as the Platform Trust Service (PTS) that is capable of handling the trusted measurement functionality of the loader. See figure 4 for the transitive trust chain when the PTS is employed.

.

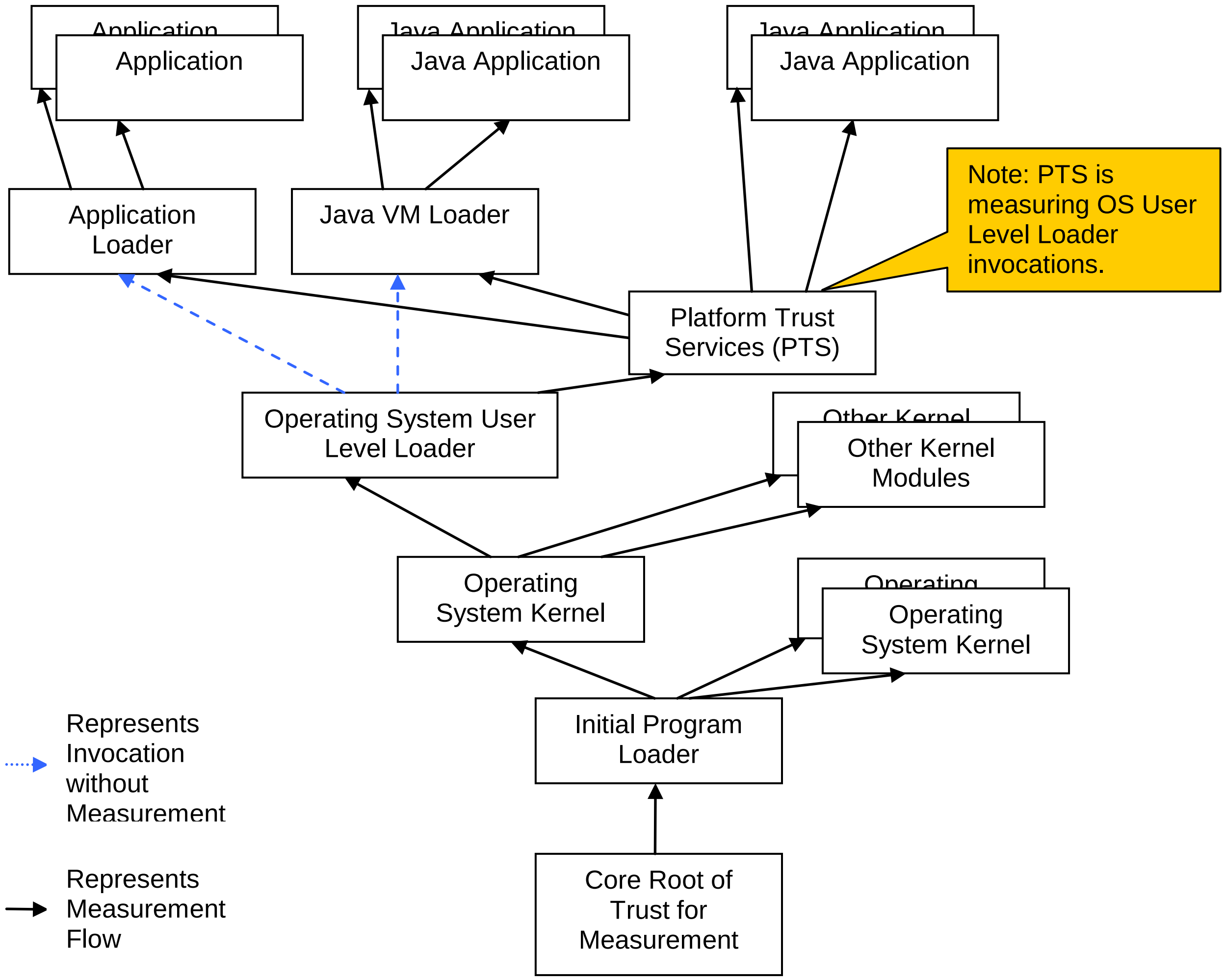


Figure 4: Transitive Trust Chain/Tree with PTS

In the PTS model, a remote challenger might request attestation evidence from the PTS and its dependencies and establish whether it is trustworthy to answer more granular questions about components that it might have measured. Once establishing trust in the PTS has occurred future attestations might request attestation evidence about components of the system above the PTS in the graph and only request the transitive trust chain down to the PTS. This enables a shorter report to be generated at a decrease in endpoint processing and network bandwidth thus improving scalability and performance.

2.5.2 TPM Quote

This section summarizes how the TPM's ability to quote PCRs provides assurance that the attestation evidence provided is trustworthy and protected from local tampering. The TPM provides a number of shielded storage locations and ordinals to perform transformations on data using the contents of these locations. For example, the TPM houses cryptographic keys that are only usable inside the TPM. One such key is the Attestation Identity Key (AIK) which is limited by the TPM to only be used to sign the contents of the TPM's PCRs (which are protected in TPM shielded storage). On a platform where the PCRs are set to reflect the operational content of the system, this combination of protected PCRs and AIK enable an attestation mechanism to be verifiable by remote parties.

Specifically, a remote challenger can request attestation information about a system and obtain an AIK signed set of “attestation evidence” that is cryptographically verifiable as having been generated using a TPM-resident AIK and PCRs. By empowering the remote challenger to be able to retrieve this signed set of PCRs with proof that they were resident inside a TPM, this mechanism allows the challenger to have confidence that it can determine (potentially using the measurement log and policy) what software has been run on the endpoint without fear of being spoofed by malware.

The TPM specification defines two variations of the quote ordinals: TPM_Quote and TPM_Quote2. Both ordinals are very similar in purpose but operate on slightly different information. When a remote challenger obtains TPM quote information as part of an attestation, it needs to verify that the quote information is authentic and came from the TPM on the endpoint. The challenger requires some or all of the following information to perform this verification:

- AIK Certificate
    - Contains public key associated with AIK private key held by TPM
    - Signed by trustworthy party that has reason to believe the TPM holds the AIK private key
    - See TCG Credentials specification for more information
- TPM version information (TPM_CAP_VERSION_INFO) when TPM_Quote2 used
    - See sections 3.10 and 3.11 for attribute details
- Set of PCRs to include in the quote
    - See section 3.15.1 for attribute details
- Hash algorithm used to create PCR values (SHA-1 for TPM 1.2 and prior)
    - See section 3.9.2 for attribute exchange details
- Current PCR values associated with requested PCR set
    - See section 3.15.1 for attribute details
- Hash algorithm used to hash together current PCR values (composite hash just for TPM_Quote)
    - See section 3.9.2 for attribute exchange details
- Secret session data in external data carrying assessment unique value (see D-H exchange below)
    - See section 3.8 for attribute exchange details
- Ordinal (whether TPM_Quote or TPM_Quote2 is used)
- Locality (for TPM_Quote2)
- TPM_Quote signature
    - Signature performed over TPM_QUOTE_INFO or TPM_QUOTE_INFO2 structure
    - See section 3.15.2 for attribute details

It’s the responsibility of the attestation protocol to enable the challenger to obtain the needed information from the above list such that the challenger can perform the necessary transforms to determine whether the TPM quote signature is authentic. The AIK certificate asserts that the signing key is held inside a TPM and assuming the TPM is authentic, will only allow the key to be used to perform a quote on other TPM resident information (PCRs).

The following sequence summarizes the information exchange required to have a TPM-based attestation:

- Challenger and remote platform exchange values and establish a shared secret for the attestation session (anti-replay)
- Challenger requests attestation evidence about a set of functional components on the platform
- PTS or another trusted measurement entity creates the attestation evidence
    - Some evidence may have previously been created such as during the boot while other could be measured at request time
- PTS sends the following attestation evidence to challenger when requested:
    - PCR identifiers used to hold measurements of requested functional component
    - PCR before and after values
    - Measurement (hash) of requested component
    - TPM_CAP_VERSION_INFO (TPM version information)
    - AIK certificate
    - Information about hash algorithm and PCR size (for hash agility)
    - TPM_Quote signature
- Challenger expected to perform:
    - Construct the equivalent TPM_QUOTE_INFO or TPM_QUOTE_INFO2 structure
        - This includes many of the elements above including: PCR set, PCR values, shared secret and potentially TPM version information and locality
    - Hash the structure using appropriate hash algorithm (for later comparison)
    - Use AIK public key to reverse TPM_Quote signature of structure hash
    - Compare computed hash value with one from signature
        - If hashes match then private key associated with AIK cert was used and the attestation evidence data matches what was in the TPM during TPM_Quote
        - If hashes do not match, something is not consistent and information should be disregarded

Based on the set of steps outlined above, the challenger can determine whether the set of attestation evidence it has received is authentic and can be checked against policy to determine the level of trustworthiness of the remote platform. Simply knowing the attestation evidence is authentic only means it accurately describes the remote system, not that the remote system is considered trustworthy by this challenger. The challenger may also wish to verify more granular information. This can be done by retrieving the integrity measurement log and optionally having it extended into one of the PCRs included in the quote.

2.5.3 Integrity Measurement Log

This section discusses the integrity measurement log and the types of meta-data that are desired to be recorded in order to facilitate a usable attestation. The integrity measurement log (IML), also known as stored measurement log in other TCG specifications, is a log of information about what was extended into each of the TPM's PCRs. Generally, each PCR will have its own log to ease separation of the entries since typically a remote challenger will require just the entries for a particular set of PCRs in the assessment. The specific contents of the IML are defined in the platform specifications; however this specification expects enough information to be present to allow the remote challenger to establish trust in the log and then learn more detailed information about the measured entity than is naturally present in just a measurement (hash) itself.

This protocol enables the remote challenger to request entries in the IML associated with a particular functional component (and optionally with its dependencies) that are extended into a particular PCR.

The desired contents of the IML are discussed in section 3.25, but include some indication of how recent the measurement was (date/time measurement taken), description of the functional component, type of measurement (hash of code) and of course the measurement value. This information can be useful to the remote challenger when comparing with its trustworthy configurations policy database. Because the IML entries can be used as part of the trust decision process, the IMLs need to be protected from modification by malware.

2.6 Supported Use Cases

This section describes the PTS Protocol use cases that must be supported. The primary usage of the PTS protocol is to enable the challenger (e.g. TNC Server) to initiate and iteratively send a series of one or more queries to obtain measurement information about the attested system (e.g. TNC Client). Because the underlying IF-T transport protocols offer different capabilities (e.g. number of roundtrips possible), it is expected that the PTS protocol exchange may need to fit into a small number of roundtrips and there may be restrictions on maximum data size.

The PTS Protocol binding to IF-M is fundamentally trying to allow a remote system to obtain verifiable (spoof resistant) information about a set of components installed and/or running on the endpoint. The remote system does not know what components are present, so must determine this through a series of queries. The attesting system has a set of policies that indicate what it considers to be acceptable posture, but the number of possible acceptable combinations of values is likely to be very large so isn't generally part of the query. For instance the PTS-IMV might ask for information and evidence about the OS boot loader component but wouldn't ask a very specific question such as whether the boot loader is the 'Vendor Loaders-r-Us boot loader version 1.2 build 19'. Instead it would ask for information about the OS boot loader and then compare the result with its database of acceptable values (likely stored in Reference Manifests).

The following are the use cases that the PTS protocol binding to IF-M must support:

Single Round Trip – Constrained Environment Assessment

1. Client-Initiated: Requestor proactively starts an assessment in conjunction with its request for access to a protected resource or service (e.g. IP connectivity). In order to optimize bandwidth and roundtrips, the attested system sends posture attributes proactively based upon its local policy or history of previous assessment from this challenger. The challenger receives these posture attributes and decides whether the system has provided adequate information. If adequate posture was received to make a decision, the challenger decides whether access should be granted.
2. Server-Initiated: Challenger initiates assessment requesting several posture attributes about the requestor. The attested system responds with these attributes (if allowed by privacy policy). The challenger decides whether the system is trustworthy.

Multiple Round Trips – TCP/IP Connected Assessment

3. Client-Initiated: Requestor proactively starts an assessment in conjunction with its request for access to a protected resource or service. The requestor sends posture attributes proactively based upon its local policy or history of previous assessment from this challenger. The challenger receives these posture attributes and decides whether the system has provided adequate information. If more information is required, the challenger sends one or more attestation evidence requests to obtain the needed additional posture. Upon receiving the additional attestation evidence, the challenger may decide to request another set of attributes. This process will continue until the challenger has received sufficient attestation evidence to make a trust decision.
4. Server-Initiated: Challenger initiates assessment and requests posture attributes about the attested system. The attested system responds with these attributes (if allowed by privacy policy). The challenger determines that additional queries are required to assess the attested system so sends additional requests for information and processes the responses until sufficient information has been obtained to make a policy decision.

Single and Multiple Round Trip Sub-cases for Attribute Encodings

a. PTS-IMC supports only TLV-based posture attributes and PTS-IMV supports both XML and TLV-based attributes. PTS-IMV needs to be able to determine that attested system can only respond to TLV-based encodings.

b. PTS-IMC system supports only XML-based posture attributes and PTS-IMV supports both XML and TLV-based attributes. PTS-IMV needs to be able to determine that attested system can respond only to XML-based encoded attributes.

c. PTS-IMV supports only TLV-based posture attributes and PTS-IMC supports both XML and TLV-based attributes. This sub-case can be resolved by PTS-IMV requesting attributes with only a TLV-based encoding.

d. PTS-IMV supports only XML-based posture attributes and PTC-IMC supports both XML and TLV-based attributes. This sub-case is least likely to happen since the IF-M base protocol is TLV encoded and can be resolved by PTS-IMV asking only for XML encoded attributes.

Note: in order to ensure interoperability, both the PTS-IMC and PTS-IMV will be required to support a core set of the TLV-based attributes so an attestation exchange is possible. The support for the non-core TLV-based attributes and the XML-based attributes is recommended but not required for base-level interoperability. Therefore, some of the attribute encoding usages described above are not mandatory to support but included for completeness.

2.7 Non-supported Use Cases

None

2.8 Requirements

Here are the requirements that IF-M must meet in order to successfully play its role in the TNC architecture.

- Flexibility

  The PTS protocol binding to IF-M MUST be able to carry attestation values in TLV, XML or a mixed encoding of attributes. Individual attributes SHOULD be of a single encoding but the protocol MUST support carrying both within the same IF-M message.

  The PTS protocol binding to IF-M MUST enable a multi-roundtrip dialog. Certain constrained network environments (e.g. EAPoL) might limit the size and number of roundtrips that may occur in a dialog. The PTS-IMC and PTS-IMV MUST be implemented to allow for multiple round trips when the network connection allows. The PTS-IMV SHOULD consider limitations of the underlying transport factoring in its policies when requesting attestation evidence so it can maximize the effectiveness of the assessment (e.g. don't request large Integrity Reports when running over a bandwidth constrained network connection). If no such limitations are apparent, the PTS-IMC and PTS-IMV SHOULD take advantage of the available bandwidth and roundtrips.

- Secure

  The PTS protocol binding to IF-M MUST be capable of protecting attestation evidence messages end to end between the PTS-IMC and PTS-IMV. This protection MUST guard against active and passive attackers by offering bi-directional user or system-level authentication, detection of alteration or replay of the messages, and confidentiality of the message contents as mandated by deployment policy. IF-M security protections enable PTS-IMV to verify the PTS generated information is received in its entirety, potentially using TPM-resident keys. Note that the PTS-IMV MUST NOT require a local TPM to be used to verify the attestation evidence.

- Privacy

  The PTS protocol binding to IF-M MUST NOT require the challenger to request information about everything that is running or installed on a particular endpoint in order to determine its trustworthiness. The security posture of a machine normally can be assessed without knowing about every application that has been run on the system, so the attestation protocol needs to enable a model where the challenger can ask questions to understand the security relevant portions of the endpoint. In some cases, these attestation questions can be broadly expressed (e.g. 'tell me about the entire running OS kernel on the system) if a broad aspect of the system could impact security of many applications.

- Efficient

  The TNC architecture delays network access until the endpoint is determined to not pose a security threat to the network based on its asserted integrity information. To minimize user frustration, the PTS protocol binding to IF-M protocol MUST minimize delays and make IF-M communications as rapid and efficient as possible. Efficiency is also important when you consider that some network endpoints are small and low powered, some networks are low bandwidth and/or high latency, and some IF-T protocols only allow one packet in flight. PTS-IMC and PTS-IMV SHOULD be able to adapt to different network limitations (roundtrips, bandwidth constraints, time outs) to perform a timely assessment.

- Transport Independence

  The PTS protocol binding to IF-M message MUST be agnostic of the underlying IF-T transport protocol and thus not change the message format when different IF-T protocols are used. However, PTS-IMCs and PTS-IMVs may alter their level of verbosity (payload size) when faced with underlying protocols that are bandwidth constrained.

- Extensible

  The PTS protocol binding to IF-M attributes MUST be extensible allowing for additional types of components and new measurement information to be defined by future specifications that extend the initial protocol without backward compatibility concerns.

2.9 Non-Requirements

None

2.10 Assumptions

Here are the assumptions that the PTS Protocol binding to IF-M protocol makes about other components in the attestation architecture.

- Reliable Message Delivery

  The underlying protocol transporting the defined IF-M messages will provide in-order, timely and highly reliable delivery between the challenger and the attested requestor. In the TNC architecture, reliable communication would normally be provided by IF-TNCCS and IF-T.

2.11 Keywords

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

2.12 IF-M Message Diagram Conventions

This specification defines the syntax and semantics of the PTS protocol messages carried within the TNC's IF-M protocol. 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 field (e.g. flags) values are described referring to the position of the bit within the field. These bit positions are numbered from the most significant bit through the least significant bit so a one octet field with only bit 0 set has the value 0x80.

3 PTS Protocol

This section specifies how the PTS protocol integrates into the IF-M protocol within the TNC architecture and describes the expected message exchange. The IF-M protocol was designed to be very extensible via new standards while allowing vendors to manage their own name space alternatives. This specification leverages this extensibility by adding additional IF-M Subtypes (AKA IF-M Component Types) and IF-M Attribute Types. This section discusses the additional enumerations of both types in the context of the TCG/TNC standard name spaces.

3.1 IF-M Subtype (AKA IF-M Component Type)

The TNC IF-TNCCS protocol provides a general message batching protocol capable of carrying one or more IF-M messages between the TNC Client and TNC Server. When IF-TNCCS is carrying an IF-M message, the IF-TNCCS message headers contain a 32 bit identifier called the IF-M Subtype. The IF-M Subtype field indicates the type of component associated with all of the IF-M messages carried by the IF-TNCCS message. The core set of IF-M Subtypes are defined in the IF-M specification. In order for the TNC protocols to carry messages associated with the PTS, this specification adds the following component enumeration to table in section 4.4 of the IF-M specification using the TCG Standard name space (SMI Private Enterprise Number 0x005597):

IF-M Subtype Component Type Name	TNC Standard Component Definition	Description
Platform Trust Service (PTS)	0x00000001	Platform Trust Service software associated with PTS-IMC supporting the PTS protocol binding to IF-M protocol.

Architecturally, each TNC Client supporting the PTS Protocol includes a component known as the PTS-IMC that will receive messages sent with the PTS component type. The PTS-IMC is an IMC that is responsible for receiving IF-M messages destined for the PTS and translating and proxying the requests into a form usable by the PTS (typically using the local IF-PTS interface). The PTS-IMC also sends the properly formatted PTS protocol responses back to the TNC Server. Similarly, the PTS-IMV exists on the TNC Server and is responsible for interpreting the PTS responses and making policy decisions based upon the received information. Each IF-M message described in this specification is intended to be sent between the PTS-IMC and PTS-IMV, so will be carried in an IF-TNCCS message indicating an IF-M Subtype of PTS. It is not envisioned that the attributes defined in this specification will be applicable to other types of components (thus other IF-M Subtypes), however the existing core set of attributes may be applicable to the PTS (e.g. obtaining vendor and version information about the PTS). Note that the PTS ComponentID and the Functional Component Type described later in this specification are only visible to the PTS-IMC and PTS-IMV and are not included in the message routing field (IF-TNCCS's IF-M Subtype) included in the IF-TNCCS protocol. The IF-TNCCS protocol will always include the PTS Subtype defined in this section when carrying the PTS Protocol over IF-M.

3.2 IF-M TLV Format

The PTS protocol binding described in this document is an extension of the IF-M protocol described in the TNC Architecture. IF-M was designed to be very flexible to carry a wide variety of types of IF-M attributes (e.g. Product Information) that pertain to an enumerated set of component types (e.g. Firewall). IF-M attributes may be carried from IMC to IMV or vice versa and may carry information about state or other messages to be sent between an IMC and an IMV. Therefore the PTS Protocol binding to IF-M is largely a collection of attribute definitions relevant to the PTS-based assessment of the system.

