TCG Generic Server Specification

Specification Version 1.0
Revision 0.8
March 23rd, 2005
Published

Contact: techquestions@trustedcomputinggroup.org

TCG

TCG PUBLISHED

Revision History

Revision	Date	Comments
0.8	June 16, 2005	Initial Release

TABLE OF CONTENTS

1 Introduction

Start of informative comment:

The Trusted Computing Group's architecture and specifically the TCG architecture as described in the TCG Main Specification is a platform independent architecture to enhance trust on computing platforms. As such, the TCG Main Specification is general in specifying both hardware and software requirements. The goal of the TCG member companies is to ensure compatibility among implementations across many computing environments, such as PC Clients, Servers, and Mobile etc. It is expected that companion implementation documents will be created for each of these environments.

This document serves as a guide for implementation for the server environment. This document is a generic document which is limited in scope. As such, it does not address details which might be interesting for certain classes of servers. Additional detailed architecture specifications may exist for specific server architectures. Specifically, this document defines:

- Glossary: A glossary of terms as canonically used in a server environment. This section will only contain those terms, which are unique to the server environment.
- Server TPM Requirements: A set of requirements for a Server Embodiment of a TPM and it's incorporation into a server environment
- Main Specification Analysis: Analysis of the TCG Main Specification determining aspects of the Main Specification which are mandatory for a Server TPM implementation and those which are optional.
- TBB Definition and Requirements: The abstract definition of the components of the Trusted Building Block for a Server
- PCR Utilization: Guidelines on the usage of PCR registers in the Pre-Boot state through the transition to Post-Boot state.

This specification is based on the TCG Main Specification. The reader is expected to have an understanding of the concepts, defined functionality, and terms expressed in that document. This specification will attempt to minimize the duplication of information from that document; therefore, concepts and terms defined in the TCG Main Specification will not be defined in this document. If there is a conflict in interpretation between this and the TCG Main Specification, the concept or functional description as defined in the TCG Main Specification will take precedence.

As server architectures vary significantly, the Server Working Group is creating an abstracted definition of a TPM enabled Server (as opposed to the approach taken by the PC Client Specific working group specifying great detail on the actual implementation and interfaces). The server WG fundamentally considers that the functional specification for the TPM is at the ordinal level (as defined in the TPM Main Specification). . Additional architecture specific documents may be created where there is common architecture dependent implementation detail (such as IA32 specific architectures, bus interfaces, ECPI programming interfaces); however those documents will only serve to clarify specifics outlined in this document. Individual vendors may choose to publish their interfaces below the ordinal level, but it is not required.

The Server TPM will respond to ordinals as defined in the TPM main specifications. For each ordinal, the Server Specification will detail the need of a Server TPM to either require implementation, allow implementations, or prohibit implementation (i.e., MUST, MAY or Not allowed).. The general Server Specification will avoid enforcing implementation choices such as bus usage, interface, or physical structure.

The Server Specification will outline the fundamental requirements for the Trusted Building Block on Servers. Conformance in a Server is defined by a protection profile (the TBB PP), which is an additional output of the Server Working group for the Conformance Working Group. This server specification will define the requirements of TBB that will be needed to be conformant with the TBB protection profile. The specification will define the requirements on a systems construction such that it can meet the various protection profiles.

The Server Working group is not explicitly covering, within the scope of this specification, other implementation details that are traditionally where platform manufacturers provide differentiation. The platform manufacturer is responsible for addressing concepts such as Reliability, Availability and Serviceability (RAS), and Scalability on their Servers in a manner that does not compromise the security requirements of TCG.

End of informative comment.

2 TCG Server Context

Start of informative comment:

The concepts and descriptions of server architectures are described below in both the diagram and the descriptions. While the diagram infers physical connections, the connections and associations between the components are logical.

End of informative comment.

