TCG Infrastructure Working Group Architecture Part II - Integrity Management

Specification Version 1.0
Revision 1.0
17 November 2006
FINAL

Contacts:
THardjono@signacert.com **(Editor, co-chair)**
Ned.Smith@intel.com **(co-chair)**

TCG

Public

Copyright © TCG 2005-2006

Copyright © 2005-2006 Trusted Computing Group, Incorporated.

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.

No license, express or implied, by estoppel or otherwise, to any TCG or TCG member intellectual property rights is granted herein.

Except that a license is hereby granted by TCG to copy and reproduce this specification for internal use only.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on specification licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

Acknowledgements

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

Geoffrey	Strongin	AMD
Malcolm	Duncan	CESG
Sigi	Guergens	Fraunhofer Institute
Mark	Redman	Freescale Semiconductor
Kazuaki	Nimura	Fujitsu Limited
Seiki	Shibata	Fujitsu Limited
Ito	Hidenobu	Fujitsu Limited
Boris	Balacheff	Hewlett-Packard
Graeme	Proudler	Hewlett-Packard
Tadaoki	Uesugi	Hitachi, Ltd.
Diana	Arroyo	IBM
Lee	Terrell	IBM
Markus	Gueller	Infineon
Johann	Schoetz	Infineon
Ned	Smith (Co-Chair)	Intel Corporation
David	Grawrock	Intel Corporation
Monty	Wiseman	Intel Corporation
Ravi	Sahita	Intel Corporation
Sandy	Roddy	NSA
Michael	Willett	Seagate Technology
Robert	Thibadeau	Seagate Technology
Manuel	Offenberg	Seagate Technology
Will	Ingersoll	Sun Microsystems, Inc.
Jeff	Nisewanger	Sun Microsystems, Inc.
Paul	Sangster	Symantec Corporation
Thomas	Hardjono (Editor, Co-Chair)	SignaCert, Inc.
Wyatt	Starnes	SignaCert, Inc.
Ron	Forrester	SignaCert, Inc.
David	Bleckmann	SignaCert, Inc.
Len	Veil	Wave Systems
Greg	Kazmierczak	Wave Systems
Scott	Cochrane	Wave Systems

IWG Document Roadmap

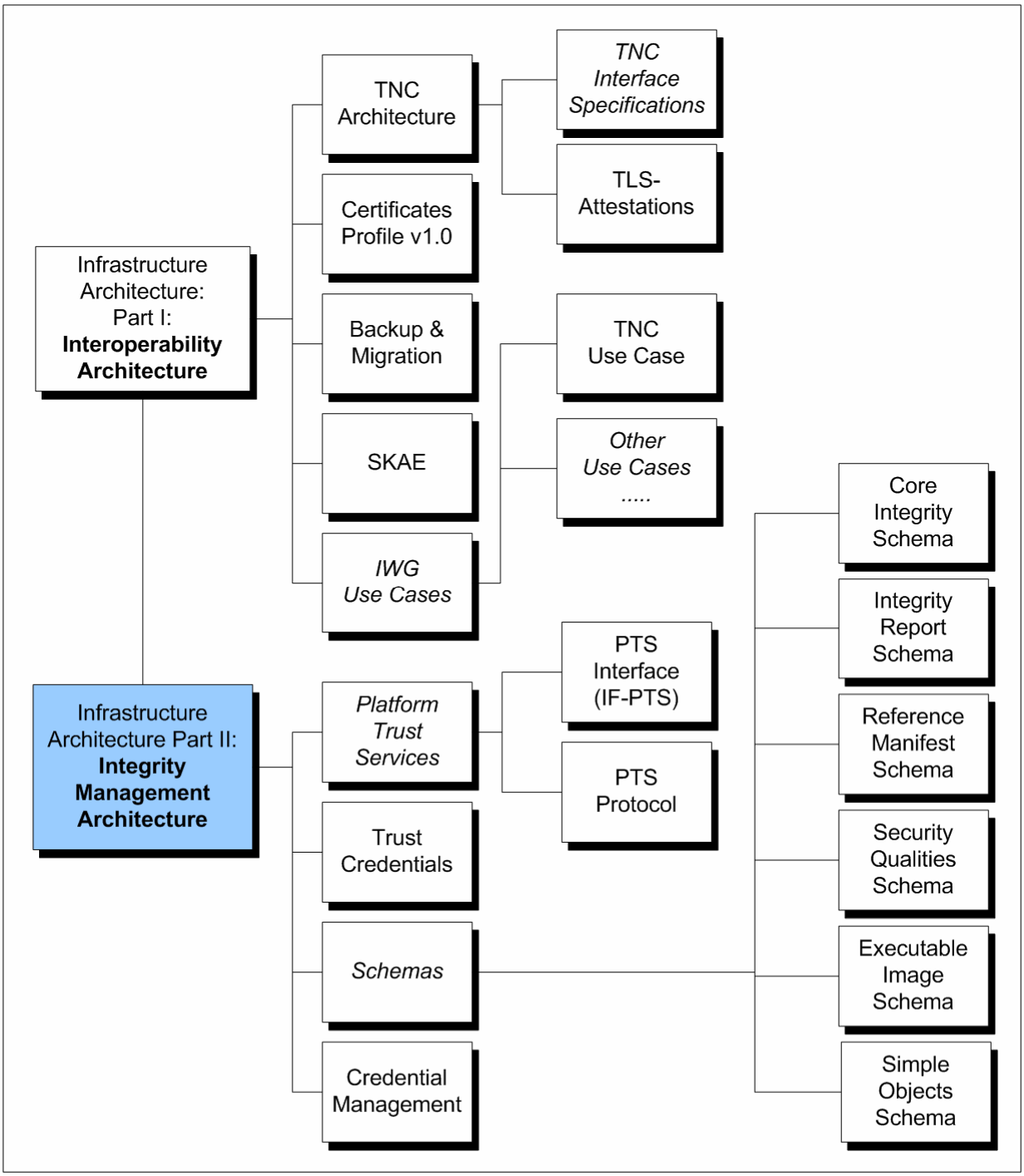


Table of Contents

1 Scope and Audience 7
2 Introduction: The TCG Integrity Management Model 8
2.1 Overview of the Integrity Management Model 8
2.2 What is Needed for Trust 9
2.3 What to capture: Transitive Trust Chains 9
2.4 Relationship of IWG Documents for Integrity Management 11
3 Basic Model for Integrity Measurement and Reporting 13
3.1 General Phases in the Integrity Management Model 14
3.2 Runtime Measurements & Reference Measurements 16
3.3 Integrity Reports and Reference Manifests 18
3.4 Comparing Snapshot and Reference Manifest Structures 19
3.5 Platform Trust Services 21
4 The IMM Processes 22
4.1 Creation (Harvesting) process 22
4.2 Collection and Publication Processes 23
4.3 Policy Authoring and Verification Processes 23
5 Runtime Measurements 25
5.1 Measurement Log Decomposition: Levels of Logs 26
5.2 The notion of Snapshots 26
5.3 An Interoperable Log Structure: Issues & Requirements 27
5.3.1 Platform Specific Integrity Values 27
5.3.2 Application Specific Integrity Values 27
5.3.3 Integrity Log Metadata 28
5.4 Measurement & Verification Interoperability 28
6 Reference Measurements 30
6.1 What is in a Reference Manifest 30
6.2 The Reference Manifest Database 30
6.3 The Role of Reference Measurements in the Supply-Chain 32
7 The Integrity Schema 35
7.1 Motivation for using XML Schemas 35
7.2 The Core Schema 36
7.3 The Integrity Report Schema 39
7.4 The Reference Manifest Schema 39
7.5 The SecurityQualities Schema 40
7.6 Executable Image Schema 41
8 Platform Trust Services 42
8.1 Basic Architecture of PTS 43
8.2 Role of PTS in the Transitive Chain Establishment 43
8.3 Role of PTS in generating Integrity Reports 44
8.4 Protocol between PTS-IMC and PTS-IMV 46
9 Policy Setting & Enforcement using Measurements 47
9.1 Role of Measurements in Policy Authoring 47
9.2 Relationship with Configuration Management & CMDB 48
9.3 Using Runtime Measurements to verify configuration 50
9.4 The TCG Policy Profile 51
10 Trust Credentials and RMs: Conceptual Model 52
10.1 Motivations 52

10.2 The Trust Credential Usage Model 54
10.3 Evolving Trust Credentials from Current TCG Certificates 54
10.3.1 Summary of Fields in TCG Credentials 1.0 55
10.3.2 Reduced EK-Certificates (Abstract) 56
10.3.3 Reduced Platform Certificates (Abstract) 57
10.3.4 Reduced AIK-Certificates (Abstract) 58
10.4 Trust Credentials: The Conceptual Model 59
11 Use Cases for the Integrity Management Model 61
11.1 Platform Authentication 61
11.2 Trusted Boot 62
11.3 Secure Boot 62
11.4 Trusted Network Connect (TNC) 63
12 Glossary 65
13 References 67

1 Scope and Audience

The TCG Infrastructure Working Group (IWG) has defined a “reference” architecture aimed at existing and new infrastructure technologies having a goal of improving interoperability among systems containing TCG technology.

Architects, designers, developers and technologists who are interested in the development, deployment and interoperation of trusted systems may find this document helpful in providing both abstract and implementation specific insights for achieving interoperation between TCG-based systems.

2 Introduction: The TCG Integrity Management Model

The broad purpose of the current document is (a) to develop further the concept of Integrity Management that has been introduced in the IWG Architecture Part I [1], and (b) to identify and/or define infrastructure services that implement the TCG Integrity Management model.

The reader is directed first to the IWG Architecture Part I [1] in order to obtain background information and context regarding the topics and issues discussed in the current document.

2.1 Overview of the Integrity Management Model

One of the key aspects of Trusted Computing is the establishment of trust – both social trust and technical trust – in platforms that possess a TPM and trusted computing features. An important part of establishing this trust is to know the origins, condition and history of components that make-up a Trusted Platform.

An important realization in the field of Trusted Computing is that the manufacturing and distribution infrastructure of today's computing devices are inadequate to assist in the establishment of trust by users in Trusted Platforms. As such, these computing industry manufacturing and distribution infrastructures need to be augmented in such a way that it allows the conveying of relevant information – *integrity information* – from one end of the supply chain (manufacturing) to the other (consumer).

Trusted Computing itself is built upon numerous processes, beginning from processes in the design and manufacture of platform components, through the processes within the supply and distribution chain, to processes used within IT organizations at the point of deployment of the platforms. If one views the creation of a Trusted Platform as clear sequence of processes that influence the trustworthiness of each component – as they are brought together to make the platform – it is evident that *process integrity* represents a crucial part of the notion of trust and even a pre-condition for the establishment of trust in the components of a Trusted Platform and of the platform as a whole.

Thus, one of the key objectives of the Trusted Computing Infrastructure is to bridge the platform manufacturing configuration management process and the IT configuration management process. Such an architecture will allow for component information – that is crucial to establish the integrity status of that component – to flow from the manufacturing configuration process at one end the spectrum to the IT configuration management process at the other end of the spectrum. Since components produced by manufacturers always change and improve, in order for the architecture to be practical and useful it is important that such changes are controllable and recorded. That is, it is highly desirable that *controlled mutability* be achieved in the architecture that bridges these two ends of the spectrum.

The approach adopted here to realize controlled mutability is to develop an *Integrity Management Model* (IMM) that defines the *transition points* of integrity information as it flows through the supply-chain, and to define *quality metrics* for each of those transition points. At each transition point in the integrity information flow the trustworthiness of the component in question (at that stage in the supply chain) can then be evaluated according to the metrics specific to the domain of use, and for a *trust score* to be computed as a result of the evaluation.

The aim of the current document, in the first instance, is to define an architecture for integrity management for Trusted Computing as a continuation of the Architecture Part I document [1]. This *Integrity Management Model* (IMM) brings together important concepts and constructs from [1]. These include the notion of trust and provenance of components of a trusted platform, the notion of trust scores and how the play a role in a Verifier's evaluation of a platform, the relevant XML schemas (see [3]) and the general use of credentials backed by supporting integrity information.

Secondly, the aim of the current architecture document is to identify and/ define infrastructure services that implement and support the Integrity Management Model. We believe that providing

a services architecture that brings together relevant elements of integrity management allows for the broadest use the IMM model and for clear domain-specific interpretations of the model.

2.2 What is Needed for Trust

The philosophy of trust underlying Trusted Platforms as defined by the TCG consists of three (3) important tenets or mandates that must be simultaneously true in order to achieve trust:

- *Unambiguous Identity*: For something to be able to be trustable it must be unambiguously identifiable. This is true for a Trusted Platform, as well as for every component that make-up that platform. Every component that make-up a Trusted Platform must be known and identifiable. In the case of a software component, a hash of that software file provides a very useful and practical means to identify that component. Other information (e.g. manufacturer, model, make, serial number, patch, etc) also provides for unique identification.
- *Unhindered operations*: Something can be trusted if it behaves in an expected manner for a particular purpose. A component of a Trusted Platform has been designed to perform a particular task and follow a designed behavior. That component must be able to operate without interference from other components or processes within the platform. In order for a given component to even begin to operate, it must not be (allowed to be) subjected to tampering within the platform.
- *Attestation*: In order for something to be trusted, there must be some way of verifying consistent good behavior of that thing. That is, for a Trusted Platform to be trustworthy, there needs to be some means for that platform to report (to the external world) its integrity state (as a whole), which is a function of the integrity state of each component that make-up that Trusted Platform.

The aim of the Integrity Management Model is to provide the infrastructure functions to ensure that that all three tenets above are supported and thus satisfied throughout the lifecycle of a Trusted Platform.

2.3 What to capture: Transitive Trust Chains

As mentioned above and in [1], attestation is an important feature and benefit of Trusted Platforms. One key aspect of a trusted platform is its ability to record the *trust relationships* of among components that make-up the trusted platform. When one trusted component attests to (or measures) the trustworthiness of a second component, trust is “transferred” transitively from that first (trustworthy) component to the second. This relationship is referred to as *transitive trust* (or inductive trust). That is, we accept that the second component is trustworthy because the first component vouches for it. The trust boundary is therefore extended from the first component to the second.

It is here that the notion of *Roots of Trust* is core to Trusted Computing in the practical world. Generally speaking, a *Root of Trust* is something (e.g. a component) that we accept as trustworthy because we have to do so for practical purposes and because we accept that its author or creator has properly designed and implemented it (ie. *technical trust*). We also accept a Root of Trust because its author or creator stands to loose (e.g. brand name, revenue, legal, etc) if they did not implement it according to its specifications. This is why trusted computing also has the aspect of *social trust* in addition to technical trust. Trusted computing recognizes that trust is ultimately rooted in people (e.g. engineers, businesses, companies, society).

There are a number of examples of Roots of Trust in specific platforms and contexts. For example, the TPM hardware is the *Root of Trust for Reporting* (RTR) because it is trusted to perform measurements (eg. hash and extend into PCRs) correctly. The TPM is trusted to operate as expected (conforms to the TCG specifications and must be uniquely bound to a single platform. The TPM functions and storage are isolated from all other components of the platform

(e.g. the CPU). We trust the TPM manufacturer to have implemented the TPM specifications correctly and honestly.

Similarly, the BIOS (in the PC platform) could be considered as the *Root of Trust for Measurement* (RTM) because it is the first piece of software that is loaded and measures other subsequent software components (e.g. drivers, kernel, etc). This is shown in Figure 1. We simply have to accept that the BIOS is correctly designed and implemented, and that the BIOS manufacturer can suffer significantly should their product behaves improperly. The RTM then measures other components (e.g. loader), which in-turn measures further components (e.g. kernel) and so on, creating a chain of (transitive) trust emanating from the RTM.

Thus, in general a transitive trust chain is only of value if it is rooted (i.e. starts from) a Root of Trust.

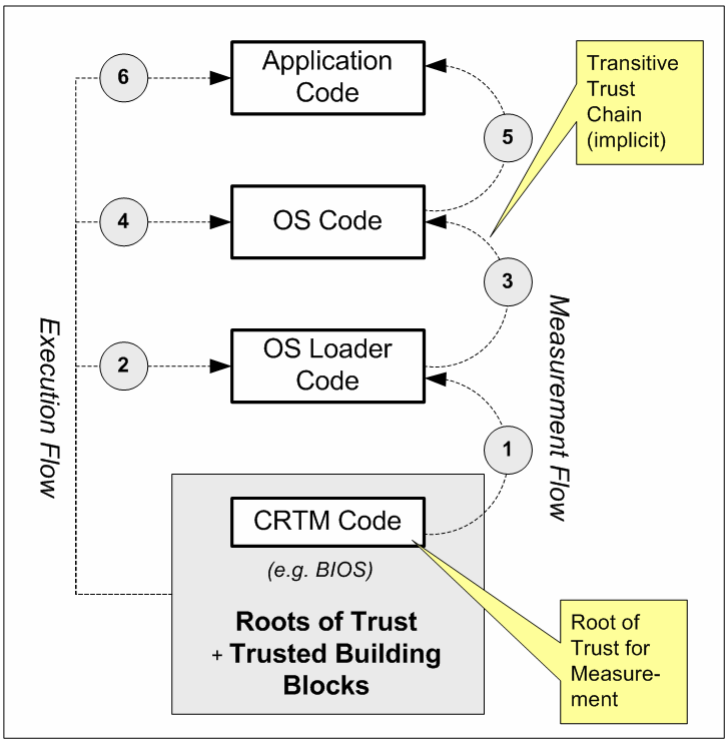


Figure 1: Example of Trust Chain emanating from Root of Trust

One crucial goal of integrity management is to record the measurements performed by components participating in a trust chain (i.e. sequence of measurements of components) in a manner that is coherent for reporting to the external world (e.g. to a verifying entity). Related to this is the correct handling of the trust of chain information within a platform and when it is reported to the external world. Protecting the recorded trust chain information of a platform (e.g. against unauthorized modification) is also crucial since the trust chain is the basis for according trust (by an external entity) to that platform. A trusted platform must be able to report its transitive trust chain in an unhindered manner lest it loses its core value of being a trustworthy platform.

As will be discussed below, integrity management also involves collecting the component-measurements (snapshots) that make-up the trust chain within a live or running platform (referred to as *runtime measurements*), and comparing these against known good measurements of those same components (referred to as *reference measurements*) as supplied by their respective manufacturers or other trusted third parties.

2.4 Relationship of IWG Documents for Integrity Management

The TCG efforts to provide the means to capture and represents integrity information – both from the supply chain side and from the runtime (platform-generated) case – is represented by the current Integrity Management Architecture document and a number of specifications that revolve around integrity-related information and their management. These specifications are shown in Figure 2.

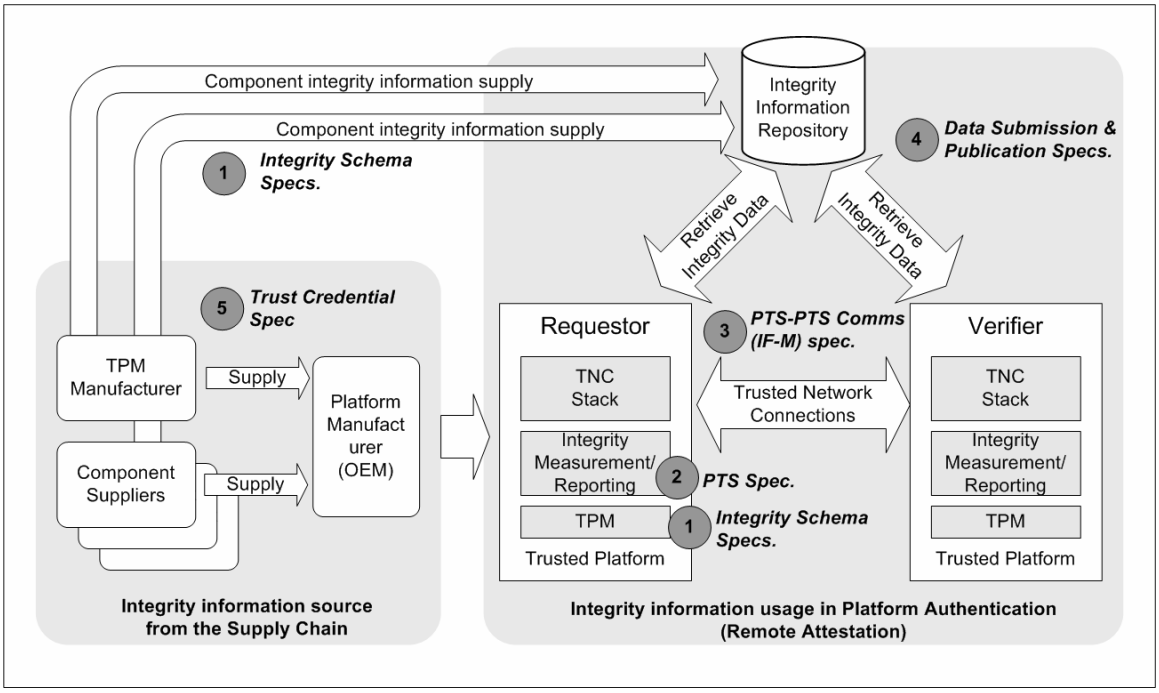


Figure 2: IWG Specifications Relationship for Integrity Management

The current set of specifications pertaining to integrity management is as follows:

- *Integrity Schemas*: The Integrity Schema set of documents pertain to the representation of integrity-related information (syntax and semantics) in the XML schema format. Both static component integrity information (i.e. Reference Measurements) and dynamic integrity information generated by the platform at runtime (i.e. Runtime Measurements) are captured and represented by the schema specifications.
- *Platform Trust Services (PTS) specification*: The PTS is the specification that provides the means to measure (capture) runtime integrity information of a given platform. The PTS is not a platform-specific specification, but is applicable across various TCG platforms. After the TPM and TSS specifications, the PTS is arguably the most important (platform-independent) specifications to be developed by the TCG, as it makes concrete the core value-proposition of remote attestation by trusted platforms.
- *PTS Communications (IF-M)*: This specification covers the messaging and functional requirements for communicating integrity information from the Requestor's PTS to the Verifier's PTS.
- *Integrity information submission and publication interfaces*: This is the interface specification to allow Supply Chain entities to submit (to a repository or database) the integrity-related information regarding their components and which allows the Verifier entity to query and retrieve information from the repository. The interface needs to be

standardized to allow various entities in the Trusted Computing Ecosystem to participate in an interoperable manner.

- *Trust Credentials specification*: This specification covers the next-generation credentials used by the entities involved in integrity management.

It is important to note that the notion of integrity management as part-and-parcel of achieving Platform Authentication (Remote Attestation) also permeates the TCG Trusted Network Connect (TNC). This is because one of the core values of the TNC approach (based on Trusted Platforms) is the platform authentication handshake that allows a Verifier (Policy Decision Point) entity to take into consideration a Requestor's integrity state as part of network access control. As such, there is a common thread running through the current set of specifications and many of those pertaining to TNC. The common thread is that of Platform Authentication (Remote Attestation) based on source-authentic integrity information regarding components that make-up the Requestor's trusted platform and regarding the Requestor's runtime integrity data (generated by that trusted platform).

3 Basic Model for Integrity Measurement and Reporting

Although the integrity management model will be further discussed in detail in the following sections, it is worthwhile at this stage to introduce the basic behavior of trusted platforms in the context of integrity management and integrity evaluation.

Integrity is a core concept in Trusted Computing. There are multiple facets to this term as used in this document and others in the area of Trusted Computing. In general the term "integrity" is used to denote the *pristine state* of a component (software or hardware) within a trusted platform (as issued by the component designer/manufacturer), and also of the Trusted Platform as a whole. This pristine state needs to be maintained all the way from the point when the component leaves the custody of the manufacturer to the actual usage of the component in daily operation of the platform. During its daily usage within a Trusted Platform, that component should remain in its original state and free from tampering (e.g. by other components or attacks). Any deviations from the pristine state should be detected (by the Trusted Platform itself) and reported to the external world (e.g. the Owner or user). The pristine state of a mutable component (ie. software) must also be safe-guarded, meaning that no modifications to that component must go undetected by the trusted platform. Hence, the use of the hash-and-extend mechanism into the PCRs and measurement log (for later verification).

Thus, the term "integrity management" refers the management of component-information throughout the supply chain to ensure their integrity (tamper-free state) and also to the management of the runtime integrity of the entire Trusted Platform through the correct management of its components, both at load-time and at runtime. Note that integrity management does not presume or guarantee that the implementation of a component is correct and bug-free. The IMM – together with the integrity-related mechanisms within the Trusted Platform (eg. RTM and transitive trust chain) – allows the detection of tamper on mutable components.

The trustworthiness of a platform is useful not only to its user or its self-management, but is a powerful means for external entities to evaluate the state of a target platform. Thus, the fundamental aim of integrity management is to support the following functions of trusted platforms:

- *Integrity Measurements*: To allow a trusted platform to perform integrity measurements (of itself). The notion of integrity measurement is core to the value proposition of trusted computing. A given Trusted Platform must not only be equipped with the necessary tools to achieve a trusted computing environment for the user (and applications as a proxy for the user), but it must also be equipped to gauge the "level" of trust it achieves (against some acceptable trust baseline). Hence, the key notion of the *Root of Trust for Measurement* (RTM) that is core to trusted computing (see [10][1]).