Figure 5 (reproduced from the IF-M specification) shows the format of an IF-M attribute TLV. Multiple IF-M attributes can be sent in a single IF-TNCCS message, each housed within an attribute TLV. The PTS protocol binding to IF-M defines one new "Attribute Type" definition (PTS) within the TCG standard "Attribute Type Vendor ID" name space and the corresponding "Attribute Value" contents expected to be understood by IMCs and IMVs receiving PTS-related messages. Note that the IF-M protocol supports up to 2^{32} attribute types per vendor name space (TCG owns one of these name spaces) and is capable of carrying messages up to 2^{32}-12 (TLV is included in the length) octets in length, so this scalability is leveraged by the PTS protocol.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |       Attribute Type Vendor ID (TCG)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Attribute Type                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Attribute Length                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Attribute Value (Variable Length)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                       . . . . . . . .                         ~
```

Figure 5: IF-M Attribute Format

As shown in figure 6, the attributes described in this specification will use the TCG Vendor ID (0x005597) indicating the Attribute Type is from the TCG standard name space. This document will specify many new Attribute Types and the associated Attribute Values that carry the PTS protocol.

3.3 PTS Messaging Exchange

This section discusses some envisioned message exchanges necessary to perform a PTS-based attestation. The following diagram shows an example PTS protocol attribute exchange including optional message exchanges and resulting in an Integrity Report.

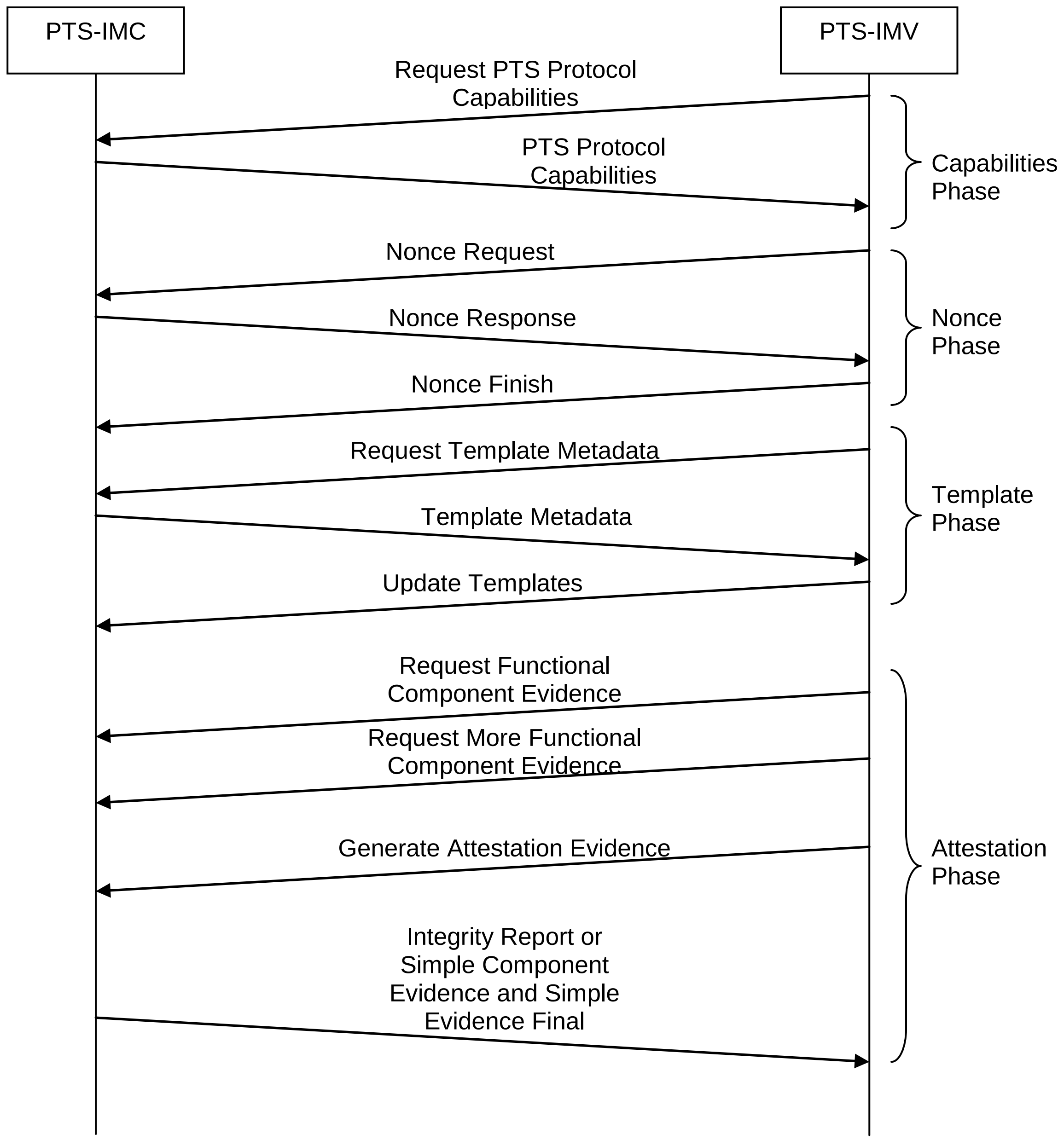

PTS-IMC
PTS-IMV
Request PTS Protocol Capabilities
PTS Protocol Capabilities
Capabilities Phase
Nonce Request
Nonce Response
Nonce Finish
Nonce Phase
Request Template Metadata
Template Metadata
Update Templates
Template Phase
Request Functional Component Evidence
Request More Functional Component Evidence
Generate Attestation Evidence
Integrity Report or Simple Component Evidence and Simple Evidence Final
Attestation Phase

Figure 6: Example PTS protocol message exchange

After the TNC protocols (IF-T transport and IF-TNCCS message brokering) have established a session between the TNC client and TNC server, the PTS-IMC and PTS-IMV are able to exchange messages to start the attestation. Figure 6 shows the logical grouping of message types into different “phases”. These phases can occur at any time during an assessment (e.g. the capabilities phase could be repeated to change an earlier selection) and in some cases a phase might not occur at all (e.g. template phase).

Initially, an optional capabilities phase exchange occurs allowing the PTS-IMV to evaluate the capabilities of the PTS-IMC to determine what optional features the PTS-IMC supports in case this impacts the attributes the PTS-IMV might wish to request.

After the capabilities phase, the PTS-IMV could initiate the nonce phase to establish a shared secret to be used by both parties for binding the attributes exchanged to this particular attestation session. After the three-way handshake, both parties now have an assessment unique value that can be factored into assessment operations such as a TPM quote (as external data).

Next, an optional template phase could occur to enable the PTS-IMV to determine if the PTS (or PTS-IMC) is using a PTS-IMV supported version of a Reference Manifest. Note that this phase isn't necessary when the assessment plans to use only TLV-based attribute (no XML reports). Once the PTS-IMC and PTS-IMV have agreed upon a Reference Manifest version to use as a template for formatting the Integrity Report, they are now ready to start the assessment.

The assessment phase begins with the PTS-IMV sending one or more (two are shown) requests for sets of attestation evidence for functional components that might exist on the endpoint. The PTS-IMC passes each of these requests off to the PTS, so it can start measuring and tracking the information to be included in the attestation. Depending on the attributes used, the PTS and PTS-IMC might respond with an attribute immediately (e.g. when a Request File Measurement attribute is used) or might wait until it receives another attribute before responding (e.g. when a Request Functional Component Evidence attribute is used). When the Request Functional Component Evidence attribute is received by the PTS, the PTS adds the described component(s) to the set of attestation evidence that will be reported later when a Generate Attestation Evidence attribute is received by the PTS. At this point the PTS can clear its state about this assessment's components, since future assessments on this session will now start with a clean slate.

Once the PTS-IMV has received attestation evidence (XML-based or TLV-based) it performs an integrity verification if a relevant TPM_Quote was provided. In order to verify that the Integrity Report (XML) or Simple Component Evidence (TLV) is authentic, the PTS-IMV will check the digital signature on the report to ensure it is correct and signed by a trustworthy PTS. Next, if the PTS-IMV is using an Integrity Report it will go through the set of measurements and verify that they in aggregate (via repeated extends), match the values included in the TPM quote. If the quote confirms the authenticity of the attestation evidence, the PTS-IMV will now compare the received attestation evidence with its Reference Manifest or measurement database to determine whether the components are compliant with policy.

When an Integrity Report or Verification Report is requested from the PTS, the PTS constructs the XML document including all the meta-data and measurements for each requested component and optionally backs up the measurements with a TPM-based quote operation. When the challenger receives the quote, it knows from processing the quote the value of a set of PCRs on the system (signed by an AIK TPM-resident private key). Inside the Integrity or Verification Report exists a copy of the integrity log for the involved PCRs. The challenger can use the measurements recorded in the integrity log to compute what the expected value would be in the PCR (if the report was valid) and compare the result against the value in the quoted PCR. This process is much more involved than merely requesting evidence in individual attributes, but should provide additional confidence in the values returned.

3.3.1 Multiple TNC Component Assessment

It's envisioned that many of the existing TNC IF-M attributes may be used during an endpoint assessment in conjunction with the PTS-based attestation evidence obtained via this protocol. These existing TNC attributes may be used in IF-M messages for other components alongside IF-M messages destined for the PTS containing the attributes defined in this specification. For example, an assessment might include posture requests for attributes about the firewall, anti-virus and the PTS integrity of the TNC software all in the same set of IF-TNCCS messages. Note that because each IF-TNCCS message pertains to a single type of component, the assessment would just include several IF-TNCCS messages containing per-component posture requests. The TNC Client distributes the requests to the IMCs that will process the requests including one associated with the PTS (known as the PTS-IMC). The PTS-IMC will respond to the attributes described in this specification. Concurrently, other IMCs on the TNC Client would process and respond to the requests pertaining to other component types on the endpoint.PTS IF-M Attributes

This section specifies the new IF-M Attribute Types required in order to support assessment of PTS-based attestation evidence using the PTS protocol. These attributes extend the core set of Attribute Types described in the IF-M specification.

3.4 PTS IF-M Attribute Enumeration

The attributes defined in this section all use the TCG SMI Private Enterprise Number (0x005597) in the Attribute Type Vendor ID field of the IF-M Attribute Header shown in section 3.2. The following table briefly describes each attribute and defines the value to be used in the Attribute Type field of the IF-M Attribute Header. The IF-M specification has grouped attributes that have a similar purpose to ease recognition by a recipient (e.g. PTS related attributes fall between 0x00100000-0xFFF00000). Later subsections provide detailed specifications for the contents of each attribute.

Attribute Purpose	**Attribute Name**	**IWG Standard Attribute Type**	**Description**
PTS Protocol Negotiations			
	Request PTS Protocol Capabilities	0x01000000	Sender requests discovery of peer’s PTS Protocol oriented capabilities.
	PTS Protocol Capabilities	0x02000000	Sender reports its support for optional PTS related capabilities.
	D-H Nonce Parameters Request	0x03000000	PTS-IMC or PTS-IMV requests that the peer send its supported D-H parameters so a D-H exchange to establish an assessment nonce can commence.
	D-H Nonce Parameters Response	0x04000000	D-H nonce responder responds with the D-H parameters desired including its supported algorithms and group
	D-H Nonce Finish	0x05000000	D-H nonce initiator selects the hash algorithm and offers its nonce values
	PTS Measurement Algorithm Request	0x06000000	Attribute sent by PTS-IMV to trigger the establishment of a hash algorithm for use with PTS measurements. This

			attribute proposes one or more desired hash algorithms to the PTS-IMV.
	PTS Measurement Algorithm Selection	0x07000000	PTS-IMC response attribute indicating the algorithm selected for use by the PTS for all future file measurement during the assessment. The PTS-IMV MAY request a change to the algorithm by sending another PTS Hash Algorithm Request attribute. If the PTS-IMC does not support the algorithms proposed by the PTS-IMV, it MUST return an IF-M Error attribute indicating TCG_PTS_HASH_ALG_NOT_SUPPORTED.
	Get TPM Version Information	0x08000000	Request for TPM Version information. This information could be included in the TPM_Quote2 so is needed for verification of the attestation.
	TPM Version Information	0x09000000	TPM Version information (TPM_CAP_VERSION_INFO structure) for evaluation of attestation evidence.
	Request Template Reference Manifest Set Metadata	0x0A000000	Attribute contains a set of functional identifiers associated with Reference Manifests held by the other party. The sender of this attribute wishes to learn what Reference Manifest level is possessed by

			the other party for each functional identifier. Higher Reference Manifest levels indicate newer Reference Manifest layouts.
	Template Reference Manifest Set Metadata	0x0B000000	Attribute contains the current Reference Manifest level information in use by the sender (e.g. endpoint PTS). This is used to determine whether the sender has a template Reference Manifest in common with the other party.
	Update Template Reference Manifest	0x0C000000	Attribute contains a Reference Manifest that the sender proposes to use as a template Reference Manifest during the attestation.
	Get Attestation Identity Key	0x0D000000	Attribute for requesting the PTS-IMC TPM’s attestation identity key (AIK).
	Attestation Identity Key	0x0E000000	Attribute containing the PTS-IMC TPM’s attestation identity key (AIK) either as a naked public key or a certificate.
PTS-based Attestation Evidence			
	Request Functional Component Evidence	0x00100000	Add component to pending attestation evidence based upon functional component name.
	Generate Attestation Evidence	0x00200000	Sent by PTS-IMV to request the generation of attestation evidence for the components previously requested.

			The report can be requested to be: an XML-based Integrity Report, an XML-based Verification Report or a TLV-based Component Evidence report.
	Simple Component Evidence	0x00300000	Contains TLV-encoded attestation evidence about the requested functional component.
	Simple Evidence Final	0x00400000	Contains a TLV-encoded set of information covering the entire set of Simple Component Evidence Reported. For example, it might include a signature over the entire set of Simple Component Evidence attributes returned.
	Verification Result	0x00500000	Attribute contains a PTS generated Verification Result XML document containing the results of local verification of the requested component.
	Integrity Report	0x00600000	Attribute contains a PTS generated Integrity Report XML document containing the requested information.
	Request File Metadata	0x00700000	Attribute contains the filename that the sender wishes to retrieve its metadata (e.g. file size) for subsequent evaluation. The file name MUST be an absolute (full) path so it is unambiguous for the PTS. The PTS MUST return a TCG_PTS_INVALID_PATH

			error if the path is not a full path or TCG_PTS_FILE_NOT_FOUND if the path refers to a non-existent file.
	Windows-Style File Metadata	0x00800000	Attribute contains the Windows oriented metadata about a particular file requested. Metadata includes properties of the file including its owner, size and file type.
	Unix-Style File Metadata	0x00900000	Attribute contains the Unix oriented metadata about a particular file requested. Metadata includes properties of the file including its owner, size and file type.
	Request Registry Value	0x00A00000	Attribute requests the value of a particular registry key. This attribute is envisioned to be Windows only, so other endpoint operating systems might respond with a TCG_PTS_REG_NOT_SUPPORTED Error attribute. If the requested registry value is not present on the endpoint, a TCG_PTS_REG_KEY_NOT_FOUND MUST be returned in a IF-M Error attribute.
	Registry Value	0x00B00000	Attribute includes the value(s) contained in the requested registry key.
	Request File Measurement	0x00C00000	Attribute contains the filename that the sender wishes to

			retrieve its measurement (hash of file contents on disk) for subsequent evaluation. The file name MUST be an absolute (full) path so it is unambiguous for the PTS. The PTS MUST return a TCG_PTS_INVALID_PATH if the path is not a full path or TCG_PTS_FILE_NOT_FOUND if the path refers to a non-existent file.
	File Measurement	0x00D00000	Attribute contains the measurement (hash) of the particular file requested. The measurement will use the negotiated PTS Measurement Hash Algorithm.
	Request Integrity Measurement Log	0x00E00000	Attribute requests the Integrity Measurement Log entries associated with the criteria (e.g. PCR) included.
	Integrity Measurement Log	0x00F00000	Attribute contains a subset of the IML with the core values necessary to attest the endpoint based on the PCR values.

Figure 7: PTS Related IF-M Attribute Types

The following sections discuss the usage, format and semantics of the Attribute Value field for each of the PTS oriented attribute types defined above. These fields are included in the Attribute Value portion of the Attribute TLV.

3.5 Attribute Support Requirements

This section defines the requirements for which attributes are to be supported by the PTS-IMC and PTS-IMV. Generally, the PTS Protocol allows for several optional capabilities to be supported and negotiated during an assessment, so many of the attributes' requirements factor in whether a particular capability is being offered by a party (see conditional situation in table below). For example, if a PTS and PTS-IMC choose to support XML-based attestation evidence reporting, the PTS and PTS-IMC MUST support the Integrity Report attribute. However, if the PTS and PTS-IMV are not supporting XML-based reporting, the Integrity Report attribute is not supported. Therefore, the "conditional situation" column is included to describe the circumstances when the requirement is to apply.

Attribute Name	**Component Involved**	**Requirements**	**Conditional Situation**
Request PTS Protocol Capabilities	PTS-IMC or PTS PTS-IMV	Send: MUST NOT Receive: MUST Send: MUST Receive: MUST NOT	
PTS Protocol Capabilities	PTS-IMC or PTS PTS-IMV	Send: MUST Receive: MUST NOT Send: MUST NOT Receive: MUST	
D-H Nonce Parameters Request	PTS-IMC or PTS PTS-IMV	Send: MUST NOT Receive: SHOULD Send: SHOULD Receive: MUST NOT	Nonce phase is optional to support. However, if not supported then this attribute is not required.
D-H Nonce Parameters Response	PTS-IMC or PTS PTS-IMV	Send: SHOULD Receive: MUST NOT Send: MUST NOT Receive: SHOULD	Nonce phase is optional to support. However, if not supported then this attribute is not required.
D-H Nonce Finish	PTS-IMC or PTS PTS-IMV	Send: MUST NOT Receive: SHOULD Send: SHOULD Receive: MUST NOT	Nonce phase is optional to support. However, if not supported then this attribute is not required.
PTS Measurement Algorithm Request	PTS-IMC or PTS PTS-IMV	Send: MUST NOT Receive: MUST Send: MUST Receive: MUST NOT	
PTS Measurement Algorithm Selection	PTS-IMC or PTS PTS-IMV	Send: MUST Receive: MUST NOT Send: MUST NOT Receive: MUST	
Get TPM Version Information	PTS-IMC and/or PTS	Send: MUST NOT Receive: MUST	Support for TPM-backed PTS attestation evidence SHOULD be supported.

	PTS-IMV	Send: MUST Receive: MUST NOT	However, if not supported then this attribute is not required.
TPM Version Information	PTS-IMC and/or PTS PTS-IMV	Send: MUST Receive: MUST NOT Send: MUST NOT Receive: MUST	Support for TPM-backed PTS attestation evidence SHOULD be supported. However, if not supported then this attribute is not required.
Request Template Reference Manifest Set Metadata	PTS-IMC or PTS PTS-IMV	Send: MUST NOT Receive: SHOULD Send: SHOULD Receive: MUST NOT	Template phase is optional to support and only necessary when XML-based attestation evidence is to be used. However, if XML-based attestation evidence is not supported, then this attribute is not required.
Template Reference Manifest Set Metadata	PTS-IMC or PTS PTS-IMV	Send: SHOULD Receive: MUST NOT Send: MUST NOT Receive: SHOULD	Template phase is optional to support. However, if not supported then this attribute is not required.
Update Template Reference Manifest	PTS-IMC or PTS PTS-IMV	Send: MUST NOT Receive: SHOULD Send: SHOULD Receive: MUST NOT	Template phase is optional to support. However, if not supported then this attribute is not required.
Get Attestation Identity Key	PTS-IMC or PTS PTS-IMV	Send: MUST NOT Receive: MUST Send: MUST Receive: MUST NOT	Support for TPM-backed PTS attestation evidence SHOULD be supported. However, if not supported then this attribute is not required.
Attestation Identity Key	PTS-IMC or PTS PTS-IMV	Send: MUST Receive: MUST NOT Send: MUST NOT Receive: MUST	Support for TPM-backed PTS attestation evidence SHOULD be supported. However, if not supported then this attribute is not required.
Request Functional	PTS-IMC or PTS	Send: MUST NOT Receive: MUST	

Component Evidence	PTS-IMV	Send: MUST Receive: MUST NOT	
Generate Attestation Evidence	PTS-IMC or PTS PTS-IMV	Send: MUST NOT Receive: MUST Send: MUST Receive: MUST NOT	
Simple Component Evidence	PTS-IMC or PTS PTS-IMV	Send: MUST Receive: MUST NOT Send: MUST NOT Receive: MUST	
Simple Evidence Final	PTS-IMC or PTS PTS-IMV	Send: MUST Receive: MUST NOT Send: MUST NOT Receive: MUST	
Verification Result	PTS-IMC or PTS PTS-IMV	Send: SHOULD Receive: MUST NOT Send: MUST NOT Receive: SHOULD	Support for XML-encoded PTS attestation evidence SHOULD be supported. However, if not supported then this attribute is not required.
Integrity Report	PTS-IMC or PTS PTS-IMV	Send: SHOULD Receive: MUST NOT Send: MUST NOT Receive: SHOULD	Support for XML-encoded PTS attestation evidence SHOULD be supported. However, if not supported then this attribute is not required.
Request File Metadata	PTS-IMC or PTS PTS-IMV	Send: MUST NOT Receive: SHOULD Send: SHOULD Receive: MUST NOT	
Windows-Style File Metadata	PTS-IMC or PTS PTS-IMV	Send: SHOULD Receive: MUST NOT Send: MUST NOT Receive: SHOULD	This attribute is applicable to Windows-based PTS-IMC, so this requirement does not apply when the PTS-IMC is running on another operating system.
Unix-Style File Metadata	PTS-IMC or PTS	Send: SHOULD Receive: MUST NOT	This attribute is applicable to Unix-based PTS-IMC, so this requirement does not

	PTS-IMV	Send: MUST NOT Receive: SHOULD	apply when the PTS-IMC is running on another operating system.
Request Registry Value	PTS-IMC or PTS PTS-IMV	Send: MUST NOT Receive: SHOULD Send: SHOULD Receive: MUST NOT	
Registry Value	PTS-IMC or PTS PTS-IMV	Send: SHOULD Receive: MUST NOT Send: MUST NOT Receive: SHOULD	This attribute is applicable to Windows-based PTS-IMC, so this requirement does not apply when the PTS-IMC is running on another operating system (e.g. Unix) that doesn't support a registry.
Request File Measurement	PTS-IMC or PTS PTS-IMV	Send: MUST NOT Receive: SHOULD Send: SHOULD Receive: MUST NOT	
File Measurement	PTS-IMC or PTS PTS-IMV	Send: SHOULD Receive: MUST NOT Send: MUST NOT Receive: SHOULD	
Request Integrity Measurement Log	PTS-IMC or PTS PTS-IMV	Send: MUST NOT Receive: SHOULD Send: SHOULD Receive: MUST NOT	
Integrity Measurement Log	PTS-IMC or PTS PTS-IMV	Send: SHOULD Receive: MUST NOT Send: MUST NOT Receive: SHOULD	Support for TPM-backed PTS attestation evidence SHOULD be supported, therefore this attribute should also be supported. However, if TPM-backed attestation evidence is not supported, then this attribute SHOULD still be supported to cover the PTS's created/managed Integrity Measurement Logs.

3.6 Request PTS Protocol Capabilities

This attribute describes the PTS protocol related capabilities of the PTS-IMV while requesting the PTS-IMC to respond with a PTS Protocol Capabilities attribute selecting the capabilities to be used for this exchange. Because the PTS Protocol is somewhat complex, this attribute exchange allows implementations that do not support some of the optional functionality to discover what functionality the peer entity is able to support. Every PTS-IMC and PTS-IMV MUST NOT require the other party to support any of the optional features in order to perform a PTS based assessment.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Reserved                  |C|V|D|T|X|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by compliant implementations.
X – XML-based Evidence Support	The sender supports use of XML-based attributes for attestation evidence exchange (e.g. Integrity Report) 0 – Does not support 1 – Does support
T – Trusted Platform Evidence	The sender supports the use of the local Trusted Platform and is able to (willingness is subject to policy) provide TPM-based attestation evidence. 0 – Does not support 1 – Does support
D – DH Nonce Negotiation Support	The sender indicates whether it is capable of supporting the Diffie-Hellman Nonce Negotiation (see section 3.8) to establish a shared secret bound to this particular session. This feature is optionally present to help detect man-in-the-middle attacks. The IF-T layer of the TNC architecture may also provide for the creation of a session layer secret, so these attributes may not be required for some deployments. 0 – Does not support 1 – Does support
V – Verification Support	The sender requests to know whether the PTS and other measurement agents are able to perform and enforce local verification of running processes against policy. For example, if the PTS were integrated with the process loader, the PTS could measure and compare against policy any newly loaded processes and prevent non-compliant software from running. 0 – Not requested 1 – Request to know if supported

C – Current (in-memory) Evidence	The sender requests to know whether the PTS and other measurement agents are able to take measurements of the requested software component in memory (assuming its already running). It is envisioned that many measurement agents might only be able to measure software as its being started, so this capability would indicate this more sophisticated measuring. 0 – Not requested 1 – Request to know if supported

PTS-IMC supporting this specification MUST support reception and processing of this attribute, while the PTS-IMV MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.7 PTS Protocol Capabilities

This attribute describes the selected PTS protocol related capabilities for this assessment but does not request a response from the peer describing its capabilities. This attribute is expected to be used after a party receives a Request PTS Protocol Capabilities attribute, and the PTS-IMC MUST select only capabilities offered in the Request PTS Protocol Capabilities attribute. Because the PTS Protocol is somewhat complex, this protocol allows implementations to dynamically determine if the PTS-IMC and PTS-IMV support an optional capability before attempting to use it. PTS protocol implementations MUST NOT require the support of optional protocol capabilities in order to interoperate.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Reserved                |C|V|D|T|X|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by compliant implementations.
X – XML-based Evidence Support	The sender selects the use of XML-based attributes for attestation evidence exchange (e.g. Integrity Report) for this particular assessment. 0 – Not selected (don’t use during assessment) 1 – Selected
T – Trusted Platform Evidence	The sender is capable of sendingTPM-based attestation evidence. 0 – Not selected (don’t use during trusted platform evidence) 1 – Selected (trusted platform evidence available)
D – DH Nonce Negotiation Support	The sender selects whether to use the Diffie-Hellman Nonce Negotiation (see section 3.8) to establish a shared secret bound to this particular session. This feature is optionally present to help detect man-in-the-middle attacks. The IF-T layer of the TNC architecture may also provide for the creation of a session

	layer secret, so these attributes may not be required for some deployments. If a D-H Nonce Negotiation occurs, it replaces the current shared secret between the PTS-IMC and PTS-IMV and establishes the new Secret-Assessment-Value to be used for TPM quote operations. 0 – Not selected (don't use during assessment) 1 – Selected (request usage of DH-PN)
V – Verification Support	The sender indicates whether the local PTS and other measurement agents are able to perform and enforce local verification of running processes against policy. For example, if the PTS were integrated with the process loader, the PTS could measure and compare against policy any newly loaded processes and prevent non-compliant software from running. 0 – Does not support (or no answer if not requested) 1 – Does support
C – Current (in-memory) Evidence	The sender indicates whether the local PTS and other measurement agents are able to take measurements of the requested software component in memory (assuming its already running). Its envisioned that many measurement agents might only be able to measure software as its being started, so this capability would indicate this more sophisticated measuring. 0 – Does not support (or no answer if not requested) 1 – Does support

Note that this attribute is currently identical to the Request PTS Protocol Capabilities attribute. The primary difference is the semantic of how the recipient is expected to respond which is housed in the Attribute Type difference. PTS-IMV supporting this specification MUST support reception and processing of this attribute, while the PTS-IMC MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.8 Diffie-Hellman Nonce Negotiation Attributes

This section describes the optional attributes involved in negotiating the D-H parameters and a nonce exchange that results in a shared, fresh secret for use during this assessment. This exchange provides establishment of a shared secret between the PTS-IMC and PTS-IMV where a man-in-the-middle is unable to learn the value. This shared secret can be integrated into the attestation evidence (e.g. TPM quote operation) to provide proof-of-knowledge where the intermediary can't learn the value. Note that some IF-T transport layer protocol may support establishing a per-session secret and provide this information to the PTS-IMC and PTS-IMV. In these cases, the following attributes may not be required for a deployment since the parties already possess the Secret-Assessment-Value unless the PTS-IMV desires a more recently generated value. If the D-H Nonce Negotiation is performed successfully, the new computed Secret-Assessment-Value replaces any previously negotiated (or inherited from IF-T layer) value.

3.8.1 D-H Nonce Parameters Request

This attribute contains the start of the Diffie-Hellman (D-H) negotiation to establish a shared secret nonce for this session. This secret nonce can be factored into the TPM quote operation to associate the attestation evidence provided by the PTS with this particular assessment. The D-H nonce differs from the standard nonce in that the value used in the TPM quote operation will contain a mix of entropy from both values provided by the PTS-IMV and PTS-IMC and will be unknown to intermediaries so is helpful for avoiding MITM attacks.

In order to avoid potential replay attacks, the shared secret SHOULD only be used for a single assessment exchange and PTS-IMC and PTS-IMV MUST NOT repeat use of a particular nonce value within a short period of time.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Reserved   | Min. Nonce Len|         D-H Group Set         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by compliant implementations.
Min Nonce Len	The initiator can send a minimum acceptable length for the nonce in bytes. This value should be set to 0 if there is no minimum required.
D-H Group Set	Bit field indicating the initiator’s supported D-H groups. See section 3.8.6 for descriptions of the D-H groups and their representation in this field.

PTS-IMC supporting the optional Diffie-Hellman exchange SHOULD support reception and processing of this attribute, while the PTS-IMV supporting the Diffie-Hellman exchange SHOULD support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.8.2 D-H Nonce Parameters Response

This optional attribute contains the D-H Nonce parameters response information from the PTS-IMC responding to the D-H Nonce Parameters Request attribute. The PTS-IMC will select a supported D-H Group from the list sent by the initiator and verify the required minimum nonce length is acceptable based on local policy. If no acceptable D-H Group is offered then the PTS-IMC MUST return an IF-M Error attribute indicating TCG_PTS_DH_GRPS_NOT_SUPPORTED and abort the assessment. Similarly if the proposed minimum nonce length is not acceptable, then the PTS-IMC MUST send an IF-M Error attribute indicating TCG_PTS_BAD_NONCE_LENGTH and abort the assessment.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Reserved                  | Nonce Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Selected D-H Group        |      Hash Algorithm Set       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      D-H Responder Nonce  …                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     D-H Responder Public Value …              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by compliant implementations.

Nonce Length	Length of the nonce field in bytes. This value MUST be greater than 16 and MUST be greater than or equal to the Min Nonce Len specified by the access requestor's D-H Nonce Parameters Request attribute.
Selected D-H Group	Selected D-H Group (single bit) from set offered in the D-H Nonce Parameters Request attribute. See section 3.8.6 for description of the D-H groups and their representation in this field.
Hash Algorithm Set	Bit field indicating the set of supported hash algorithms. See section 3.8.5 for a description of the defined hash algorithms and their representation in this field.
D-H Responder Nonce	High entropy random data used to assure the freshness of the session. Nonces MUST NOT be repeated or be predictable by other parties.
D-H Responder Public Value	Responder's public value for this D-H exchange. The size of this field is determined by the authenticator selected D-H group to use. See section 3.8.6 for the lengths used for each D-H group.

This attribute MUST only be sent in response to a D-H Nonce Parameters Request attribute by a PTS-IMC. PTS-IMV supporting the optional Diffie-Hellman exchange SHOULD support reception and processing of this attribute, while the PTS-IMC supporting the Diffie-Hellman exchange SHOULD support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.8.3 D-H Nonce Finish

This optional attribute contains the D-H Nonce final parameter information from the PTS-IMV. The PTS-IMV will select a supported hash algorithm from the list sent by the PTS-IMC and verify the required minimum nonce length used is greater than the minimum allowable. If no acceptable hash algorithm is offered then the PTS-IMV MUST return an IF-M Error attribute indicating TCG_PTS_HASH_ALG_NOT_SUPPORTED and MAY abort the assessment. Similarly if the proposed minimum nonce length was not complied with, then the PTS-IMV MUST send an IF-M Error attribute indicating TCG_PTS_BAD_NONCE_LENGTH and abort the assessment.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Reserved   | Nonce Length  |    Selected Hash Algorithm    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 D-H Initiator Public Value  …                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   D-H Initiator Nonce  …                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by compliant implementations.

Nonce Length	Length of the nonce field in bytes. This value MUST be greater than 16 and MUST match the length used by the D-H Nonce responder's nonce.
Selected Hash Algorithm	Selected hash algorithm (single bit) from offered set for use later in D-H computation. See section 3.8.5 for a description of the defined hash algorithms.
D-H Initiator Public Value	D-H initiator's public value for this D-H exchange. The size of this field is indicated by the selected D-H group.
D-H Initiator Nonce	High entropy random data used to assure the freshness of the session (nonces MUST NOT be repeated or be predictable.)

This attribute MUST only be sent in response to a D-H Nonce Parameters Response attribute by a PTS-IMV. PTS-IMC supporting the optional Diffie-Hellman exchange SHOULD support reception and processing of this attribute, while the PTS-IMV supporting the Diffie-Hellman exchange SHOULD support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.8.4 Calculation of TPM_Quote ExternalData Value

After the successful completion of the D-H Nonce exchange the PTS-IMC and PTS-IMV compute a shared secret that is freshly bound to the assessment with the nonce values. The computation is the following:

Secret-Assessment-Value = HASH ("1" | D-H Initiator Nonce | D-H Responder Nonce | Computed D-H Shared Secret)

Note: "|" represents concatenation of the values into a single byte sequence. The value "1" represents the ASCII byte value for the character "1" (0x31).

The D-H Shared Secret is computed using the negotiated D-H group and the exchanged D-H public values. The HASH algorithm is the algorithm selected in the D-H Nonce Finish attribute. If the hash algorithm returns a value >20 bytes this value must be truncated to fit into the ExternalData argument of the TPM_Quote ordinal. Therefore the most significant 20 bytes of the HASH algorithm output MUST be stored in the Secret-Assessment-Value and any remaining bytes are discarded.

The Secret-Assessment-Value will be used in all TPM_Quote operations associated with this assessment until either the assessment completes or a new D-H Nonce is requested by the either party.

3.8.5 Diffie-Hellman Hash Algorithm Values

This section defines the values for the Hash Algorithm Set and Hash Algorithm Selected fields for the various hashing algorithms supported in the D-H exchange. The values are as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 2 3 R R R R R R R R R R R R R|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

1 – SHA-1 [FIPS-180-1]

2 – SHA-256 [FIPS-180-1]

3 – SHA-384 [FIPS-180-2]

R – Reserved for future use

All implementations MUST support the use of SHA-1 and SHA-256 as a hashing algorithm for any supported attributes including hashing support. All implementations SHOULD support the use of SHA-

384 for any supported attributes including hashing support. All implementations MUST default to SHA-256 algorithm if a hashing algorithm was not negotiated for this assessment.

Implementations compliant with this specification MUST ignore bits set that they are unable to support. Such implementations MUST NOT set hash algorithm values that they are unable to support.

3.8.6 Diffie-Hellman Group Values

This section defines the bit values for the D-H Group Set and the D-H Group Selected field used in the D-H exchange attributes. The values are as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 2 3 4 5 R R R R R R R R R R R|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

1 – Indicates the use of values based on the group 2 from IKE[RFC4306].

2 – Indicates the use of values based on the group 5 from IKE.

3 – Indicates the use of values based on the group 14 from IKE.

4 – Indicates the use of values based on group 19 from IKE, Elliptic Curve Diffie-Hellman group using the NIST curve with 256-bit prime modulus

5 - Indicates the use of values based on group 20 from IKE, Elliptic Curve Diffie-Hellman group using the NIST curve with 384-bit prime modulus

R – Reserved for future use

All implementations supporting the optional Diffie-Hellman exchange MUST support the use of Diffie-Hellman group 19 for any supported attributes including hashing support. All implementations supporting the optional Diffie-Hellman exchange MUST default to group 19 if no Diffie-Hellman group was previously negotiated for this assessment.

Implementations compliant with this specification MUST ignore bits set that they are unable to support. Such implementations MUST NOT set D-H Group values that they are unable to support.

3.8.6.1 Diffie-Hellman Group 2 Definitions

This section defines the Diffie-Hellman algorithm values that MUST be used when using group 2 (bit 1 above) of the D-H PN. This group is taken from group 2 of IKE.

The public values exchanged when using this group MUST be 128 bytes in length.

The Diffie-Hellman generator (g) MUST be 2.

The prime modulus is the 128 byte value:

```
2^1024 - 2^960 - 1 + 2^64 * { [2^894 pi] + 129093 }
```

that has a hexadecimal value of:

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08
8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B
302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9
A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6
49286651 ECE65381 FFFFFFFF FFFFFFFF
```

3.8.6.2 Diffie-Hellman Group 5 Definitions

This section defines the Diffie-Hellman algorithm values that MUST be used when using group 5 (bit 2 above) of the D-H PN. This group is based on group 5 from IKE MODP Groups[IKE-MODP].

The public values exchanged when using this group MUST be 192 bytes in length.

The Diffie-Hellman generator (g) MUST be 2.

The prime modulus is the 192 byte value:

```
2^1536 - 2^1472 - 1 + 2^64 * { [2^1406 pi] + 741804 }
```

that has a hexadecimal value of:

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA237327 FFFFFFFF FFFFFFFF
```

3.8.6.3 Diffie-Hellman Group 14 Definitions

This section defines the Diffie-Hellman algorithm values that MUST be used when using group 14 (bit 3 above) of the D-H PN. This group is based on group 14 from IKE MODP Groups[IKE-MODP].

The public values exchanged when using this group MUST be 256 bytes in length.

The Diffie-Hellman generator (g) MUST be 2.

The prime modulus is the 256 byte value:

```
 2^2048 - 2^1984 - 1 + 2^64 * { [2^1918 pi] + 124476 }
```

that has a hexadecimal value of:

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
15728E5A 8AACAA68 FFFFFFFF FFFFFFFF
```

3.8.6.4 Diffie-Hellman Group 19 Definitions

This section defines the Diffie-Hellman algorithm values that MUST be used when using group 19 (bit 4 above) of the D-H PN. This group corresponds to group 19 from IKE ECP Groups [RFC4753].

The public values exchanged when using this group MUST be 64 bytes in length.

The Diffie-Hellman public value consists of two components, x and y, corresponding to the coordinates of an elliptic curve point. Each component must be 32 bytes in length. The Diffie-Hellman public value is obtained by concatenating the x and y values. The format of the Diffie-Hellman shared secret value is the same as that of the Diffie-Hellman public value.

The curve is based on the integers modulo the generalized Mersenne prime p given by

```
p = 2^(256)-2^(224)+2^(192)+2^(96)-1
```

The equation for the elliptic curve is:

```
y^2 = x^3 - 3 x + b (mod p)
```

The group prime/irreducible polynomial has a hexadecimal value of:

```
 FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF
 FFFFFFFF FFFFFFFF
```

Parameter b of the group curve has a hexadecimal value of:

```
 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6
 3BCE3C3E 27D2604B
```

The group order has hexadecimal value:

```
FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84
3B9CAC2 FC632551
```

The generator for this group is given by g=(gx,gy) where gx has hexadecimal value:

```
6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0
F4A13945 D898C296
```

and gy has hexadecimal value:

```
4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE
CBB64068 37BF51F5
```

3.8.6.5 Diffie-Hellman Group 20 Definitions

This section defines the Diffie-Hellman algorithm values that MUST be used when using group 20 (bit 5 above) of the D-H PN. This group corresponds to group 20 from IKE ECP Groups [RFC4753].

The public values exchanged when using this group MUST be 96 bytes in length.

The Diffie-Hellman public value consists of two components, x and y, corresponding to the coordinates of an elliptic curve point. Each component must be 48 bytes in length. The Diffie-Hellman public value is obtained by concatenating the x and y values. The format of the Diffie-Hellman shared secret value is the same as that of the Diffie-Hellman public value.

The curve is based on the integers modulo the generalized Mersenne prime p given by

```
p = 2^(384)-2^(128)-2^(96)+2^(32)-1
```

The equation for the elliptic curve is:

```
y^2 = x^3 - 3 x + b (mod p)
```

The group prime/irreducible polynomial has a hexadecimal value of:

```
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFFFF FFFFFFFE FFFFFFFF 00000000 00000000 FFFFFFFF
```

Parameter b of the group curve has a hexadecimal value of:

```
B3312FA7 E23EE7E4 988E056B E3F82D19 181D9C6E FE814112
0314088F 5013875A C656398D 8A2ED19D 2A85C8ED D3EC2AEF
```

The group order has hexadecimal value:

```
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
C7634D81 F4372DDF 581A0DB2 48B0A77A ECEC196A CCC52973
```

The generator for this group is given by g=(gx,gy) where gx has hexadecimal value:

```
AA87CA22 BE8B0537 8EB1C71E F320AD74 6E1D3B62 8BA79B98
59F741E0 82542A38 5502F25D BF55296C 3A545E38 72760AB7
```

and gy has hexadecimal value:

```
3617DE4A 96262C6F 5D9E98BF 9292DC29 F8F41DBD 289A147C
E9DA3113 B5F0B8C0 0A60B1CE 1D7E819D 7A431D7C 90EA0E5F
```

3.9 PTS Measurement Algorithm Selection

This section discusses the attribute exchange used to select a single measurement hashing algorithm for files and other measurements. This algorithm MUST be used for all File Measurement attributes and other attestation evidence measured by the PTS-IMC. Note that TPM performed hashes are outside the scope of this measurement algorithm as they are not PTS performed hashes. The PTS-IMV starts this exchange when it plans to request file measurements or attestation evidence (this might not occur in all situations). The PTS-IMV MUST successfully complete the PTS measurement algorithm exchange before sending any requests for file measurement or attestation evidence attributes in order to select the desired hashing algorithm.

3.9.1 PTS Measurement Algorithm Request

This attribute is sent by the PTS-IMV to establish (or change) the PTS used file or attestation evidence measurement hashing algorithm to be used during this assessment.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Reserved           |      Hash Algorithm Set       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by compliant implementations.
Hash Algorithm Set	Bit field indicating the PTS-IMV proposed set of supported hash algorithms. See section 3.8.5 for a description of the defined hash algorithms and their representation in this field.

PTS-IMC supporting this specification MUST support reception and processing of this attribute, while the PTS-IMV MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.9.2 PTS Measurement Algorithm Selection

This attribute is sent by the PTS-IMC to select the current file or attestation evidence measurement hashing algorithm that it will use during this assessment (unless changed via another PTS Measurement Algorithm Request). If the PTS-IMC does not see an acceptable or supported hash algorithm in the set proposed by the PTS-IMV, the PTS-IMC MUST send an IF-M Error attribute indicating TCG_PTS_HASH_ALG_NOT_SUPPORTED to indicate the exchange has failed and file measurements are not currently available.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Reserved           |   Selected Hash Algorithm     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by compliant implementations.

Selected Hash Algorithm	This field MUST contain a single bit representing the selected hash algorithm from the set proposed by the PTS-IMV. See section 3.8.5 for a description of the defined hash algorithms.

PTS-IMV supporting this specification MUST support reception and processing of this attribute, while the PTS-IMC MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.10 Get TPM Version Information

This attribute is sent by the PTS-IMV to retrieve the TPM Version information. This is particularly useful in conjunction with a TPM based attestation where the PTS-IMV needs to verify a TPM_Quote2 operation.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                    1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Reserved                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by compliant implementations.

PTS-IMC supporting the TPM-backed attestation evidence MUST support reception and processing of this attribute, while the PTS-IMV supporting the TPM-backed attestation evidence MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.11 TPM Version Information

This attribute is sent by the PTS-IMC to indicate the TPM Version information.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                    1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             TPM Version Information (Variable Length)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
TPM Version Information	This field contains the TPM Version information as retrieved by the TPM_GetCapability ordinal for the TPM. The specific contents are defined in the TPM Structure specification section 21.6 (TPM_CAP_VERSION_INFO) [TPM1.2].

PTS-IMV supporting the TPM-backed attestation evidence MUST support reception and processing of this attribute, while the PTS-IMC supporting the TPM-backed attestation evidence MUST support sending this attribute. If the PTS-IMC is unable or unwilling to obtain the TPM_CAP_VERSION_INFO, it MUST respond with an IF-M Error attribute indicating TCG_PTS_TPM_VERS_NOT_SUPPORTED. Other TNC architecture components MUST NOT support sending this attribute.

3.12 Get Attestation Identity Key

This attribute is sent by the PTS-IMV to request a copy of the AIK certificate (or naked public key) to use with the quote operations performed during this attestations.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Reserved                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by compliant implementations.

PTS-IMC supporting the TPM-backed attestation evidence MUST support reception and processing of this attribute, while the PTS-IMV supporting the TPM-backed attestation evidence MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.13 Attestation Identity Key

This attribute is sent by the PTS-IMC to provide the AIK for use with this assessment (quote).

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Flags      |  Attestation Identity Key (variable length)   ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Attestation Identity Key (variable length)           ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Flags</td><td><table>
<tr><th>Bit Encoding</th><th>Description</th></tr>
<tr><td>Bit 0 – Naked Public AIK</td><td>This flag is set to 1 if the sender only has a naked public key (no certificate), so is providing the AIK public key in binary format in the AIK Certificate field.</td></tr>
<tr><td>Bit 1-7 - Reserved</td><td>These bits MUST be set to 0 and MUST be ignored by compliant implementations.</td></tr>
</table></td></tr>
<tr><td>Attestation Identity Key</td><td>This field contains the AIK Certificate (or naked public key if indicated by Flags) of the TPM being used by the PTS for this assessment.</td></tr>
</table>

PTS-IMV supporting the TPM-backed attestation evidence MUST support reception and processing of this attribute, while the PTS-IMC supporting the TPM-backed attestation evidence MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.14 Request Attestation Evidence

This section discusses the attributes enabling requests for attestation evidence from the PTS-IMV. As discussed in section 3.3, the PTS-IMV is able to send a sequence of requests for attestation evidence for functional components on the endpoint. When the PTS-IMV is ready to receive the attestation evidence, it indicates to the PTS-IMC that it is ready for the evidence. This model enables the PTS and PTS-IMC to batch up many requests into a single report (e.g. Integrity Report) if desired. Such a model could also be a performance benefit in some network constrained environments.

The following sub-sections contain the attribute definitions for how the PTS-IMV requests evidence for functional components independent of whether the responses are in XML or TLV-based encoding. Subsequent sub-sections discuss the responses from the PTS-IMC.

3.14.1 Request Functional Component Evidence

This attribute allows the PTS-IMV to request attestation evidence for a set of components. The evidence will not be returned until the PTS-IMV has sent a Generate Attestation Evidence attribute. This allows the PTS-IMV(s) to send multiple component requests separately prior to having the attestation evidence report (e.g. Integrity Report) created. Each new request adds functional components to those included in the next attestation evidence report sent (which empties the pending component attestation list). The format of this request leverages the binary enumeration fixed length naming family (see section 5.3) to efficiently allow for multiple components to be listed in a single TLV message.