2.1 Basic Isolated Computing Environment Architectures

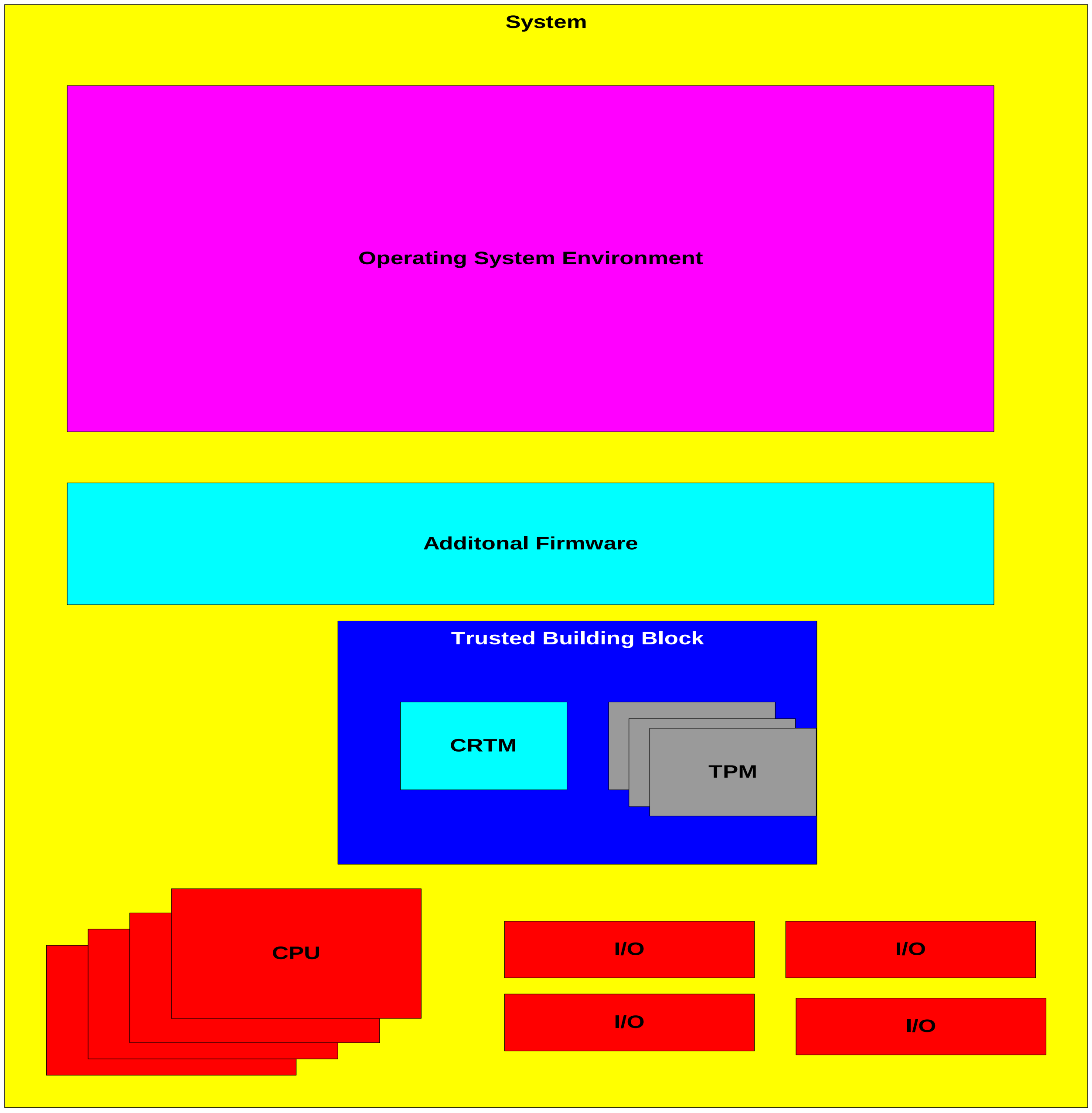


Figure 1

Start of Informative Comment

Figure 1 is a representation of a typical Multiprocessor server, which also applies to uniprocessor servers. Such servers come in many different form factors, from desk-side, rack mounted 1U systems to blade environments. The basic architectural representation is of a partition, which from a trust viewpoint is an isolated computing environment that hosts an operating system environment. Physical server hardware which may be partitioned either physically or logically divides the physical resources of a server into multiple instances of the above model (i.e., multiple partitions) as long as the separation from the trust standpoint is maintained.