  As will be described later, the term "measurement" has a dual meaning. First, *runtime measurement* refers to the hash-and-extend operation (into PCRs) performed upon software components during the operation of a Trusted Platform. Second, *reference measurement* refers to the out of-band digest collection (eg. by the manufacturer) of non-hardware components and the recording of these digest value as part of the informative metadata that accompanies that component from its manufacturer. Reference measurement data may also include metadata regarding hardware.

- *Integrity Reporting*: To allow the measurements to be reportable to the outside world. Once a trusted platform is able to achieve a trustworthy computing environment and is able to measure it in a given way (e.g. measures each and every component), the platform must also be equipped with the means to report these measurements to the external world. The ability to report one's own integrity state as part of an authentication handshake provides a very powerful addition to existing credential-based authentication mechanisms. This is also why the *Root of Trust for Reporting* (RTR) is a key concept underlying trusted computing (see [10][1]).

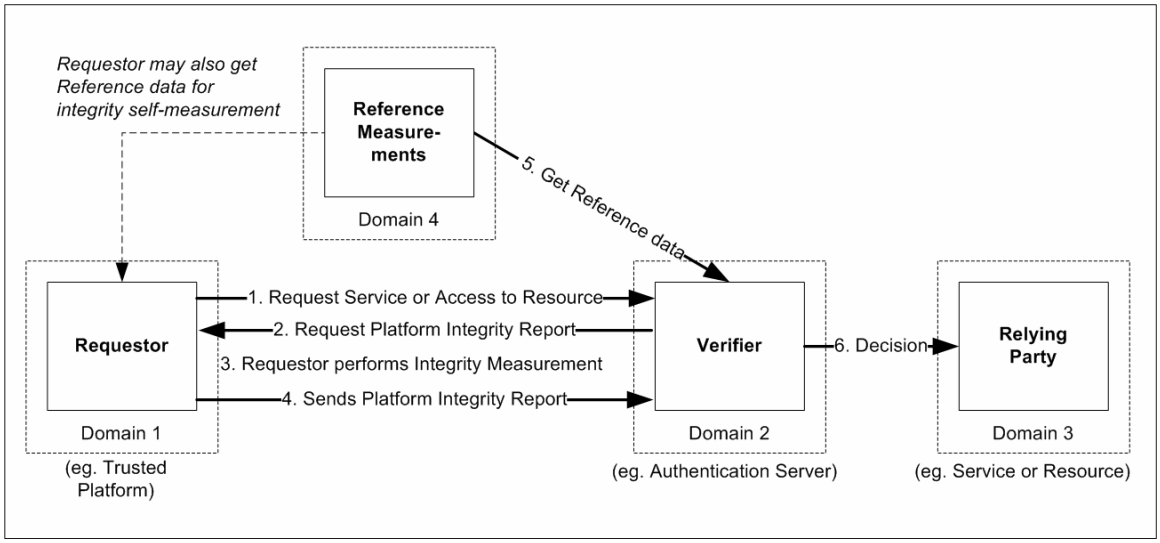


Figure 3: Basic Model for Integrity Measurement and Reporting (see [1])

In order to illustrate integrity measurement and reporting as a core part of the Integrity Management Model, Figure 3 shows the basic interaction between the Requestor and Verifier (from [1]). Here the Requestor is seeking some interaction with (e.g. services or access to resources at) the Relying Party. In the first step the Requestor indicates its request to the Verifier. In the second step the Verifier in return asks the Requestor to perform an integrity measurement of the entire platform (using the requestor's RTM and RTR). Upon completing this measurement in the third step, the Requestor returns a platform integrity report to the Verifier (using the RTM) in the fourth step.

Since the Verifier cannot be assumed to be in the same domain as the Requestor, it cannot be assumed to know all the components (hardware and software) making-up the Requestor's platform. Thus, the Verifier must seek *authoritative information* regarding each of the components of the Requestor's platform (in the fifth step). We refer to this authoritative information as the *Reference Measurements*, which is made available by the metric provider. Examples of the metric provider are the hardware manufacturer, software vendor or a trusted provider on behalf of the manufacturer and vendor.

Once the Verifier is able to identify each component of the Requestor's platform and compare the reported measurement against the expected (baseline) reference measurement value (for that component), the Verifier can gauge the level of trust of the Requestor's platform. A this stage the Verifier is thus able to make a decision with regards to Requestor's wish to access resource/services at the Relying Party. This is shown in the sixth step in Figure 3.

Although Figure 3 provides only an overview of the process and use of integrity measurements, this basic behavior remains consistent across various use cases (e.g. network end-point access control). This basic behavior will be described in further detail throughout the current document.

3.1 General Phases in the Integrity Management Model

There are a number of ways to view integrity information and their flow through the Trusted Platform lifecycle. At an abstract level – independent of any domain specific or use-case specific scenarios – the phases of integrity information management can be summarized as consisting of 5 general phases. These phases are the *creation*, *collection, communication, collation* and

evaluation of data about the components – both hardware and software – that make-up a Trusted Platform (see Figure 4):

- *Creation*: Integrity data or values regarding a given component must be created or produced by an entity that is authoritative regarding the creation, manufacturing processes and history of handling (ie. chain of custody) of that component. For example, in the context of Trusted Computing a typical creator of the integrity data would be a hardware components manufacturer, such as a TPM Manufacturer. Another example would be a TSS software manufacturer or a BIOS manufacturer. An important factor here is the *identification* of the authoritative creator of the component, and therefore the source of the integrity data for that component. Thus, although some component manufacturers are well known in the industry and have a financial stake behind their products, other lesser-known suppliers of components (or sub-components) also need to be clearly identified.
- *Collection*: When components are gathered together to constitute a Trusted Platform, not only do the constituent components (hardware, software, firmware) need to be actually collected and be assembled into a platform, the integrity data relating to each of these components need to be correspondingly collected and combined into a consistent set of integrity information about the platform. The resulting collection or set of integrity data must be semantically meaningful and express integrity meaning at a higher value level, from which trust assertions and claims can be derived in a more general manner about the platform as a whole. No integrity data must be lost in the process of collection, and an evaluator (downstream) must be able to query the existence and status of the component integrity information (e.g. to the creator or manufacturer of the integrity data). Thus, for example, when an OEM creates a Trusted Platform by assembling various components – hardware and software – from various manufacturers of these components, in addition to acquiring the components the OEM should also request the integrity data corresponding to the supplied components. The OEM (or an appointed entity on behalf of the OEM) would then make these integrity data available for evaluation by other parties (e.g. Enterprise buyer). Here, individual component manufacturers could be the publisher of integrity data. Alternatively, the OEM itself could be the publisher of data.

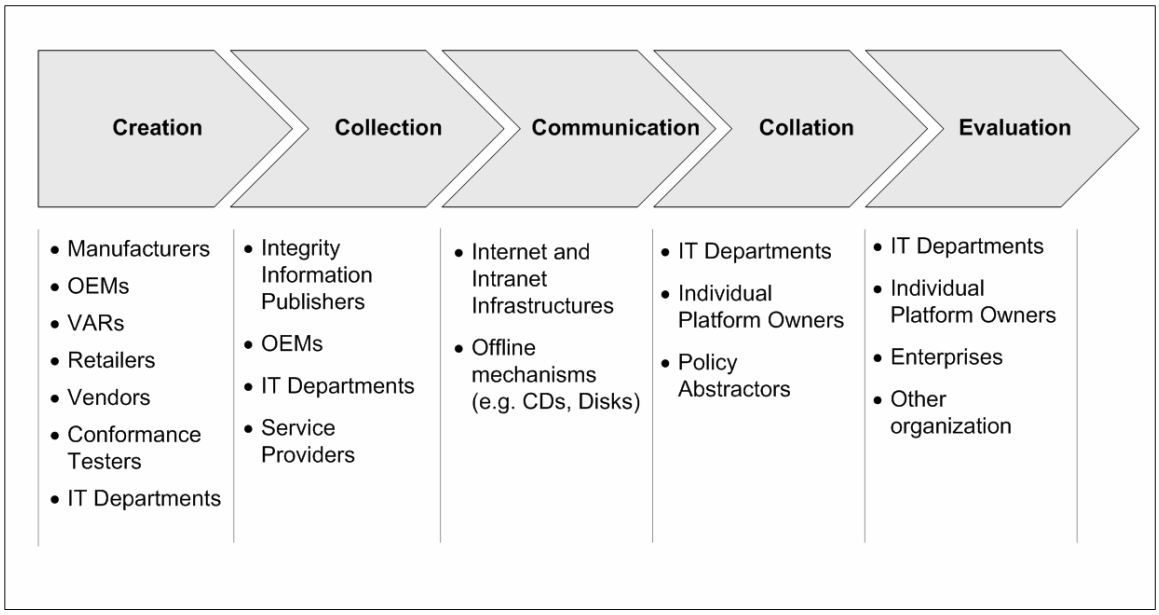


Figure 4: General Phases in Integrity Information Management

- *Communication/Publication*: When a Verifier seeks to verify or check the integrity data regarding a Trusted Platform and all its components, some method of communicating these integrity information to the Verifier must be established. There are number of ways of making the integrity data available to the Verifier. One approach could be for the creator of the component and integrity data to publish the information in a database accessible to the Verifiers. Alternatively, the integrity data could be shipped with the component (e.g. on the CD), and so on. Availability of integrity data to the Verifier is crucial to the whole value proposition of integrity management and the infrastructure services implementing them.
- *Collation*: The act of gathering together integrity data pertaining to components of a given target platform is referred to as *collation*. The collation of integrity data can be done by a number of entities, depending on the where in the TP Lifecycle the collation step occurs. However, it is important to distinguish this step from that of collection (above). The aim of "collation" is to eventually verify the integrity information, whereas the purpose of "collection" is to publish or make available the information to other consumers of the information in the TP Lifecycle.
- *Evaluation*: Any entity who needs to ascertain the integrity of a platform and therefore ascertain correctness of integrity data can be categorized as an Verifier entity. A Verifier can be performing evaluation for its own purposes, or on behalf of Relying Party. Evaluation itself is context-dependent (ie. use case dependent).

3.2 Runtime Measurements & Reference Measurements

The basic notion behind integrity information management in the context of trusted computing is to provide reliable and authentic integrity data regarding components of a Trusted Platform that can be provisioned and later be verified. Typically, a trusted platform will consists of numerous hardware and software components, which are brought together by the platform manufacturer (e.g. ODM, OEM) to form the platform.

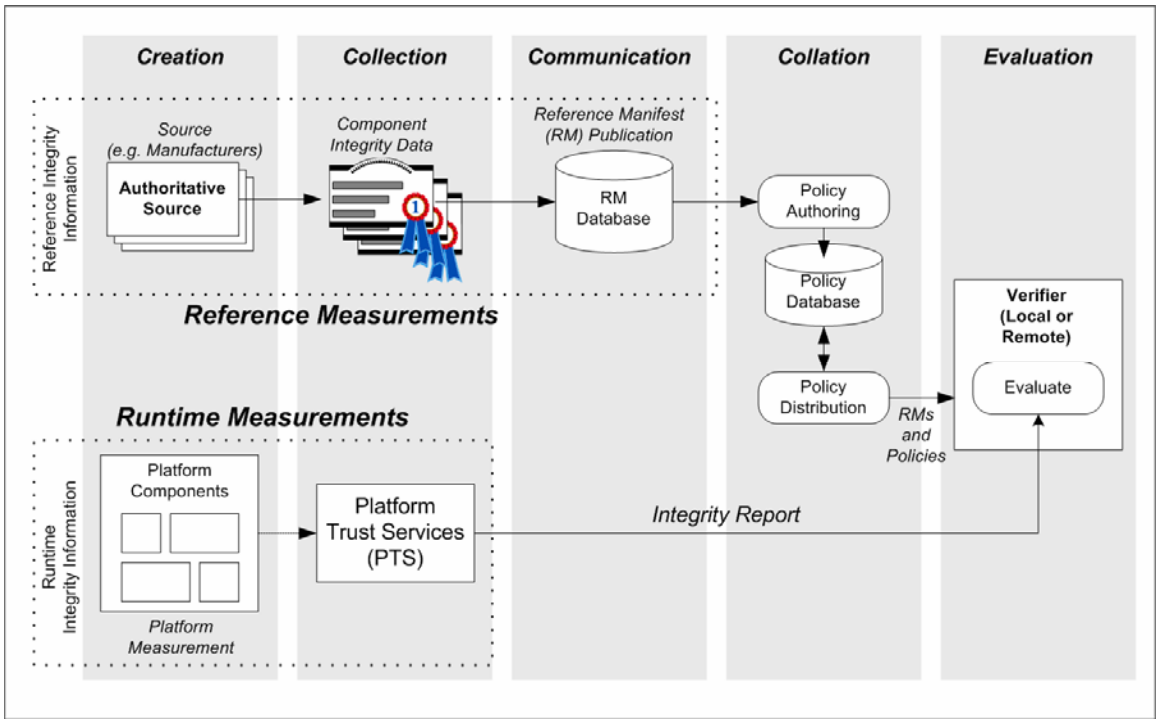


Figure 5: Runtime/Loadtime Measurements and Reference Measurement Processes

Throughout the current document and other TCG specifications pertaining to integrity management, there is a distinction made between Runtime/Loadtime measurements and Reference Measurements (Figure 5):

- *Runtime/Loadtime Measurements*: These types of measurements are those made by a platform during its bootup and operations. They refer specifically to measurements taken of components of the platform and also of the platform as a whole. Typically, this is done by a component of the platform that is a Root of Trust for Measurement (RTM), which could be embodied in different forms depending on the specific platform. Loadtime Measurement refers to the integrity measurement of a platform at the completion of a boot-up sequence, whilst runtime measurement refers to the event or act of integrity measurement of a platform at anytime during that platform's operations. The aim this kind of measurement is to obtain platform integrity state at a given time. The form used is the TCG *Integrity Report* structure as defined in [4].
- *Reference Measurements*: In order to support technical trust in a given component, the component manufacturer must support its product with information regarding the provenance of the component. That is, the manufacturer or a trusted third party must provide some static *reference* values for that component. These static reference/metric values for components are called reference Measurements, and are represented in the form of the TCG *Reference Manifest* (RM) structure, much in the traditional use of the term manifest. The manifest for a component contains information such as its Identity, Manufacturer, Model number, version number, and others. The Reference Manifest usage model is discussed further below, while the specific fields or items within a Reference Manifest are defined in [3]. Generally speaking, in order for Reference Manifest information to be useful within a given context (e.g. verification of component as part of platform authentication), the Reference Manifest data must be made available and accessible to entities in the Trusted Computing Ecosystem.

Runtime measurements generally refer to the measurement of the integrity of the components of a Trusted Platform during the operation of that platform. Runtime measurements is platform-specific in the sense that each platform type (e.g. PC-Client, Server, Mobile Phone, etc) will have unique system architectures, TPM usage, PCR allocations, and modes of booting-up and operations. Depending on the specific use-case scenarios, it is often useful to distinguish further between the platform state at the end of a boot-up sequence (Loadtime measurement) and the platform state as measured anytime during its operations (Runtime measurement). This will be the topic Section 5.

Reference measurements regarding a given component must be collected from the manufacturer and made available before the component is actually put to use within a given trusted platform. In Figure 5 this step is generally called the *Creation* of reference integrity information. The manufacturer is considered to be the authoritative source of both the Reference Manifest data and the actual component product described by the Reference Manifest.

Since not all raw data regarding a given component can be suitably called a Reference Manifest, some preparatory work must be done on that data to convert it into an acceptable Reference Manifest. In Figure 5 this step is generally referred to as the *Collection* of reference integrity information, which includes the formatting or canonicalization of the information into a standard format [3]. Included here is the digital signature over the Reference Manifest data to provide source-authentication of the data. Note that the signature over a component Reference Manifest can be provided by the manufacturer or by some trusted third party (such as a value-add service provider) on behalf of the manufacturer. In the case that a trusted third party becomes the Reference Manifest issuer, it is assumed that a business and trust relationship exists between the Reference Manifest issuer and the manufacturer/vendor of the component.

Once the Reference Manifest data regarding a component has been prepared and signed, it published into an *Integrity Database* (or Reference Manifest Database/Repository) that is accessible to the consumers of the information. Each manufacturer can stand-up its own

Reference Manifest database for all its component products, or such a database can be made available by a third party service provider.

Note that the aim of the Reference Manifest Database is to provide an authoritative source of component integrity information whose records can be read (downloaded) by a Verifier. This is done in the context of a Verifier evaluating a Requestor. The Verifier would then compare the component information (involved in a runtime measurement event) reported by the Requestor (e.g. within a Platform Authentication handshake) against the reference measurements obtained from the authoritative Reference Manifest database. In this way, the Verifier knows both the current integrity-status of the components making-up the Requestor's platform as well as the source-authenticity of those components (as coming from the manufacturer).

Reference measurements will be discussed further in Section 6.

3.3 Integrity Reports and Reference Manifests

The TCG Integrity Management Model is designed to support the integrity evaluation of entities, following the TCG basic model for Platform Authentication (see Section 4 of Architecture Part I [1]). There, a Verifier entity performs an evaluation of Requestor entity, possibly on behalf of some Relying Party entity. The term Platform Authentication is a specific term used by the TCG to denote integrity-verification of a Requestor platform as part-and-parcel of the authentication handshake by the Verifier with that Requestor platform.

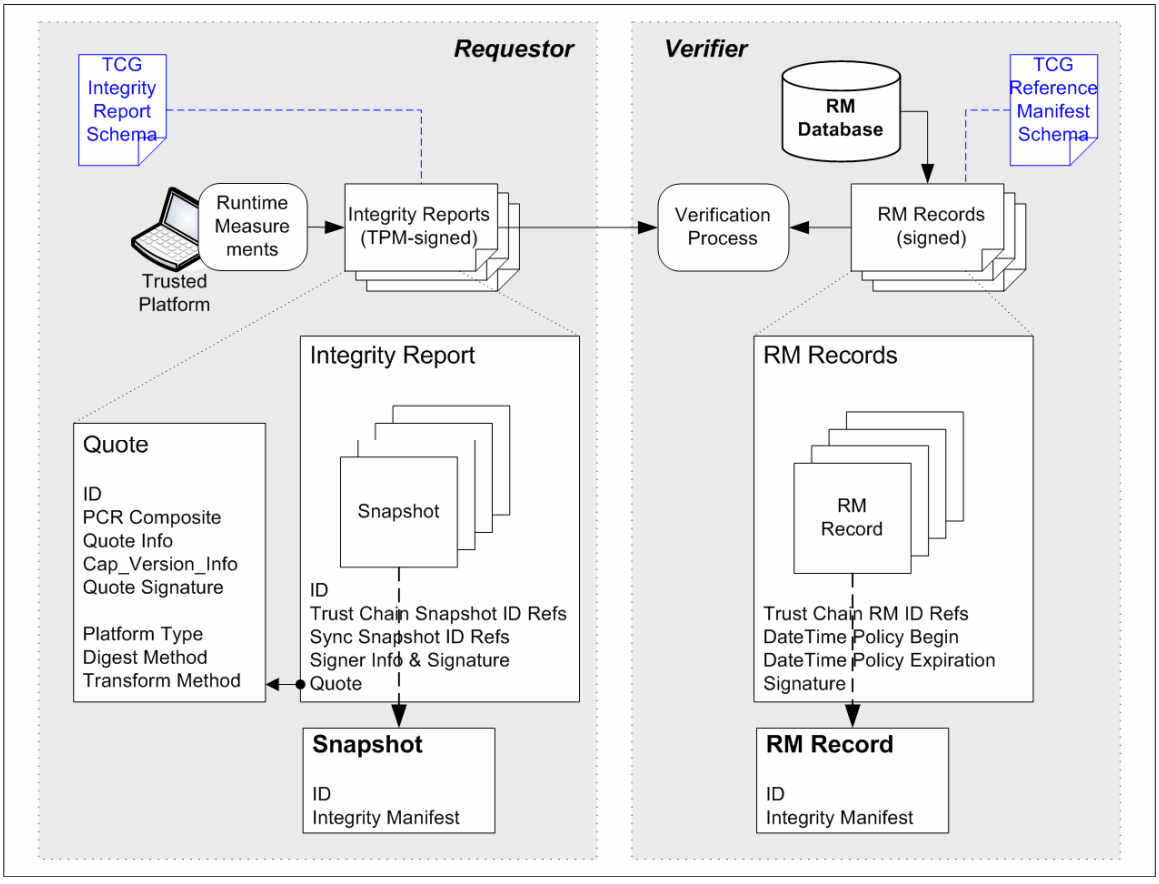


Figure 6: Overview of Integrity Reports and Reference Manifests (RM)

When a Requestor's platform needs to be evaluated by a Verifier, the Requestor must provide the Verifier with an Integrity Report [4] which is complied from a set of *snapshots* covering the

platform's components (Figure 6). The snapshot represents both the measurements and assertions about every relevant component of the platform (and possibly sub-components of these components). References (ID Refs) are used to point to information pertaining to the components reported in the Integrity Report.

The authenticity of an Integrity Report as pertaining to a given platform with a TPM hardware is achieved using the AIK-certificate (see [2]) and Quote structure. In particular, the Verifier can check the digital-signature over the Integrity Report verifying that the AIK private key used in the Integrity Report signature (i.e. xml-dsig) is the same as that found in the Quote structure, and that the AIK-certificate is tied to the EK-certificate inside the TPM hardware. In other words, the Integrity Report is therefore tied to the TPM hardware from which the root-of-trust emanates.

3.4 Comparing Snapshot and Reference Manifest Structures

It is important that the measurements collected within a snapshot during runtime correspond or match the reference measurements represented within a Reference Manifest record. That is, semantically they must possess the same essential structural information so that automated comparison is achieved. Figure 7 attempts to shows this correspondence.

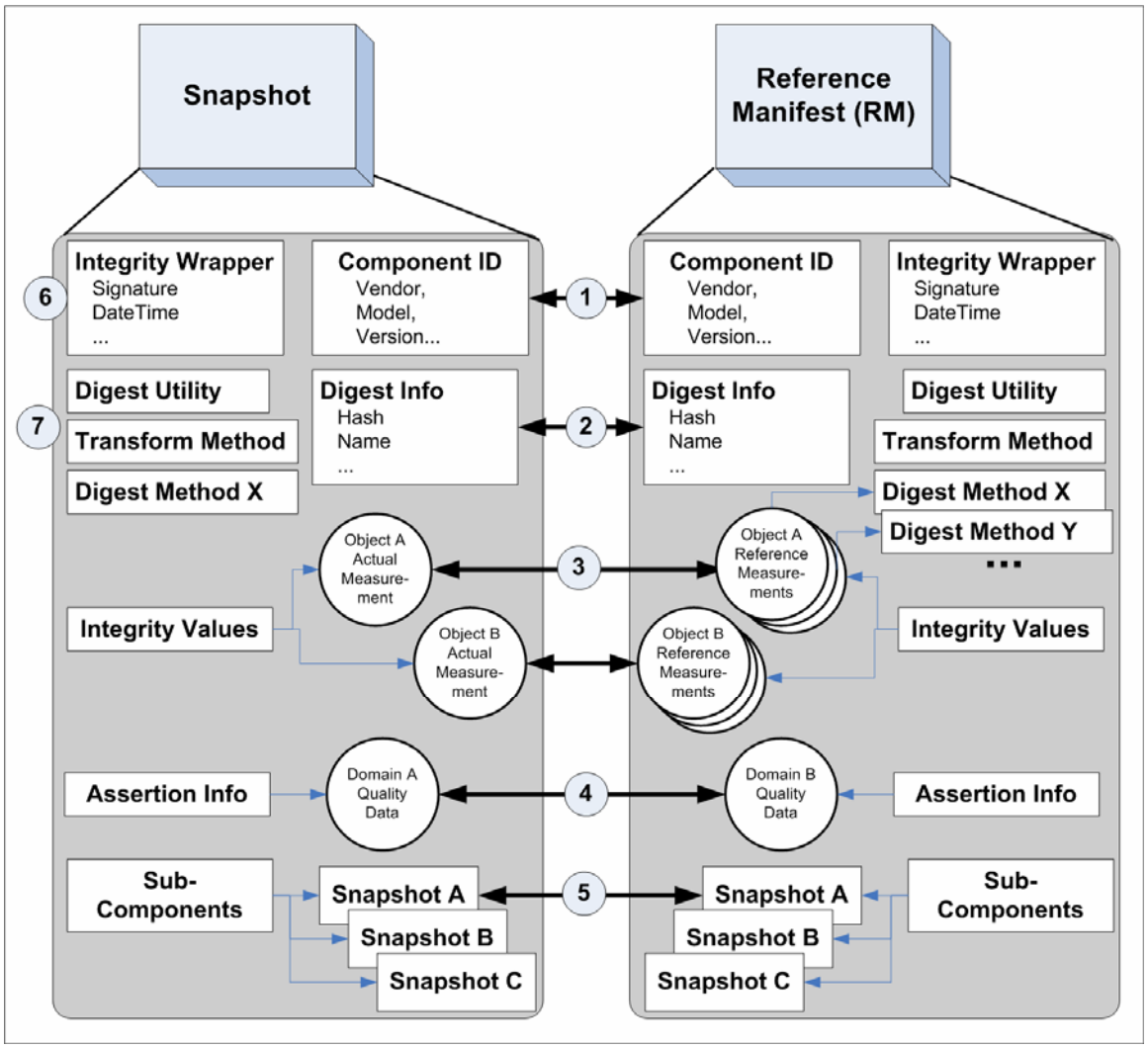


Figure 7: Using Reference Manifests to Verify Snapshots

When a Verifier receives an Integrity Report regarding a platform, the Verifier needs to parse through each snapshot within that report and compare the integrity information within that snapshot (See Figure 7 Item-3) with corresponding integrity information in the Reference Manifest. For each snapshot, the Verifier needs to obtain a Reference Manifest record containing the expected integrity data for components referenced in the snapshot.