.The Attribute Value contains the following information:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |       Sub-component Depth (for Component #1)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Component Functional Name #1                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Component Functional Name #1                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                          . . . . .                            ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |       Sub-component Depth (for Component #N)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Component Functional Name #N                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Component Functional Name #N                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

<table>
<tr><th>Header Field</th><th>Description</th></tr>
<tr><td>Flags</td><td>This field contains flags that impact the component search associated with the particular Component Functional Name.
<table>
<tr><th>Bit Encoding</th><th>Description</th></tr>
<tr><td>Bit 0 – Transitive Trust Chain</td><td>Request inclusion of transitive trust chain below Component Functional Name. If the PTS didn’t measure the requested component, the PTS provides its transitive trust chain and as much additional trust path as known to exist to the component. If the PTS is unable to determine the transitive trust chain below it or the system offers no trusted platform capability, the PTS MUST respond with an Unable to</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>Determine Transitive Trust Chain error.</td></tr>
<tr><td>Bit 1 – Verify Component</td><td>Request PTS verify components and include verification results in attestation evidence. Use of the Verify Component flag SHOULD NOT be used unless the PTS-IMV has verified that the PTS or other local measurement agent is capable of performing local verification. If the PTS-IMC receives a request including the Verify Component bit set but is unable to perform local verification, the PTS-IMC MUST respond with an Unable to Perform Local Validation error.</td></tr>
<tr><td>Bit 2 – Current Evidence</td><td>Requests that the PTS obtain current (fresh) evidence about the requested component and not use information it might have cached from previous attestations. Use of the Current Evidence flag SHOULD NOT be used unless the PTS-IMV has verified that the PTS or other local measurement agent is capable of performing measuring already executing components. If the PTS-IMC receives a request including the Current Evidence bit set but is unable to measure components running and one of the requested components is executing, the PTS-IMC MUST respond with a Unable to Collect Current Evidence error.</td></tr>
<tr><td>Bit 3 – PCR Information</td><td>This indicates whether the PTS-IMV requests the PTS return a TPM Quote structure plus accompanying PCR usage information and hash values. If the PTS is unable to determine the PCRs associated with the requested component or the trusted platform is not being used, the PTS MUST respond with an Unable to Determine PCR Information error.</td></tr>
<tr><td>Bit 4-7 - Reserved</td><td>These bits MUST be set to 0 and MUST be ignored by compliant implementations.</td></tr>
</table>
</td></tr>
<tr><td>Sub-Component Depth</td><td>This field contains the level of nested sub-components request below the Component Functional Name to include in the attestation evidence. For example, this could be used to identify a complex component and have the PTS report on measurement for all (or a limited number of) of its sub-components. While this 24 bit value is less than the 32 bits supported in IF-PTS, these bits should be considered the least significant bits of IF-PTS's 32 bit argument. If the PTS-IMV requested depth is higher than the number of sub-components for the requested component(s), the PTS-IMC MUST return those that are present on the system. In effect, this makes the Sub-Component Depth an upper bound on the number of sub-component layers.</td></tr>
<tr><td>Component Functional Name</td><td>Contains the enumerated name of the functional component requested to be in the attestation evidence. This field is four octets in length and can be repeated many times in this attribute to request multiple attributes. See section 5 for a description of the functional naming.</td></tr>
</table>

PTS-IMC supporting this specification MUST support reception and processing of this attribute, while the PTS-IMV MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.14.2 Generate Attestation Evidence

This attribute allows the PTS-IMV to request attestation evidence to be reported for all the functional components requested since the prior Generate Attestation Evidence request. This model allows the PTS and PTS-IMC to gradually build up a list of desired components and send the resulting Integrity Report or set of TLV-based attestation evidence (e.g. Simple Component Evidence) back at one time. This could be beneficial when operating over certain types of communications channels (e.g. low bandwidth or half duplex). If a PTS-IMV wishes to have attestation evidence returned for each functional component request (note that a single request could return many attestation for many components), the PTS-IMV should send this attribute immediately after each Request Functional Component Evidence attribute. Note that if the attestation evidence includes a TPM quote operation, then the most recently computed (or passed up the stack from IF-T layer) value for Secret-Assessment-Value must be used as the external data for the quote. This use of the shared secret provides man-in-the-middle detection for the assessment.

The Attribute Value contains the following information:

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Reserved                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by compliant implementations. In the future, this field might be used to provide information about the desired attestation evidence encoding.

PTS-IMV supporting this specification MUST support reception and processing of this attribute, while the PTS-IMC MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.15 TLV-based Attestation Evidence

This section discusses the TLV-based attestation evidence that can be returned by a PTS-IMC wishing to communicate with TLV encodings. These attributes are an alternative to the XML-based attributes discussed in the next section.

3.15.1 Simple Component Evidence

This attribute allows the PTS-IMC to return a TLV-encoded, simple component measurement instead of the more complex XML-based report (e.g. Integrity Report). This attribute will contain a subset of the Integrity Report so may not be applicable to some types of PTS-IMV queries. PTS-IMV must be written to request the appropriate type of response (XML full report vs. simplified measurements) based upon the information required to compare against policy. This PTS-IMV request is accomplished by sending a Request PTS Protocol Capabilities attribute.

When the PTS-IMV requests one or more Functional Components' measurements in TLV form, this will result in a sequence of this attribute being returned (one per component) plus a Simple Process Measurement Summary attribute indicating the end of the sequence and including information like the signature over the entire sequence and PCR related information if the measurements were synced to the TPM.

The ordering of the sequence of attributes is critical as it MUST start with a "top level" component and then a depth-wise traversal of the sub-component hierarchy listing each piece of the component. The sub-component depth field is used to determine how deep in the tree the component is relative to the top. For example, if a component has 2 run-time libraries (sub-components) called libA.dll and libB.dll and the libA.dll run-time library also depended on another run-time library called libA-dep.dll, then this attribute would include evidence about: the main component, the first library (libA.dll), the run-time dependency library (libA-dep.dll) and then finish with a description of the second library (libB.dll). When a second (or more) component is included, the Top Level Component flag indicates the start of a new component tree.

The Attribute Value contains the following information:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |                Sub-Component Depth            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Specific Functional Component                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Specific Functional Component                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Measure. Type |                Extended into PCR              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Hash Algorithm        | PCR Transform |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Measurement Date/Time                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Measurement Date/Time                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Measurement Date/Time                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Measurement Date/Time                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Measurement Date/Time                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Optional Policy URI Length  |  Opt. Verification Policy URI ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                  Optional Verification Policy URI             ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Optional PCR Length      |   Optional PCR Before Value   ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~             Optional PCR Before Value (Variable Length)       ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~             Optional PCR After Value (Variable Length)        ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~             Component Measurement (Variable Length)           ~
```

<table>
<tr><th>Header Field</th><th>Description</th></tr>
<tr><td>Flags</td><td>This field contains flags that impact the component evidence associated with the particular Component Functional Name.
<table>
<tr><th>Bit</th><th>Description</th></tr>
</table>
</td></tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><th>Encoding</th><th></th></tr>
<tr><td>Bit 0 – PCR Information Included</td><td>Indicates whether the optional PCR Information fields are included in this attestation evidence attribute. If a component wasn’t measured into a PCR or the PTS isn’t willing to provide this information might determine whether the attributes includes these fields. When this value is zero, it means the Optional PCR oriented fields are not present in this attribute.</td></tr>
<tr><td>Row 1-2 – Validation Results</td><td>Indicates if PTS performed validation of component and result
<table>
<tr><th>Bit Encoding</th><th>Description</th></tr>
<tr><td>00</td><td>No validation was attempted. No Optional Verification URI included.</td></tr>
<tr><td>01</td><td>Attempted Validation, unable to verify (error occurred). No Optional Verification URI included.</td></tr>
<tr><td>10</td><td>Attempted Validation and component FAILED local verification against policy referenced in Optional Verification Policy URI</td></tr>
<tr><td>11</td><td>Attempted Validation and component PASSED local verification against policy referenced in Optional Verification Policy URI</td></tr>
</table>
</td></tr>
<tr><td>Bit 3-7 - Reserved</td><td>These bits MUST be set to 0 and MUST be ignored by compliant implementations.</td></tr>
</table>
</td>
</tr>
<tr>
<td>Sub-Component Depth</td>
<td>This field indicates the depth down the subcomponent hierarchy that the Specific Functional Component describes. For instance, if a requested component uses 3 libraries and these libraries dynamically load 2 other objects, then we have a 3 level hierarchy of sub-components. Therefore the libraries would exist at a depth of 1 and the dynamically loaded objects would be at depth 2.

In order to simplify evaluation, the PTS and PTS-IMC MUST send back the set of simple measurements in a depth first manner as this allows sub-components at depth 2 to immediately follow the subcomponent at depth 1 that they are used by. So in the example discussed in the prior paragraph, this would mean a depth 0 component attribute would be sent first, followed by a depth 1 library, followed by its 2 dynamic objects, then another depth 1 library and so forth.</td>
</tr>
<tr>
<td>Specific Functional Component</td>
<td>Identifies a specific functional component that was measured on the system. While the name uses the component functional name syntax discussed in section 5, this field MUST NOT use the wildcard or unknown features of the syntax.</td>
</tr>
</table>

<table>
<tr><td></td><td>Note that if a PTS-IMC and PTS are responding to a request for a component with multiple sub-components or a wildcarded component, this would result in multiple instances of this attribute being returned to the PTS-IMV. Therefore it’s important that the PTS-IMV be able to re-associate each subcomponent with the requested component. The Component Functional Name and the Sub-Component Depth field in this attribute help with this process.</td></tr>
<tr><td>Measurement Type</td><td>This field contains an indication of the type of measurement included in this attribute. Its envisioned that over time additional types of measurements will be added besides a single hash value for the entire (sub)component. For example, a future measurement type might include a set of hashes for different aspects of a running process (stack, data, code, …).
<table>
<tr><th>Bit Encoding</th><th>Description</th></tr>
<tr><td>Bit 0 – Simple Hash</td><td>This indicates that the Component Measurement field contains a simple hash of the sub-component. Since a single hash is present the length of the Component Measurement can be determined by the Hash Algorithm used.</td></tr>
<tr><td>Bit 1-7 - Reserved</td><td>This field MUST be set to 0 and MUST be ignored by compliant implementations.</td></tr>
</table></td></tr>
<tr><td>Extended into PCR</td><td>This field indicates the number of the PCR where the functional component was extended into for the TPM. This PCR value typically is <32 but in order to allow for future TPMs which might include a much larger number of PCRs or support for virtual PCR (e.g. somehow emulated by software) where the number can be much greater, this value allows up to 2^24 PCR values to be identified.</td></tr>
<tr><td>Hash Algorithm</td><td>Hash algorithm (single bit) used to take the measurement of the (sub)component described by this attribute. See section 3.8.5 for a description of the defined hash algorithms. This hash algorithm MUST be the one selected during a PTS Measurement Algorithm negotiation.</td></tr>
<tr><td>PCR Transform</td><td>This field describes how the PTS’s hash result is stored into the PCR. This field MUST be set to 0 and ignored by recipients if the Flags field’s PCR Information Included bit is set to 0 indicating no PCR information is included in this attribute.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0 – No Transform</td><td>This field MUST be set to 0 if the attribute does not include PCR information or the transform of the hash to fit into the PCR is unknown.</td></tr>
<tr><td>1 – Hash value matched PCR size</td><td>PTS’s hash result was the same size as the PCR size, so the result was directly applied without alteration. This might occur if the PTS used SHA-1 on a TPM 1.2 system.</td></tr>
<tr><td>2 – Hash value shorter than PCR size</td><td>PTS’s hash result was shorter than the size of the PCR, so the high order bits are padded with zero to enable the hash to fit into the PCR. Note this isn’t expected to be common but if a shorter hash results (e.g. 128 bits) was determined and needs to be extended into a long (e.g. 160 bit) PCR this</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td></td><td>padding might occur.</td></tr>
<tr><td>3- Hash value longer than PCR size</td><td>PTS’s hash result was longer than the size of the PCR size, so the result was truncated in order to fit into the PCR extend. This might occur if the PTS was using SHA-256 on a system with a 20 octet PCR size.</td></tr>
<tr><td>Values 4-7 - Reserved</td><td>These values MUST be set to 0 and MUST be ignored by compliant implementations.</td></tr>
</table></td></tr>
<tr><td>Reserved</td><td>This field is reserved for future use and MUST be set to zero. Compliant implementations MUST ignore the contents of this field.</td></tr>
<tr><td>Measurement Date/Time</td><td>This field contains the date and time that the measurement was taken (if known). The Measurement Date/Time field’s date and time MUST be represented as an RFC 3339[RFC3339] 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 with the above restrictions. Leap seconds are permitted and IMVs MUST support them.
The Measurement Date/Time string MUST NOT be NUL terminated or padded in any way. If the measurement date and/or time is not known, not applicable, or cannot be represented in this format, this field MUST contain ”0000-00-00T00:00:00Z” allowing this attribute to be fixed length. Note that this reserved value is not RFC 3339 compliant (zero month).
This encoding produces an easy to read, parse and interpret string in YYYY-MM-DDTHH:MM:SSZ format that can precisely define a particular second in UTC time. For example, 9:05:00AM EST on January 19, 1995 can be represented as “1995-01-19T14:05:00Z”. The length of this field is always 20 octets.</td></tr>
<tr><td>Optional Policy URI Length</td><td>This field indicates the length (in bytes) of the Optional Verification Policy URI if present in the message. This field and the Optional Verification Policy URI are absent if the Flags indicate no verification was performed.</td></tr>
<tr><td>Optional Verification Policy URI</td><td>When present, this field includes a URI to a policy used by the PTS’s local policy verification agent. Some deployments may not support such network retrieval of the policy so could treat this policy as a policy name that the challenger can correlate with its own policy database. As such, the syntax/semantic of this field are left to deployers’ discretion. This URI MUST be converted to a UTF-8 sequence of octets and then percent encoded where necessary [RFC3986].</td></tr>
<tr><td>Optional PCR Length</td><td>This field indicates the length (in bits) of the PCR value for the Optional PCR Before Value and later the Optional PCR After Value fields. This length allows for PCR values other than 20 bytes (fixed in TPM 1.2 and earlier). This field and the Optional PCR Before and Finish Values are absent if the Flags indicate no PCR inclusion in the message.</td></tr>
<tr><td>Optional PCR Before Value</td><td>This variable length value includes the PCR’s content before the TPM extend operation was performed. Having this value could help in reconstructing the ordering of sub(component) extends or to verify with the subsequent quote</td></tr>
</table>

	information. The Optional PCR Before and Finish Value fields repeat until every PCR that was extended during the measurement of this component is listed.
Optional PCR After Value	This variable length value includes the PCR's content after the TPM extend operation was performed. Having this value could help in reconstructing the ordering of sub(component) extends or to verify with the subsequent quote information.
Component Measurement	This field contains the measurement (hashing result) of the particular (single) described (sub)component. This follows the last occurrence of the Optional PCR After Value which might be absent (if not reported) or might have been repeated several times.

PTS-IMV supporting this specification MUST support reception and processing of this attribute, while the PTS-IMC MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.15.2 Simple Evidence Final

This attribute is sent by the PTS-IMC at the conclusion of attestation evidence reporting when the Simple Component Evidence attribute is used. This attribute is not intended for use with any other form of attestation evidence reporting (e.g. not needed with an Integrity Report attribute). The purpose of this attribute is to provide any summary information involving the set of individual Simple Component Evidence attributes reports (which could be many). For example, the PTS-IMC can provide a signature over the entire attestation evidence. This attribute also signals the end of the TLV-based attestation evidence report.

The Attribute Value contains the following information:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |    Reserved   |  Optional Composite Hash Alg  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Optional TPM PCR Composite Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~           Optional TPM PCR Composite (Variable Length)        ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Optional TPM QUOTE Signature Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~         Optional TPM QUOTE Signature (Variable Length)        ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~           Optional Evidence Signature (Variable Length)       ~
```

<table>
<tr><th>Header Field</th><th>Description</th></tr>
<tr><td>Flags</td><td>This field contains flags that impact the component evidence associated with the particular Component Functional Name.
<table>
<tr><th>Bit Encoding</th><th>Description</th></tr>
<tr><td>Bit 0-1 – TPM Info</td><td>Indicates whether the Optional TPM PCR Composite and Optional TPM Quote Signature fields are included in this</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>Included</td><td>attribute.
<table>
<tr><th>Bit Encoding</th><th>Description</th></tr>
<tr><td>00</td><td>No Optional TPM PCR Composite nor Optional TPM Quote Signature fields included</td></tr>
<tr><td>01</td><td>Optional TPM PCR Composite and Optional TPM Quote Signature fields included using TPM_QUOTE_INFO to compute composite hash</td></tr>
<tr><td>10</td><td>Optional TPM PCR Composite and Optional TPM Quote Signature fields included using TPM_QUOTE_INFO2 to compute composite hash. A TPM_CAP_VERSION_INFO was NOT appended.</td></tr>
<tr><td>11</td><td>Optional TPM PCR Composite and Optional TPM Quote Signature fields included using TPM_QUOTE_INFO2 to compute composite hash. A TPM_CAP_VERSION_INFO structure was appended.</td></tr>
</table>
</td></tr>
<tr><td>Bit 2 – Evidence Signature</td><td>Indicates whether the optional Evidence Signature is included in this attribute. This signature might not be present if the PTS (or PTS-IMC) is unwilling or unable to sign all of the provided evidence.</td></tr>
<tr><td>Bit 3-7 - Reserved</td><td>This field MUST be set to 0 and MUST be ignored by compliant implementations.</td></tr>
</table>
</td></tr>
<tr><td>Reserved</td><td>This field is reserved for future use and MUST be set to zero. Compliant implementations MUST ignore the contents of this field.</td></tr>
<tr><td>Optional Composite Hash Alg</td><td>When present, this field contains an indication of the composite hash algorithm used during the quote operation. The composite hash algorithm is used to reduce the size of the TPM_PCR_COMPOSITE list value down to TPM_COMPOSITE_HASH a smaller form used in the signature operation. See section 3.8.5 for a description of the defined hash algorithms.</td></tr>
<tr><td>Optional TPM PCR Composite Length</td><td>When present, this field indicates the length of the TPM PCR Composite field. This length is necessary since this attribute might contain multiple variable length fields.</td></tr>
<tr><td>Optional TPM PCR Composite</td><td>When present, this field contains the TPM's current values for all PCRs involved in the attestation evidence reported in the Simple Component Attestation attributes for this report. This value is a direct copy of the TPM_PCR_COMPOSITE value returned by the TPM as part of the TPM Quote operation and is included in the TPM Quote signature computation.</td></tr>
<tr><td>Optional TPM</td><td>When present, this field indicates the length of the TPM Quote Signature field.</td></tr>
</table>

Quote Signature Length	This length is necessary since this attribute has multiple variable length fields.
Optional TPM Quote Signature	When present, this field contains the signature computed during the TPM Quote operation (sig argument) over the PCR composite information. The recipient needs to verify this signature across the PCR composite information prior to trusting the values contained in the TPM PCR Composite information field for policy decisions. NOTE: the TPM Quote signature also includes the external data value established during the D-H Nonce exchange defined above. The value MUST already be known by both parties and is not included in this message as part of the anti-replay protection.
Optional Evidence Signature	When present, this field contains a signature across all of the base64 encoded Simple Component Evidence attributes returned during this attestation reporting. The length of this field can be computed based upon the overall attribute's length minus the lengths of any other optional fields included (see above TPM Quote and PCR Composite fields).

Generally the way this attribute is processed when the TPM PCR and Quote information is provided is:

1. PTS-IMV already knows the set of PCRs used during this sequence of Simple Component Evidence attributes since the PCR to Extend is included. Therefore it can combine together all the PCR to Extend fields from each of the received Simple Component Evidence attributes into All-Used-PCRs. Similarly, it can keep a list of the current PCR values based on the PCR Final Value fields received. These can be transformed into an appropriate (depending on quote version) PCR selection map.
2. PTS checks the Flag field of this attribute to be aware whether a TPM_Quote or TPM_Quote2 was used by the PTS. The details of the structure signed during the quote operation differ so the PTS-IMV will need to construct the appropriate version to correctly verify the signature.
3. PTS-IMV constructs either a TPM_QUOTE_INFO or TPM_QUOTE_INFO2 using the All-Used-PCRs as the targetPCRs, the correct tag (QUOT or QUT) for version and the final PCR After Values included in the Simple Component Evidence attributes which last used each PCR composited using the Composite Hash Algorithm into a value analogous to TPM_COMPOSITE_HASH. Finally the locality is included in using TPM_QUOTE_INFO2. This information should match the PTC-IMC sent TPM_PCR_INFO_SHORT or TPM_PCR_COMPOSITE to verify that the correct information was used during the quote.
4. If using TPM_Quote2 and the Flags indicate that the TPM_CAP_VERSION_INFO is required, the TPM version information for the attested party's TPM can be retrieved using the Get TPM Version Information attribute and added to the TPM_QUOTE_INFO2 structure.
5. Verify the TPM Quote Signature using the AIK public key and compare the result with the hash of the structure built in the previous step. If they match, then the TPM on the endpoint constructed the same structure so the data can be considered authentic.
6. Verify that the list of PCR before/after values in the Simple Component Measurement attributes result in a hashed value equal to the present one included in the validated Optional TPM PCR Composite value received in the Simple Evidence Final attribute.
7. Compare the individual hash values to policy possibly factoring in the associated Integrity Log contents for the PCR.

Note that in the completely degenerate case where none of the optional fields are present, the value of this attribute is only 2 octets long. This attribute MUST only be sent by the PTS-IMC and MUST be ignored if received from another TNC architecture component.

PTS-IMV supporting this specification MUST support reception and processing of this attribute, while the PTS-IMC MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.16 XML-based Attestation Evidence

This section defines the attributes used by the PTS-IMC (and PTS) to respond with attestation evidence using an XML-based annotation. The XML reports and schemas are defined in other specifications from the IWG, so these attributes generally provide generic fields that are used to carry the reports.

3.16.1 Verification Result

This attribute contains the Verification Report describing the local verification performed on the system pertaining to the requested components.

The Attribute Value contains the following information:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                 Verification Result (Variable Length)         ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Verification Result	This field includes a full XML document as described in [VERIFY-RESULT] describing the verification performed by the system of the components requested by the PTS-IMV.

PTS-IMV supporting XML-based attestation evidence MUST support reception and processing of this attribute, while the PTS-IMC supporting XML-based attestation evidence MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute. Integrity Report

This attribute contains the Integrity Report for the components requested.

The Attribute Value contains the following information:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                 Integrity Report (Variable Length)            ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Integrity Report	This field includes a full XML document as described in [INT-REPORT] describing the measurements of the components requested by the PTS-IMV.

PTS-IMV supporting XML-based attestation evidence MUST support reception and processing of this attribute, while the PTS-IMC supporting XML-based attestation evidence MUST support sending this attribute. Other TNC architecture components MUST NOT support sending this attribute.

3.17 File Based Metadata

This section defines how the PTS-IMV may request information about the properties of an individual file/directory or every file within a particular directory. Using the ability to fetch information about every file in a directory, the PTS-IMV can iteratively request directory contents to walk an entire filesystem. However this protocol does not intend to provide a network filesystem sort of capability, so offers a subset of all the available file attribute metadata for simplicity.

3.17.1 Request File Metadata

This attribute contains a generic request for information about the attributes of a particular file or directory. The goal for this attribute is to be as generic in syntax as possible to allow for file pathnames for other operating systems besides the initial set specified in this document (Windows and Unix-like).

The attributes (or metadata) refer to the information describing the contents of a file or directory. Frequently this information is visible when doing a directory listing (e.g. file size).

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |   Delimiter   |          Reserved             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~        Fully Qualified File Pathname (Variable Length)        ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Flags</td><td>This field contains flags that modify the target filename requested.
<table>
<tr><th>Bit Encoding</th><th>Description</th></tr>
<tr><td>Bit 0 – Directory Contents</td><td>This value indicates when the specified Fully Qualified File Pathname field contains the pathname to a directory, whether the metadata about each file in the directory is desired to be returned or metadata about the directory node itself. If this bit is set to 0, this indicates that metadata about the directory is to be returned. If this bit is set to 1, this indicates that the recipient should return metadata about every file in the directory (not the directory itself). The Directory Contents bit MUST NOT be set to one when in an attribute referencing a non-directory.</td></tr>
<tr><td>Bit 1-7 - Reserved</td><td>This field MUST be set to 0 and MUST be ignored by compliant implementations.</td></tr>
</table>
</td></tr>
<tr><td>Delimiter</td><td>This field contains the UTF-8 encoding of the character used to delimit the filename given in the Fully Qualified File Pathname field. For example, the ‘\’ character is typically used on Windows systems to separate the directory names in a path while Unix oriented systems use the ‘/’ character. The Delimiter field MUST contain the UTF-8 character used to separate</td></tr>
</table>

	directory names in the Fully Qualified File Pathname field or an IF-M Error attribute indicating TCG_PTS_INVALID_DELIMITER is sent by the recipient. This field is included in the request in case the sender does not yet know what type of operating system is present on the endpoint. This is not envisioned to be common. PTS-IMC MUST convert the included Delimiter to the one used by the local operating system in order to respond to the request. PTS-IMV SHOULD use an appropriate Delimiter for the endpoint's operating system once discovered during an assessment.
Reserved	This field is reserved for future use and MUST be set to zero. Compliant implementations MUST ignore the contents of this field.
Fully Qualified File Pathname	This variable length field contains the UTF-8 encoding of the fully qualified path to a particular file or directory on the system being attested (requestor). For detailed information about the required syntax for this field see section 3.17.1.1. For example "C:\tcg\iwg\example" might be used while "iwg\example" is not a full path so it not allowed. The Fully Qualified File Pathname field MUST specify an unambiguous, fully qualified file system location. The PTS-IMC (or the PTS) SHOULD be flexible in the Delimiters it supports even when the delimiter does not match the one used by the local operating system. For example a Unix-based PTS might support the Windows oriented file pathnames. If the PTS receives an unknown file pathname, it MUST respond with a TCG_PTS_FILE_NOT_FOUND error code.

PTS-IMC supporting this specification SHOULD support reception and processing of this attribute, while the PTS-IMV SHOULD support sending this attribute. Other TNC architecture components besides IMVs MUST NOT support sending this attribute. PTS-IMVs sending this attribute need to be aware that they could receive either Windows-Style File Metadata or Unix-Style File Metadata attributes in response attributes.

3.17.1.1 Fully Qualified Pathname Syntax

This section defines the required syntax for the Fully Qualified File Pathname field in the Request File Metadata attribute. The Fully Qualified File Pathname is a UTF-8 string that MUST use a pathname syntax of:

Pathname ::= Directory-Pathname | File-Pathname ;

Directory-Pathname ::= [Disk-Designator] , Delimiter , { Sub-Directory , Delimiter }

File-Pathname ::= [Disk Designator] , Delimiter , { Sub-Directory , Delimiter } , Filename

Where:

Disk-Designator – optional physical or logical storage volume id (e.g. "c:" in Windows). This value is a UTF-8 string up to but not including the Delimiter value.

Delimiter – separator between components of the pathname (e.g. between sub-directories).

Sub-Directory – name of directory in the pathname. This value is a UTF-8 string up to but not including the Delimiter value.

Filename – final component of pathname to a particular file and is not used for identifying a directory. The filename is a UTF-8 string up to the end of the pathname.

Note:

Most operating systems limit the set of allowable characters that can be used in file and directory names so PTS-IMV should use care when using such characters since it will just result in TCG_PTS_FILE_NOT_FOUND errors. However, this protocol does not impose character restrictions beyond those described above since the restrictions vary by target operating system.

The Disk Designator, Sub-Directory and Filename components of the pathname MUST NOT include the Delimiter value defined in this attribute. As shown in the syntax above, the filename form of the pathname MUST NOT include a Delimiter as the final byte where the directory form of the pathname MUST use a Delimiter as the final UTF-8 character.

Note that some of the provided example directory name paths include (using “\” as a delimiter): c:\, \, \foo\, a:\example\directory\, while some example file name paths include (using “/” as a delimiter): c:/file, /file, /example/file.

3.17.2 Windows-Style File Metadata

This attribute contains the information about the properties of a particular file or directory requested by the PTS-IMV. The properties (or metadata) refer to the information frequently shown when doing a directory listing (e.g. file size). Note that this attribute is capable of including metadata for multiple files in case the request was for the contents of a directory.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Number of Files Included                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Number of Files Included                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     File Metadata Length      |     Type      |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         File Size                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         File Size                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        File Create Time                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        File Create Time                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Last Modify Time                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Last Modify Time                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Last Access Time                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Last Access Time                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         File Owner ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         File Owner ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         File Group ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         File Group ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Filename (Variable Length)  . . .           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    File #2 Metadata Length    |    Type #2    |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         File Size #2                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         File Size #2                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        . . . . . . . .                        |
```

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Number of Files Included</td><td>This value indicates the number of files being reported in this attribute. Normally this value is set to one, but when the request is for an entire directory, the value may be between zero and 2^64-1. This is the only field in this attribute that is not repeated when multiple files are included.</td></tr>
<tr><td>File Metadata Length</td><td>This field indicates the number of bytes including this field taken up by the metadata describing the next file including both the fixed length metadata and the variable length filename. When only one file is described in this attribute, the length can be computed (since only one field is variable length). However when multiple files are described in this attribute the length field is used to locate the start of the next file metadata (since the filename is variable length).</td></tr>
<tr><td>Type</td><td>This field indicates the type of file being described. This value might be computed using the GetFileAttributeEx call on Windows. However not all Windows types of file modes are explicitly defined in this specification (e.g. FILE_ATTRIBUTE_OFFLINE). Vendors wishing to include non-standard file types are encouraged to create vendor specific file metadata attributes based upon this attribute to report those values and use this attribute when responding with standard file types.

The following enumeration defines those values standardized by this specification:<b>.</b> The PTS-IMC MUST send one of these values and the PTS-IMV MUST treat any reserved values as equivalent to 0 (Other).

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0 – Other</td><td>This file type indicates a value that is either unknown or different from all of the standardized file types.</td></tr>
<tr><td>1 - Directory</td><td>This value indicates that the Filename is a directory. This corresponds to file attribute FILE_ATTRIBUTE_DIRECTORY's 2nd byte on Windows systems.</td></tr>
<tr><td>1 - Reserved</td><td>This value MUST NOT be used and MUST be ignored by compliant implementations.</td></tr>
</table>
</td></tr>
</table>

	2 - 3 - Reserved	This value MUST NOT be used and MUST be ignored by compliant implementations.
	4 – Device	This value indicates that the Filename is a device special file. This corresponds to file attribute FILE_ATTRIBUTE_DEVICE's 2nd byte on most Windows systems.
	5 - 7 Reserved	This value MUST NOT be used and MUST be ignored by compliant implementations.
	8 - Regular	This value indicates that the Filename is a standard on disk file. This corresponds to the file attribute FILE_ATTRIBUTE_NORMAL's 2nd byte on Windows systems. Windows file attributes that indicate information about how the file is stored or encoded should use this value. For example, Windows file attributes FILE_ATTRIBUTE_COMPRESSED, FILE ATTRIBUTE_ARCHIVE, FILE_ATTRIBUTE_ENCRYPTED or FILE_ATTRIBUTE_TEMPORARY should use this value.
	9 - Reserved	This value MUST NOT be used and MUST be ignored by compliant implementations.
	10 – Reparse Point	This value indicates that the Filename is a reparse point (symbolic link) file. This value corresponds to file attribute FILE_ATTRIBUTE_REPARSE_POINT on supporting Windows systems.
	11 - 255 Reserved	These values MUST NOT be used and MUST be ignored by compliant implementations.
Reserved	This field is reserved for future use and MUST be set to zero. Compliant implementations MUST ignore the contents of this field.	
File Size	This field includes an eightoctet value representing the number of bytes in the file. This value can be obtained on some Windows operating systems using the GetFileAttributeEx call and using the nFileSizeHigh and nFileSizeLow values. The nFileSizeHigh value should be placed in the first four octet field while the nFileSizeLow is placed in the second four octet portion of the File Size field. For reporting of metadata about a directory this field MUST be set to zero.	
File Create Time	This field includes a four octet value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC). This value can be obtained on some Windows operating systems using the GetFileAttributeEx call and using the ftCreationTime value. Some Windows operating systems do not support this	

	time value so will return a zero value.
Last Modify Time	This field includes a four octet value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC). This value can be obtained on some Windows operating systems using the GetFileAttributeEx call and using the ftLastWriteTime value. Some Windows operating systems do not support this time value so will return a zero value.
Last Access Time	This field includes a four octet value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC). This value can be obtained on some Windows operating systems using the GetFileAttributeEx call and using the ftLastAccessTime value. Some Windows operating systems do not support this time value so will return a zero value.
File Owner ID	This field indicates the operating system's identity of the owner of the file. For Windows based system, this value MUST be set to the sidOwner of the file if supported by the system, otherwise it MUST be set to 2^64-1. This information can be obtained using the GetFileSecurity to get the file's security descriptor and then retrieving the file owner using the GetSecurityDescriptorOwner call.
Group Owner ID	This field indicates the operating system's identity of the owner of the file. For Windows based system, this value MUST be set to the sidGroup of the file if supported by the system, otherwise it MUST be set to 2^64-1. This information can be obtained using the GetFileSecurity to get the file's security descriptor and then retrieving the file owner using the GetSecurityDescriptorGroup call.
Filename	This field includes a UTF-8 encoded string containing the file name associated with the requested file. This name is only the last component of a fully qualified file path name. For example, "c:\tcg\iwg\example.txt" path name would have a file name of "example.txt".

PTS-IMV supporting this specification on a Windows platform SHOULD support reception and processing of this attribute, while the PTS-IMC running on a Windows platform SHOULD support sending this attribute. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

3.17.3 Unix-Style File Metadata

This attribute contains the typical 32 or 64 bit Unix filesystem stored properties about a particular file or directory requested by the PTS-IMV. The properties (or metadata) refer to the information normally visible when performing a directory listing (e.g. file size). Note that this attribute is capable of including metadata for multiple files in case the request was about the contents of a directory.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                 Number of Files Included                      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                 Number of Files Included                      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     File Metadata Length      |     Type      |    Reserved   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                         File Size                             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                         File Size                             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        File Create Time                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        File Create Time                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        Last Modify Time                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        Last Modify Time                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        Last Access Time                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        Last Access Time                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        File Owner ID                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        File Owner ID                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                         File Group ID                         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                         File Group ID                         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                    Filename (Variable Length)  . . .          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Number of Files Included	This value indicates the number of files being reported in this attribute. Normally this value is set to one, but when the request is for the contents of an entire directory, the value may be between zero and 2^64-1. This is the only field in this attribute which is not repeated when multiple files are included.
File Metadata Length	This field indicates the number of bytes including this field taken up by the metadata describing the next file including both the fixed length metadata and the variable length filename. When only one file is described in this attribute, the length can be computed (since only one field is variable length). However when multiple files are described in this attribute the length field is used to locate the start of the next file metadata (since the filename is variable length).
Type	This field indicates the type of file being described. This value might be computed by the PTS-IMC using the S_IFMT & st_mode field from a stat() call. However not all Unix implementation file modes are explicitly defined in this specification (e.g. Doors on Solaris) as this list parallels the Unix

standard set from the Open Group. Vendors wishing to include non-standard file types are encouraged to create vendor specific file metadata attributes based upon this attribute to report those values.

The following enumeration defines those values standardized by this specification. The PTS-IMC MUST send one of these values and the PTS-IMV MUST treat any reserved values as equivalent to 0 (Other).

Value	Description
0 – Other	This file type indicates a value that is either unknown or different from all of the standardized file types.
1- FIFO	This value indicates that the Filename field refers to a FIFO or local named pipe communication special file. The meaning of FIFO/pipe special file is operating system dependent and might not be supported by all operating systems. This corresponds to file of type S_IFIFO on Unix.
2 - Character Special	This value indicates that the Filename field refers to a character special file. The meaning of character special is operating system dependent and might not be supported by all operating systems. This corresponds to file of type S_IFCHR on Unix.
3 - Reserved	This value MUST NOT be used and MUST be ignored by compliant implementations.
4 - Directory	This value indicates that the Filename is a directory. This corresponds to file of type S_IFDIR on Unix.
5 - Reserved	This value MUST NOT be used and MUST be ignored by compliant implementations.
6 - Block Special	This value indicates that the Filename field refers to a block special file. The meaning of block special is operating system dependent and might not be supported by all operating systems. This corresponds to file of type S_IFBLK on Unix.
7 - Reserved	This value MUST NOT be used and MUST be ignored by compliant implementations.
8 - Regular	This value indicates that the Filename is a standard on disk file. This corresponds to file of type S_IFREG on Unix.
9 - Reserved	This value MUST NOT be used and MUST be ignored by compliant implementations.

<table>
<tr><td></td><td>
<table>
<tr><td></td><td></td></tr>
<tr><td>10 - Symbolic Link</td><td>This value indicates that the Filename is a symbolic link to another Filename. This corresponds to file of type S_IFLNK on Unix.</td></tr>
<tr><td>11 - Reserved</td><td>This value MUST NOT be used and MUST be ignored by compliant implementations.</td></tr>
<tr><td>12 - Socket</td><td>This value indicates that the Filename field refers to a socket communication special file. The meaning of socket special file is operating system dependent and might not be supported by all operating systems. This corresponds to file of type S_IFSOCK on Unix.</td></tr>
<tr><td>13 - 255 Reserved</td><td>These values MUST NOT be used and MUST be ignored by compliant implementations.</td></tr>
</table>
</td></tr>
<tr><td>Reserved</td><td>This field is reserved for future use and MUST be set to zero. Compliant implementations MUST ignore the contents of this field.</td></tr>
<tr><td>File Size</td><td>This field includes an eightoctet value defining the overall size of the file in bytes. This field is able to handle 32 and 64 bit operating systems (future >64 bit operating system would require a new attribute). 32 bit operating systems MUST place the 32 bit length in the least significant 32 bits of this field and zero out the upper 32 bits.</td></tr>
<tr><td>File Create Time</td><td>This field includes an eightoctet value defining the number of seconds since the Unix epoch (mid-night Jan 1st, 1970 UTC) when the file was created. This field is intended to match the Unix standard st_ctime field in the stat structure. For operating systems using 32 bit st_ctime values, the initial four octets MUST be set to zero while the final four octets are set to the st_ctime value associated with the file.</td></tr>
<tr><td>Last Modify Time</td><td>This field includes an eightoctet value defining the number of seconds since the Unix epoch (mid-night Jan 1st, 1970 UTC) when the file was last modified. This field is intended to match the Unix standard st_mtime field in the stat structure. For operating systems using 32 bit st_mtime values, the initial four octets MUST be set to zero while the final four octets are set to the st_mtime value associated with the file.</td></tr>
<tr><td>Last Access Time</td><td>This field includes an eightoctet value defining the number of seconds since the Unix epoch (mid-night Jan 1st, 1970 UTC) when the file was last accessed. This field is intended to match the Unix standard st_atime field in the stat structure. For operating systems using 32 bit st_atime values, the initial four octets MUST be set to zero while the final four octets are set to the st_atime value associated with the file.</td></tr>
</table>

File Owner ID	This field includes an eightoctet value defining the identity of the owner of the file. This field is intended to match the Unix standard st_uid field in the stat structure. For operating systems using 32 bit st_uid values, the initial four octets MUST be set to zero while the final four octets are set to the st_uid value associated with the file.
Group Owner ID	This field includes an eightoctet value defining the identity of the owner of the file. This field is intended to match the Unix standard st_gid field in the stat structure. For operating systems using 32 bit st_gid values, the initial four octets MUST be set to zero while the final four octets are set to the st_gid value associated with the file.
Filename	This field includes a UTF-8 encoded string containing the file name associated with the requested file. This name is only the last component of a fully qualified file path name. For example, "/tcg/iwg/example.txt" path name would have a file name of "example.txt". The length of this variable length field can be computed by subtracting the length of the fixed length portion of the metadata from the File Metadata Length field. If the resulting value is zero this means that no Filename was included.

PTS-IMV supporting this specification on a Unix platform SHOULD support reception and processing of this attribute, while the PTS-IMC running on a Unix platform SHOULD support sending this attribute. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

3.18 Registry Based Metadata

The registry has a structure comparable to the structure of a filesystem on disk. Keys are comparable with directories whereas values are comparable with files. A key may contain other sub-keys or values. Values store data, however each key does also have an unnamed value that is marked as "(Default)" in the graphical GUI viewer. When requesting this key using the PTS protocol, the value-name MUST be set to the NULL string (zero length string) to indicate the default value. A value in the context of the Windows registry consists of the triple {value-name, value-type, value-data}. This triple is the one shown in the right tab of the GUI that appears when running "regedit" under Windows.

Example

The following example illustrates the difference between keys and value-names, value-types, and value-data. Under Windows 7, there is a registry key called "HKEY_CURRENT_USER\Software\Microsoft\Keyboard\Native Media Players\WMP". This key consists of three value triplets {value-name, value-type, value-data}:

1. There is a value with the value-name "(Default)" and a value-type of REG_SZ. The value-data carries the string "(value not set)".

2. There is a value with the value-name "AppName" of value-type REG_SZ and with value-data set to "Windows Media Player"

3. There is a value with value-name set to "ExePath", value-type set to REG_SZ and value-data set to "C:\Program Files\Windows Media Player\ wmplayer.exe"

3.18.1 Request Registry Key Metadata

This attribute contains the registry key indicating the location of the desired value in the registry. The use of the registry is primarily supported on Windows operating systems, so other endpoint operating systems are unlikely to support this attribute type.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |                    Reserved                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~  Fully Qualified Registry Object Pathname (Variable Length)   ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Flags</td><td>This field contains flags.
<table>
<tr><td>Bit Encoding</td><td>Description</td></tr>
<tr><td>Bit 0 – Key Contents</td><td>When the Fully Qualified Registry Object Pathname is a pathname to a registry key, this flag indicates whether metadata about possible sub-keys is desired to be returned or metadata about the key itself.</td></tr>
<tr><td>Bit 1 – 7 – Reserved</td><td>This field MUST be set to 0 and MUST be ignored by compliant implementations</td></tr>
</table></td></tr>
<tr><td>Reserved</td><td>This field is reserved for future use and MUST be set to zero. Compliant implementations MUST ignore the contents of this field.</td></tr>
<tr><td>Fully Qualified Registry Object Pathname</td><td>This variable length field contains the UTF-8 encoding of the fully qualified path to a particular key or value triplet {value-name, value-type, value-data} on the system being attested (requestor). For detailed information about the required syntax for this field see section 3.18.1.1. For example “KEY_CURRENT_USER\Software” might be used while “Software” is not a full path, so it is not allowed.

If the PTS receives an unknown key pathname, it MUST respond with a TCG_PTS_REG_KEY_NOT_FOUND error code.</td></tr>
</table>

PTS-IMC supporting this specification SHOULD support reception and processing of this attribute, while the PTS-IMV SHOULD support sending this attribute. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

3.18.1.1 Fully Qualified Key Pathname Syntax

This section defines the required syntax for the Fully Qualified Object Pathname field in the Request Registry Value and Metadata attribute. The Fully Qualified Object Pathname is a UTF-8 string that MUST use the pathname syntax of:

Pathname ::= Key-Pathname **|** Value-Pathname;

Key-Pathname ::= Predefined-Key , {Delimiter, Sub-Key}+, Delimiter

Value-Pathname ::= Predefined-Key , {Delimiter, Sub-Key}, Delimiter, Value-Name

Where:

Predefined-Key – This value is a UTF-8 string identifying the registry root key up to but not including the Delimiter value. The current predefined keys in Windows are: HKEY_CLASSES_ROOT, HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE, HKEY_USERS, and HKEY_CURRENT_CONFIG.

Delimiter – separator between components of the pathname "\" in the Windows registry

Sub-Key – name of Key in the pathname. This value is a UTF-8 string up to but not including the Delimiter value.

Value-Name – final component of pathname to a particular registry key value and is not used for identifying a Key. The Value-Name is a UTF-8 string up to the end of the pathname.

Example:

"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer" would be a valid example for a Key-Pathname, whereas "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Version" would be a valid example for a Value-Pathname.

Note:

Most operating systems limit the set of allowable characters that can be used in value and Key names, so PTS-IMV should use care when using such characters since it will just result in TCG_PTS_REG_KEY_NOT_FOUND errors. However, this protocol does not impose character restrictions beyond those described above since the restrictions vary by target operating system.

The Predefined-Key, Sub-Key and Value-Name components of the pathname MUST NOT include the Delimiter value defined in this attribute. As shown in the syntax above, the Value-Name form of the pathname MUST NOT include a Delimiter as the final byte where the Key-Pathname form of the pathname MUST use a Delimiter as the final UTF-8 character.

3.18.2 Registry Key Metadata

This attribute is sent as response to a Request Registry Key Metadata attribute. The metadata can be retrieved using the Windows RegQueryInfoKey function.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Number of Keys Included                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Number of Keys Included                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Key Metadata Length                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Reserved                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Classname Length                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Number of Sub-Keys                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Maximum Sub-Key Name Length                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Maximum Sub-Key Classname Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Number of Key Values Triplets               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Maximum Value-Name Length                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Maximum Value-Data Size                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Owner ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Owner ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Group ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Group ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Last Write Time                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Last Write Time                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Key-Name (variable Length)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               User Defined Class (variable Length)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Key #2 Metadata Length                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Reserved                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Classname Length #2                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Number of Sub-Keys #2                      |
|                         …………                                  |
```

Field	Description
Number of Keys Included	This value indicates the number of keys being reported in this attribute. Normally this value is set to one, but when the request is for a directory containing Sub-Keys, the value may be between zero and 2^64-1. This is the only field in this attribute that is not repeated when multiple keys are included.
Key Metadata Length	This field indicates the number of bytes including this field taken up by the value of the next key including both the variable length metadata and the variable length key-name. When multiple keys are described in this attribute the length field is used to locate the start of the next key metadata (since there are variable length attributes included). If the PTS-IMV requested the metadata of a single key, the PTS-IMC MAY return the metadata without the key-name by setting this field to the length of the variable length metadata fields and zero for the key-name. The variable length of the metadata is the sum of the length of all attributes returned by RegQueryInfoKey, which includes a string of variable length.
Reserved	This field is reserved for future use and MUST be set to zero. Compliant implementations MUST ignore the contents of this field.
Classname Length	This field includes the length of the user-defined class name of the requested key. This value is determined using the lpcClass value returned by RegQueryInfoKey. lpcClass counts also the terminating NULL character, however in this message, Classname Length is the length of the user-defined class name excluding the terminating

	NULL character. This value may be NULL.
Number of Sub-Keys	This field includes the 32 bit number of Sub-Keys within the requested key. This value is determined by calling RegQueryInfoKey and using the lpcSubKeys value. This parameter could be NULL.
Maximum Sub-Key Name Length	This field contains the size of the Sub-Key with the longest name. The length counts the number of Unicode characters, not including the terminating NULL character. This value is obtained by calling the RegQueryInfoKey method and using the lpcMaxSubKeyLen value. The parameter can be NULL.
Maximum Sub-Key Classname Length	This field contains the size of the longest classname of all Sub-Keys of this key, not including the terminating NULL character. This value is obtained by calling RegQueryInfoKey and using the lpcMaxClassLen value. This value can be NULL.
Number of Values Triplets	This field includes the 32 bit number of Key Values Triplets of the requested key. This value can be obtained by calling RegQueryInfoKey and using the lpcValues value. It may be NULL.
Maximum Value-Name Length	This field contains the size of the longest Value-Name of the requested key, not including the terminating NULL character. This value may be obtained by calling RegQueryInfoKey and using the lpcMaxValueNameLen value. The parameter can be NULL.
Maximum Value-Data Size	This field carries the size of the longest Value-Data of the requested key, in bytes. This value may be obtained by calling ReQueryInfoKey and using the lpcMaxValueLen value. It may be NULL.
Owner ID	This field indicates the operating system’s identity of the owner of the requested key. This information can be obtained by calling RegGetKeySecurity to get the key’s security descriptor and then retrieving the owner using the GetSecurityDescriptorOwner call.
Group ID	This field indicates the operating system’s identity of the group of the requested key. This information can be obtained by calling RegGetKeySecurity to get the key’s security descriptor and then retrieving the group using the GetSecurityDescriptorGroup call.
Last Write Time	This field contains the last write time of the requested key. This 64 bit value can be obtained by calling RegQueryInfoKey and using the lpftLastWriteTime value. The parameter can be NULL
Key-Name	This field includes a UTF-8 encoded string containing the Key-Name associated with the requested key. This name is only the last component of a fully qualified key pathname. For example “HKEY_CURRENT_USER\Software\TCG” pathname would have a key-name of “TCG”.
User Defined Class	This field contains a UTF-8 encoded string containing the user defined class associated with the requested key. This value may be obtained by calling RegQueryInfoKey and using the lpClass value. It can be NULL. This field does not include the terminating NULL

	character.