The Snapshot contains the *actual* integrity measurements for a given component identified by Component-ID (See Item-1). The Reference Manifest contains *reference* integrity measurements for the component identified by Component-ID. For each component within the verifier's platform, the Component-ID is used to match both the snapshot and the Reference Manifest records. The Component-ID (Item-1) contains several fields pertaining to the component that together, identifies a distinct object that can be placed under configuration management. (e.g. Vendor, model, version, etc). The Vendor-ID portion of the Component ID identifies the vendor that manufactured the component and also applies a revision control process to the component. It is the vendor's revision control process that makes it possible for a verifier to have confidence that the code or logic contained in the component will behave as expected. This property is fundamental to trusted computing. See Section 7 for additional details regarding the Core Integrity Manifest.

Both Reference Manifest and Snapshot can represent a summary of integrity measurements by computing a hash of various elements in the document called Digest Info (Item-2). Digest Info also has attributes describing how the digest was computed. The digest-info of the snapshot (Item-2) has attributes specific to TPM PCR registers that assist in reconstructing TPM hash-and-extend operations (upon a given component within the trusted platform). The Digest Info in a Reference Manifest is not required to match that of a Snapshot. These are for internal consistency only. However, in some situations it may be advisable to construct a Reference Manifest such that it exactly matches an anticipated Snapshot so that verification is simplified.

Reference Manifests and Snapshots both contain integrity measurements for individually identifiable objects that make up a component. These could be a collection of libraries and executable files of a software utility or package. The Reference Manifest integrity values (Item-3) shows multiple reference measurements for a given object. The Reference Manifest creator has chosen to compute several reference measurements for the same object using different Digest Methods so that the digest method selected by the snapshot creator can be validated without pre-negotiation.

Assertions pertaining to esoteric attributes of the component are represented in Insertion Info (Item-4). Assertions do not have to be the same for verification to be successful. The verifier considers assertion info when applying a verification policy and to apply compensating transactions for risk mitigation. The assertion is taken to be a statement by the signer.

Reference Manifests and Snapshots are often amalgamations of a collection of other Components (Item-5). The relationship of Component to sub-component helps describe the nature and structure of the parent Component. If a measurement process constructs a Snapshot that has missing or extra sub-components, this can be an indication of tampering or improper configuration.

The Reference Manifest or Snapshot documents can be protected with an Integrity Wrapper (Item-6) that contains a confidence value, digital signature and date-time or random nonce. These attributes are used by a verifier to establish the validity and veracity of the document. It is unlikely that integrity wrapper information would be the same between Reference Manifest and Snapshot.

The Digest Utility (Item-7) refers to the component that performed the measurement function. Digest utility, in the case of a Reference Manifest refers to the tool used to harvest reference measurements and in the case of Snapshot, refers to the measurement engine collecting attestation evidence.

3.5 Platform Trust Services

In order for runtime measurements to be collected, a process within a Trusted Platform must be defined to do so. The TCG Integrity Management Model has defined Platform Trust Services (PTS) as a standard IMM component which provides all the necessary services which an *Integrity Measurements Collector* (IMC) [9], or other measurement controlling agent, can interface with to measure components, parse Reference Manifests, create snapshots, synchronize snapshots, and generate Integrity Reports. PTS is then the primary IMM component responsible for interfacing with the TPM when this trusted hardware is available on a client platform. The specific capabilities which are available to an IMC or other measurement controlling agent are defined in the TCG IF-PTS specification [8].

PTS is intended to be integrated within the Trusted OS components for creating these measurements and assertions in a trustworthy fashion. Alternate implementations of PTS, outside of a trusted OS but within the transitive trust chain, are also possible. The PTS is further described in Section 8.

4 The IMM Processes

The Integrity Management Model (IMM) includes a number of key processes to complete the relationship between the Requestor, Verifier and Reference Manifest Database. These processes are described in the following (Figure 8).

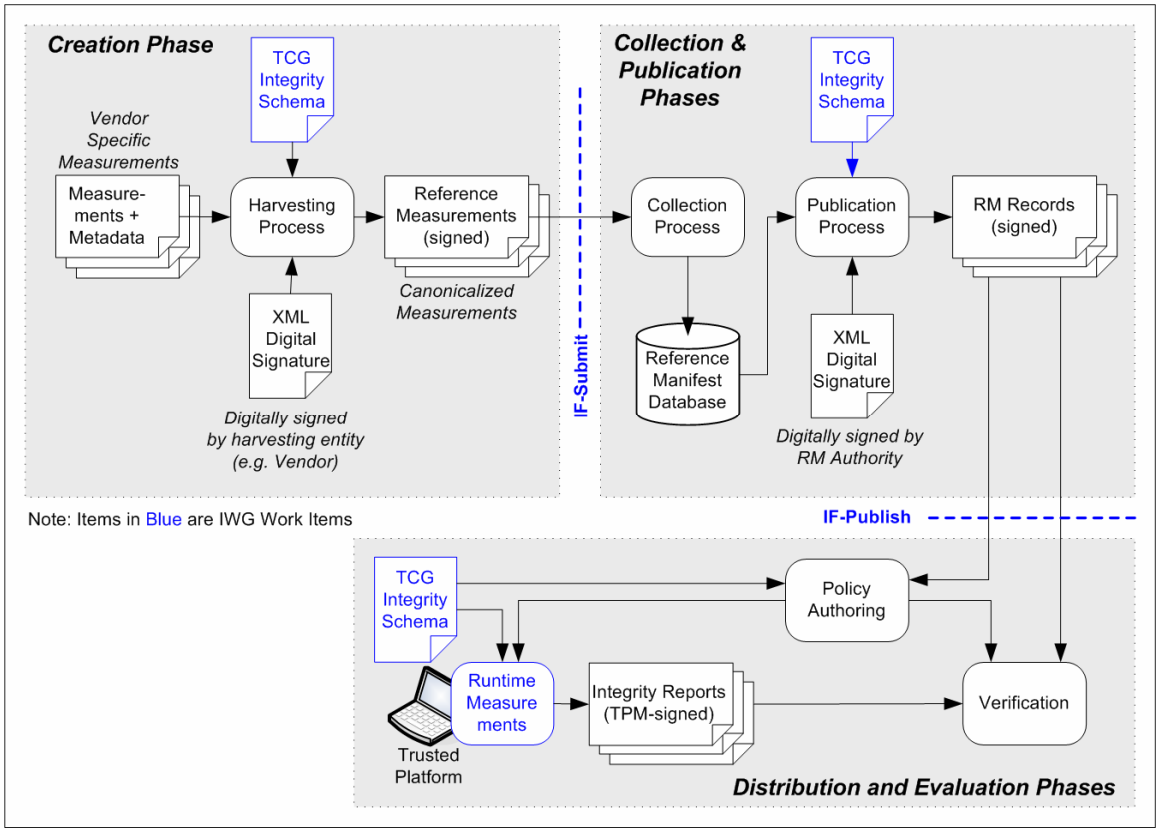


Figure 8: IMM Processes

4.1 Creation (Harvesting) process

The creation or "harvesting" process revolves around the gathering of vendor-specific reference measurements and metadata pertaining to a specific component (from that vendor). Preferably, harvesting should occur at the source of the component, namely the vendor or manufacturer, as this would achieve the highest level of trust that can be accorded to the component and its accompanying metadata. Note that source-harvesting can be done either by the vendor or manufacturer itself, or by a trusted third on behalf of the vendor or manufacturer under some legal business relationship. This legal relationship is one manifestation of social trust, which is accompanied by the technical trust manifested by the mechanisms used to harvest the measurement and metadata for the component in question.

In Figure 8 a number of elements/entities are shown to interact with the harvesting process. Raw measurements and metadata that are vendor-specific and are expressed in a vendor proprietary format are used as input into the harvesting process. The output of the process is the reference measurements that are expressed according to the TCG Integrity Schema [3]. Since the expectation is that there will be numerous vendors participating in the Trusted Computing Lifecycle, each vendor will need to digitally-sign their reference measurements data.

It is worthwhile to note that although Figure 8 has demarcated processes as being carried-out by distinct entities, in practice this demarcation may not (and need not) exist. Thus, the entire harvesting process could in fact be done by the same entity that carries out the collection and publication of the measurements (e.g. vendor or trusted third party). Similarly, it is not inconceivable that an organization (e.g. Enterprise IT) may perform all the above process. That is, it performs self-harvesting of the measurements and metadata from the available software (i.e. binaries) at their disposal and makes these available internally to the organization only.

4.2 Collection and Publication Processes

The next step in the IMM is the collection and publication of the reference measurements from various sources (e.g. vendors, manufacturers) into a single point of distribution.

As mentioned above, in the harvesting process a vendor gathers vendor-specific measurements and metadata pertaining to a specific component (from that vendor). Although this approach may be sufficient from the perspective of data availability, the fact that a trusted platform consists of numerous components (from numerous vendors) makes the task of obtaining measurements cumbersome. That is, it would impractical for the Verifier (in a transaction with a Requestor) to communicate with every vendor (of the components) related to the trusted platform.

As such, in addition to individual vendors performing the collection and publication processes, the IMM recognizes the usefulness of *aggregation services* – possibly by third-party (value-add) service providers – in the collection of measurements and metadata (collection process) from disparate sources, and publishing these from a single-point of availability through a publication process. The interface to the collection process is referred to in Figure 8 as *IF-Submit* (i.e. submission interface). This is the interface used by the harvesting entity (e.g. vendor) to submit measurements and metadata to the collection process in order for the information to be stored in the *Reference Manifest* (RM) Database. It is the Reference Manifest Database that is viewed upon as being the single point of availability for the consumers of the measurements and metadata (namely the Verifier or Relying Party).

The process of publication involves making measurements and metadata (in the Reference Manifest database) available to the Verifier and other entities through a standard interface. This interface is referred to as the IF-Publish (i.e. publish interface). The Verifier retrieves *Reference Manifest Records* (or simply “RMs”) from the database through the IF-Publish interface. The Reference Manifest record format is defined by the TCG Core Integrity Schema [3] (more specifically, the TCG Reference Manifest Schema [5]).

The idea here is that a Reference Manifest Record pertains to a single component. Where a component is made-up of one or more sub-components, the Reference Manifest Record would have pointers to the records of those sub-components (each defined in the same TCG Reference Manifest schema).

Note that all Reference Manifest Records must be digitally signed by its issuer (either the vendor or the value-add service provider). The term applied to the signer of these records is *Reference Manifest Authority* or Integrity Authority.

4.3 Policy Authoring and Verification Processes

Since one of the key goals of the IMM is to allow for the trustworthiness of a platform (Requestor) to be evaluated by a Verifier or Relying Party, an important role of the IMM is to make the reference measurements and runtime measurements available to the use-case specific policies. This is the role of the Policy Authoring process in Figure 8.

The term “policy authoring” in the context of IMM specifically means the use of measurements (reference and runtime) as input into the policy-setting function for the given environment of use-case.

Thus, for example, in the context of network access control as defined by the Trusted Network Connect (TNC) architecture [9], an IT Administrator can use the measurements as one (of many)

rules or conditions for granting access to the network. A Trusted Platform that wishes to connect to the network will need to be integrity-evaluated by the Policy Decision Point (PDP) entity. This involves the Trusted Platform (as the Network Access Requestor or NAR) reporting its runtime measurements to the PDP (e.g. AAA Server), who in-turn retrieves the corresponding Reference Manifest Records and uses the records to evaluate the NAR in the Verification process. Although the IMM is not a policy-setting tool, it is expected that the measurement information (made available by the infrastructure implementing the IMM) be used as input by policy-setting tools.

Figure 8 also shows the TCG Integrity Schema as defining the format of the Integrity Report as produced by the Trusted Platform as part of runtime measurements. The same schema feeds into the Policy Authoring process so that the policy authoring tools can provision policies on both the Requestor-side and the Verifier-side, in such a way that a match may occur during a Verification process or event. Thus, the IMM and the TCG Integrity Schema provides both the integrity semantics and syntax respectively into the policy-authoring tools, providing a way for these tools to make use of the powerful features of trusted platforms.

The topic of Policy Authoring will be discussed further in Section 9.

5 Runtime Measurements

The TCG Integrity Management Model (IMM) covers the production, collection, communication, storage and evaluation of integrity information pertaining to trusted platforms, with the aim of achieving and maintaining the trustworthy state of the platform. As mentioned above, the IMM covers two sides of the equation of trust evaluation: runtime measurements and reference measurements. See Figure 9.

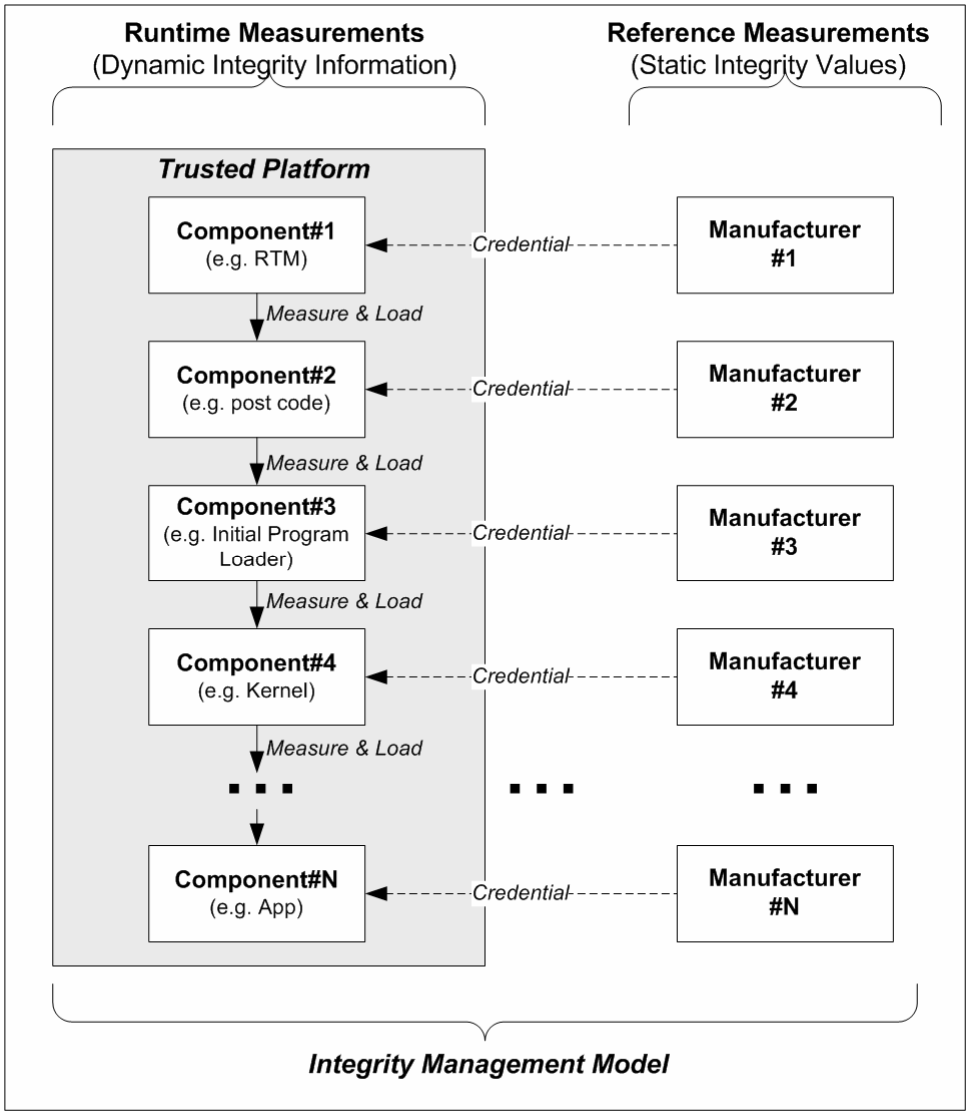


Figure 9: Runtime Measurements and Reference Measurements

Since the overarching aim of the IWG architecture is for interoperability, it is important that both dynamic and static integrity information be defined and presented in a standardized format. In particular, the semantics and syntax must be useful to the participants in the trusted computing ecosystem, notably here the producers of static integrity values (e.g. manufacturers, OEMs, vendors), the platform measurements and reporting agents and the platform verification agents. That is, the measurement log structures produced by a Trusted Platform and manufacturer produced integrity values should be based on a common format specification.

Note that a number of TCG specifications make several references to integrity measurements, metrics, event or logs that in some way contribute to the value of PCR registers inside the TPM. Since the entries of a PCR changes as measurements are entered (and extended), some mechanisms to store these measurements are needed so that the correct evaluation of the PCR (which requires a recalculation of the PCR entries throughout a specified time) can be performed. We thus refer to the logical structure that represents all necessary measured data as the *Integrity Measurement Log* (IML).

In the following we discuss further the structure of the measurements and their representations.

5.1 Measurement Log Decomposition: Levels of Logs

In order to understand the structure of the log that is needed to store entries found in PCR, it is useful to breakdown the PCR values into its basic form.

- *Logging by PCR Index* (First Order Decomposition): A first order decomposition of a measurement log is simply by PCR index. Each PCR register corresponds to a distinct log structure. Hence, it may be appropriate to describe the integrity measurement log at it highest level as a collection of logs (each corresponding to a PCR index).
- *Logging of Extended PCRs* (Second Order Decomposition): Second order log decomposition occurs when the PCR state of one PCR is hashed into the PCR state of another PCR. Extending PCR results into another PCR may be done to initialize a starting state for resetible PCRs or anytime trust dependencies that cross layering boundaries need to be made explicit.

The lifetime of measurement logs correlates with that of a PCR register. PCRs are expected to contain temporal values. They typically do not survive across system power cycles and may not survive across some low power states. Some PCRs are resetible indicating ever shorter lifetimes. Resetting a PCR value (programmatically or by system restart) implies the integrity log is also reset. Thus, in defining log management schemes the issue of lifetime of logs must be addressed in order to maximize resource utilization and to prevent stockpiling of stale integrity data. In general, each PCR verification operation should also anticipate a method for obtaining and destroying the corresponding log.

5.2 The notion of Snapshots

For any given PCR there is a *starting state*, one or more *intermediate states* and an *ending state*. Thus, a log can be in fact be also sequentially partitioned into a structure corresponding to these three states. The starting state (which exactly matches the PCR starting state) can be represented as a hash value. A series of data values ordered according to the sequence in which they were measured can represent the intermediate states. Finally, the computation of the hash-extend operation results in the ending state.

It is important to note that the purpose or identifying these three states is to allow easy comparison at a later time between PCR and integrity log structure. Thus, we introduce the notion of *snapshots* as representing some portion of the log (corresponding to a number PCR states in sequence). A snapshot of an integrity log can be constructed by selecting a common starting state (between PCR and log) and a common ending state. This concept is shown in Figure 10.

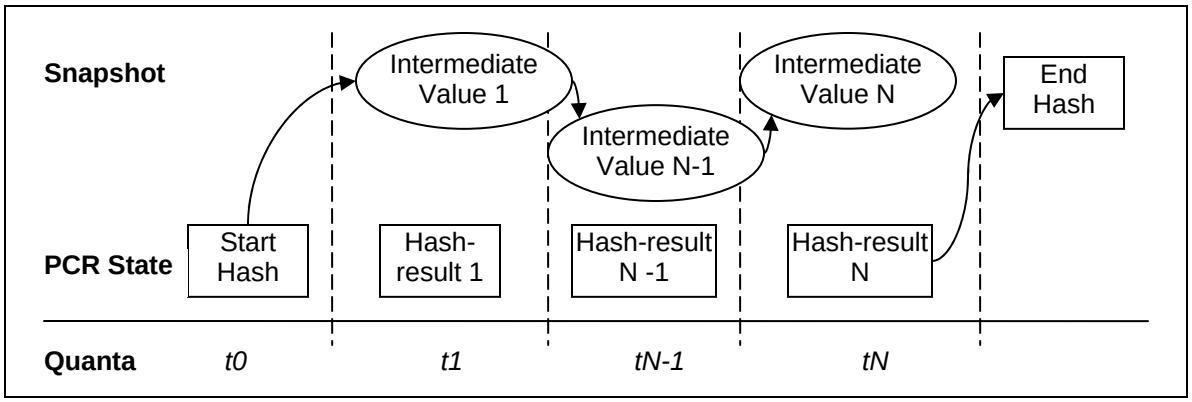


Figure 10 - Snapshot of an Integrity Log over time quanta showing PCR state changes

5.3 An Interoperable Log Structure: Issues & Requirements

An interoperable log structure should comprehend all the ways in which logs may be used. The variety of usage modes should be reflected in log metadata such that verifier's are able to reproduce meaningful hash-extend sequences.

5.3.1 Platform Specific Integrity Values

The TCG has determined that certain integrity values are specific to particular hardware architectures (e.g. PC-Client, Mobile Phone, etc). Each architecture brings with it platform specific constraints regarding format and structure of integrity values as well as measurement sequences. TCG platform specific working groups work to explicitly define and characterize integrity values and states such that a verifier may evaluate platform integrity. This approach requires verifiers to maintain platform specific interpretation logic.

An interoperable integrity log structure can address some platform specific constraints by defining a "hardware architecture" type value to indicate platform specific encodings. Type values should be defined in terms of TCG namespace using an interoperable namespace mechanism such as XMLNamespaces and XRI.

Platform-specific specifications (e.g. PC-Client) may also define classes of integrity values that may be measured into particular PCRs and define initial sequences on a per PCR basis. For static sequences platform specific structures may be defined that hold integrity values. For example, the PC-Client defines an ACPI table structure to capture information from selected PCRs.

Some platform specific values may be measured that are not defined by a platform specific specification (such as option ROMs, removable daughter cards or peripherals). For these the integrity log should contain metadata describing measurement sequence and data semantics or a reference to an integrity database that contains data semantics. See section 6 below.

5.3.2 Application Specific Integrity Values

It is an onerous requirement for application specific integrity values to be defined in application specific specifications to achieve interoperability. Rather, application integrity values should be expressed in an industry standard format (such as XML) and reported in that format.

The integrity log should contain metadata describing measurement sequence and data semantics or references to an integrity database containing semantics. This approach permits out of band collection of known good integrity values without mandating in band exchange during an integrity challenge.

5.3.3 Integrity Log Metadata

Integrity logs must contain metadata that captures log structure semantics described above. In particular, it must capture platform architecture type, PCR index(s), measurement sequence, snapshot start/end, snapshot sequence, snapshot reference (if remote), platform-specific log structures, semantics associated with integrity values (or reference) and well-known snapshots.

Other optimizations associated with managing integrity logs may be needed. Compression of logs should be lossless. Compression should not diminish interoperability due to a verifier's inability to decompress logs. Metadata should make provisions for log compression in the form of uncompressed metadata that indicates compression algorithm and any associated parameters.

5.4 Measurement & Verification Interoperability

A goal of interoperation is to establish a match between measurement processes and verification processes. The complexity of the verification process roughly approximates the complexity of measurement processes. Thus, the importance of the integrity log cannot be underestimated because its structure will determine the effectiveness and efficiency of measurement processes and verification processes in a trusted platform.

A model for partitioning measurement processes may be helpful when considering verification engine design. In Figure 11, the bottom layer shows a relationship between Root of Trust for Measurement (RTM), Root of Trust for Storage (RTS) and Root of Trust for Reporting (RTR) with a Verifier entity.