PTS-IMV supporting this specification SHOULD support reception and processing of this attribute, while the PTS-IMC running on a Windows platform SHOULD support sending this attribute. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

3.18.3 Request Registry Key Value Data

This attribute enables the PTS-IMV to request the value associated with a particular registry key. This attribute is a peer to the Request Registry Metadata attribute that returns only metadata about the registry key, but not its content {value-name, value-type, value-data}.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |                  Reserved                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~   Fully Qualified Registry Object Pathname (Variable Length)  ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Flags</td><td>This field contains flags. In order to maintain consistency with the Request Registry Metadata attribute (and for future use), this field was left in the attribute despite not being used.
<table><tr><th>Bit Encoding</th><th>Description</th></tr><tr><td>Bit 0 – 7 – Reserved</td><td>This field MUST be set to 0 and MUST be ignored by compliant implementations.</td></tr></table></td></tr>
<tr><td>Reserved</td><td>This field is reserved for future use and MUST be set to zero. Compliant implementations MUST ignore the contents of this field.</td></tr>
<tr><td>Fully Qualified Registry Object Pathname</td><td>This variable length field contains the UTF-8 encoding of the fully qualified path to a particular key or value triplet {value-name, value-type, value-data} on the system being requested. For detailed information about the required syntax for this field see section 3.18.1.1. For example “KEY_CURRENT_USER\Software” might be used while “Software” is not a full path, so it is not allowed.
If a path to a key is given, the IMC has to send the value-data of all values defined under that key which could consist of multiple triplets. If this field contains a path to a value-triplet, the IMC will return the value-data of that triplet only.
If the PTS receives an unknown key pathname, it MUST respond with a TCG_PTS_REG_KEY_NOT_FOUND error code.</td></tr>
</table>

PTS-IMC supporting this specification SHOULD support reception and processing of this attribute, while the PTS-IMV SHOULD support sending this attribute. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

3.18.4 Registry Key Value Data

This attribute returns the value of the requested registry key or value. The use of the registry is primarily supported on Windows operating systems, so other endpoint operating systems are unlikely to support this attribute type. The value can be requested using the Windows RegGetValue function.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Number of Key Value Triplets Included             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Number of Key Value Triplets Included             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Key Value Triplet Length                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Value-Type                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Reserved                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Value-Data Length                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                     Value-Data (variable length)              ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                     Value-Name (variable length)              ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Key Value Triplet Length #2               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Value-Type #2                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Reserved                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                   Value-Data (variable length) #2             ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                   Value-Name (variable length) #2             ~
```

Field	Description
Number of Key Values Triplets Included	This value indicates the number of key value triplets {value-name, value-type, value-data} being reported in this attribute. Normally this value is set to one, but when the request is for an entire key, the value may be between zero and 2^64-1. This is the only field in this attribute that is not repeated when multiple files are included.
Key Value Triplet Length	This field indicates the number of bytes including this field taken up by the value triplet of the next key including both the possibly variable length value-data and the variable length value-name. When multiple key value triplets are described in this attribute the length field is used to locate the start of the next key value triplet (since the value-name as well as the value-data are of variable length).

<table>
<tr>
<td>Value-Type</td>
<td>This field indicates the value-type of the value triplet being described. This value might be obtained using the RegGetValue call on Windows.
The following enumeration defines the current type values:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0 – Other</td><td>This key value type indicates a value that is either unknown or different from all types of the standardized key value types.</td></tr>
<tr><td>1 – REG_NONE</td><td>This key value type indicates no value type.</td></tr>
<tr><td>2 – REG_SZ</td><td>A key value of type REG_SZ is a fixed length Unicode string.</td></tr>
<tr><td>3 – REG_EXPAND_SZ</td><td>This key value type represents a variable length Unicode string.</td></tr>
<tr><td>4 – REG_BINARY</td><td>A key value of type REG_BINARY carries binary data of arbitrary length.</td></tr>
<tr><td>5 – REG_DWORD</td><td>This key value type indicates a 32 bit number.</td></tr>
<tr><td>6 – REG_DWORD_LITTLE_ENDIAN</td><td>This key value type indicates a 32 bit number with the lowest byte first.</td></tr>
<tr><td>7 – REG_DWORD_BIG_ENDIAN</td><td>This key value type indicates a 32bit number with the highest byte first.</td></tr>
<tr><td>8 – REG_LINK</td><td>This key value type indicates a 16-bit Unicode symbolic link.</td></tr>
<tr><td>9 – REG_MULTI_SZ</td><td>This key value type indicates an array of Unicode NULL-terminated strings. The array itself is terminated with two NULL characters.</td></tr>
<tr><td>10 – REG_RESOURCE_LIST</td><td>A key value of type REG_RESOURCE_LIST carries a hardware resource description.</td></tr>
<tr><td>11 – REG_FULL_RESOURCE_DESCRIPTOR</td><td>A key value of type REG_FULL_RESOURCE_DESCRIPTOR carries a</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>hardware resource description.</td></tr>
<tr><td>12 – REG_RESOURCE_REQUIREMENTS_LIST</td><td>This key value type indicates resource requirements.</td></tr>
<tr><td>13 – REG_QWORD</td><td>This key value type indicates a 64 bit number.</td></tr>
<tr><td>14 – REG_QWORD_LITTLE_ENDIAN</td><td>This key value type indicates a 64 bit number with the lowest byte first.</td></tr>
<tr><td>15 – REG_QWORD_BIG_ENDIAN</td><td>This key value type indicates a 64 bit number with the highest byte first.</td></tr>
<tr><td>16 – 255 Reserved</td><td>These values MUST NOT be used and MUST be ignored by compliant implementations.</td></tr>
</table>
</td></tr>
<tr><td>Reserved</td><td>This field is reserved for future use and MUST be set to zero. Compliant implementations MUST ignore the contents of this field.</td></tr>
<tr><td>Value-Data Length</td><td>This field carries the length of the value-data. This length MAY be set to NULL if the length of the data is determined by its value-type, but for variable length value-data this field MUST be set to the length of value-data field of this triplet excluding the terminating NULL.</td></tr>
<tr><td>Value-Data</td><td>This field includes the value-data portion of the value triplet determined by Value-Name and of type Value-Type excluding the NULL character.</td></tr>
<tr><td>Value-Name</td><td>This field includes a UTF-8 encoded string containing the value-name associated with the requested key or the name of the key value triplet requested. This name is only the last component of a fully qualified key pathname. For example “HKEY_CURRENT_USER\Software\TCG” pathname would have a key or value name of “TCG”.</td></tr>
</table>

PTS-IMV supporting this specification SHOULD support reception and processing of this attribute, while the PTS-IMC running on a Windows platform SHOULD support sending this attribute. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

3.19 File Measurement Attributes

This section defines the attributes used to request the PTS perform a file or process measurement. All file measurements use the PTS measurement algorithm currently negotiated for this assessment.

3.19.1 Request File Measurement

This attribute is sent by the PTS-IMV to request the measurement of a particular regular file or directory. A request for measurement of other types of files (e.g. device instance) is operating system dependent on when the PTS-IMC will return a TCG_PTS_OPERATION_NOT_SUPPORTED or to perform the operation. When the measurement request indicates a directory and the Directory Contents flag is set, the PTS-IMC will respond with a File Measurement attribute including the measurement of every file in the directory (not a hash of the directory internal structure).

This attribute closely resembles the Request File Metadata attribute in that it includes an operating system agnostic (flexible file delimiter) fully qualified file pathname to maximize the usefulness of the attribute in different situations.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |   Reserved    |          Request Id           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Delimiter                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~        Fully Qualified File Pathname (Variable Length)        ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Flags</td><td>This field contains flags that modify the target filename requested.
<table>
<tr><th>Bit Encoding</th><th>Description</th></tr>
<tr><td>Bit 0 – Directory Contents</td><td>When the specified Fully Qualified File Pathname field contains the pathname to a directory, this value indicates whether the measurement of each file in the directory is desired to be returned or the measurement of the directory node itself (not all operating systems will support measuring a directory).

If this bit is set to 0, this indicates that the measurement of the directory itself is to be returned.
If this bit is set to 1, this indicates that the recipient should return measurements of every file in the directory (not the directory itself). The Directory Contents bit MUST NOT be set to one when in an attribute referencing a non-directory.</td></tr>
<tr><td>Bit 1-7 - Reserved</td><td>This field MUST be set to 0 and MUST be ignored by compliant implementations.</td></tr>
</table>
</td></tr>
<tr><td>Reserved</td><td>This field is reserved for future use and MUST be set to zero. Compliant implementations MUST ignore the contents of this field.</td></tr>
<tr><td>Request Id</td><td>This field contains a unique number selected by the PTS-IMV used to match up responses with requests. The PTS-IMC merely copies this into the responses so is unaware of the contents, therefore it’s up to the PTS-IMV to select a value that will allow it to determine which request the response received is associated with. It is suggested that a monotonic counter could be a good approach, but no approach is required since interoperability is not impacted.</td></tr>
</table>

Delimiter	This field contains the UTF-8 encoding of the character used to delimit the filename given in the Fully Qualified File Pathname field. For example, the '\' character is typically used on Windows systems to separate the directory names in a path while Unix oriented systems use the '/' character. The Delimiter field MUST contain the UTF-8 character used to separate directory names in the Fully Qualified File Pathname field or a TCG_PTS_INVALID_DELIMITER error is sent by the recipient. This field is included in the request in case the sender does not yet know what type of operating system is present on the endpoint. This is not envisioned to be common. PTS-IMC MUST convert the included Delimiter to the one used by the local operating system in order to respond to the request. PTS-IMV SHOULD use an appropriate Delimiter for the endpoint's operating system once discovered during an assessment.
Fully Qualified File Pathname	This variable length field contains the UTF-8 encoding of the fully qualified path to a particular file or directory on the system being attested (requestor). For detailed information about the required syntax for this field see section 3.17.1.1. For example "C:\tcg\iwg\example" might be used while "iwg\example" is not a full path so it not allowed. The Fully Qualified File Pathname field MUST specify an unambiguous, fully qualified file system location. The PTS-IMC (or the PTS) SHOULD be flexible in the Delimiters it supports even when the delimiter does not match the one used by the local operating system. For example a Unix-based PTS might support the Windows oriented file pathnames. If the PTS receives an unknown file pathname, it MUST respond with a TCG_PTS_FILE_NOT_FOUND error code.

PTS-IMC supporting this specification SHOULD support reception and processing of this attribute, while the PTS-IMV SHOULD support sending this attribute. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

3.19.1 File Measurement

This attribute contains the measurement of a particular file or directory requested by the PTS-IMV. The hash algorithm used to create the measurement MUST use the algorithm selected during the PTS Measurement Algorithm attribute exchange. Note that this attribute is capable of including measurements of multiple files in case the request was for the contents of a directory.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                    1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Number of Files Included                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Number of Files Included                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Request Id           |      Measurement Length       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Measurement (Variable Length)  . . .         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Filename Length         | Filename (Variable Length) . .|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Filename (Variable Length)  . . .          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Measurement #2 (Variable Length)  . . .        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Filename Length #2      | Filename #2 (Variable Length) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             . . . . . . .                     |
```

Field	Description
Number of Files Included	This value indicates the number of files being reported in this attribute. Normally this value is set to one, but when the request is for an entire directory, the value may be between zero and 2^64-1. This is the only field in this attribute that is not repeated when multiple files are included.
Request Id	This field indicates the Request Id provided by the PTS-IMV associated with this response. The PTS-IMC MUST copy and not interpret the Request Id provided by the PTS-IMV in the Request File Measurement attribute associated with this response.
Measurement Length	This field indicates the number of bytes used by the Measurement variable length field for each file for the entire attribute. Because every measurement uses the same PTS measurement algorithm, the measurement lengths will be the same for every file, so this field is not repeated. Note that this length does not include the Filename field just the Measurement field.
Measurement	This field contains the measurement (hash) of the contents of the file. The measurement algorithm used for each file listed in this attribute MUST be the algorithm selected during the most recent PTS measurement algorithm attribute exchange. The length of this variable length field is indicated by the Measurement Length.
Filename Length	This field indicates the number of bytes used by the Filename variable length field. If this attribute contains the measurement of only one file and it is the exact file requested by the PTS-IMV then the PTS-IMC MAY set this field to zero and not include the filename (since it is already known by the PTS-IMV).
Filename	This field includes a UTF-8 encoded string containing the file name associated with the requested file. This name is only the last component of a fully qualified file path name. For example, “c:\tcg\iwg\example.txt” path name would have a file name of “example.txt”.

PTS-IMC supporting this specification SHOULD support reception and processing of this attribute, while the PTS-IMV SHOULD support sending this attribute. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

3.20 Template Reference Manifests

This section describes the role of the template Reference Manifest (AKA Reference Integrity Measurement Manifest) and how it is used during an attestation. The concept of a template Reference Manifest exists in order to allow a PTS-IMV to request that a PTS create an integrity report in a particular format and structure to ease comparison against a Reference Manifest (with trusted reference measurements). By allowing the PTS-IMV to request a particular component/sub-component ordering (structure), this allows the PTS-IMV to walk through the report sequentially comparing each reported (sub)component measurement and meta-data against its similarly structured Reference Manifest containing reference measurements and meta-data considered acceptable by the policy.

For example, a PTS-IMV wishes to obtain an integrity report for an endpoint's operating system kernel components and sub-components. An endpoint's operating system's kernel is comprised of a complex set of components many of which are made up of a hierarchical set of sub-components that PTS might need to describe in an integrity report. In order for the PTS to create a report with a particular structure and component ordering, it needs a "map" for how to describe the composition of the kernel. Without a template Reference Manifest, the PTS could describe the components and sub-components making up the kernel in an enormous number of orderings making the PTS-IMV job more difficult. By having a template Reference Manifest, the PTS can walk sequentially through the template Reference Manifest and use it as policy for how to order and structure the report, since the template Reference Manifest will list a single ordering for describing the set of components and the associated sub-components for each portion of the operating system.

Now that we have the notion of a template Reference Manifest that drives the format and ordering of the integrity report created by the PTS, we can associate other meta-data with the template manifest to allow detection of newer template Reference Manifests, a top level functional component indicator (e.g. this is an operating system kernel without all the specifics potentially found in a ComponentID) and an indication of the vendor that created the component. Both of these values are described in the following subsections and how they enable a template Reference Manifest discovery and update to occur.

3.20.1 Template Reference Manifest Schema

The template Reference Manifest describes the structure and content that is to be used by the PTS when creating an integrity report associated with the particular component or product. There are a number of possible formats for describing the policy used by the PTS when creating the integrity report, so in order to enable interoperability and to minimize the amount of implementation and deployment work required, the TCG standard template Reference Manifest will have the same XML schema as specified for the Reference Integrity Measurement Manifest.

In fact, a Reference Manifest for a component can serve double duty and also act as a template Reference Manifest. In order to leverage the existing Reference Manifest schema, it is important to understand that a typical Reference Manifest contains a significant amount of information that is not required to fulfill the role of the template Reference Manifest. For example a Reference Manifest contains reference measurements for each (sub)component. When the PTS is creating an integrity report of the component(s) described by a template Reference Manifest, it is taking the actual measurements that exist on the endpoint and therefore doesn't have a need for the reference measurement value elements (unless it is performing local validation) in the Reference Manifest. Therefore in order to ease the challenge of creating and managing template Reference Manifests (in addition to other Reference Manifests) it is envisioned that the PTS and PTS-IMV would use already provisioned Reference Manifests that they consider trustworthy as template Reference Manifests.

For example a PTS-IMV might have a Reference Manifest describing the expected set of measurements and meta-data about an anti-virus security component in the kernel. The PTS-IMV will check any measured values sent from the endpoint against the contents of this Reference Manifest to determine if the endpoint is compliant with policy for kernel anti-virus features. At the same time, the PTS-IMV would want the endpoint's PTS to create the integrity report in the same format and component ordering as in this anti-virus Reference Manifest, so it would negotiate and potentially send the contents of the kernel anti-virus Reference Manifest to the endpoint to be used as its template Reference Manifest during the attestation. The PTS would ignore the fields in the Reference Manifest that aren't relevant to creating an integrity report and focus on the general structure and set of (sub)components listed in the Reference Manifest. When the PTS-IMV receives the integrity report it can sequentially walk through the integrity report and Reference Manifest comparing actual measurements with expected reference measurements for each (sub)component.

3.20.2 Multiple Template Reference Manifests

Each endpoint typically consists of components from a potentially large number of sources. For example, the trusted building blocks, operating system, security software, backup software, remote management, browser and other applications frequently come from a variety of different vendors. Therefore, there are several possible deployment models for the template Reference Manifest.

The first model might occur in an enterprise setting where the IT department has strict controls over the configurations and contents of every endpoint. In this case, an IT department might issue a single or small number of template Reference Manifests that aggregate the components and sub-components of most or all of the different vendor products allowed to some of the enterprise's endpoints. Multiple of these IT minted template Reference Manifests might exist over time for the same configuration build allowing for a gradual transition between versions of vendor products (e.g. allowing the latest or last week's operating system patch levels). This approach might require frequent Reference Manifest issuance because of newly approved patches being included the standard build thus requiring a new Reference Manifest including those measurements. Note that if the patches do not change the component hierarchy, a new template Reference Manifest isn't required to be issued but issuance of the new Reference Manifest to all verifiers would be necessary.

Another model recognizes environments where endpoints may fill different roles or have different ownership, so a single entity does not create a single template Reference Manifest describing all of the possible sets of functional component versions allowed on the endpoint. Instead, a more scalable solution might be to issue a potentially larger set of smaller template Reference Manifests each describing one or a small set of related features or a smaller product. In this case, the PTS might require storage of a larger set of template Reference Manifests and must select and use the appropriate template Reference Manifests depending upon which products/components are requested by the PTS-IMV during an attestation. The PTS can create an integrity report corresponding to each template Reference Manifest and then concatenate several together when responding to an attestation of several products or different components.

In both of these general models, the PTS might have more than one template Reference Manifest, so an identifier is needed that describes what product/component the template Reference Manifest is describing in addition to the level described earlier.

3.20.3 New Reference Manifest Elements

As updates and new releases of the operating system occur, the set of components and sub-components that make up the operating system's kernel may change. As new kernel modules are added or removed or the sub-component hierarchy changes, these changes need to be reflected in new revisions of the template Reference Manifest. Therefore, the template Reference Manifest (and thus base Reference Manifest) contains three new identifiers: FunctionalComponentID, ComponentVendorID and TemplateVersion identifying the top level functional component (e.g. operating system kernel's firewall) described in the Reference Manifest and what version of the Reference Manifest structure is included by the Reference Manifest. The FunctionalComponentID indicates what type of component is described by the Reference Manifest while the TemplateVersion is incremented each time a change to the (sub)component ordering occurs. The level enables the PTS and PTS-IMV to do comparisons to know which party's TemplateVersion is more recent.

This section assumes the following elements are added to the Reference Manifest schema:

- FunctionalComponentID – Unsigned 64-bit integer format that indicates the functional component (e.g. firewall) described by the Reference Manifest. This will match the component functional naming used by this protocol.
- ComponentVendorID – Unsigned 24-bit integer value that indicates the Private Enterprise Number (PEN) of the vendor who defined the FunctionalComponentID functional component described by this Reference Manifest. This is effectively the namespace ID for FunctionalComponentID.
- TemplateVersion – Unsigned 32-bit integer value that initially MUST be set to 1 by Reference Manifest creators and MUST be monotonically incremented only when changes to the component format and ordering (not when changes to the reference values or meta-data for a particular component) occur.

3.20.4 Template Reference Manifest Level Negotiation

In order to ease the comparison of the components included in an integrity report with those included in a Reference Manifest (containing reference measurements from a trusted source), the PTS-IMV needs the ability to determine what template Reference Manifests exist on the endpoint and provide an update when the PTS-IMV has a different (potentially newer) set then the endpoint (or vice versa). The following sub-sections describe several attributes that enable the PTS-IMC or PTS-IMV to request the active set of Reference Manifests that the other party possesses and trusts for use during an attestation. Similarly either party can propose the “update” of the set of template Reference Manifests held by the other party if allowed by its policy.

For example, a PTS-IMV may wish to assess the trusted building blocks, kernel-based firewall and the PTS on an endpoint. In order to achieve this, the PTS-IMV may send a Request Template Reference Manifests Set attribute listing a set of products or components it plans to include in the attestation. The PTS-IMC (and PTS) would (if allowed by local policy) return the set of template ComponentVendorID and TemplateVersion values associated with each of the requested components. The PTS-IMV would compare the responses with the set of Reference Manifests it is prepared to accept from the endpoint and if they are different could use the Update Template Reference Manifest Set attribute to send the endpoint one or more template Reference Manifest(s) it prefers particularly if the PTS-IMV held a newer version.

3.21 Request Template Reference Manifest Set Metadata

This attribute can be sent by a PTS-IMV in order to request metadata (e.g. version level) about the set of template Reference Manifest information supported by the recipient of the message. The recipient of this message would inspect the set of template Reference Manifests it possesses for each Component Functional Name and would be willing to use in an attestation with the sending party and responds with the Template Reference Manifest Set Metadata attribute.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Component Functional Name #1                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Component Functional Name #1                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                         . . . . .                             ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Component Functional Name #N                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Component Functional Name #N                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Header Field	Description

Component Functional Name	Contains the enumerated name of the functional component described in the requested template Reference Manifest. This value is compared by the recipient against the set of Reference Manifest in its possession. This field is four octets in length and can be repeated many times in this attribute to request multiple attributes. See section 5 for a description of the functional naming.

PTS-IMC supporting XML-based attestation evidence SHOULD support reception and processing of this attribute, while the PTS-IMV supporting XML-based attestation evidence SHOULD support sending this attribute. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

3.22 Template Reference Manifest Set Metadata

This attribute contains a short description (metadata) about each of the template Reference Manifests held and trusted by the sender. This attribute is sent in response to a Request Template Reference Manifest Set Metadata attribute so includes metadata associated with the requested components. This manifest set could contain multiple different revisions of the same functional component’s Reference Manifests that are supported by the PTS-IMC (e.g. if a sender has multiple usable Reference Manifests covering the requested component). The Template Reference Manifest Set Metadata attribute sent MUST not include metadata about Reference Manifests that do not include any of the requested functional components, but MAY contain Reference Manifests that include a requested component plus other components.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Component Functional Name #1                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Component Functional Name #1                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   RM Flags    |          Component Vendor Name #1             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Component Template Version #1                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                         . . . . .                             ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Component Functional Name #N                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Component Functional Name #N                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   RM Flags    |          Component Vendor Name #N             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Component Template Version #N                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Header Field	Description
Component Functional Name	Contains the enumerated name of the functional component described in the requested template Reference Manifest. This value is compared by the recipient against the set of trusted Reference Manifests in its possession. This field is four octets in length and can be repeated many times in this attribute to request multiple attributes. See section 5 for a description of the functional naming. The contents of this field are copied from the template

<table>
<tr><td></td><td>Reference Manifest's FunctionalComponentID element.</td></tr>
<tr><td>RM (Reference Manifest) Flags</td><td>This field contains flags that are associated with the vendor, level or usage of a particular described Reference Manifest.
<table>
<tr><th>Bit Encoding</th><th>Description</th></tr>
<tr><td>Bit 0 – Updatable</td><td>This Reference Manifest may be updated for use with this attestation. This flag helps the recipient understand whether it should attempt to update the Reference Manifest if it possesses a newer level.</td></tr>
<tr><td>Bit 1-7 - Reserved</td><td>This field MUST be set to 0 and MUST be ignored by compliant implementations.</td></tr>
</table>
</td></tr>
<tr><td>Component Vendor Name</td><td>This field contains the IANA assigned three octet SMI private enterprise number (PEN) associated with the vendor that implemented this functional component. The contents of this field are copied from the template Reference Manifest's ComponentVendorID element.</td></tr>
<tr><td>Component Template Version</td><td>This field contains the four octet revision number of the template Reference Manifest in use by the sender. The level indicates how many times the order and format of the components in this Reference Manifest have changed so the parties can determine if they have a Reference Manifest with the same layout. The contents of this field are copied from the template Reference Manifest's TemplateVersion element.</td></tr>
</table>

PTS-IMV supporting XML-based attestation evidence SHOULD support reception and processing of this attribute, while the PTS-IMC supporting XML-based attestation evidence SHOULD support sending this attribute. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

3.23 Update Template Reference Manifest

This attribute allows the PTS-IMV to transmit a full Reference Manifest to the recipient for use during the attestation. The PTS-IMC SHOULD validate the signature on the Reference Manifest to ensure it is authentic (however the values aren't used for template purposes). All PTS-IMC and PTS-IMV SHOULD consult the Reference Manifest Flags field in the Template Reference Manifest Set attribute to determine whether the other party has a willingness to update the Reference Manifest before sending this attribute.

This attribute contains a single template Reference Manifest, so in cases where the sender wishes to update multiple Reference Manifests then multiple of this attribute would be sent. This was done to help reduce the size of the individual Update Template Reference Manifest attributes in case there are underlying transport limitations regarding attribute sizes.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                    1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~         Template Reference Manifest (Variable Length)         ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description

Template Reference Manifest	This field includes a full XML document as described in [RM] describing the ordering of components and subcomponents that are desired in the attestation. The recipient MAY choose to use this template Reference Manifest as an actual manifest (if appropriate) after validating the integrity, authenticity and trustworthiness of the Reference Manifest's contents. If the recipient is unable to update or wishes to reject the Template Reference Manifest sent, the recipient SHOULD send a TCG_PTS_RM_ERROR message.

PTS-IMC supporting XML-based attestation evidence SHOULD support reception and processing of this attribute, while the PTS-IMV supporting XML-based attestation evidence SHOULD support sending this attribute. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

3.24 Request Integrity Measurement Log

This attribute allows the PTS-IMV to request the Integrity Measurement Log (IML) entries associated with a particular PCR and functional component. It is envisioned that after a functional component has been attested using evidence associated with a TPM PCR that the challenger might wish to obtain the IML details involving the component. This attribute allows for fetching either a PCR's IML or can additionally allow the IML to be filtered down to just entries associated with a particular functional component (when supported by the PTS or PTS-IMC).

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |               PCR Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Optional Sub-Component Depth                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Optional Component Functional Name              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Optional Component Functional Name              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Flags</td><td>This field identifies whether optional fields are included in this attribute.
<table>
<tr><td>Bit Encoding</td><td>Description</td></tr>
<tr><td>Bit 0 – Component Filter</td><td>Indicates whether the Optional Component Functional Name and Sub-Component Depth fields are present.</td></tr>
<tr><td>Bit 1-7 - Reserved</td><td>This field MUST be set to 0 and MUST be ignored by compliant implementations.</td></tr>
</table>
</td></tr>
</table>

PCR Number	This field identifies the PCR number of the Integrity Measurement Log requested by the sender. This must be a single value; however the PTS-IMV may send additional requests with other PCR numbers in the future.
Optional Sub-Component Depth	This field indicates the depth down the subcomponent hierarchy below the Optional Functional Component Name that is requested. For instance, if a requested component uses 3 libraries and these libraries dynamically load 2 other objects, then we have a 3 level hierarchy of sub-components. Therefore the libraries would exist at a depth of 1 and the dynamically loaded objects would be at depth 2.
Optional Functional Component Name	This field filters down the entries in the PCR's IML such that only entries associated with a component or sub-component of the identified functional component are included. This field is four octets in length and can be repeated many times in this attribute to request multiple attributes. See section 5 for a description of the functional naming.

PTS-IMC SHOULD support reception and processing of this attribute, while the PTS-IMV SHOULD support sending this attribute. PTS-IMC supporting TPM-backed attestation evidence SHOULD support inclusion of the trusted platform's integrity measurement log messages with the appropriate PTS-based log messages. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

3.25 Integrity Measurement Log

This attribute contains the key set of values expressed in the Trusted Platform's Integrity Measurement Log (IML) associated with the requested PCR and optionally Functional Component Name. This IML (also known as the Event Log for PC Client Trusted Platforms) is typically constructed from the early boot through later RTM like the PTS. As we boot more of the platform, the higher layer RTMs may be able to record more detailed information about the components being measured. For example, the PTS is likely to be aware of the functional component hierarchy, since this attestation protocol leverages this naming scheme. Other lower level RTM (e.g. BIOS) may not leverage this naming, but because a higher layer attestation agent (PTS-IMC and PTS) is likely to be handling the attestation responses, it can infer many of the lower level component's functional names based on the PCR uses and the Event Log (if available).

The IML representation in this attribute likely will not match the version stored in the platform. It's hoped that this TLV representation is easy to derive from what is stored and will enable an interoperable format that can be sent between vendors. It's critical that relying parties have a canonical representation, because the IML is used to compute what components and metadata are composed into the more general PCR values.

The desired IML entry would include:

> *<Date/Time> <Functional Component Name> <Event Type> <Component Measurement>*
> *<PCR Before Value> <PCR After Value> <Other Data>*

Where the date/time time indicates when the entry was created in the IML, the PCR values show component's measurement and its impact (both before and after values) as it is extended into the associated PCR. Finally other data allows for additional information about the component to be captured.

The following diagram shows the contents of the Attribute Value field for this attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |               PCR Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           PCR Length          |      PCR Hash Algorithm       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PCR Transform |                 Reserved                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Number of IML Entries in this Batch              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         IML Entry Date/Time                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         IML Entry Date/Time                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         IML Entry Date/Time                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         IML Entry Date/Time                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         IML Entry Date/Time                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Event Type                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Component Functional Name                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Component Functional Name                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                 Component Measurement (Variable Length)       ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                 PCR Before Value (Variable Length)            ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                  PCR After Value (Variable Length)            ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                         Other Data Length                     ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                    Other Data (Variable Length)               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Note that the initial 128 bits are the preamble header that occurs before each of the IML entries is described. Therefore, the remainder (after the initial 128 bits) of the diagram above can be repeated multiple times allowing the sender to return a batch of IML entries in a single attribute. The sender can also send multiple attributes to carry a large set of IML entries using the Flags field to indicate when more entries are forthcoming.

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Flags</td><td>This field identifies whether optional fields are included in this attribute.
<table>
<tr><th>Bit Encoding</th><th>Description</th></tr>
<tr><td>Bit 0 – First Entry Batch</td><td>First bit. This represents the first bunch of IML entries that were requested. This bit MUST be set to 1 only for the first set of entries returned. Subsequent batches this flag MUST be set to 0.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>Bit 1 – More Entries</td><td>More bit. This bit MUST be set to 1 when the sender has more entries to send after this attribute (more entries remaining after this batch). This value MUST be set to 0 when this attribute contains the last set of entries requested.</td></tr>
<tr><td></td><td>Bit 2 – Functional Name Present</td><td>This bit indicates whether any of the entries in this attribute include the optional functional component type. It's envisioned that most PCR logs would eventually include this name for most or all of its entries. If a log contains a few unknown functional names, the name can indicate "unknown" (see section 5.2 for encoding of unknown in a functional name) for those entries. In some cases a log may completely lack functional names, so this flag allows the Optional Component Functional Name fields to not be present (saving bandwidth).

If this value is set to 0, an Optional Component Functional Name field MUST NOT be present in any of the entries. If this field is set to 1, then every entry MUST have an Optional Component Functional Name field indicating a component name or unknown if the component can't be determined.</td></tr>
<tr><td></td><td>Bit 3 – PC Client Other Data Content</td><td>This bit indicates whether the Other Data field (when present) contains the 'event' value from the PC Client platform event log. If this value is set to 1, the Other Data field MUST contain information as defined by the PC Client specification. If this value is set to 0, it should contain self descriptive information so the recipient can process it.</td></tr>
<tr><td></td><td>Bit 4-7 - Reserved</td><td>This field MUST be set to 0 and MUST be ignored by compliant implementations.</td></tr>
<tr><td>PCR Number</td><td colspan="2">This field identifies the PCR number of the Integrity Measurement Log returned by the sender. This must be a single PCR value matching the PCR number requested by the PTS-IMV.</td></tr>
<tr><td>PCR Length</td><td colspan="2">This field indicates the length (in bits) of the PCR Before Value(s), PCR Stop Value(s) and Component Measurement for all entries included in this attribute. This length allows for PCR lengths other than 160 bits (fixed in TPM 1.2 and earlier). If there are no PCR Before and PCR Stop values included in this attribute, this value MUST be set to 0.</td></tr>
</table>

<table>
<tr><td>PCR Hash Algorithm</td><td>Hash algorithm (with only one bit selected from the set offered) used to take the measurement of the components described by all IML entries in this attribute. See section 3.8.5 for a description of the defined hash algorithms. If there are no PCR Before and PCR Stop values included in this attribute, this field MUST be set to 0.</td></tr>
<tr><td>PCR Transform</td><td>This field describes how the PTS’s hash result is stored into the PCR. If the hash value is the same size as the PCR (e.g. 20 octets on TPM 1.2) then no transform is required.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0 – No Transform</td><td>This field MUST be set to 0 if transform of the hash to fit into the PCR is unknown.</td></tr>
<tr><td>1 – Hash value matched PCR size</td><td>PTS’s hash result was the same size as the PCR size, so the result was directly applied without alteration. This might occur if the PTS used SHA-1 on a TPM 1.2 system.</td></tr>
<tr><td>2 – Hash value shorter than PCR size</td><td>PTS’s hash result was shorter than the size of the PCR, so the high order bits are padded with zero to enable the hash to fit into the PCR. Note this isn’t expected to be common but if a shorter hash results (e.g. 128 bits) was determined and needs to be extended into a long (e.g. 160 bit PCR) this padding might occur.</td></tr>
<tr><td>3- Hash value longer than PCR size</td><td>PTS’s hash result was longer than the size of the PCR size, so the result was truncated in order to fit into the PCR extend. This might occur if the PTS was using SHA-256 on a system with a 20 octet PCR size.</td></tr>
<tr><td>4-7 - Reserved</td><td>These values are reserved for future use and MUST NOT be used by senders compliant with this specification.</td></tr>
</table></td></tr>
<tr><td>Reserved</td><td>This field is reserved for future use and MUST be set to zero. Compliant implementations MUST ignore the contents of this field.</td></tr>
<tr><td>Number of IML Entries in this Batch</td><td>This field indicates the number of IML entries included in this response attribute. This allows the sender to group a bunch of IML entries together into a single attribute response. Note that a batch can be any number of entries and merely represents the number of entries included in this particular attribute. The sender may send additional entry batches in subsequent attributes immediately following this attribute or may send all the PCR’s entries in a single larger message.</td></tr>
</table>

IML Entry Date/Time	This field contains the date and time that the IML log entry was recorded (if known). The IML Entry Date/Time field’s date and time MUST be represented as an RFC 3339 [RFC3339] 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 with the above restrictions. Leap seconds are permitted and IMVs MUST support them. The IML Entry Date/Time string MUST NOT be NUL terminated or padded in any way. If the measurement date and/or time is not known, not applicable, or cannot be represented in this format, this field MUST contain ”0000-00-00T00:00:00Z” allowing this attribute to be fixed length. Note that this reserved value is not RFC 3339 compliant (zero month). This encoding produces an easy to read, parse and interpret string in YYYY-MM-DDTHH:MM:SSZ format that can precisely define a particular second in UTC time. For example, 9:05:00AM EST on January 19, 1995 can be represented as “1995-01-19T14:05:00Z”. The length of this field is always 20 octets.
Event Type	Defined in PC Client Implementation section 11.3.1. Future versions of this specification may define additional event types which aid in interpreting the other fields in this entry.
Functional Component Name	This field identifies the functional component (if known) of the entry associated with this IML entry. This field is four octets in length. See section 5 for a description of the functional naming. If the functional component name for an integrity measurement log entry cannot be determined (not included by the measurement agent) and this attribute includes functional component names, this field should be set to unknown state as per the functional component name enumeration.
PCR Before Value	This variable length value includes the PCR’s content before the TPM extend operation was performed. Having this value could help in reconstructing the ordering of sub(component) extends or to verify with the subsequent quote information. The PCR Before and Finish Value fields repeat until every PCR that was extended into during the measurement of this component is listed.
PCR After Value	This variable length value includes the PCR’s content after the TPM extend operation was performed. Having this value could help in reconstructing the ordering of sub(component) extends or to verify with the subsequent quote information.
Component Measurement	This variable length value includes the actual measurement of the component.

Other Data Length	This field indicates the number of bytes comprising the Other Data field. This field should be set to 0 if no Other Data is included in this IML entry.
Other Data	This field contains any additional data included with the IML entry. If the PC Client Other Data Content Flag bit is set in the Flags field, this field carries the PC Client platform defined 'event' information. It's envisioned that in the future, additional content bits will be defined to carry other types of IML entry information.

PTS-IMV SHOULD support reception and processing of this attribute, while the PTS-IMC SHOULD support sending this attribute. PTS-IMC supporting TPM-backed attestation evidence SHOULD support inclusion of the trusted platform's integrity measurement log messages with the appropriate PTS-based log messages. Other TNC architecture components besides IMCs MUST NOT support sending this attribute.

4 PTS Attestation Errors

This section contains a set of error codes and associated attribute contents for PTS-based attestation protocol errors. These errors are carried within the standard IF-M Error attribute (see IF-M specification for details). The format of the extensible IF-M Error attribute is as follows:

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Reserved   |         Error Code Vendor ID                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Error Code                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Error Information (Variable Length)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           . . . . . . .                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The highlighted portions of the above error code format indicate the portions that need to be set in order to convey PTS protocol related errors. For all error codes defined in this specification, the Error Code Vendor ID is set to the TCG SMI Private Enterprise Number (0x005597). Vendor specific PTS oriented errors SHOULD change the Error Code Vendor ID to the SMI of the vendor defining the error.

When a PTS error attribute is received, the recipient MUST NOT respond with an IF-M error, because this could result in an infinite loop of errors. Instead, the recipient MAY log the error, modify its behavior to attempt to avoid the error (attempting to avoid loops or long strings of errors), ignore the error, terminate the assessment, or take other action as appropriate (as long as it is consistent with the requirements of this specification).

4.1 PTS Error Code Values

This section defines the Error Code enumeration used for each type of error defined within the PTS attestation protocol. These values are placed in the Error Code field of the IF-M Error attribute. The following table briefly describes each. Later subsections provide detailed specifications for the Error Information associated with each Error Code.

Error Friendly Name	**TCG Standard Error Code Value**	**Description**
Reserved Error	0	This value is reserved and MUST NOT be sent in an IF-M Error attribute (TCG_PTS_RESERVED_ERROR).
Hash Algorithm Not Supported	1	This value indicates the sender found the proposed DH-PN or File Hash Algorithm unacceptable or unsupported (TCG_PTS_HASH_ALG_NOT_SUPPORTED).
Invalid Path	2	This value indicates that the sender requested information about a filesystem path that is not valid such as a not fully qualified path (TCG_PTS_INVALID_PATH).
File Not Found	3	This value indicates that the sender requested a file that was not found on the endpoint (TCG_PTS_FILE_NOT_FOUND).

Registry Not Supported	4	This value indicates that the sender requested a registry entry from an endpoint without registry support (TCG_PTS_REG_NOT_SUPPORTED).
Registry Key Not Found	5	This value indicates that the sender requested a registry key that was not present on the endpoint (TCG_PTS_REG_KEY_NOT_FOUND).
D-H Group Not Supported	6	This value indicates that the sender does not support any of the offering D-H groups during the DH-PN (TCG_PTS_DH_GRPS_NOT_SUPPORTED).
DH-PN Nonce Not Acceptable	7	This value indicates the sender found the proposed DH-PN Nonce size to be unacceptable (TCG_PTS_BAD_NONCE_LENGTH).
Invalid Functional Name Family	8	This value indicates the sender received an attribute containing an invalid functional name family (TCG_PTS_INVALID_NAME_FAM).
TPM Version Information Unavailable	9	This value indicates the sender was unable or unwilling to retrieve the TPM version information requested (TCG_PTS_TPM_VERS_NOT_SUPPORTED).
Invalid File Pathname Delimiter	10	This value indicates an attribute was received with an invalid pathname delimiter (TCG_PTS_INVALID_DELIMITER).
PTS Operation Not Supported	11	This value indicates that the PTS operation requested by the PTS-IMV is not supported on this particular endpoint (TCG_PTS_OPERATION_NOT_SUPPORTED).
Unable to Update Reference Manifest	12	This value indicates that the sender was unwilling or unable to update the provided Reference Manifest (TCG_PTS_RM_ERROR).
Unable to Perform Local Validation	13	The PTS-IMV requested local verification of a component, but the PTS and other measurement agents are unable to perform local validation or were unable to validate the particular component requested.
Unable to Collect Current Evidence	14	The PTS-IMV requested current evidence be collected for a component that is currently executing and the PTS and other measurement agents were unable to perform such a measurement.
Unable to Determine	15	The PTS-IMV requested the transitive trust chain to a component and the PTS was unable

Transitive Trust Chain		to determine the appropriate transitive trust chain.
Unable to Determine PCR	16	The PTS-IMV requested PCR information about a particular component, but the PTS was unable to determine the appropriate PCR values (e.g. the component wasn't measured).

4.2 PTS Error Information Values

The following subsections show the supplemental Error Information that MUST be included in the Error Information field for each TCG standard IF-M Error attribute. This information frequently involves sending a copy of the original IF-M message, so the recipient can determine which message caused the error and the messages content.

4.2.1 Errors Including Original Attribute

In many cases the appropriate response in an Error attribute is to return the first 1024 bytes of the received attribute that caused the error. It's envisioned that this copy can be used by recipients along with the error code to determine what went wrong. With the exception of the Reserved Error above, all TCG standard error codes defined in section 4.1 that do not have a specific Error Information content description in the remainder of this section MUST return a copy of the first 1024 bytes of the IF-M attribute that caused the error.

4.2.2 Hash Algorithm Not Supported Information

This section describes the Error Information field's content for a Hash Algorithm Not Supported error code. Rather than returning the entire request attribute, this error code returns the set of hash algorithms supported by the sender in order to avoid future errors.