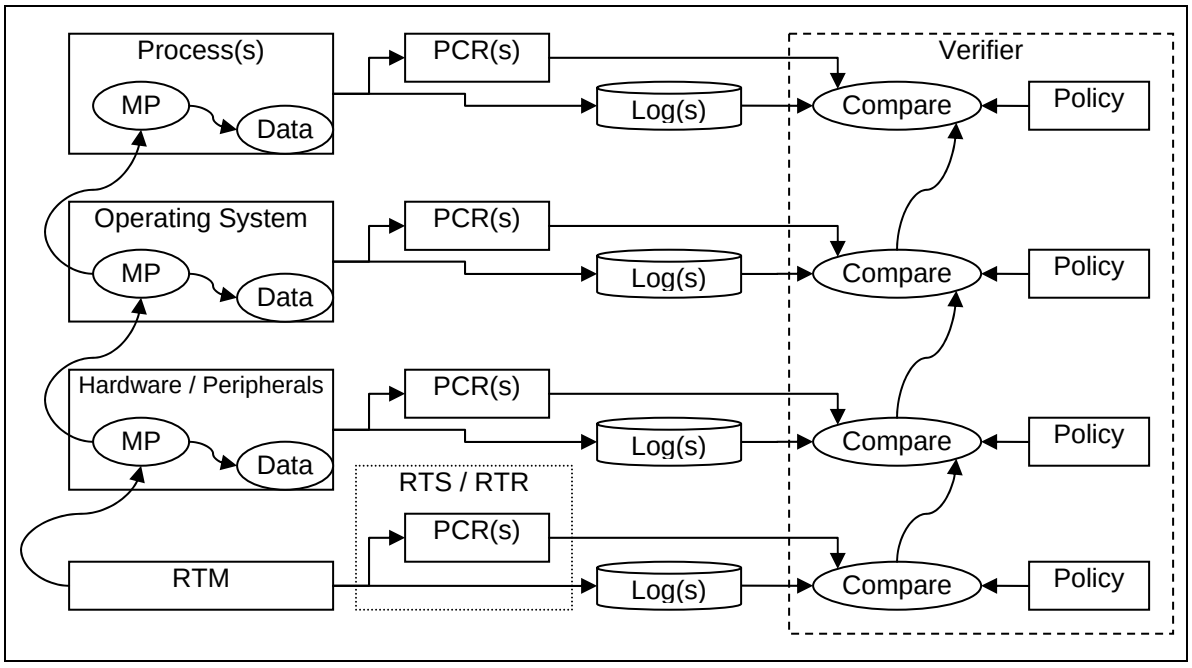


Figure 11 - Model for measurement and verification interoperation

The model demonstrates verification of acceptable hardware configuration given as policy. The model can be extended into the various layers of a platform architecture including operating system and applications. In each layer there may be a *measurement process* (MP) used to measure the next layer. Corresponding PCRs and log files also are used to satisfy storage and reporting functions. The measurement path followed by RTM and subsequent MPs captures layering semantics that are also followed by Verifiers when determining policy compliance.

Measurement Processes are free to measure any data value that may be of interest to Verifiers, with the caveat that measured data must be written into a measurement log (in a form recognizable by the Verifier).

It is incumbent upon the lower layers to provide the trusted environment upon which the functions of measurement, storage and reporting may be founded. There is an implicit assumption that the MP at layer n+1 trusts the execution, storage and reporting environment established by the lower layers. If an MP trusts a particular layering structure, it should take steps to seal itself to these layers.

Each layer has the option of abstracting storage and reporting functionality. Agents and services that depend on such an environment should determine if the environment meets its trust expectations by inspecting the trust engines (that the agents and service are built upon) before performing sensitive operations. Thus, if questions arise regarding the trustworthiness of lower level functions, then the lower layers must provide a way for further details to be obtained regarding the trustworthiness of the trust engines.

6 Reference Measurements

As mentioned above, the Integrity Management Model consists of a number of phases covering both static and dynamic integrity information, obtained through in-band and/or out-of-band mechanisms. In this section, we discuss the reference measurements as obtained through the IMM Processes (see Section 4), and their use in the context of verifying an Integrity Report obtained during a runtime measurement event. The aim of this section is to show the close relationship and correspondence between the elements of a runtime measurement (as represented by an Integrity Report made-up of integrity-snapshots) and the elements of the reference measurements.

6.1 What is in a Reference Manifest

There are a number of fields of information that is harvested into a reference measurement as represented by a Reference Manifest structure. The Reference Manifest structure itself is defined inside the TCG Integrity Schema [3].

The purpose of the Reference Manifest is to capture information regarding components (which may or may not itself be made-up of sub-components). The information captured within a Reference Manifest includes items pertaining to the component itself, such as the component ID, its manufacturing date, the manufacturer identification information and – in the case of software – the digest information.

Information regarding the component item includes its simple name, the model information (model name, number, class) which are generic to the model, as well a serial number which could be unique per component instance.

Various version data is maintained within a Reference Manifest to clearly identify the component in question. This is necessary since one version of a component may not have obtained some conformance status (e.g. against a standard), but a new patch issued for that version may bring the version into conformance. The version data maintained includes major/minor version numbers, build information, build date and patch levels.

Lastly, a Reference Manifest's usefulness may be dependent on who issues the Reference Manifest. Thus, a digital signature over the Reference Manifest must be applied, together with some pointer or reference to the certificate of the signer.

An important design-decision in the development of the TCG Integrity Schema [3] is the need for both the Integrity Report structure (to be delivered by the Requestor) and the Reference Manifest structure (to be compared against the Integrity Report structure by the Verifier) to share the same fundamental building block, namely the Component-ID substructure. A shared Component-ID substructure allows the Integrity Report to precisely identify which component (e.g. BIOS, Loader, Kernel, drivers, etc) was involved in a runtime measurement for a given use-case (e.g. Trusted Boot) and to allow that entity (or another) to retrieve the corresponding Reference Manifest structure for those involved components (for comparison/verification).

The TCG Integrity Schema is discussed further in Section 7.

6.2 The Reference Manifest Database

The process of making the contents of a Reference Manifest Database available is referred to as the *Publication* process (see Figure 8). In looking at the Reference Manifest Database, it is useful to distinguish Reference Manifest Databases provided by vendors and manufacturers (for their products and components) and Reference Manifest Databases provided by third-party (value-add) service providers or aggregators.

Figure 12 shows two general approaches for providing Reference Manifest Databases. In the first case (a), each vendor harvests their reference measurements (for their component products) and make them available to the Verifier in the vendor's Reference Manifest Database. For each component of interest, the Verifier must locate the correct vendor's Reference Manifest Database

and obtain the Reference Manifest Records from that database. Here, it is the vendor that digitally signs their Reference Manifest Records, and thus the vendor is the Reference Manifest Issuer.

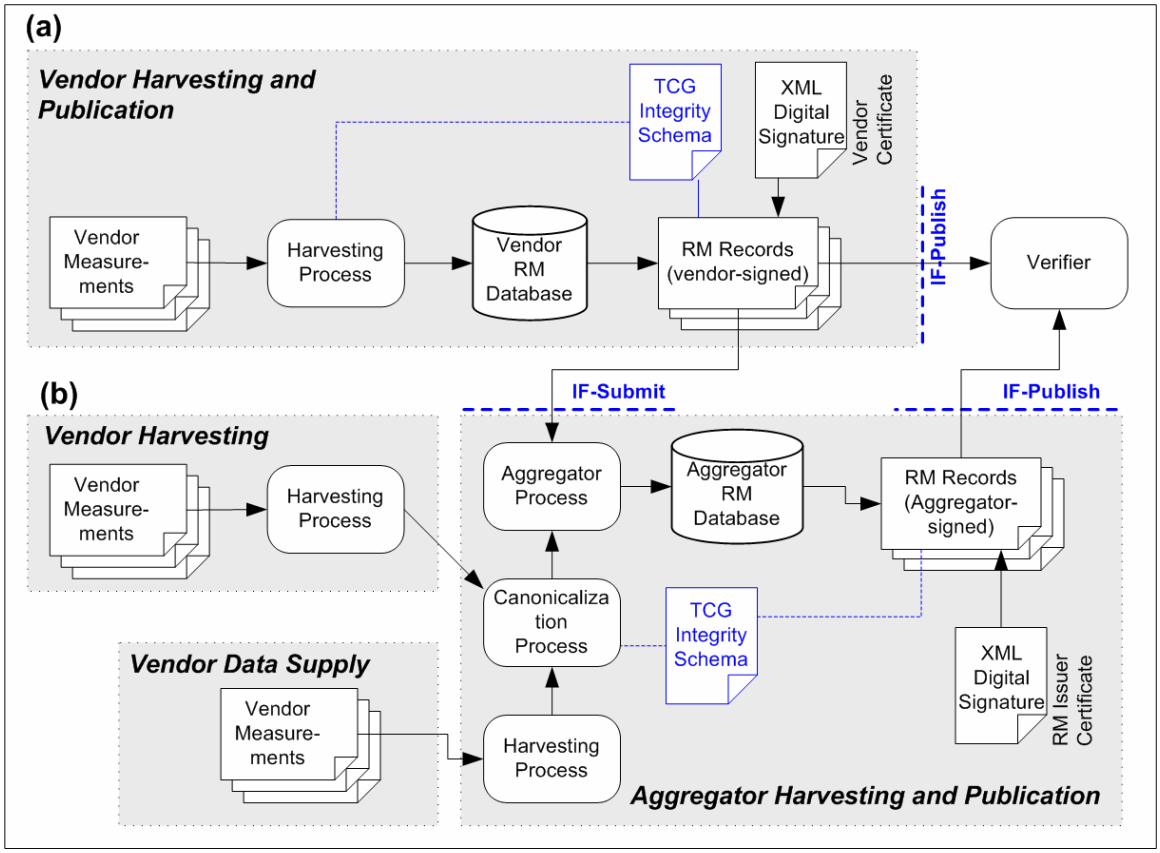


Figure 12: Reference Manifest Database approaches

In Figure 12 (b), a third party service provider acts as an Aggregator. Here, the Aggregator can play at least two roles. In the first case, the Aggregator can simply obtain Reference Manifests (in TCG canonicalized format) and re-publish or propagate them to the Verifier. The Aggregator can either choose to propagate the Reference Manifest Records in their original format (thus making the Aggregator a content availability entity or even a Content Distribution Network), or the Aggregator can digitally-sign the received Reference Manifest Records from the vendor. Note that when Aggregator digitally-signs Reference Manifest Records received from a vendor, it is assumed that the two parties have a legal agreement and that the Aggregator verifies the correctness of the Reference Manifest Records.

Another variation of the second approach is the situation where the Aggregator also harvests the measurements direct from the vendor. The vendor simply makes its "raw" measurements available to the Aggregator to be harvested. This is shown at the bottom of Figure 12 (b).

It is important to note that for interoperability the submission and publications interfaces (IF-Submit and IF-Publish) need to be used by entities in the supply chain. The Aggregator may need to implement both, since it may be receiving data into its Aggregator Process already in the Reference Manifest format. Furthermore, the use of a standardized submission and publications interfaces promotes the adoption of this aggregator role by existing data aggregators and content publisher existing today.

The current IMM model and architecture does not define the mechanisms used to implement or realize the Reference Manifest database. However, there are some aspects of the Reference Manifest Database that must be standardized in order to ensure interoperability. These are as follows:

- *Standardized data representation*: In order for the Reference Manifest to be useful for the entire computer and networking industry, a standardized data representation is required to ensure interoperability between Reference Manifest Providers and their consumers (Verifiers). It is for this reason the TCG has developed the Reference Manifest Schema specifications in [5].
- *Standardized web-based access interface*: Together with a standardized data representation is the need to define a standardized interface to access the Reference Manifest Database service, which should be independent of whether the Reference Manifest Database is provided by a manufacturer (e.g. for their components) or operated by the third-party (non-vendor). To this extent, the TCG IWG is in the processes of developing basic web-services based interface that will allow access to the Reference Manifest Database.
- *Standardized Practices Statement*: When the Reference Manifest Database is provided as a third-party service by a non-vendor, it is expected that the Reference Manifest Provider (or Reference Manifest Authority) provides a *Practices Statement* reminiscent to that found in the PKI industry (ie. for Certificate Authorities – see RFC2527). This is particularly important when the Reference Manifest Provider also performs the harvesting process and is the issuer (signer) of the Reference Manifests. The Practices Statement is needed because (analogous to Classic CAs vouching human certificates), the Reference Manifest provider is vouching for the correctness and accuracy of the information captured in the Reference Manifest. Such a Practices Statement is useful for the consumer of the service to understand the level of service provided and the liabilities covered by the Reference Manifest Provider.

6.3 The Role of Reference Measurements in the Supply-Chain

The use of Reference Manifest records is not limited to use-cases where runtime measurements need to be compared against Reference Manifest records. The TCG recognizes that an important part of achieving trust in platforms is to know the origins of the platform and all its components (both software and hardware). The term often used in TCG specifications and other documents is *provenance*, while another similar term (used often in the computer forensics domain) that can be applied is *Chain of Custody*.

The notion of using reference measurements in the equipment supply-chain interaction is described – albeit in a simplified form – in Figure 13. Here, the IT manager produces a purchase order for equipment, as determined by the organizational policy using the information supplied by the platform manufacturer or vendor (Steps 1 and 2). The manufacturer creates the platform and corresponding Reference Manifests (Step-3), and ships it to the organization to fulfill the order (Step-4). Upon receiving the platform and Reference Manifests, the IT manager performs verification that indeed the Reference Manifests match the shipped platform (Step-5). The IT manager then completes the internal order-tracking system to indicate that the order has been fulfilled, and stores the Reference Manifests that accompanied the platform in its local Reference Manifest repository (Step-6).

Note it is here that the Reference Manifest schema [5] plays an important role as a standardized format to represent the component information from the various suppliers in the supply chain. The product information in the form of an accompanying (signed) Reference Manifest from a supplier indicates both the supplier's endorsement of the component product (by digitally-signing the Reference Manifest) and the provenance of that component product. From the pure supply-chain mechanics perspective, the standardized Reference Manifest format facilitates the order-tracking

mechanism on the part of the platform manufacturer. From a platform trust establishment perspective, having Reference Manifests for every component that goes into a platform allows the platform manufacturer to convey the same Reference Manifests (plus its own platform-Reference Manifest) to the end-consumer (e.g. Enterprise). The burden of proving provenance for the entire platform (covering all components) is then shared by the entire computing industry supply-chain, and not only by the platform manufacturer or OEM.

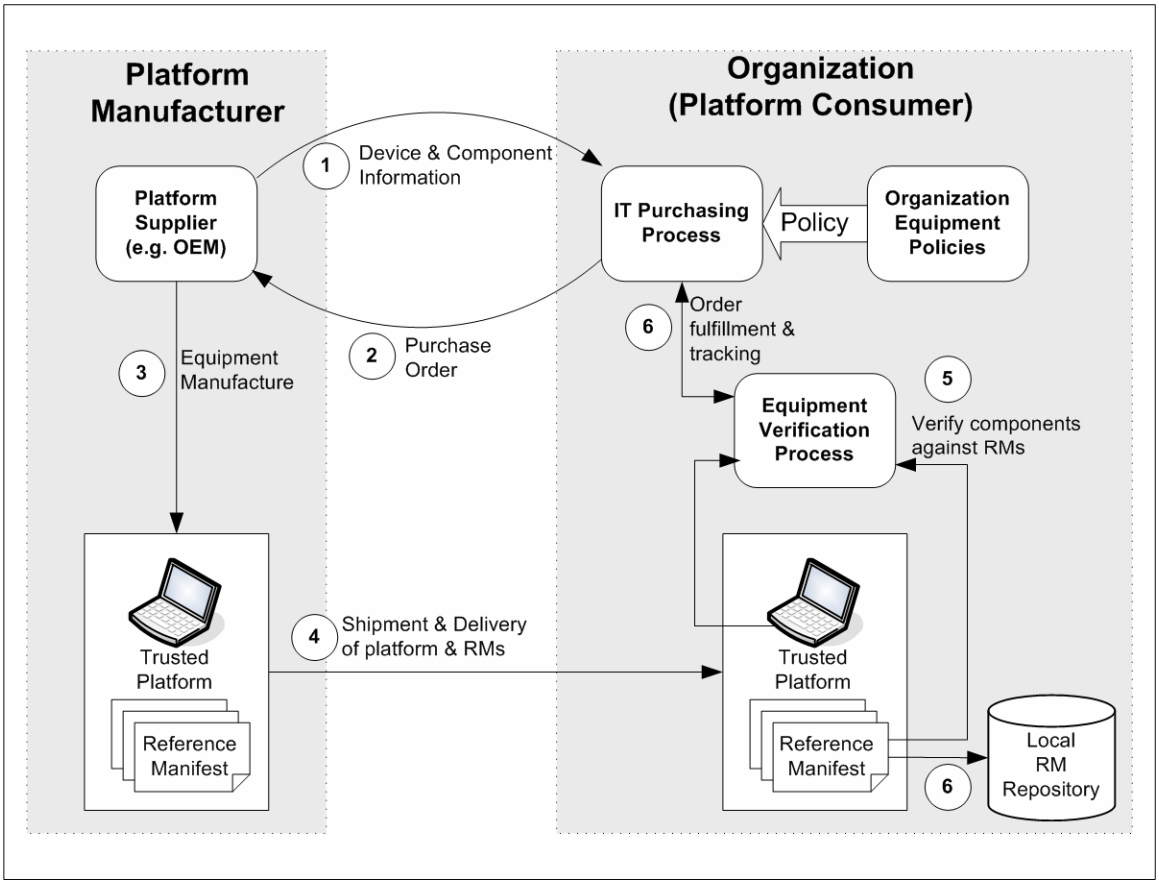


Figure 13: The use of Reference Measurements in the equipment Supply-Chain

The TCG understands that requiring component-level Reference Manifests for every piece of component making-up a trusted platform is a revolutionary undertaking. As such, the TCG Integrity Schema – notably the portions dealing with components – allows different granularity of components to be defined. That is, the Integrity Schema defines a *component* to be made of multiple *sub-components* (which is in-turn defined to be *components*). This nested (but uniform) structure allows supply-chain participants to begin addressing Reference Manifests from a granular level initially. For example, a network NIC hardware vendor (with various supplied sub-components) may initially ship only a Reference Manifest for that hardware without other sub-component Reference Manifests. Thus, the NIC vendor may initially depend on its business relationship with its suppliers and not demand Reference Manifests from these individual suppliers. In essence the NIC vendor is endorsing the product on behalf of its suppliers (which may be sufficient for certain platform markets). However, over time, the NIC vendor may eventually require its component suppliers to also deliver a Reference Manifest for their component, especially if the same suppliers also cater for other (competing) NIC vendors.

It is important to note that the need for finer level component Reference Manifests (i.e. many of Reference Manifests for every sub-component) may be dictated by the target market of the platform (e.g. SMB Enterprise, large enterprise, Health industry, Government, Defense, etc). The role of the TCG in this case is to make the standards for Reference Manifests available and allow each market to determine its level of need for Reference Manifests. The crucial aspect here is that the Reference Manifests allows the *communication of provenance information* throughout the supply-chain (regardless of the granularity level of the information). Establishing industry standards and facilitating this deployment in the industry is indeed one of the key aims of the TCG as an industry consortium.

7 The Integrity Schema

In order to capture and represent runtime measurements and reference measurements, the IWG has developed a number for specifications defining data formats in the form of XML schemas. The reason the XML schema was selected was driven by the realization that increasingly semantically-rich transactions on the internet were occurring at the "web layer" (i.e. above HTTP) and that the TCG must rely on existing and future infrastructures – which are increasingly being developed as web-services. Although defining formats for runtime measurements and reference measurements in XML schema does not necessarily imply an automatic use or reliance on web-services, it was considered wise to select XML schema to ensure a high degree of interoperability of components through integration with existing web-services related services standards. This choice also reflects the realization that many older service infrastructures (e.g. EDI, PKI, etc) are being migrated to newer services infrastructures (e.g. EDI-XML, eb-XML, XKMS, SAML, etc) with added functionalities and other benefits.

The set of XML schemas in the TCG pertaining to runtime measurements and reference measurements are referred to as the *TCG Integrity Schema* [3], which consists of a number of derivative schemas. This is shown in Figure 14.

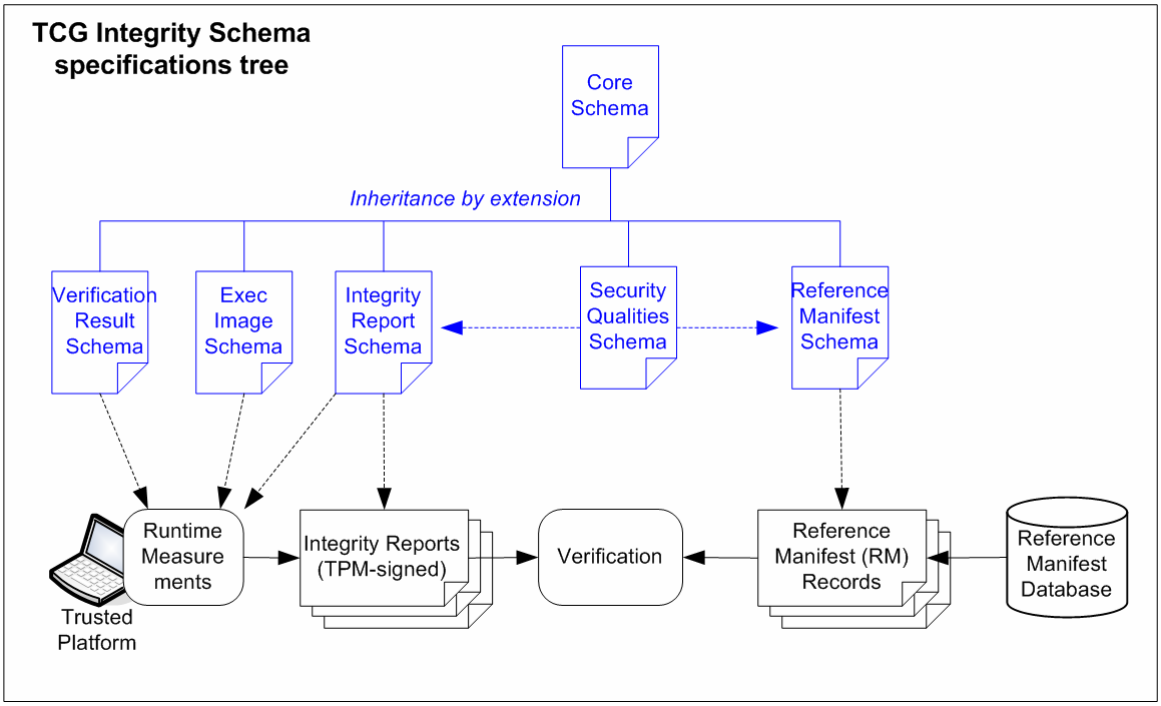


Figure 14: TCG Integrity Schema specifications and relationship

7.1 Motivation for using XML Schemas

There a number of benefits to using a standardized schema definition for component information:

- *Interoperability*: By providing a standardizes schema, a Requestor and Verifier engaging in an integrity check handshake (as part of Platform Authentication) can attain some assurance that integrity data being exchanged in the handshake will be syntactically and semantically meaningful to each other. Vendors of components therefore derive a direct benefit from standardization of the format, as they will only need to publish integrity-related data regarding their products once (in the standard format).

- *Extensibility*: The TCG Integrity Schema set for representing integrity data is extensible through the inherent properties of schemas based on XML. This extensibility allows the Integrity Schema to evolve following the evolution of the technologies and industries supplying components for trusted platforms.
- *Manageability*: Since the Integrity Schema is based on XML and since XML is increasingly becoming the high-level language for expressing industry-specific needs for standardized formats and processing, numerous tools and XML-processing capabilities have been developed. This provides the best opportunity for the Integrity Schema to be integrated into those existing tools and environments.

The TCG Integrity Schema consists of several schemas, as shown in Figure 14. These are discussed in the following.

7.2 The Core Schema

The *Core Integrity Manifest Schema* [3] (or Core Schema for short) defines code and configuration integrity management primitives that support the *Integrity Report Schema* and *Reference Manifest Schema*. Other schema designers may find these primitives helpful when defining application specific schemas that are extensions of integrity reports or Reference Manifests.

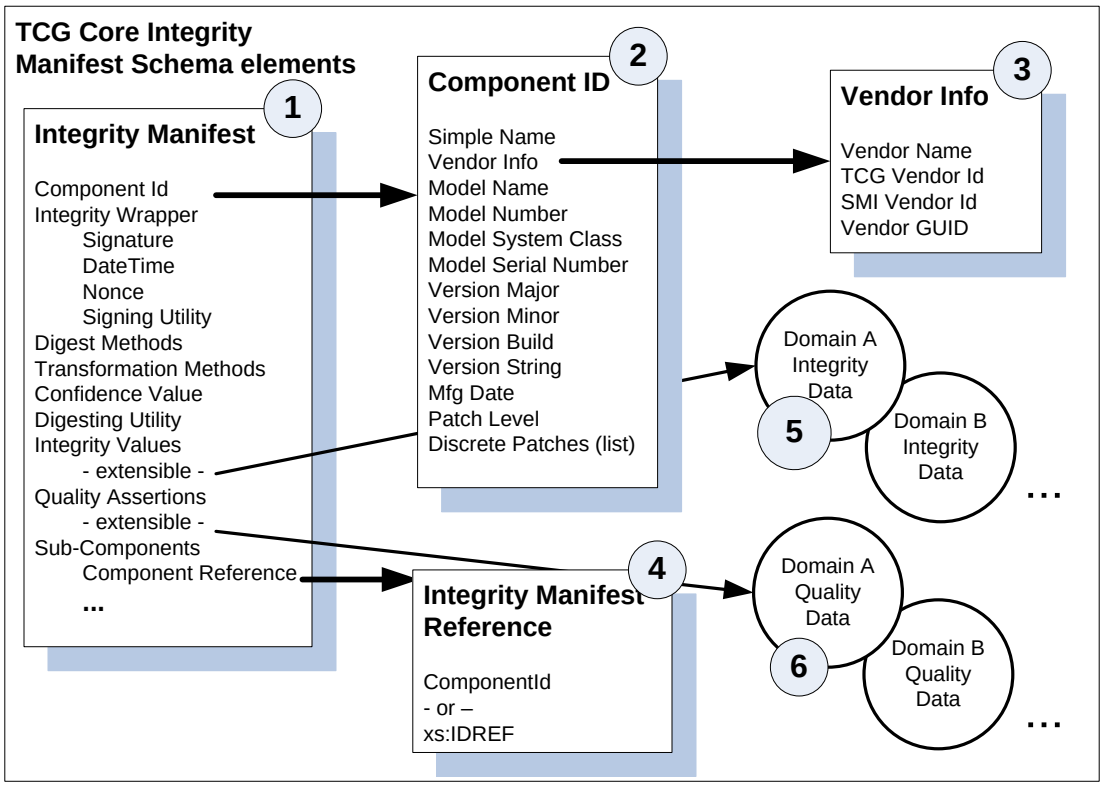


Figure 15 - Core Integrity Manifest

The *Core* schema, as a common element to the Integrity Report and Reference Manifest schemas satisfies an important requirement that verification proceeds according to a common understanding of the platform model. A verifier relies on the cooperation between measurement agents on the platform that collect actual measurements with manufacturers and platform configuration data to agree to on syntax and semantics of integrity data to be effective. The Core

schema provides a common understanding of measurements and how they relate to platform components.

The Integrity Manifest complex type defines the integrity manifest type that is inherited by Snapshot and Reference Manifest schemas. The Integrity Manifest (see Figure 15) contains a number of elements.

The core schema ComponentID (Item-2) is used to identify a manufactured item or any object that can be placed under revision or configuration control. The intent of a ComponentID is to associate an identifier with hardware, software and configuration data. A ComponentID need not be specific to an instance of a component (e.g. item serial number) since quality controlled mass production techniques generally provides assurance that one instance of the component is the same as another.

The ComponentID has the following attributes and element:

- Record Instance identifier
    - *ID*: Uniquely identifies the record instance such that it can be distinguished from other records possibly having the same ComponentID value – recommended globally unique.
- Component model information:
    - *SimpleName*: String version information for simple compare operations.
    - *ModelName*: Model name with which the component is marketed.
    - *ModelNumber*: Alphanumeric model number with which the component is identified
    - *ModelSystemClass*: Vendor-specific system type or environment with which the component is associated
    - *ModelSerialNumber*: Alphanumeric model serial number with which the component is identified. Serial numbers are included only in rare cases where customizations have been applied that are unique to an instance of the component. (e.g. pin on an IC has been clipped)
- Component Version Information:
    - *VersionMajor / VersionMinor*: Major/Minor version numbers of the component
    - *VersionBuild*: Build number of the component
    - *VersionString*: String with which the component's version may be identified
    - *PatchLevel:* Patch level of the component.
    - *DiscretePatches*: Identifiers of isolated patches that have been applied to the component.
- Manufacturing Date:
    - *MfgDate*: Date and time that the component was manufactured
- Vendor Identity:
    - *Name*: A string representation of the component vendor name.
    - *TcgVendorId*: A TCG assigned vendor ID.
    - *SmiVendorId*: A SMI Network Management Private Enterprise Code http://www.iana.org/assignments/enterprise-numbers
    - *VendorGUID*: A globally unique identifier that is used to identify non-vendor entities (e.g. IT departments) that may construct custom platforms

The Core schema (see Figure 15) can represent a platform model at varying levels of granularity. A platform being composed of other hardware and software components relies on manufacturing processes that in turn revision controls the sub-components (Item-4) that make up the platform. The sub-components may further be constructed of sub-components and so on. There is no theoretical limit to the number of layers of hardware and software sub-components that defines a platform. Platforms may evolve during the course of their deployment lifecycle resulting in the addition or modification of sub-components. The IWG Integrity Manifest therefore, allows for an arbitrary number of references to sub-components that in turn conform to the IWG Integrity Manifest schema.

A primary objective of the Core schema is to represent integrity information that falls into two classes; a) cryptographic hash of code or data (Item-5) and b) quality assertions (Item-6) about the component that a verifier can use to better asses risks associated with transactions involving the platform. Because the structure of code, data and quality assertions can be highly domain specific, the integrity manifest schema is extensible. Domain specific schemas can be incorporated into the manifest structure as needed. In cases where there is sufficient interest in a particular domain the IWG may define domain specific schemas. Schemas that appear to have broad applicability include:

- *IWG Simple Object Schema* [7] which defines a representation for measurements of list of files.
- *IWG Security Qualities Schema* [6] which defines quality attributes germane to trusted computing.

An additional schema – referred to as the Executable Image Schema, which defines a representation for binary formats such as ELF and COFF – will be specified by the IWG.

The Core schema contains boiler plate elements (Item-1) that support collection and reporting of integrity values. These include the following:

- Message Digest Creation:
    - *DigestMethod:* For each hash algorithm used to compute a digest, an algorithm identifier is specified.
    - *TransformMethod:* For each hash computation a transformation method may be applied to ensure verifiers can exactly recreate the hash value.
    - *Collector:* The utility performing the hash computation is described in terms of its ComponentID and may be included in the manifest for forensic and chain of custody purposes.
- Manifest Issuer:
    - *SignerInfo:* The manifest should be signed to allow verifiers the ability to determine if the measurement is from a reliable source.
    - *SigningComponent:* The utility applying the signature is described in terms of its ComponentID and may be included in the manifest for forensic and chain of custody purposes.
    - *ConfidenceValue:* The signer / collector may have performed the collection process according to procedures having varying degrees of assurance. Confidence value allows the collector to apply a weighting factor to the report. Additional details related to specific processes followed can be referenced.
    - *PlatformClass:* The PlatformClass information qualifies the component according to the type of platform to which the component belongs.

7.3 The Integrity Report Schema

The *Integrity Report Schema* [4] is a highly flexible XML schema which defines the instantiation of interoperable integrity reports and snapshots including TPM Quote-related data structures, for the purpose of a relying party determining provable trustworthiness of a particular cone of execution. Collectors nominally will generate reports for their corresponding Verifier and provide these reports in the interoperable Integrity Report Schema syntax. The PTS client-side service is responsible for the creation of these integrity reports based on a Collectors requests, and the PTS server-side service is responsible for validation of the integrity reports based on coordination with the corresponding Verifier. The Integrity Report Schema is derived from the core schema [3].

Snapshot structures are defined which contain measurements and assertions for Components, where a Component is a set of one or more items which a Verifier is interested in evaluating. Integrity Reports contain a combination of snapshot data and TPM Quote data, linking the snapshot data to a trusted measurement process through the use of the TPM. Components described by snapshots may be logically related by tree structures, and thus may cover a cone of measurement of trust and execution. Components described in the Integrity Reports may include all of portions of the transitive trust chain, or all or portions of a application cone of execution. Additionally, Components may be instantiated which describe the TPM, PCRs, the platform, or other unique capabilities of a trusted platform, with the ability for a measurement agent to qualify the reports with security or quality assertions.

One use of integrity reports and structures is in the Trusted Network Connect (TNC) [9] use models whereby the Platform Trust Service (PTS) creates integrity reports and snapshots to be sent by IMCs to their corresponding IMVs for verification of acceptable platform state prior to network access. Similarly, other services may use this interoperable report format to query and validate the state of clients, such as Web Services that are interested in the integrity of their corresponding client side components.

The Integrity Report schema has been defined in such a fashion as to allow minimalist implementations with only basic required report data, or very complex representations with high degrees of detail. The usage of these optional elements is left to the system implementer.

7.4 The Reference Manifest Schema

The *Reference Manifest* (RM) schema [5] – which inherits by extension from the core schema – defines the structure to hold reference measurements that are obtained through the IMM harvesting processes (see Section 4).

The Reference Manifest schema can roughly be understood as consisting of two sets of information (see Figure 16). The first pertains to the information about individual components, and covers attributes such as the component model, name, version, serial-number, and so on. This information is captured in the *ComponentIDtype* structure. Surrounding this component-specific information is the metadata that pertains to the capturing (harvesting) of the component information. This is represented by the *IntegrityManifestType* structure in the schema. The metadata captured includes the collector (harvester) method used, the Reference Manifest signature (and associated signer/issuer information), digest value(s) over the component (for software), confidence levels, assertions made and others.

It is important to note that here that a component (as identified by a ComponentID) may be made-up from one or more sub-component, each of which is defined using the same *ComponentIDtype* structure. This nested approach allows essentially a “component tree” to be constructed, starting at the root of the tree (the largest component unit, such as the entire platform) down to the smallest unit of components on the platform (e.g. theoretically the transistors on the mainboard). The nested approach provides a path for industry participants to begin to issue Reference Manifests from the root of the “component tree” towards its leaves. That is, from the component aggregators (i.e. OEMs) upstream to the component suppliers. Thus, an OEM could begin issuing a Reference Manifest for its complete product, without demanding its suppliers upstream to also

begin shipping components with Reference Manifests. In doing so, the OEM is essentially standing on behalf of its suppliers in stating that the Reference Manifest (and assertions) it is producing represents its endorsement of its product.

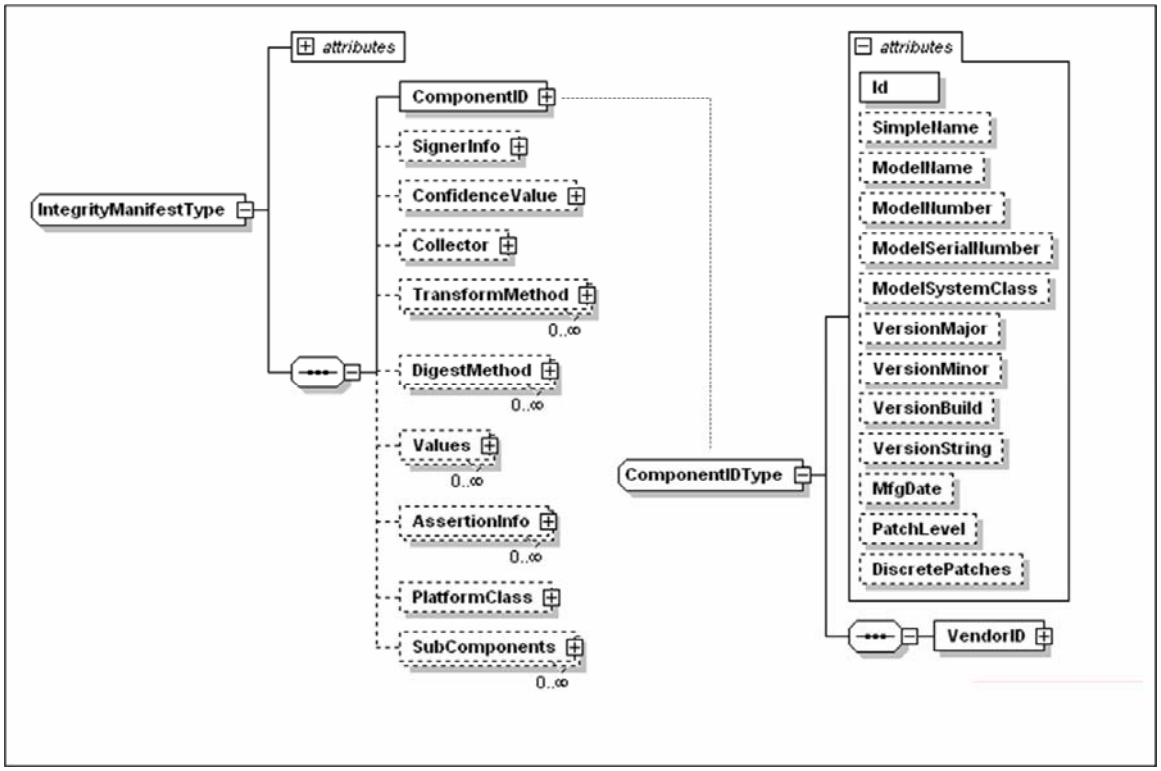


Figure 16: Summary of component information represented in a Reference Manifest

7.5 The SecurityQualities Schema

The *SecurityQualities* Schema [6] defines a method for including a set of quality assertions about a component with its reference measurements (i.e. in Reference Manifest). These quality assertions enable a knowledgeable party (e.g. manufacturer) to make protected claims to a verifier about how the component was created and offer evidence of the components trustworthiness for aspects of the component that can not be measured by cryptographic operations. The SecurityQualities Schema is designed to be used in conjunction with other schemas derived from the Core Integrity Schema using the IntegrityManifestType's AssertionInfoType element.

For example, a manufacturer creating a set of Reference Manifests for a new product release might wish to assert that the product's manufacturing process was compliant with ISO 9000 and that the resulting product has received an EAL-4 Common Criteria certification against a particular protection profile. Clearly this information can not be measured by digesting of the product's code into a Reference Manifest, so the manufacturer would include these qualities via this schema.

The SecurityQualities schema makes use of XML to allow for future extensibility. The initial version of the schema enables assertion of the following qualities about a component:

- Common Criteria evaluation circumstances and results
- ISO 9000 manufacturing process metrics employed

- FIPS Compliance

7.6 Executable Image Schema

The Executable Image Schema will define the format for reporting the contents of an executable binary file (e.g. COFF or ELF) used by many major operating systems. A typical binary file is internally composed of a number of different sections that tell the run-time loader how to load the file into memory and properly execute it. For example, a binary file may contain sections including: executable code, comments, string tables, symbol tables, relocation information and pre-initialized data. The loader needs to understand and make use of these sections to properly load and prepare the binary for execution. Some of these sections affect the trustworthiness of the execution of the file (e.g. the executable code) while others may not (e.g. comments.) This schema allows the verifier to have more granular visibility into binary files to assess only those that affect the trust decision.

It's envisioned that this schema might be built upon in the future to define the current state of a running process so that a verifier may inspect the currently running state of the system (not just its load-time state.) This scanning of the state of currently running processes is very operating system loader specific and requires a great deal of care to specify correctly yet generically. The state of a running process involves more information than what is statically available within a COFF or ELF file. Processes may dynamically load libraries or use platform specific optimizations to improve operation which might affect its measurement in ways that are difficult to predict in a manufacturer's Reference Manifest. As a result, the memory-based scanning and reporting of running processes is left as a topic for a future specification. Architecturally, the IWG believes it has defined an environment allowing vendors to innovate such solutions but does not yet define a single operating system neutral approach/representation.

This schema will be designed to be used in conjunction with other schemas derived from the Core Integrity Schema using the IntegrityManifestType's ValueType element. This element provides a hook for applying a schema to describe the measurements for a wide variety of types of components including the contents of an executable file.

8 Platform Trust Services

As mentioned above, integrity management involves collecting the component-measurements (snapshots) that make-up the trust chain within a live or running platform (referred to as runtime measurements), and comparing these against known good (reference) measurements of those same components as supplied by their respective manufacturers or other trusted third parties. The function of collecting the component-measurements and reporting them within an integrity report is assigned to a special component within the trust platform known as the Platform Trust Services (PTS).

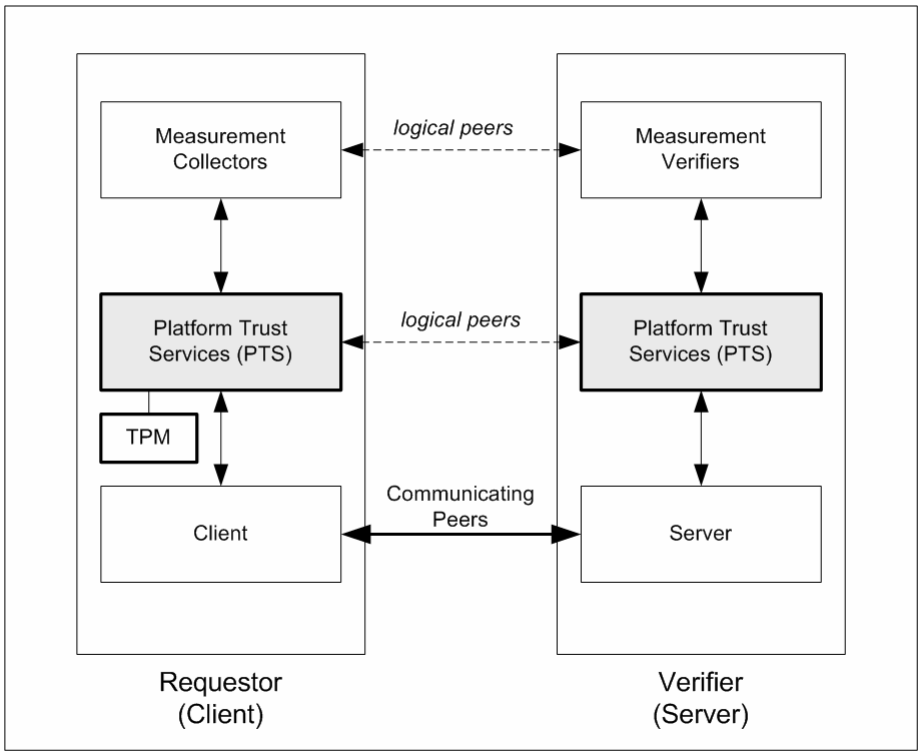


Figure 17: Architecture of PTS

The PTS has the task, among others, of performing the following:

- *Inclusion of components in the trust chain*: The PTS provides the capability to select hardware, firmware and software components (that make-up a trusted platform) that participate in the transitive trust chains computed for that platform. The platform Owner (e.g. IT administrator) may also use the PTS to perform local verification at boot and/or runtime to restrict what components are allowed to be used (e.g. software allowed to execute) on the platform.
- *Computation and collection of integrity measurements application components*: The PTS has the specific task of computing the measurements of requested components and the creation of an integrity report containing these measurements. An example use of these features might occur within the Trusted Network Connect (TNC) architecture [9]. The TNC architecture includes *Integrity Measurement Collectors* (IMC) that collect application specific measurements, and *Integrity Measurement Verifiers* (IMV) that verify these measurements. It is possible for an IMC and IMV pair to exist that leverage the PTS to provide TPM anchored assurances of about the security posture of the client system and the trustworthiness of the other TNC components.

- *Formatting of integrity measurements*: The PTS also has the task of formatting the integrity measurements collected from the measured components into a standardized format (namely the Integrity Report schema) for the purposes of reporting these measurements to a remote relying party. In the specific context of the TNC use-case using Platform Authentication, the PTS running on the Client (i.e. Network Access Requestor) will collect the measurements for the client components with the aim of being consumed by the PTS component on the Server side (i.e. Policy Decision Point or PDP).

8.1 Basic Architecture of PTS

The basic architecture of the PTS is shown in Figure 17. The figure is not symmetrical in that it shows only the Requestor (client-side) being the platform whose integrity is evaluated by the Verifier (Server-side). In order to have mutual Platform Authentication (with mutual integrity verification), the roles need to be reversed. Note also that the Verifier entity does not require TPM hardware in order to be able to evaluate the Integrity Report sent by the Requestor.

There are a number of components to the architecture:

- *Measurement Collectors and Verifiers*: The Collector is responsible for commanding the measurement collection on a candidate (target) platform or providing measurements and/or assertions to PTS for inclusion in Integrity Reports. Verifiers are responsible for requesting measurement evaluation or performing measurement evaluation for one or more candidate platforms, respectively.
- *PTS Client-side and Server-side*: The PTS client-side component is responsible for the actual measurements, as requested by one or more collectors, and aggregating the measurements into a standard snapshot or Integrity Report format. The PTS server side component is responsible for validating the authenticity and integrity of an Integrity Report against a collection of Reference Manifests, and verifying the data within the snapshot or Integrity Report as requested by a Verifier.
- *TPM*: The TPM provides a set of protected Platform Configuration Registers (PCR) which are used to hold the measurements made during establishment of the transitive trust chain, and providing the quoting keys used to Quote the PCR structures and signing keys for snapshots and Integrity Reports.

8.2 Role of PTS in the Transitive Chain Establishment

It is important to understand the place of the PTS in the transitive trust chain of a trusted platform in order to understand its role during that platform's runtime in performing measurements (i.e. runtime measurements).

Since the PTS is the component that will be used (by other components designed to measure themselves and other components), it is important that the PTS is designed and implemented correctly, and that the PTS itself is trustworthy. Thus, ideally the PTS component should be measured as part as Load-Time Measurement within the Trusted OS, prior to user-initiated processes running. This allows the PTS component measurement to be stored inside the Integrity Measurement Log in a position that is relatively "early" in the sequence of stored entries in the log, thereby allowing application level services to make use of a PTS which was validated as part of the transitive trust chain. The goal is to have the PTS component always be part of the transitive trust chain established at boot-up.

There are a number of benefits to having the PTS component be configured as part of the transitive trust chain established at boot-up. In the first instance, when the Owner (e.g. IT Administrator) configures the PTS to be part of the load-time components to be measured, the Owner has assurance that at each boot-up the PTS will be measured regardless of which subsequent process or code calls the PTS to perform other further measurements.

Secondly, having the PTS as part of the "permanent" set of components (that make-up the transitive trust chain) at boot-up allows other processes (e.g. Measurement Collectors) to have a

dependency on the PTS without concern that it will not have yet been started. This provides the benefit of efficiency in the creation of Integrity Reports. A Measurement Collector code or agent has the assurance that it does not need to re-compute the transitive trust chain (from the RTM), since it can trust the PTS to retrieve the measurements (pertaining to the transitive trust chain created at load-time) from the *Integrity Measurement Log* (IML). See Figure 18. Furthermore, it can trust the PTS to perform further measurements of other components on the platform (including the Measurement Collector itself).

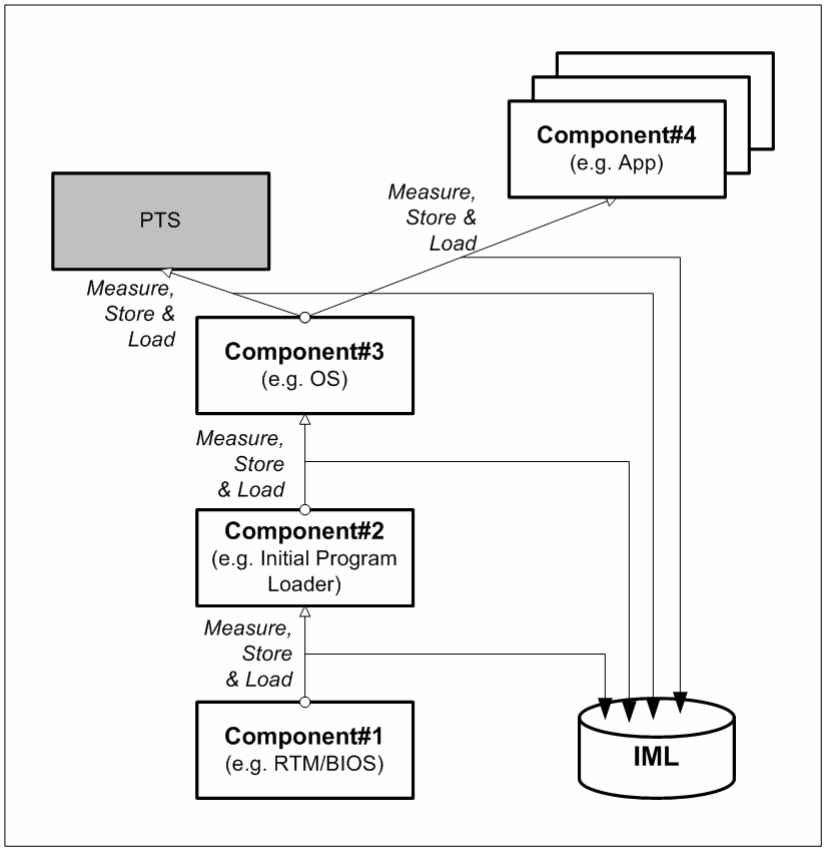


Figure 18: The place of the PTS in load-time measurements

8.3 Role of PTS in generating Integrity Reports

The role of the PTS in reporting the integrity measurements of a platform (to an external party or verifier) is illustrated by an example in Figure 19. Here, the Client-Server interaction is used, in which the Server (i.e. Verifier) seeks to evaluate the integrity of the Client (i.e. Requestor). A Measurement Verifier entity (on the Server) instructs a Measurement Collector (on the Client) to collect measurements regarding the client platform and to provide it with an Integrity Report pertaining to the client platform. The general steps involving the PTS are described in the following:

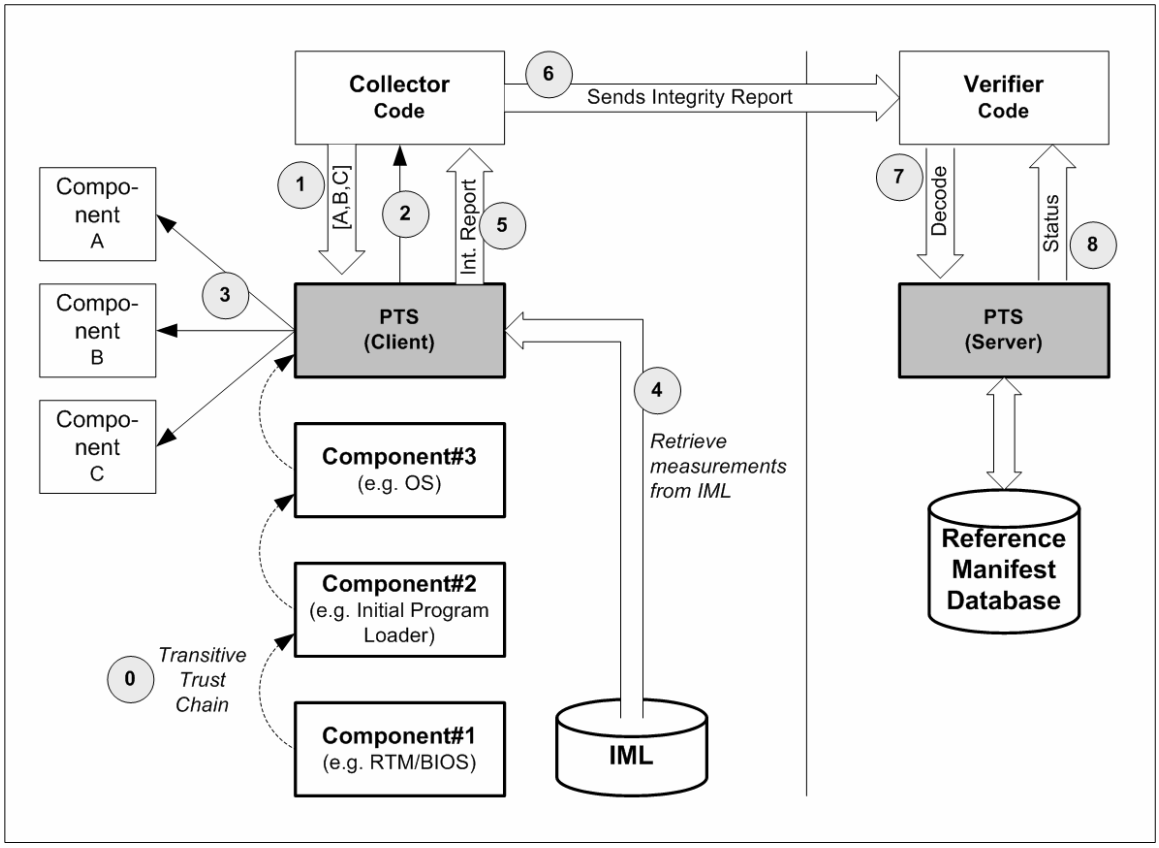


Figure 19: The Role of PTS in runtime Integrity Report generation

Step-0: The Requestor's machine is assumed to be in operation, and the Verifier is assumed to have instructed the Requestor to provide it with an integrity report (see Figure 3). Before the PTS on the Verifier can be of use to other components (i.e. on the client platform), the platform itself must have booted into a secure state and have stored the load-time measurements (that make-up the transitive trusts chain so far) into the IML and the TPM. Here, it is assumed that the PTS itself is part of the load-time measurements.