The following diagram shows the contents of the Error Information field for this IF-M Error attribute:

```
                    1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Reserved           |      Hash Algorithm Set       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by compliant implementations.
Hash Algorithm Set	Bit field indicating the sender's set of supported hash algorithms. See section 3.8.5 for a description of the defined hash algorithms and their representation in this field.

4.2.3 Registry Not Supported Information

When the Registry Not Supported error code is returned, the error code itself is sufficient information for the sender to understand why an error occurred. Therefore, when sending a Registry Not Supported error code, the Error Information MUST be empty (zero length). If an entity receives this IF-M attribute error code and the attribute contains an Error Information, the receiver MUST ignore the Error Information.

4.2.4 D-H Group Not Supported Information

This section describes the Error Information field's content for a D-H Group Not Supported error code. Rather than returning the entire request attribute, this error code returns the set of Diffie-Hellman groups supported by the sender in order to avoid future errors.

The following diagram shows the contents of the Error Information field for this IF-M Error attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Reserved           |         D-H Group Set         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by compliant implementations.
D-H Group Set	Bit field indicating the initiator's supported D-H groups. See section 3.8.6 for descriptions of the D-H groups and their representation in this field.

4.2.5 DH-PN Nonce Not Acceptable Information

This section describes the Error Information field's content for a DH-PN Nonce Not Acceptable error code. Rather than returning the entire request attribute, this error code returns the range of DH-PN nonce sizes that would be acceptable by the sender in order to avoid future errors.

The following diagram shows the contents of the Error Information field for this IF-M Error attribute:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Min Nonce Len         |         Max Nonce Len         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
Min Nonce Len	Minimum acceptable length for the nonce in bytes. This value should be set to 0 if there is no minimum required.
Max Nonce Len	Maximum acceptable length for the nonce in bytes. This value should be set to 2^16-1 (all ones) if there is no maximum required.

5 PTS Protocol Component Functional Name

This section describes the Component Functional Name namespace supported by the PTS protocol.

5.1 Component Functional Name Structure

The Component Functional Name has the following structure:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Component Functional Name Vendor ID     |Fam| Qualifier |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Component Functional Name                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Component Functional Name Vendor ID</td><td>The SMI Private Enterprise Number of the vendor who controls the name space for the Fam, Qualifier and Component Functional Name fields. This field enables vendor and standards based attributes to be used without potential collisions since TCG will have an isolated name space from each vendor.

Any TCG standard functional component names MUST use the TCG SMI Private Enterprise Number (0x005597) in this field. Vendor-defined functional names MUST use the SMI Private Enterprise Number of the vendor who defined the functional names.

When the Component Functional Name Vendor ID is zero (IETF name space), the Component Functional Name field will include the PA sub-type value as defined in the PB protocol.</td></tr>
<tr><td>Fam (Functional Name Encoding Family)</td><td>This field indicates the naming family for the Component Functional Name. This family indicates how the Component Functional Name is encoded so MUST be consulted prior to evaluating the contents of the Component Functional Name field.

<table>
<tr><th>Bit Encoding</th><th>Description</th></tr>
<tr><td>Bits 00 – Binary Enumeration</td><td>This family is the fixed length 64 bit field containing the binary enumeration of the functional name. For version 1.0 of this specification, this is the only standard expression of the functional name.</td></tr>
<tr><td>Bits 01-11 - Reserved</td><td>These bit values are reserved for future use and MUST not be sent by version 1.0 compliant implementations.   If new naming families require variable length or >64 bits then some additional new attributes will be required since some attributes are designed to work with the binary enumeration.</td></tr>
</table>

Implementations compliant with the 1.0 specification MUST only</td></tr>
</table>

	use bits 00 in the Fam field and MUST generate an IF-M Error attribute indicating TCG_PTS_INVALID_NAME_FAM when receiving a name using other Fam values within the TCG standard name space.
Qualifier	This field contains the qualifier on the Component Functional Name field expressing the category of component named. This allows for some components to have multiple sub-components with the same functional name (e.g. network service with a kernel component and user-level component with the same Component Functional Name). The TCG standard enumeration for the Qualifier field is shown in section 5.2. The name space of the qualifier field is indicated by the Component Functional Name Vendor ID's name space allowing vendors to define their own qualifiers outside of the TNC standard name space.
Component Functional Name	This field contains an identifier representing a particular functional component. The name is functional as it describes the component by its function (e.g. firewall module) as opposed to other forms of component naming such as the PTS Component ID which could include version and vendor information. The Fam field indicates the syntax and semantic of how this field identifies the functional component. The TCG standard values for this field are enumerated in section 5.3. Version 1.0 of this specification defines a single mandatory to support Fam encoding which is a binary enumeration. When the Component Functional Name Vendor ID is zero (IETF name space), the Component Functional Name field MUST include a value from the PA sub-type enumeration as defined in the PB protocol.

5.2 Component Functional Name's Qualifier Field

This section describes the syntax, semantics and enumeration of the Qualifier field in the binary enumeration naming family (Fam is 00). The qualifier field could be a useful hint to the PTS about where to find (or take) measurements of the requested component or to provide for wildcarding. In order to support both semantics, two special values for the entire Qualifier field are reserved:

Reserved Qualifier Value	Description
Bits 000000 (Unknown)	This value indicates sender does not know what qualifier is appropriate for the component so this aspect should be ignored by the recipient. This value should be used by the PTS when it is reporting information about a functional component but does not know the appropriate qualifier.
Bits 111111 (Wildcard)	This value indicates that sender wishes to match all functional components that correspond to the other fields of the functional component name. This value should only be used in queries by the PTS-IMV.

When the qualifier doesn’t match one of the reserved values, it contains the following structure:

```
 0 1 2 3 4 5
+-+-+-+-+-+-+
|K|S| Type  |
+-+-+-+-+-+-+
```

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>K (Kernel) Flag</td><td>This flag indicates if the Component Functional Name refers to a kernel-level or user-level software. For example, this could refer to a device driver in the kernel of the attested system.

If K flag is 1 – Component is a kernel module or driver
If K flag is 0 – Component is a non-kernel feature

NOTE: If the entire Qualifier field is set to 0 (including the K Flag) this indicates the sender does not know the category of component requested (as opposed to stating it’s non-kernel).</td></tr>
<tr><td>S (Sub-Component) Flag</td><td>This flag indicates if the Component Functional Name is a nested sub-component of a prior Component Functional Name. The S flag MUST only be present on the non-first element of a hierarchical functional name.

If S flag is 1 – Component is a sub-component of the prior name
If S flag is 0 – Component is not a sub-component.

When a functional name is expressed in an attribute, the S flag MUST be set to 0 for the first name in the component hierarchy. If a sub-component is being specified the S flag MUST be set to 1 and the sub-component MUST follow the identification of the parent component (which might also be a sub-component).

For example, if the IMC portion of the TNC Client was being specified, the functional naming might be:
<K=0,S=0,Type=0x05 (Networking),Value=X (TNC Client)>
<K=0,S=1,Type=0x06 (Library), Value=Y (AV IMC)>

NOTE: If the entire Qualifier field is set to 0 (including the S Flag) this indicates the sender does not know the category of component requested (as opposed to stating it is not a sub-component).</td></tr>
<tr><td>Type</td><td>This field indicates additional information about the category of component being described in the Component Functional Type. The following table shows the names and values of each enumeration of the Type sub-field:

<table><tr><th>Category Name</th><th>TCG Standard Type Value</th><th>Description</th></tr><tr><td>Unknown</td><td>0x0</td><td>Component described</td></tr></table></td></tr>
</table>

				is of unknown category. This value should be accompanied by a 0 value in the K and S Flags. Use of Unknown category with a 1 value in K or S Flags MUST NOT be sent.
		Trusted Platform	0x1	Component described is a part of the transitive trust chain of the system or associated with a trusted root.
		Operating System	0x2	Component is a portion of the operating system
		Graphical User Interface	0x3	Component is part of the windowing environment.
		Application	0x4	Component is an application program. This value should not be used when the K flag is set to 1.
		Networking	0x5	Component is associated with the networking stack on the system.
		Library	0x6	Component is a library sub-component of another component.
		TNC Defined Component	0x7	The Component Functional Name contains a TNC defined component identifier. This identifier is the IF-M Subtype as defined in the IF-TNCCS specification.
		Reserved for future use	0x8 – 0xE	Reserved for future use so MUST NOT be used in an attestation.
		All matching components	0xF	Wildcard matching all components of

				particular type. The K and S Flags are interpreted with this value allowing for matching of ‘all kernel modules’.	

5.3 Component Function Name Binary Enumeration

The following table enumerates the TCG standard Component Functional Names.

Component Functional Friendly Name	TCG Standard Component Functional Name Value	Description
Ignore	0x0000	This value indicates that the Qualifier field identifies the desired components. This value MUST only be used when the Qualifier field is 0x0 or 0xF. The result is that this field is always ignored when it contains a zero value.
CRTM	0x0001	Core Root of Trust for Measurement responsible for measuring the initial code segment loaded normally from BIOS.
BIOS	0x0002	Basic I/O System which handles interrupt handling for input and output (in addition to other features).
Platform Extensions	0x0003	ROMs containing firmware code frequently present on motherboard
Motherboard Firmware	0x0004	ROMs containing firmware besides those included in the BIOS and Platform Extensions
Initial Program Loader	0x0005	Code that executes immediately following the platform boot to load software to start the operating system boot
Options ROMs	0x0006	Code stored in ROMs contained on the non-platform adapters.
Reserved for future use	0x0007 – 0x000E	Reserved for future use so MUST NOT be used in an attestation.

6 Security Considerations

This section discusses the major types of potential security threats relevant to the PTS protocol binding to IF-M message protocol and summarizes the expected security protections that should be offered by either the PTS protocol or the underlying IF-M protocol. Ultimately, the deployer decides whether each particular security protection is necessary for a particular deployment environment, so the expected security protections discussed in this section highlight the need for PTS for IF-M protocol implementations to be capable of offering the security property.

6.1 Trust Relationships

Initially, let's examine the envisioned trust relationships between the major components of the PTS Protocol architecture. Some implementations may choose to offer strong security checking (e.g. encryption services between PTS-IMC and PTS-IMV) in order to reduce the amount of assumed trust in a deployment and possibly increase the product's suitability for different usages.

6.1.1 PTS-IMC

The PTS-IMC is trusted by PTS-IMV to:

- Not tamper with and report upon attestation information it collects or receives from the PTS (or other measurement agent) consistent with local security and privacy policies
- Interact with the local PTS conveying the attestation requests and evidence
- Accurately report current information associated with the type of component for the IF-M message
- Not act maliciously including not launching denial of service attacks against the PTS-IMV or PTS

Note that the use of TPM based attestation evidence allows for PTS-IMV detection of modification of the reporting by the PTS-IMC. Therefore, the amount of blind trust required is reduced, since the threat turns into a denial of service attack as opposed to a more significant attack possible when the TPM is not used.

6.1.2 PTS-IMV

The PTS-IMV is trusted by PTS-IMC to:

- Only request information necessary to assess the security state of the endpoint
- Make assessment decisions based on deployer-defined integrity policies
- Return the correct IMV Action Recommendation to the attesting component (e.g. TNCS) and when necessary the PTS-IMC
- Manage collected information consistent with its data retention and privacy policies
- Not act maliciously to TNCS and IMC including not launching denial of service attacks against their operation

6.1.3 PTS

The PTS is trusted by the PTS-IMC to:

- Accurately and reliably perform the requested attestation evidence collection
- Follow the requested reporting scheme (XML-based or TLV-based)
- Not to falsify the attestation evidence
- Report recently collected attestation evidence for component (when possible)
- Not act maliciously to the PTS-IMC including launching of denial of services attacks

6.1.4 IF-M Message Processing

The PTS-IMC and PTS-IMV trust the underlying IF-M protocol stack to:

- Provide a reliable transport for IF-M messages
- Deliver PTS protocol messages only to those IMCs and IMVs that have registered for them
- Not disclose any provided attributes to parties outside of the integrity assessment
- Not act maliciously to drop, duplicate or flood registered IMC and IMV with unnecessary messages
- Not to observe, fabricate or alter the contents of an IF-M message (this trust could be minimized with an IF-M security protocol)
- Properly expose the identity of the peer PTS-IMC or PTS-IMV for use by IMC to make policy decisions

6.2 Security Threats and Countermeasures

Beyond the trusted relationships assumed in section 6.1, the PTS and IF-M protocols faces a number of potential security attacks that could require targeted security countermeasures.

Generally, the PTS Protocol operating over the TNC protocol stack (specifically over IF-M) faces a number of threats that are described within the appropriate TNC protocol specification’s security considerations. The IF-M specification currently does not offer strong security protections of the individual messages between the IMC (PTS-IMC) and IMV (PTS-IMV) instead relying on transport layer security in IF-T or its underlying carrier protocol (e.g. TLS) which protect against MITM attacks for data in transit over the network. This section will focus on threats between the PTS and the IF-M protocol (PTS-IMC). These attacks exist locally on the endpoint (between the PTS and PTS-IMC) in order to disrupt, replay or alter the assessment.

6.2.1 Attestation Evidence Theft

Malware present on the endpoint may be capable of observing the operation of the PTS (or appropriate measurement agent) and making a copy of the evidence for future use. The malware could analyze the information for well known vulnerabilities and share this information with a botnet or other network based vulnerability collectors. Because the malware is already running on the endpoint, many of the attributes it could compute by itself if it possessed knowledge of where the appropriate metadata or evidence was stored, so the attestation evidence theft attack is merely a convenience. In some implementation architectures, the malware might only have unprivileged access to the platform, so products implementing the PTS SHOULD strive to limit the attack surface where attributes could be stolen. For example, a PTS could be run in a different VM that is capable of obtaining the necessary posture for another VM being assessed (and including the malware). Since the PTS is outside the execution environment of the malware, its address space and posture collection data storage would be protected from unauthorized access. A similar approach would be to run all of most of the PTS in kernel space and hide the posture accumulated within the kernel (again to potentially defeat application space malware). Such protections would raise the bar enough where malware writers might feel the need to accumulate the posture information on their own causing the malware to be larger and more CPU intensive which could increase the likelihood of it being detected.

6.2.2 Message Fabrication

Malware present on the endpoint could attack the PTS, PTS-IMC or the communications path between them to introduce false posture information in order to hide its existence and/or to alter the assessment result. Network Access Control (NAC) systems in general do not contain countermeasures against local malware altering or falsifying attestation evidence. However with the use of the trusted platform and specifically the capabilities of the TPM, endpoints can provide attestation evidence that the malware is unable to alter without detection by the remote challenger.

In a NAC environment (e.g. TNC), the IMV requests measurements of the endpoint and compares the measurements against policy to determine if the endpoint is trustworthy. Clearly malware on an infected endpoint that wasn't aware of the NAC assessments would be detected during admission to the network and NAC remediation could eradicate the malware. Therefore, it's envisioned that over time malware will evolve to detect and attack NAC agents on the endpoint. The PTS protocol leveraging the TPM allows for the creation of attestation evidence that cannot be undetectably modified by malware, thus preventing it from hiding it existence. As discussed above, this occurs by the PTS-IMV requesting TPM quotes of the boot components to ensure a safe base trusted platform is started. Next, the PTS-IMV can request information about the PTS and its run-time dependencies and request another TPM quote confirming this information is authentic (unmodified). At this point the PTS can be trusted to properly operate and report about the endpoint, so the PTS-IMV can attest the PTS-IMC to PTS communication path and the TNC client components backed by another TPM quote. At this point we have established a trustworthy path to send the information, so fewer TPM quote operations are necessary. The smaller the attack surface is against the PTS, PTS-IMC, and TNC client, the fewer components that will require a TPM backed attestation evidence, thus making verification less complex.

6.2.3 Man-in-the-Middle Attack

A number of attacks could be attempted against the TNC protocol stack including the Asokan attack discussed in the IF-T Binding to Tunneled EAP Methods. This attack requires active man-in-the-middle (MitM) malware on the endpoint to perform the TNC handshakes and replay posture obtained from another clean system assessment. The attributes discussed in this specification, carried by IF-M, require MitM protection by the TNC protocol stack. However, the TNC transport protocol IF-T may offer a countermeasure by establishing a shared, per-session secret between the parties that needs to be included within any TPM quote operations performed during the assessment (as proof of knowledge bound to the TPM quote). Since a man-in-the-middle attack could potentially happen at any time, the PTS Protocol described in this specification (see section 3.8) also allows for the per-assessment secret to be created (if not done by IF-T) or be replaced with a fresher version. Both the PTS Protocol and IF-T address this attack by establishing a shared secret that is included in the TPM quote (in ExternalData) which links the TPM PCR values to the shared secret. In order for this to be effective, it is important that client system not offer another protocol service where a remote party can request a quote and provide the ExternalData or this would allow a MitM could use this to have the clean system create TPM quotes responses for it to use on other assessments. Having both parties (remote challenger and client) provide entropy into the creation of the ExternalData for a TPM Quote provides the MitM protection (see discussion of Asokan attack in 5.4.5 of the IF-T Binding to Tunneled EAP Methods specification).

6.2.4 Attribute Insertion

Similar to the attribute modification attacks, endpoint malware wishing to include one or more PTS protocol attributes carried within IF-M inside a valid assessment may be able to insert the attributes without detection by the recipient. Even if authentication of the parties is present during the IF-T protocol exchange, if no per-message and per-session integrity protection is present an attacker can add information to the assessment possibly causing incorrect assessment results. For example, an attacker could prepend PTS protocol attributes to the front of an IF-M message to cause an assessment to succeed even for a non-compliant endpoint if it knew that the recipient ignored repeated PTS protocol attributes found later in a message. Similarly if an IMC or IMV always generated an error if it saw unexpected PTS protocol attributes, the attacker could cause failures and denial of service by adding attributes or messages to an exchange. Therefore, it is recommended that the PTS to PTS-IMC protocol include a feature to detect alteration of the attribute flow between them.

6.2.5 Denial of Service

A variety of types of denial of service attacks are possible against the PTS-IMC and PTS software when faced with malicious privileged code running on the endpoint. If the PTS attribute exchange is left unprotected along the communication path between the PTS-IMC and PTS, then malware could alter, corrupt or destroy important protocol attributes, thus causing the assessment or remediation to fail. Similar local attacks by privileged local malware could prevent portions of the PTS and PTS-IMC to not properly operate (e.g. causing the PTS to repeatedly crash). If the PTS is unable to operate the endpoint might not be able to pass a network assessment. As a countermeasure to these types of attacks, endpoints should leverage the local trusted platform to employ local verification (in addition to measurement) of software prior to execution and to run anti-malware scanning of disk and memory.

6.2.6 Alteration of Integrity Measurement Log

The attestation protocol leverages the Integrity Measurement Log that is created by the PTS and potentially other measurement agents on the platform. The platform integrity measurement log isn't required to be integrity protected, so could be subject to alteration by malware. Similarly, the PTS's equivalent integrity measurement log may also be subject to modification by malware. Therefore, the PTS SHOULD include provisions to isolate and protect or at least detect unexpected alterations. The more trustworthy the measurement log is on a platform, the more reliable the information is to the remote challenger as part of the attestation process. Because the integrity log provides granular information about what was measured, it could be very useful for a remote challenger when assessing portions of the platform that could impact its trustworthiness for a particular function.

7 Privacy Considerations

The PTS Protocol Binding to IF-M protocol is designed to allow for controlled disclosure of security relevant information about an endpoint specifically for the purpose of enabling an assessment of the endpoint's compliance with network policy leveraging the underlying IF-M protocol feature set. The purpose of this protocol is to provide PTS and TPM rooted attestation evidence about the state of the protective mechanisms on the endpoint in order for the PTS-IMVs and TNCS to determine whether the endpoint is up to date and thus having the best chance of being resilient in the face of malware threats. One risk associated with providing visibility into the contents of an endpoint is the increased chance for exposure of privacy sensitive information without the consent of the user.

While this protocol does provide the PTS-IMV the ability to request specific information about the endpoint, the protocol is not open ended, bounding the PTS-IMV to only query specific information (attributes) about specific security features (component types) of the endpoint. Discretionary components used by the user to create or view content are not on the list as they are more likely to have access to privacy sensitive information. Similarly, PTS Protocol messages contain a set of specific information that describes the requested component. This combination of limited set of security related components with non-user specific attributes greatly reduces the risk of exposure of privacy sensitive information. Vendors that choose to define additional component types and/or attributes within their name space are encouraged to provide similar constraints.

Even with the bounding of standard information, it is possible that individuals might wish to share less information with different networks they wish to access. For example, a user may wish to share more information when connecting or being re-assessed by the user's employer network than he does when connecting to the local coffee shop wireless network. While these situations do not impact the protocol itself, they do suggest that PTS-IMC implementations should consider supporting a privacy filter allowing the user and/or system owner to restrict access to certain attributes based upon the target network. The underlying IF-T protocol authenticates the network's TNCS at the start of an assessment, so identity could be made available to the PTS-IMC so per-network privacy filtering is possible. Network owners should make available a list of the attributes they require to perform an assessment and any privacy policy they enforce when handling the data. Users wishing to use a more restricted privacy filter on the endpoint may risk not being able to pass an assessment and thus not gain access to the requested network or resource.

8 References

8.1 Normative References

[IF-M] Trusted Computing Group, "TNC Architecture for Interoperability", http://www.trustedcomputinggroup.org/resources/tnc_ifm_tlv_binding_specification, January 2010.

[IF-PTS] Trusted Computing Group, "TCG Infrastructure Working Group Platform Trust Services Interface Specification (IF-PTS)", http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_platform_trust_services_interface_specification_ifpts_version_10, November 2006.

[KEYWORDS] S. Bradner, "Keywords for use in RFCs to Indicate Requirement Levels", http://www.ietf.org/rfc/rfc2119.txt, IETF, March 1997.

[RFC2279] F. Yergeau, "UTF-8, a transformation format of ISO 10646", http://www.ietf.org/rfc/rfc2279.txt, IETF, January 1998.

[RFC3339] G. Klyne, C. Newman, "Date and Time on the Internet: Timestamps", **http://www.ietf.org/rfc/rfc3339.txt**, IETF, July 2002.

[RFC3986] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", **http://www.ietf.org/rfc/rfc3986.txt**, IETF, January 2005.

[RFC4306] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol", http://tools.ietf.org/rfc/rfc4306.txt, IETF, December 2005.

[RM] Trusted Computing Group, "TCG Infrastructure Working Group Reference Manifest (RM) Schema", http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_reference_manifest_rm_schema_specification_version_10, November 2006.

[TPM1.2] Trusted Computing Group, "TPM Main Part 2 TPM Structures", http://www.trustedcomputinggroup.org/files/resource_files/E14876A3-1A4B-B294-D086297A1ED38F96/mainP2Structrev103.pdf, July 2007.

8.2 Informative References

[IF-IWG-ARCH] Trusted Computing Group, "Architecture for Part II - Integrity Management", **http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_architecture_part_ii__integrity_management_version_10**, November 2006.

[IF-TNC-ARCH] Trusted Computing Group, "TNC Architecture for Interoperability", **http://www.trustedcomputinggroup.org/resources/tnc_architecture_for_interoperability_specification**, May 2007.

[RFC4753] D. Fu, "ECP Groups for IKE and IKEv2", http://www.ietf.org/rfc/rfc3526.txt, January, 2007.

[IKE-MODP] T. Kivinen, M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", http://tools.ietf.org/rfc/rfc4753.txt.

[INT-REPORT] Trusted Computing Group, "Core Integrity Schema", **http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_core_integrity_schema_specification_version_101**, November 2006.

[VERIFY-RESULT] Trusted Computing Group, "Verification Result Schema", http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_verificat ion result schema specification version 1 0, May 2007.