Step-1: A Measurement Collector instructs the PTS to perform measurements of components A, B and C (indicated by Component-ID) that may comprise an application or OS component of interest to the Verifier. Note that in order to make the PTS's measurements of value to an external verifier, the PTS itself needs to prove its own trustworthiness by showing that it is part of the load-time transitive trust chain.

Step-2: In this example, the first task that the PTS performs prior to measuring components identified by the Measurement Collector (ie. component A, B and C) is to measure the Measurement Collector itself.

Step-3: After measuring the Measurement Collector, the PTS proceeds to measure components A, B and C, as requested earlier by the Measurement Collector. Note that all measurements by the PTS are stored in the IML.

Step-4: The PTS then proceeds to retrieve measurements (i.e. "snapshots") from the IML and organize these as part of the Integrity Report. Note that the report will be formatted according to the Integrity Report Schema [4].

Step-5: The PTS then returns the Integrity Report to the calling Measurement Collector.

Step-6: The Measurement Collector sends the Integrity Report to the Measurement Verifier entity on the Server.

Step-7: The Measurement Verifier on the Server then uses its corresponding server-side PTS to parse through the Integrity Report. That PTS may communicate with the Reference Manifest Database in order to obtain Reference Manifest records corresponding to components A, B and C (and others) that are of interest to the Measurement Verifier.

Step-8: After the server-side PTS completes its evaluation of the Integrity Report, it reports the status of the components to the Measurement Verifier.

8.4 Protocol between PTS-IMC and PTS-IMV

Because the verification of the integrity of a remote client system may be a complex decision, the verifier may need to evaluate the measurements of a large number of components running on the system. For example, to evaluate whether a client system is trustworthy to perform a web transaction, the verifier may need to evaluate the clients browser and its dependent libraries, portions of the operating system, the kernel and the transitive trust chain including the PTS. In this instance, the verifier may wish to ask the collector to perform multiple integrity reports each covering different aspects (components) of the client system. By standardizing an interoperable protocol which allows for communications between PTS-IMC and PTS-IMC [8] components, this will allow server-side verification component to make specific requests of the client side PTS for presentation of static transitive trust measurements, periodic re-measurement of transitive trust measurements, executable measurements, and rule-based self validation, amongst other capabilities. This protocol is the subject of a future specification.

9 Policy Setting & Enforcement using Measurements

Besides communicating component provenance information, measurements as represented by Reference Manifests and Integrity Reports can also be used to enhance existing policy setting and enforcement scenarios (as determined by the specific use-cases). More specifically, reference measurements can be used during the process of policy authoring and during policy evaluation (decision making).

9.1 Role of Measurements in Policy Authoring

The detailed knowledge and information regarding components that make-up a platform allows the policy authority to specify rules that include specific platform component information. Thus, for example, using Reference Manifest information about components – such as TPM hardware, BIOS, OS, 802.1X Supplicants, VPN clients – the policy author is able to write network access rules that call-out specific platform configurations that must be on a platform before that platform is allowed on the network. Similarly, expected runtime measurements can be input into the policy authoring process.

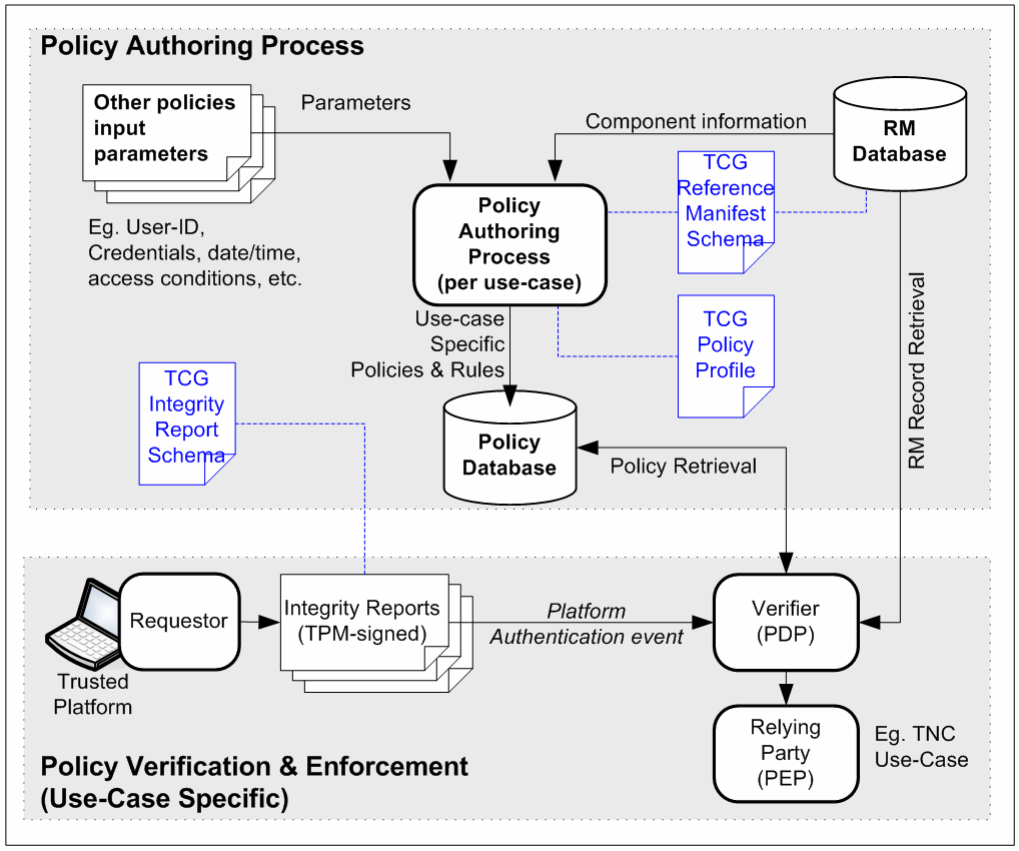


Figure 20: Role of Measurements in Policy Setting

In the context of trusted computing the combined use of runtime measurements and reference measurements allows for at least two types of measurements-based policy setting:

(a) *Verification of the existence of particular components*: Policy authors can specify that platforms must possess certain components (hardware and/or software). Later, during policy verification and enforcement, TCG technologies (e.g. such as the PTS [8]) can be used to validate that the platform in question indeed possesses the required components.

(b) *Verification of the runtime state of a given platform*: Policy authors can specify that a given platform must be in a particular runtime state, as reported by trustworthy measurement processes in the platform.

As an example, reference measurements enhance device access control policies because they allow the policy author finer granularity of component identification and usage in policy-setting. Thus, to take the network access control use-case (e.g. TNC), the network administrator (as the network policy author) can specific that only platforms containing a specific NIC model and containing a certain version of an OS can gain access to the corporate network. Later, when a client platform (Requestor) seeks network access, using the runtime measurements (and the PTS – see below) the client platform can prove that it possesses and is using the specified NIC model, and that is running a given OS version.

9.2 Relationship with Configuration Management & CMDB

One of the attractive uses of measurements is in relation to *Configuration and Change Management.* More specifically, the availability of information regarding components (reference measurements) and the expected integrity state of a good platform (via runtime measurements) complements configuration management by allowing a finer level of control over the configuration of a trusted platform – both at initial deployment time and throughout its lifecycle. Runtime measurements can thus be used to verify that a platform is in the expected configuration.

Today there are a number of tools and mechanisms used in organizations (e.g. Enterprises) to plan and control the deployment of software across numerous target platforms. One of the aims of Configuration Management is to allow for new target platforms to be deployed quickly and efficiently with the so called “gold image” that contains the policy-approved set of softwares (e.g. OS, drivers, applications, etc). In addition, it is also a goal to allow for efficient change management, whereby new patches and new softwares can be applied to the target platform(s) from a remote management point (e.g. admin console). Central to configuration and change management is the *Configuration Management Database* (CMDB), which contains information regarding the gold-image files, updates and changes done to the target platforms. Note that in many instances, a CMDB solution may include the so-called *Definitive Software Library* (DSL), which defines the authorized softwares that is permitted (by organizational policy) to be run on the platforms in the organization.

It cannot be overemphasized enough that change and configuration management is absolutely crucial to the system operations within an organization, and thus to the business survival of that organization. Indeed, configuration management is today a growing sector within the broader computing and network industry. It is the TCG’s aim to make-use of existing standards and solutions for configuration and change management in order to manage trusted platforms and their unique properties.

Figure 21 attempts to show a simplified relationship between configuration management and policy authoring/enforcement, with measurements (runtime and reference) as representing a common aspect of both. The process of policy authoring (which is dependent on the use-case) may use data as found in the Reference Manifest Database (for component information) and in the CMDB (for how those components are configured).

(1) *Policy Authoring Process*: Policy authoring/setting involves taking both platform configuration information (from the CMDB/DSL) and authentic component information (from the Reference Manifest Database) – together with other related policy parameters – into the overall policy creation for a given use-case. Thus, for example, in the network access control (TNC) use-case, the configuration information could be settings and config-files that must be installed on the client (Requestor) and server (Verifier). The Reference Manifest data would pertain to the hardware and software that make-up the platform. The policy authoring might mandate that certain components (e.g. certain version of drivers and NIC cards) must be used in the organization with a given configuration setting. Other policy parameters may include rules concerning user-IDs, credentials, time-of-access, allowable VLANs, etc.

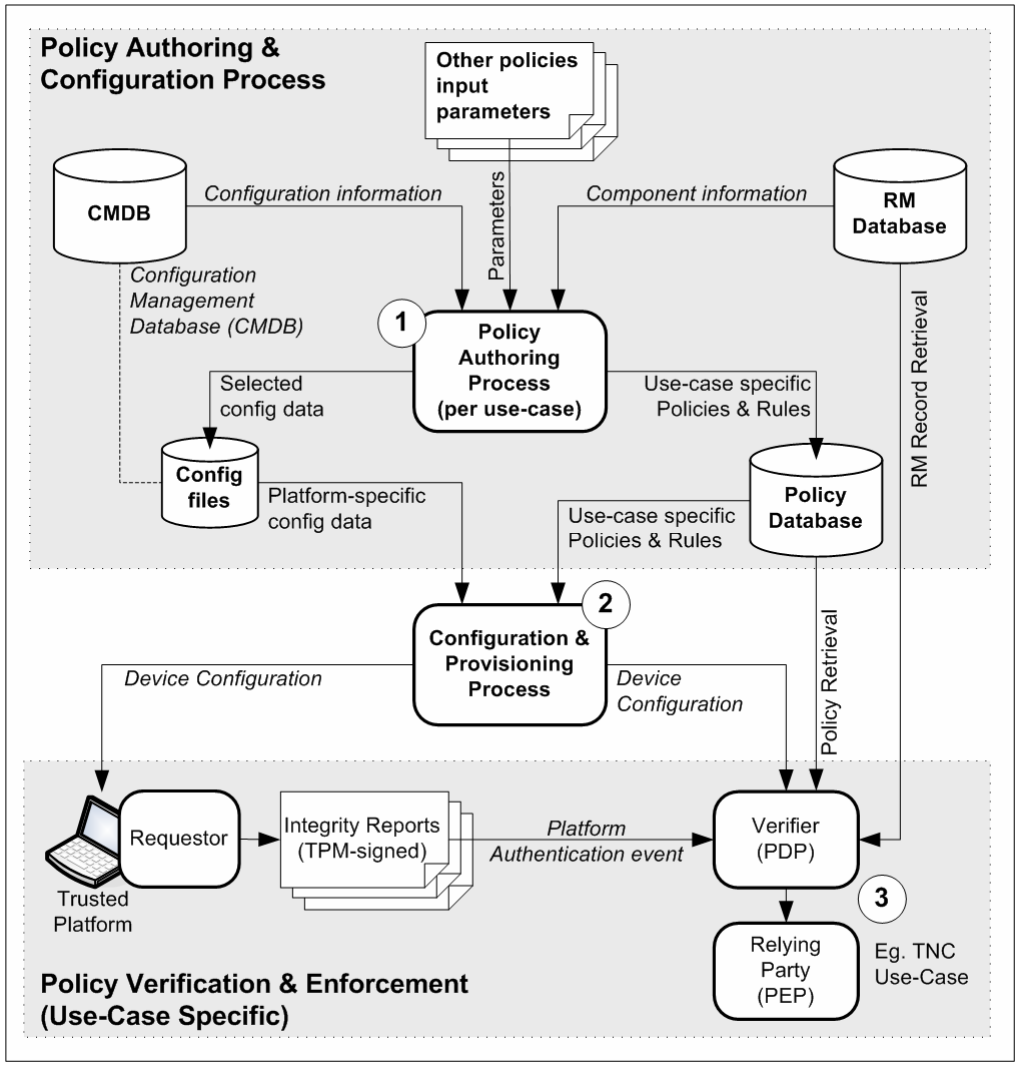


Figure 21: The Role of Reference Measurements in Policy Setting & Enforcement

(2) *Configuration and Provisioning process*: The process of provisioning of platforms can take many forms, as such only a high level description can really be shown in Figure 21. Here, the idea is to separate the notion of policy authoring from that of provisioning and configuration management. Policy authoring should be agnostic to the mechanism used to make the stated policies/rules present on the platforms. Provisioning and configuration management should be agnostic to the rules (of the policies) that it is pushing-down (i.e. provisioning) to the target platforms. In Figure 21 the aim is to show that configuration and provisioning reads from the policy database/repository, which in-turn contains rules that call-out certain components (as identified through the Reference Manifests). Thus, configuration management is enriched by the fact that precise component information is available at hand (in the Reference Manifest Database) and that, by using the appropriate tools, platforms can be configured to behave in a trustworthy manner for a given use-case.

(3) *Configuration Verification and Policy Verification process*: The process of configuration verification refers to the checking of the configuration information by a trusted platform. When a given platform is configured according to some policy (e.g. corporate policy),

during the day-to-day use of the platform, periodic re-verification of the configuration – hardware and software – needs to be performed. This is because unauthorized hardware can be added (e.g. consciously by the employee) and unauthorized software can inadvertently be present on the platform (e.g. malware, spyware, etc). Policy retrieval and verification is typically use-case specific (e.g. network access control), and is driven by the PDP entity (Verifier) on behalf of a PEP entity (Requestor).

9.3 Using Runtime Measurements to verify configuration

It is worthwhile to point-out that configuration self-checking (or remote checking by an administrator) is a feature that is a by-product of runtime measurement that is often understated. Part of runtime measurement is to report the (software) components that are present and running on the platform. Through mechanisms such as the PTS (see Section 8), a comprehensive verification of the entire platform can be performed, with the integrity report produced by the PTS covering all the software components on the platform. The integrity report can then be validated against the configuration file or list that was generated earlier (e.g. by the IT administrator) during the policy authoring process.

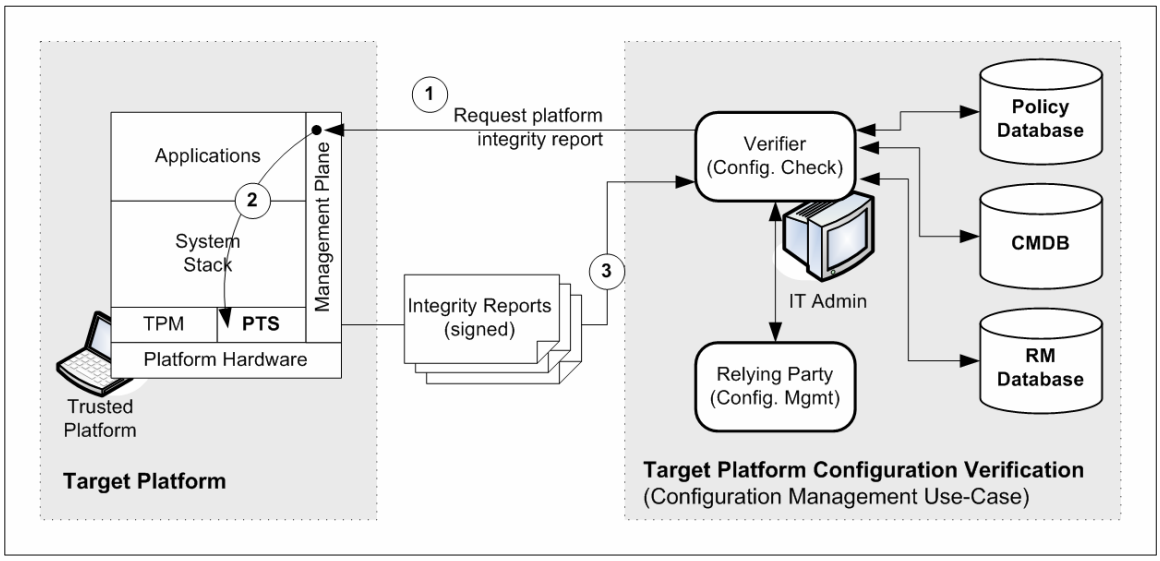


Figure 22: Using Runtime Measurement for Configuration Verification

Runtime measurement for configuration verification is very useful for system management architectures that deploy separate planes for management/control and user/data (Figure 22). More specifically, there are emerging today new system management paradigms that provide separated/isolated management hardware and software (on the same platform) specifically for remote management purposes. This allows the IT administrator to remotely connect to the control portion of the platform without interrupting the user process on the platform.

Runtime measurement as provided by the PTS under the domain of the management/control plane allows the IT administrator to remotely run the PTS on a (client) platform to obtain an integrity report regarding that platform. The integrity report can then be used by the IT administrator to verify the correct configuration of the platform. The difference between the PTS and other collection-agents today running on computer systems is the fact that the PTS is rooted in provable trust based on the TPM hardware and verifiable through the transitive trust chain. That is, the integrity report produced by the PTS is signed by a cryptographic key that is bound uniquely to the TPM ensuring that the integrity report has not been modified in-transit, and

transitive trust provides the strongest level of assurance that the PTS itself has not been tampered with.

Note that the integrity report has been defined by the TCG following the XML-based Integrity Schema specification. This allows the integrity report to easily interoperate with the growing number of web-services based architectures and protocols (e.g. WS-Manage, CIM) that have been produced by other standards organizations and industry consortiums (e.g. DMTF, Oasis, WS-I, etc). As far as possible, the TCG will not produce its own platform management specifications, but will rely instead on and re-use these other management-related standards. In certain specific cases, TCG will work with existing standards organizations and industry consortiums to ensure consistent and comprehensive support for TCG technologies.

In summary, runtime measurement complements configuration management because it allows platform configuration checking to be performed, either by the platform itself or by a remote management entity.

9.4 The TCG Policy Profile

Currently there are a number of policy expression grammars or languages being deployed by the industry through various device-management tools and use-case specific services. Often these languages are so rich in expressivity and flexibility that an additional target-specific policy profile is needed to instantiate the policy for a given use-case scenario.

The TCG will not develop its own policy language, but will instead rely on existing standards and where necessary create *policy profiles* for use that will allow easier integration of TCG technologies with specific scenarios that are policy-driven.

Thus, for example, in the TNC use-case, the PDP entity (Verifier) is assumed to be policy-aware, even though the TNC Architecture [9] does not specify any policy languages of choice. In order for the TNC use-case to be realized in a given deployment scenario (which most likely will be policy-driven), an additional policy profile will be needed to integrate measurements (both integrity reports and Reference Manifests) into the existing policy language of the target environment.

10 Trust Credentials and RMs: Conceptual Model

Credentials play an important role in expressing both technical trust and social trust in the context of trusted computing. Indeed the area of public key certificates and PKI can be seen as an example of a bridge between technical trust (strength of cryptography) and social trust (that a Certificate Authority vouches for a human person or device). Within trusted computing itself credentials have played a crucial role in a number of instances, chief among them in the use of EK-certificates (issued by a TPM Manufacturer) as means for a TPM manufacturer to endorse its product (social trust) as well as a mechanism – among others – to identify the unique TPM hardware instance (technical trust).

In the current section, we discuss the conceptual model underlying the notion of Trust Credentials. We abstract the existing TCG certificate profiles 1.0 [2] and proceed to split these into two halves: the Reference Manifest structure and a "reduced" certificate. The purpose of this abstraction-and-split is to illustrate firstly the fact that many information fields in the current TCG certificate profiles 1.0 are in fact component-specific details that should be better placed inside Reference Manifests. Secondly, the purpose is to show that all the "reduced" forms of the three TCG certificate profiles 1.0 are very similar in structure that a single common structure can be used in their stead. We call this single common structure as the *Trust Credential*.

The current section does not define any new certificate profiles, but instead point to the future need for the single common structure (i.e. Trust Credential) in order to provide a flexible method for entities in the Trusted Computing Ecosystem to issue Reference Manifests and Trust Credentials.

10.1 Motivations

One of the important needs of the IMM model is the expansive use and deployment of credentials as a means for other suppliers of trusted computing technologies to express their endorsement of components they manufacture that make-up a trusted platform (beyond only the TPM Manufacturer). Thus, when a vendor produces a given component (e.g. BIOS, Loader, Kernel) and issues reference measurement data (e.g. in the form of a Reference Manifest record) there needs to be a flexible mechanism and supporting infrastructure for these vendors to digitally sign the reference measurement data and for their corresponding consumers (e.g. other manufacturers, OEMs and end-users) to verify these credentials.

The fact that a trusted platform is made-up from numerous components points to the need for a extensible credential structure (Figure 23). That is, there needs to be a credential structure that is flexible enough to express endorsement of components that are in fact made-up of other sub-components (e.g. NIC card product from vendor A is made-up of Ethernet hardware supplied from manufacturer B and Modem hardware from manufacturer C). Currently, the credential structure or format used for EK-credentials is defined in the *TCG Certificate Profiles 1.0* specification [2]. This specification covers only EK-certificates (for TPM manufacturers), Platform Credentials (for OEM manufacturers) and AIK-certificates (for the Platform-CA).

The need for extensible credentials is driven by a number of requirements:

- *Supply-chain coverage*: Firstly, manufacturers and suppliers need to be able to issue signed reference measurement for their component products. This can be achieved today using existing X.509 PKI infrastructure, although additional certificate extensions will be needed to capture and express trust-related information from these manufacturers.
- *Extensibility*: Secondly, a flexible credential is needed to respond to the oft-changing needs and situations of manufacturers and vendors. Given that many components are in fact made-up of sub-components, the credential structure needs to allow manufactures and suppliers to point to reference measurements (Reference Manifests) of sub-components (within their component product) from other manufacturers. In addition, the credential must be easily re-issued when sub-components of a product change (e.g. as response to changes in business relationships and supply-demand dynamics).

- *Compatibility*: Thirdly, the credential needs to be compatible with the TCG Integrity Schema, which provides the XML syntax and semantics for expressing runtime measurements and reference measurements within the IMM. The credential must also be useful in other mediums of business transactions which may make use of the trust-related information captured in the credential.

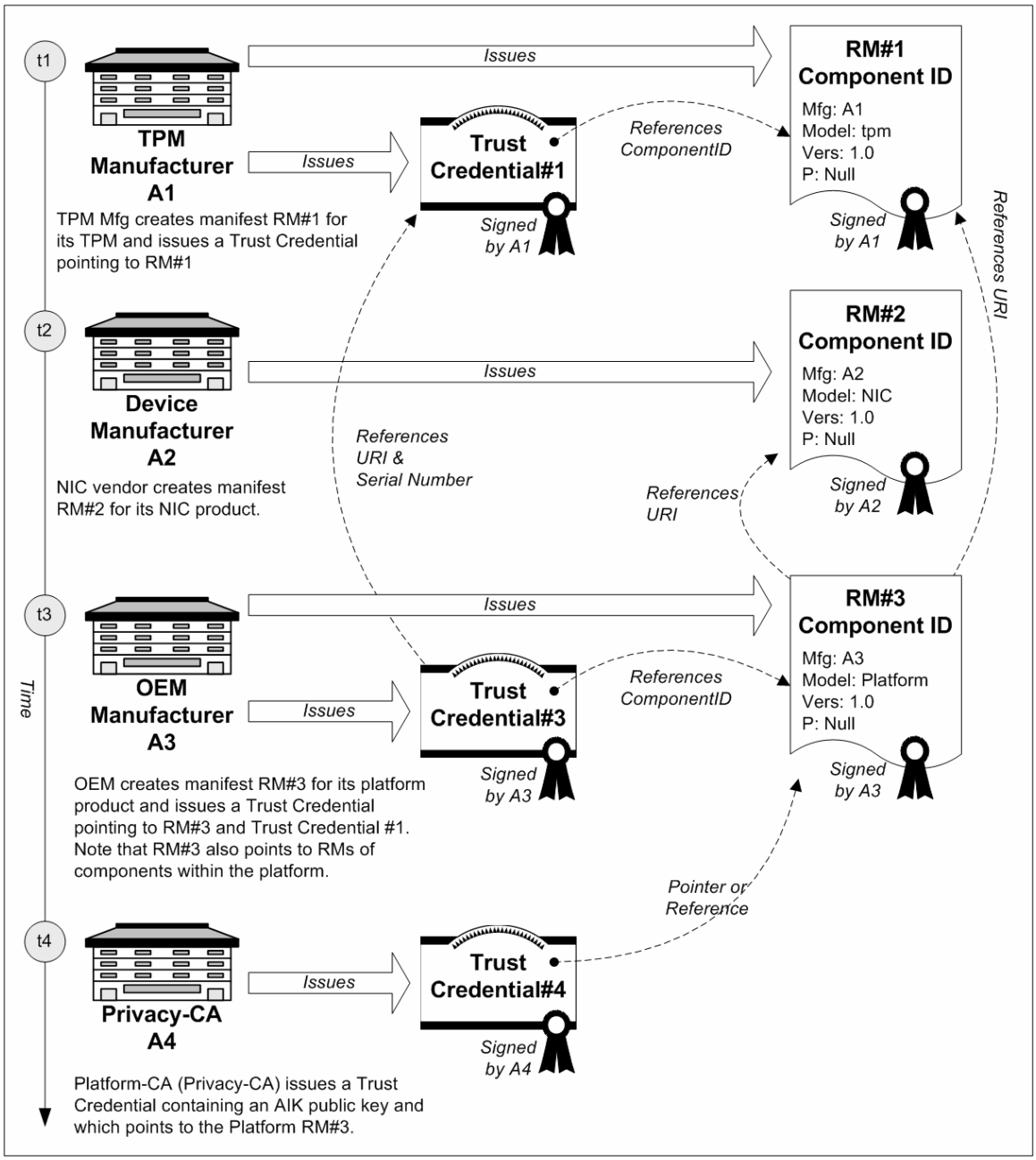


Figure 23: The Trust Credential Model

10.2 The Trust Credential Usage Model

The current set of TCG credentials provides sufficient capabilities for early deployment of platforms containing a TPM. However, the certificate structure leaves little opportunity for other trusted computing ecosystem entities (e.g. component vendors) to participate and report the integrity information for their contribution (i.e. component that makes-up a trusted platform). In order to understand the difference between the current usage of TCG certificates 1.0 and the notion of Trust Credentials, in this section we describe the Trust Credential Model and it usage with Reference Manifests. The aim is describe the desired usage scenario, and then understand how to evolve the current TCG Certificates into Trust Credentials.

Figure 23 attempts to show the Trust Credential Model and its usage, albeit in a simplified manner. It illustrates the need of a flexible new Trust Credential structure (see Section 10.4) that carries the core information pertaining to the assertions required at different phases in the platform creation supply-chain.

In Figure 23 when a platform is being created, there is a stage-by-stage accumulation of components by the platform manufacturer. Each component manufacturer produces both a signed Reference Manifest and a signed Trust Credential for the component it produces. The credential for a component may contain references to the Reference Manifest for that component, providing the separation of the credential's assurance qualities from its quantitative attribute qualities (as found in the Reference Manifest).

In Figure 23 the following sequence of events occur:

- At time T1 a TPM manufacturer A1 creates an integrity manifest (RM#1) for its TPM product and at the same time issues a Trust Credential that references the manifest. The credential contains, among others, the Component ID of the TPM hardware.
- At time T2 a device manufacturer A2 produces a NIC card product and issues an integrity manifest (RM#2) for its product. Note that the device manufacturer in this example does not issue any Trust Credentials, and thus makes no trust assertion about its product.
- At time T3, an OEM A3 incorporates the TPM from manufacturer A1 and the NIC card from manufacturer A2 into its Mainboard product. The OEM creates an integrity manifest (RM#3) for its platform product which references the integrity manifests of its components, namely RM#1 and RM#2 respectively. In addition, in this example the OEM makes Mainboard trust assertions by issuing a Trust Credential that references the Trust Credential for the TPM.
- At time T4, a Privacy-CA issues a Trust Credential containing an AIK public key resident within the TPM on a specific Mainboard. The Trust Credential here points to the RM#3 for the platform (issued earlier by the OEM).

The above example points to the need for a new type of credentials – referred to as *Trust Credentials* – which allow for rich references to various Reference Manifests containing reference measurements (e.g. from manufacturers) pertaining to components of a trusted platform. The availability of supportive integrity information (in the form of Reference Manifests) from suppliers upstream allows downstream entities, such as OEMs, Privacy-CAs and IT manager, to determine with a high degree of accuracy the integrity level of platforms with which they are dealing.

10.3 Evolving Trust Credentials from Current TCG Certificates

One of aims of Trust Credentials is to provide a light-weight credential that carries only the minimal necessary information regarding the issuer entity (free of component-related information), while at the same time having the capability to provide component-related information available through Reference Manifests. A second aim of Trust Credential is to define it in such a way that it is not bound or dependent on specific formats (e.g. X509 v2 or V3, SPKI, XML, etc). Indeed, it has been the aim of the TCG certificates to be format-neutral, while at the same understanding that there are already existing infrastructures and services for PKI and certificates in the industry

today. This is reflected in the current TCG Certificate profiles specification [2], which defines the mandatory and optional information in the current set of TCG certificates in Section 1 and 2 of [2], but also providing an X.509 specific definition in Section 3 of specification [2].

In order to illustrate the evolution of the Trust Credentials as coming from the TCG Certificate Profiles, in the following sections we will illustrate the combination of Trust Credentials and Reference Manifests to achieve the same effect as the current TCG Certificate Profile [2] for EK-certificates, Platform-Certificates and AIK-certificates. Note that these are illustrative examples only, and that further specifications will be developed to define the Trust Credentials more precisely.

10.3.1 Summary of Fields in TCG Credentials 1.0

In order to understand the motivation and need for Trust Credentials, it is useful to look at the current TCG certificate profiles 1.0 specification [2] in order to understand its limitations. Figure 24 shows a summary of the three certificate profiles (EK-certificates, Platform Certificates and AIK-certificates) and identifies the fields (in grey) that are present in all three certificates.

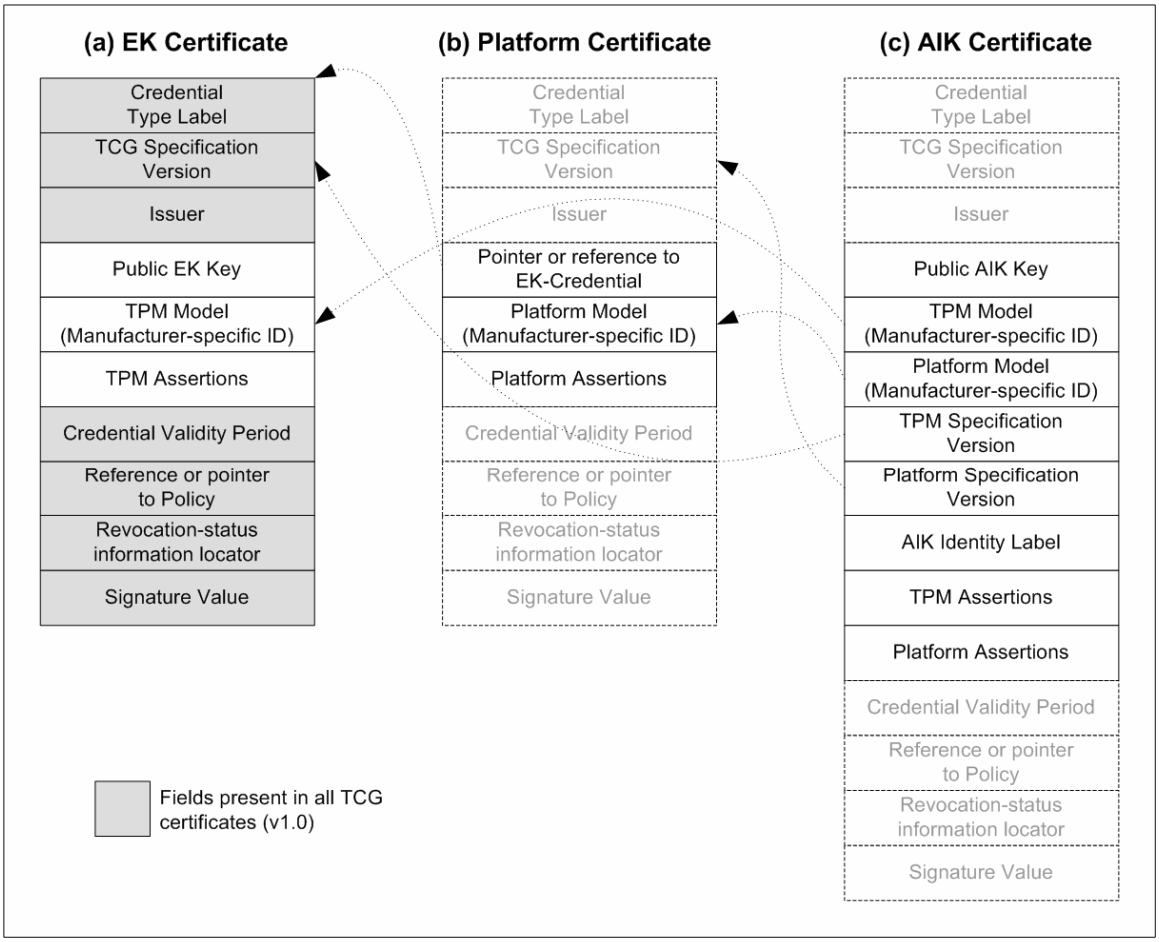


Figure 24: Summary of Fields in TCG Credentials (after [2])

The common fields are as follows:

- *Credential type label*: The label enables the Issuer to sign the credential with a key that is not reserved exclusively for a particular credential type.

- *Issuer*: Identifies the entity that signed and issued the credential
- *TCG specification version*: Identifies the TPM or platform-specific specifications implemented by the TPM or platform TBB which is represented by the credential
- *Signature value*: The signature of the issuer over the other fields in the credential

All other fields in a TCG credential type are called, collectively, "information fields." Some information fields are mandatory and some are optional [2].

Looking briefly into Figure 24 it is clear that one of the limitations of these credentials is the fact that item-specific information (e.g. TPM or Platform manufacturing information) are mixed with certificate-specific syntax.

As we will see in the following section, one of the aims of Trust Credentials is the separation of component information into Reference Manifests and the identification of a set of core fields that make-up a Trust Credential. In the following sections we use the term "Reduced" certificate to refer to the core (stripped-down) version of the TCG certificates (EK-certificate, Platform-Certificate and AIK-certificate).

10.3.2 Reduced EK-Certificates (Abstract)

One of the key purposes of the EK-certificate is to capture the binding of a TPM hardware instance with a unique EK key pair. Figure 25 illustrates a Reduced EK-Certificate, which is obtained from separating the TCG EK-Certificate in Figure 24 (a) into portions pertaining to component information and those pertaining to credentialing. The reader is encouraged to compared Figure 25 with the EK-certificate in Figure 24 (a).

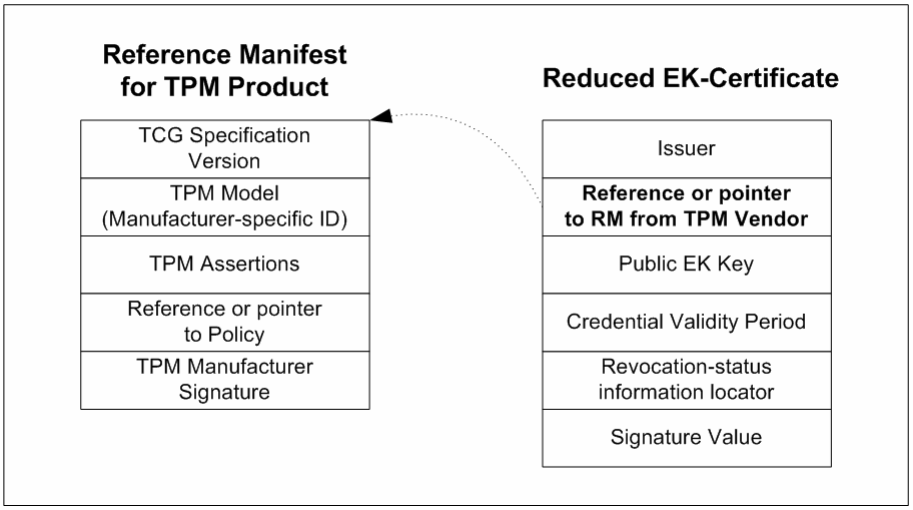


Figure 25: Splitting EK-Certificate into a Reference Manifest and a Reduced EK-Certificate

A key feature of the Reduced EK-Certificate in Figure 25 is the separation of TPM-related information (into the Reference Manifests) from the certificate issuer information. In the context of the TPM and EK-keys specifically, this approach provides a number of advantages. Firstly, the approach recognizes that the EK certificate issuer may not necessarily be the TPM manufacturer. The approach allows other entities downstream in the supply chain (e.g. OEM, IT Administrator) to be the issuer of the certificate. This was referred to as "late-binding" in [1].

Secondly, this approach allows the TPM manufacturer to focus on its core business (producing good TPM products) and not on certificate issuance and PKI management for the TPM. This approach provides a more flexible solution that recognizes current market conditions by making EK certificate issuance an optional task for the TPM manufacturer.

Thirdly, the TPM manufacturer can still stand behind its product by issuing the signed Reference Manifest corresponding to its TPM product. The TPM manufacturer can still indicate its compliance and conformance levels through the TPM-Assertions information in the Reference Manifest.

Finally, the separation of TPM-related information (into the Reference Manifests) from the certificate issuer information allows different expiration dates. That is, the Reference Manifest record pertaining to the TPM product can be longer lived compared to the Reduced EK-Certificate.

10.3.3 Reduced Platform Certificates (Abstract)

The benefit of Reference Manifests becomes clear in the context of trusted platforms and the task to evaluate it. The purpose of Reference Manifests is indeed to provide a complete and accurate manifest of the components that make-up a trusted platform. Currently, the TCG Platform Certificate does not capture sufficient information regarding components. The exact knowledge of components is especially important for evaluation organizations and also the Privacy-CA that must issue AIK-Certificates.

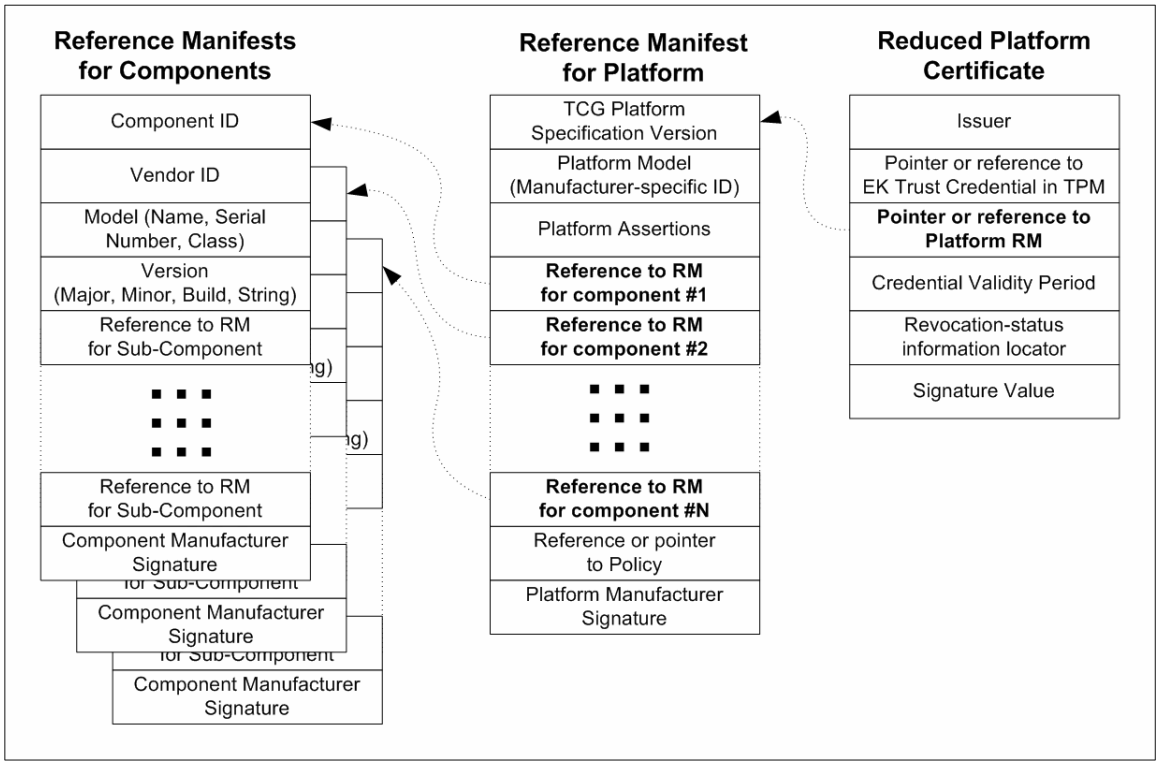


Figure 26: Splitting Platform-Certificate into RMs and a Reduced Platform-Certificate

Similar to the Reduced EK-Certificate (in Figure 25), the Platform Certificate can also be reduced into a Reference Manifest and a *Reduced Platform Certificate*, as shown in Figure 26. The figure illustrates the use of Reference Manifests to capture platform component information, and to represent this information is a systematic manner. Each component vendor provides a

Reference Manifest for each of the components that they manufacture, while the platform manufacturer (e.g. OEM, end-user, etc) issues the Platform-RM. There are a number of entities that could issue the Reduced Platform Certificate, including the OEM, IT Administrator and compliance bodies.

One of the advantages of this approach is the separation of tasks pertaining to component information collection from tasks pertaining to the issuance of the Reduced Platform Certificate. Thus, this approach allows the issuer (of the Reduced Platform Certificate) to trust the platform RM signer (e.g. the OEM) that it has correctly audited and accounted for every component that made-up a platform. The Reference Manifest truly becomes a manifest in the ordinary sense of the word. By digitally signing the Reference Manifest for the platform, the signing entity (e.g. OEM) legally states correctness of the manifest.

An additional advantage of this approach is the ability to replace the pointer or reference to the EK Trust Credential with the actual EK Public Key, thereby allowing the platform certificate issuer to directly incorporate TPM specific references into the Platform Certificate. This eliminates the need for an additional Reduced EK Certificate, decreasing the number of certificates required to effectively validate the operational state of a platform, and simplifying the infrastructure necessary to manage and evaluate platform assertions.

10.3.4 Reduced AIK-Certificates (Abstract)

One of the many tasks (and values) of a Classic Certificate Authority (CA) is to vouch that a digital certificate corresponds to a person, entity or device. In practice, a CA is aided by a number of services provided by organizations and bodies. Thus, in issuing a human certificate (of high grade), the CA may rely on a person's driver's license or passport as a means of identifying the person. These other means of identification are provided by other entities in the world.

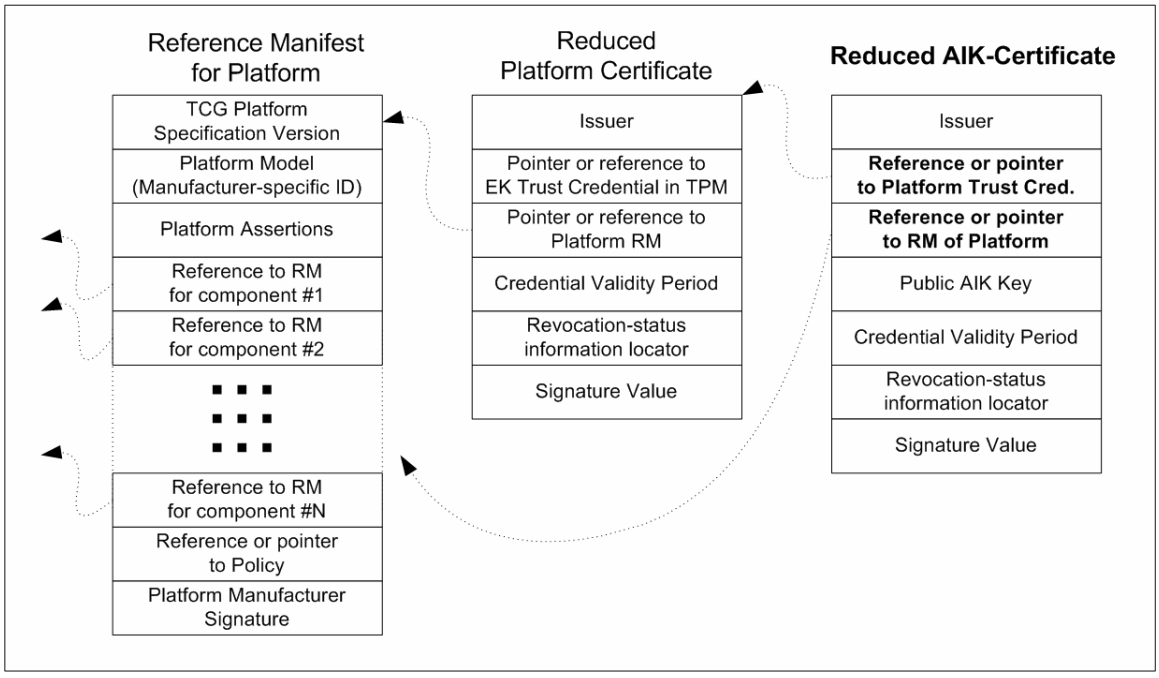


Figure 27: Splitting the AIK-Certificate into RMs and a Reduced AIK-Certificate

Similarly, a Privacy-CA (Platform-CA) cannot reasonably be expected to verify that a given platform possesses the correct hardware and software components. The Privacy-CA must rely on the platform Reference Manifest signer and the (Reduced) Platform Certificate issuer to vouch for

the correctness and accuracy of the manifest for that platform. Figure 27 shows the separation of the TCG AIK Certificate into a Reduced AIK-Certificate (that carried the AIK public key) plus the Reference Manifest for the platform (that carries pointers to component Reference Manifests). Note that here the Reduced AIK-certificate also (optionally) points to the Reduced Platform Certificate (from Figure 26). This optional inclusion is typically driven by the Privacy-CA policies.

By separating-out component information into Reference Manifests, the Privacy-CA's task is simplified through a distribution of duties (to component-RM issuers and platform-RM issuers). In issuing a (Reduced) AIK-Certificate (Figure 27), the Privacy-CA really needs to see only two types of certificate. The first is the Reduced Platform Certificate (from Figure 26), while the second is the Reduced EK-Certificate containing the EK public key (from Figure 25).

Analogous to the human certificates issued by Classic-CAs, the Privacy-CA can now rely on the services provided by other entities (e.g. OEMs) in the Trusted Computing Ecosystem who vouch for the correctness of the Reference Manifests and other certificates pertaining to the platform in question.

10.4 Trust Credentials: The Conceptual Model

The purpose of the previous section – and the exercise of splitting TCG certificates into Reference Manifests and Reduced Certificates – is to illustrate that the three Reduced Certificates can in fact be combined into one common structure, called the *Trust Credential*. The Trust Credential will have fields that are present (ie. Mandatory) for all occasions of use throughout the supply-chain, and will have certain fields that are context-specific and issuer-dependent.

Figure 28 shows the Trust Certificate as the combination of the Reduced EK-certificate (from Figure 25), the Reduced Platform Certificate (from Figure 26) and the Reduced AIK-Certificate (from Figure 27). Note that the Trust Credential has some fields that are present in all instantiations of the Trust Credential (e.g. Issuer, Signature, etc).

There are a number of crucial points to note when looking at the Trust Credential in Figure 28:

- *Common structure with RMs*: The Trust Credential closely resembles the Reference Manifest structure, with the presence of a cryptographic key (in the Trust Credential) being the crucial differentiating factor. In fact, one might argue that a Trust Credential is a Reference Manifest (and vice-versa).
- *Context-specific fields*: The optional (context-dependent) fields determine the purpose of a given Trust Credential. Depending on the instance of use, Trust Credential could be (for example) an EK Trust Credential or an AIK Trust Credential. The differentiating factor would be the issuer entity and the references/pointers in the Trust Credential. Thus, in the case of an EK Trust Credential, the issuer could be a well-known TPM manufacturer, while one of the pointers refers to a Reference Manifest specifically for a TPM. By following the reference/pointer to the Reference Manifest, the Verifier entity can obtain further information regarding the Trust Credential it is evaluating. Similarly, in the case of an AIK Trust Credential, there will be a reference/pointer to one or more Reference Manifests pertaining to the platform and/or components of the platform (including a TPM component Reference Manifest).
- *Backward compatibility*: The fields of the Trust Credential structure are compatible with existing industry certificate standards, notably with the X.509 certificate structure. Thus, although the semantics of the Trust Credential has been the focus here, the aim is to develop Trust Credential specifications that can be implemented in X.509, XML or other mechanisms according to needs to the use-case. Particular use-case specific implementations of the Trust Credential may add further fields beyond those shown in Figure 28 (e.g. adding Subject field and Certificate Serial Number in X.509).

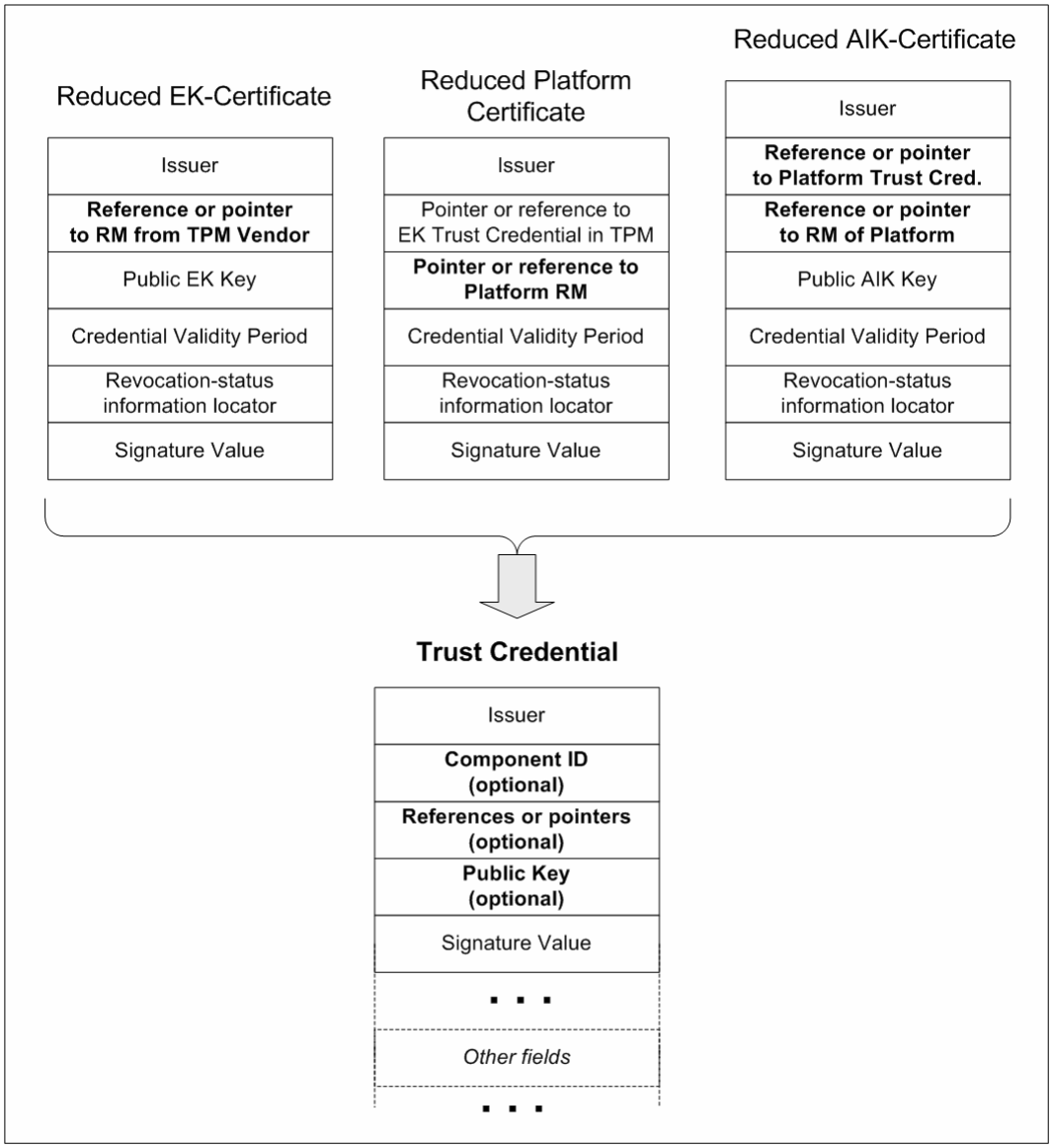


Figure 28: Combining the abstract (Reduced) Certificates into a Trust Credential structure

An important consideration that has underlie the current discussion of Trust Credentials is the effort and level of difficulty in getting entities in the Trusted Computing Ecosystem to participate by providing Reference Manifests for their products, in addition to issuing credentials for products that require keying material. A factor is the interest and ability of vendors to deploy digital-signature technology (i.e. PKI) to sign Reference Manifests and Trust Credentials. Another factor is the increase in use of web-services technologies, which are slowly replacing the legacy supply-chain IT infrastructure such as EDI and others.

The recognition of these factors have motivated the design of the Trust Credentials to be implementation-agnostic, allowing vendors to either use existing X.509 PKI infrastructures or to use web-services related technologies and standards (e.g. SAML, XACML, XKMS, etc). The TCG believes that providing vendors with the freedom to choose syntax-specific (but standards-based) implementations provides the best way forward for the adoption of Reference Manifests and Trust Credentials.

11 Use Cases for the Integrity Management Model

The TCG Integrity Management Model (IMM) is the basis for evaluation trust in entities in a number of use-cases. The IMM is closely related to the basic authentication model for platform authentication described in the Architecture for Interoperability Part I [1]. In this section we discuss this relationship and provide some use cases for the deployment of the IMM.

11.1 Platform Authentication

In [1], a Requestor seeks to carry-out a transaction with the Relying Party through the mediation (direct or indirect) of the Verifier. Here, the Relying Party relies on the Verifier to evaluate the assertions or claims presented by the Requestor. Furthermore, the Verifier performs the evaluation of the Requestor's assertions based on some set of criteria or rules, which are understood by all three parties and have been established through some out-of-band method. Recall that from [1] it is important that all three parties understand the same criteria (both syntax and semantics) in order for all three to communicate meaningful assertions. The outcome of the Verifier's evaluation of the Requestor's assertions can be binary (True or False), or it can be a score (within a range of values) based on the agreed criteria for evaluation. The notion of a "score" is intended to reflect the fact that many transactions in the real world have results that cannot map easily (or even logically) into a binary value.

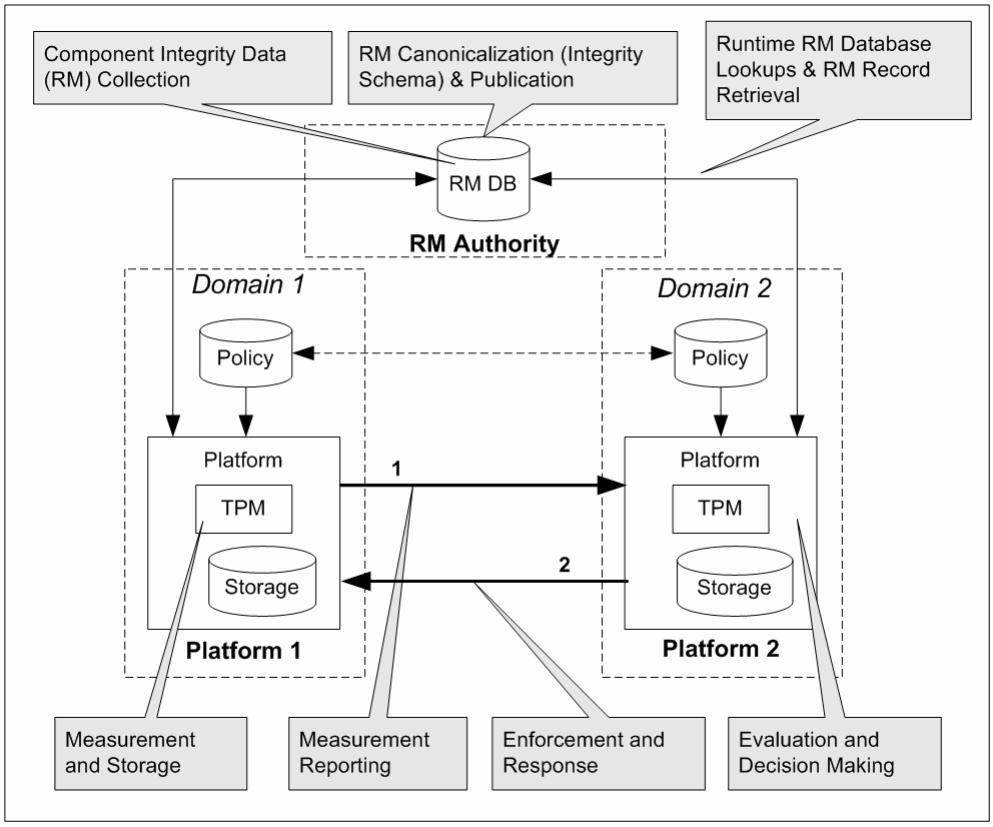


Figure 29: The IMM in the context of Platform Authentication defined in [1]

In Figure 29, two platforms are engaged in platform authentication as defined in [1]. Here, the Requestor (Platform-1) is being authenticated by the Verifier (Platform-2) who may be doing so in behalf of a Relying Party. Reminiscent of and continuing from Figure 7 in [1], the Requestor performs platform integrity measurements (e.g. at boot-up time) and stores them in the

measurement log. During a platform authentication event, the Requestor provides the Verifier with a measurement report to be evaluated by the Verifier as part of its decision-making. In the process of evaluating the Requestor, the Verifier may obtain component integrity data (regarding the Requestor's platform components) from the Reference Manifest Authority which operates the Reference Manifest Database. The IA may be the manufacturer or a Trusted Third Party providing the service. Once the Verifier has sufficient information from the Requestor and from the Integrity Authority to perform an evaluation, it does so and factors the result into the broader decision making based on the prevailing policies. The result is then enforced and a response in provided to the Requestor.

11.2 Trusted Boot

Trusted boot is the process of measuring platform integrity, commencing before an operating system is loaded and including the measurement of the OS components, for reporting of the integrity of the pre-boot and OS environments to a verifier. Measurement code that is included into the pre-OS environment can make assertions about the pre-OS environment, principally by extending values into the appropriate TPM Platform Configuration Registers (PCRs), and thru transitive trust properties, have the pre-OS environment make assertions about the integrity of OS components. One important component of the pre-OS measurement is the measurement of the initial program loader (IPL code) or boot image, which is the component used to measure the OS image.

Trusted boot verification then relies on the measured OS components to provide an Integrity Measurement Log (IML) to a remote verifier, which is capable of parsing the IML to ensure that the platform booted in a trusted fashion into a valid state. In the event that a remote verifier determines that the platform has booted correctly, according to pre-established security policies, the platform is then granted services within the networked environment. If the remote verifier determines that the platform has booted into an unacceptable state, the remote verifier may send the platform into a remediation process to update the necessary components, and request the platform to reboot, re-measure, and re-attest to its remediated state. This process may continue until the remote verifier determines that a valid boot process has been achieved, or until remediation process is determined to no longer be effective.

PTS is nominally used as part of the measurement process. Once enough of the OS has been loaded and is executing, PTS can be commanded to complete the measurement process and create the IML. This IML can then be sent to the verifier using integrity messaging protocols.

11.3 Secure Boot

Secure boot is the process of measuring platform integrity, commencing before an operating system is loaded and including the measurement of the OS components, for local verification on the platform. By interleaving measurement steps with verification steps, a boot process can self-validate according to a pre-established set of security policies which are resident on the client. Effectively, this is similar to the trusted boot process except that verification is done locally and the set of reference measurements and policies must be available to the client platform during the local validation process.

One potential implementation would be for the CRTM to measure and verify the POST code, and for the POST code to measure and verify the Option ROMs, and then for the POST code to measure and verify the IPL code, and the IPL code to measure and verify the kernel OS components (including PTS). Once the kernel and PTS components are loading and executing, they may then measure and verify other components which are deemed critical by the security policies of the platform, using local rules which the PTS may enforce.

If at any time during the measure/verify process it is determined that a component is out of security policy compliance, the platform may choose to halt execution and require manual remediation, or it may choose to boot into a valid remediation image which is capable of updating components and/or security policies.

Secure boot may have the benefit of preventing out of compliance machines from ever executing rogue code or from presenting compromised services to the network or from allowing trojan login sessions to users from occurring..

Secure boot requires the verification policy to be securely provisioned to the platform in such a way to ensure policy can only be interpreted by the proper verification code. TPM non-volatile storage can be used to store reference measurements such that only trusted verification code may access it. In addition, verification code must be protected from rogue code that might reverse or hide the results of verification. Therefore, the verification engines must be protected to an equal degree as the measurement engines.

11.4 Trusted Network Connect (TNC)

The Trusted Network Connect (TNC) use-case [9] provides a good illustration of the benefits of runtime measurements and reference measurements. In the context of network access control, the TNC architecture is the only approach today that makes use of the trusted hardware (TPM) on the client (Network Access Requestor) device in order to provide a strong integrity status report (regarding the Client device) to the Server (Network Access Authority). This is summarized in Figure 30.

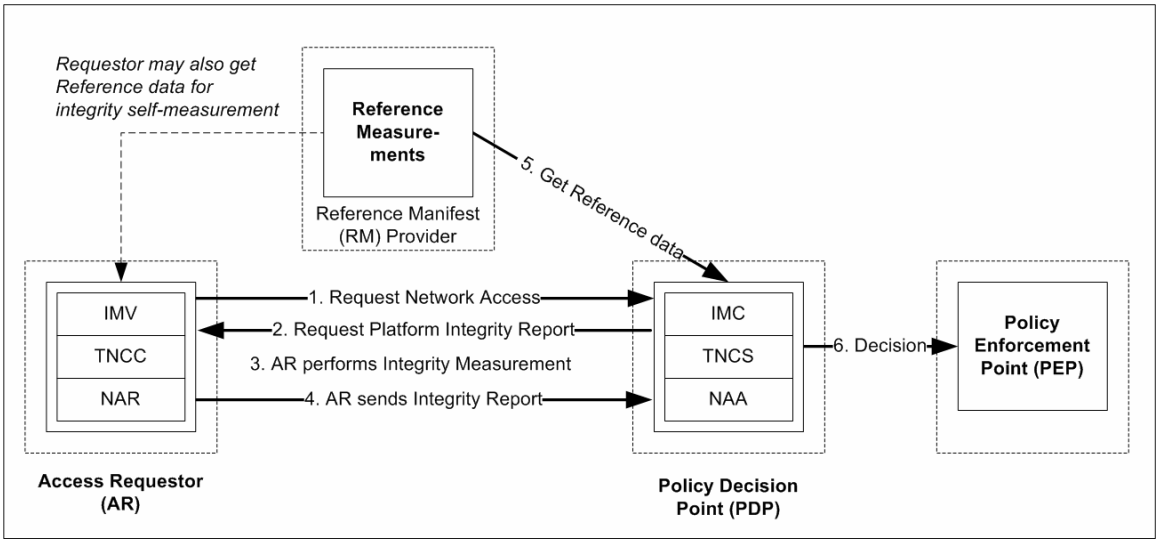


Figure 30: The TNC Use-Case with TPM-based integrity measurements and reporting

The IMM plays an important role because it provides the underlying model for the infrastructure that support the use of the TNC with the TPM, with measurements done by the PTS (i.e. runtime measurements) and verification against Reference Manifests (reference measurements). Thus, the true value of the TNC using the TPM can only be realized if integrity measurements can be captured and reported (via the PTS), and can be verified by an Authentication Server (PDP) against source-authentic Reference Manifests.

Although the PTS provides invaluable support for device integrity self-verification use-cases (e.g. Trusted Boot), the PTS plays an even more important role when it comes to transactions involving other parties (of which the TNC is an example). Integrity Reports produced by the PTS at the AR entity obtains source-authenticity by virtue of the TPM's Quote function, which "binds" the integrity report to the origin TPM (ie. the AR's TPM). The verifying entity (PDP) can use the AIK-certificate or Trust Credential of the AR in order to verify the signature on the integrity report from the Client.

It is here that the Integrity Schema plays an important role in providing a common syntax for the AR to provide an integrity report (through the PTS at the AR) and for the PDP to parse through the integrity report and to retrieve the appropriate Reference Manifests in order to verify all the elements inside the integrity report. See Figure 31.

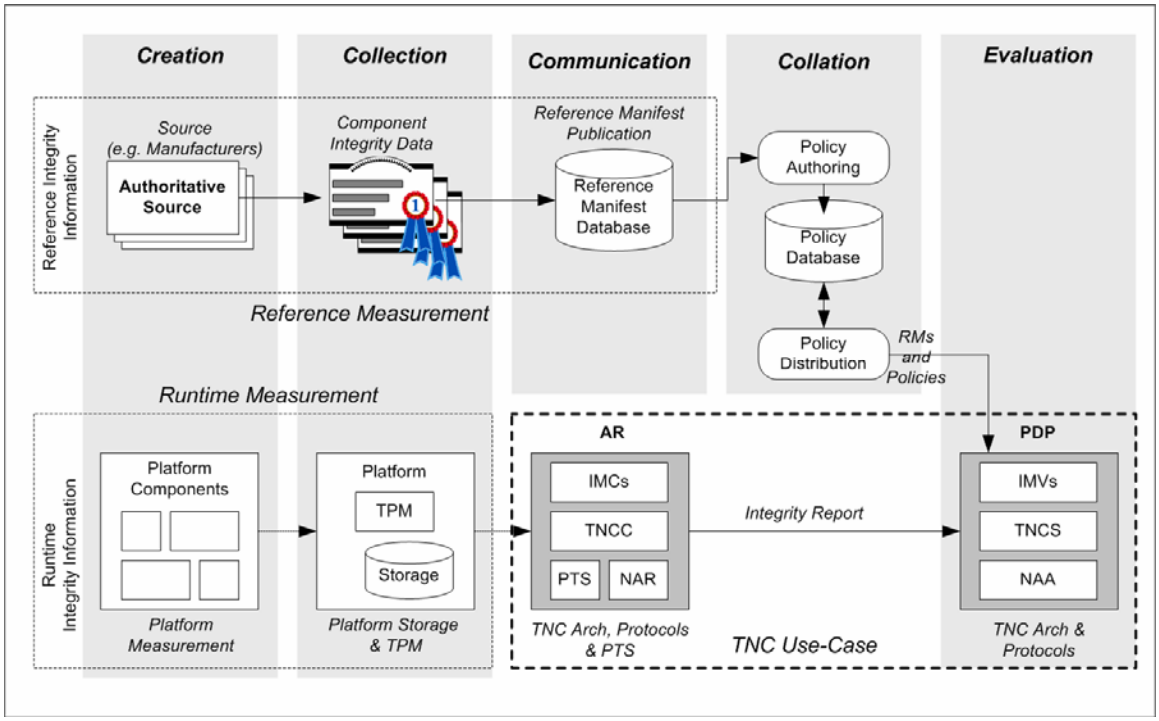


Figure 31: IMM in the context of TNC

12 Glossary

Term	Definition
Component	An instance of code or logic.
Component ID	A unique identifier that refers to a Component. Generally it is derived from product release and change control processes.
Integrity Information	The set of platform specific information that make-up a Trusted Platform. These range from information about a platform's hardware components, TPM information (e.g. versions), PCRs, peripherals, Trusted Building Blocks, OS/Kernel, drivers, Applications, Anti-Virus information and others. Each specific use-case determines which information set will be of interest. As such, it is expected that for a given situation these will be pre-determined or pre-configured by an authorized entity (e.g. IT administrator).
Integrity Measurement Log (IML) and Stored Measurement Log (SML)	Look at TCG Glossary.
Integrity Measurements	The informational output from applying an Integrity Measurement process.
Integrity Report:	Structure containing one or more snapshots and quote information. The Integrity Report is a platform-independent representation of the Stored Measurement Log (or event log).
Logical Snapshot	Inverse of a PCR Snapshot – i.e. a snapshot not tied to a PCR.
Most Authoritative Entity (MAE)	The entity that signs a RIM. The entity can be the component manufacturer, a Trusted Third Party, an IT Manager or others.
PCR Snapshot	Snapshot structure whose resultant hash value is extended into a PCR.
Platform Authentication	The act of verifying both the proof-of-identity and integrity-status of a given platform.
Platform Trust Services (PTS)	The Platform Trust Services (PTS) is a system service that exposes trusted platform capabilities to TNC components that reside on a Trusted Platform containing a TPM. PTS services include protected key storage, asymmetric cryptography, random numbers, platform identity, platform configuration reporting and integrity state tracking.
Policy Decision Point (PDP)	The PDP is an entity evaluating the status of a TNC Client (seeking network connectivity) and deciding upon some network-related action to be enforced by the PEP. The PDP embodies the security and integrity related policies governing the network.
Policy Enforcement Point (PEP)	The PEP is a component within the TNC Architecture that controls access to a protected network, whose policies are implemented through a Policy Decision Point (PDP). The PEP enforces the decision of the PDP.
Reference Integrity Measurement (RIM)	Single instance of a digest/hash regarded as the baseline. ("blessed" hash)

Reference Manifest (RM)	Data structure that contains collections of reference measurements that is used by the Verifier to compare against the set of integrity reports (containing snapshots) submitted by the Requestor in a given platform authentication event.
Snapshot	Measurements, assertions and descriptive log data of a Component and it's itsconstituent elements represented in XML.
Sync Snapshot (aka PCR Snapshot)	The collection of measurements over other snapshots with resultant hash extended into a PCR. Sync Snapshot is a PCR snapshot + collection of measurements of other snapshots.
Transitive Trust Path	References to each node of a Transitive Trust Chain.
Trust Chain	The dependency graph from a chosen node to a Root of Trust for Measurement.
Trust Graph	The structure of trust dependencies in a system of Measurement Engines

13 References

[1] Trusted Computing Group, *Reference Architecture for Interoperability* (Part I), Specification Version 1.0, TCG Published, June 2005.

[2] Trusted Computing Group, *TCG Credential Profile for v1.0*, Specification Version 1.0, TCG Published, June 2005.

[3] Trusted Computing Group, *Core Integrity Schema*, Specification Version 1.0, TCG Published, September 2006.

[4] Trusted Computing Group, *Integrity Report Schema*, Specification Version 1.0, TCG Published, September 2006.

[5] Trusted Computing Group, *Reference Manifest (RM) Schema*, Specification Version 1.0, TCG Published, September 2006.

[6] Trusted Computing Group, *Security Qualities Schema*, Specification Version 1.0, TCG Published, September 2006.

[7] Trusted Computing Group, *Simple Object Schema*, Specification Version 1.0, TCG Published, September 2006.

[8] Trusted Computing Group, *Platform Trust Services (PTS),* Specification Version 1.0, TCG Published, September 2006.

[9] Trusted Computing Group, *Trusted Network Connect (TNC) Architecture*, Specification Version 1.1, TCG Published, May 2006.

[10] Trusted Computing Group, *TPM Specifications v1.2*, October 2003.

[11] Trusted Computing Group, *TSS Specifications v1.1*, August 2003.

[12] Trusted Computing Group, *TCG Design Guidelines, (Glossary section)*, June 2004. Work in Progress.

[13] W3C, *Extensible Markup Language* (XML) 1.1, W3C Recommendation, 4th February 2004. See www.w3c.org.