Figure 2a is a diagram that represents this view that each "partition" can be considered separately, and therefore maintains its own TPM. A partition is equivalent to a physically or logically distinct machine. Partitions provide the environment where a single instance of an operating system executes in a manner that is INDEPENDENT of other partitions, just as physical machines' operating systems execute independently of each other.

End of informative comment

. .

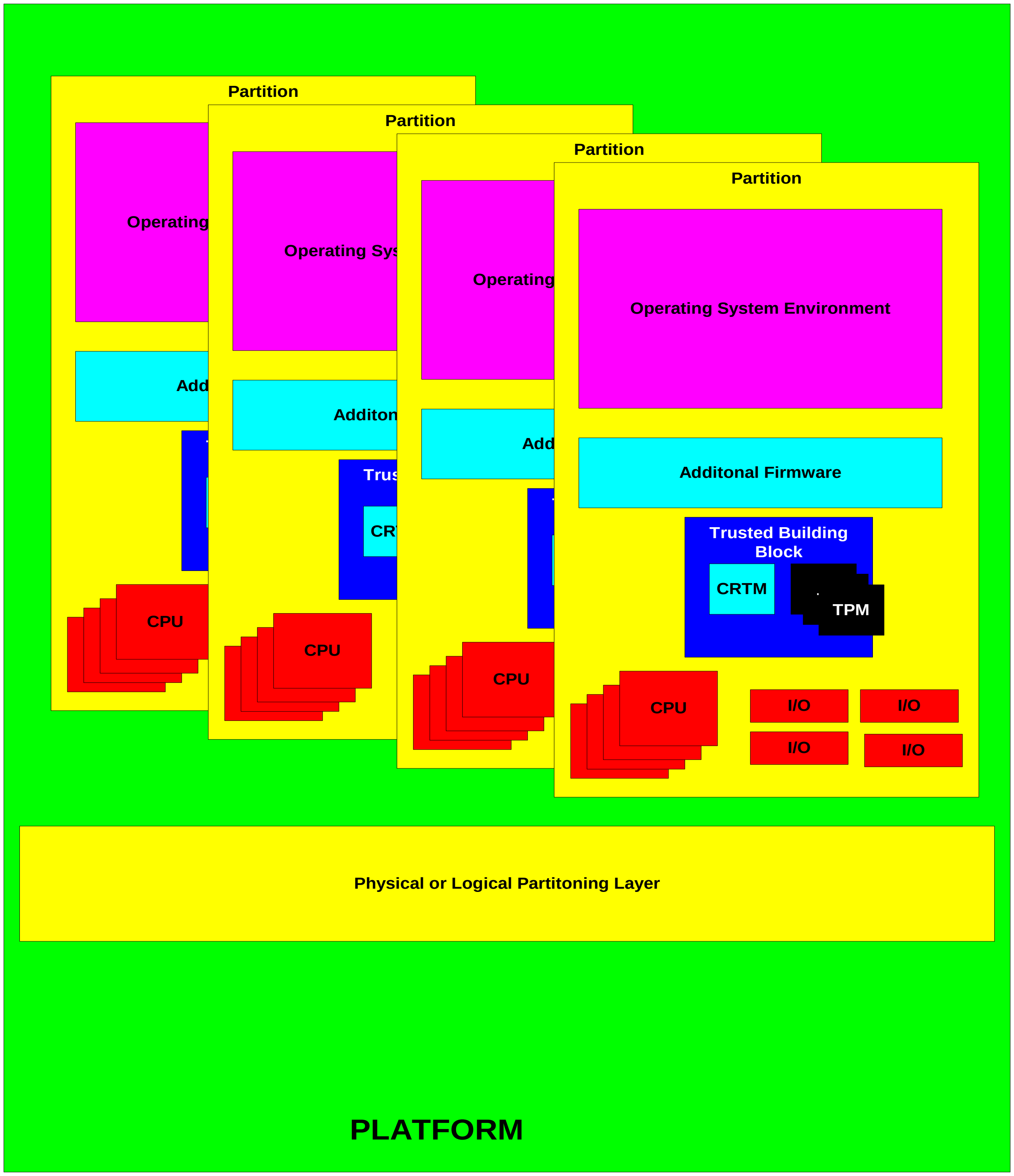


Figure 2a: Generic Server Platform

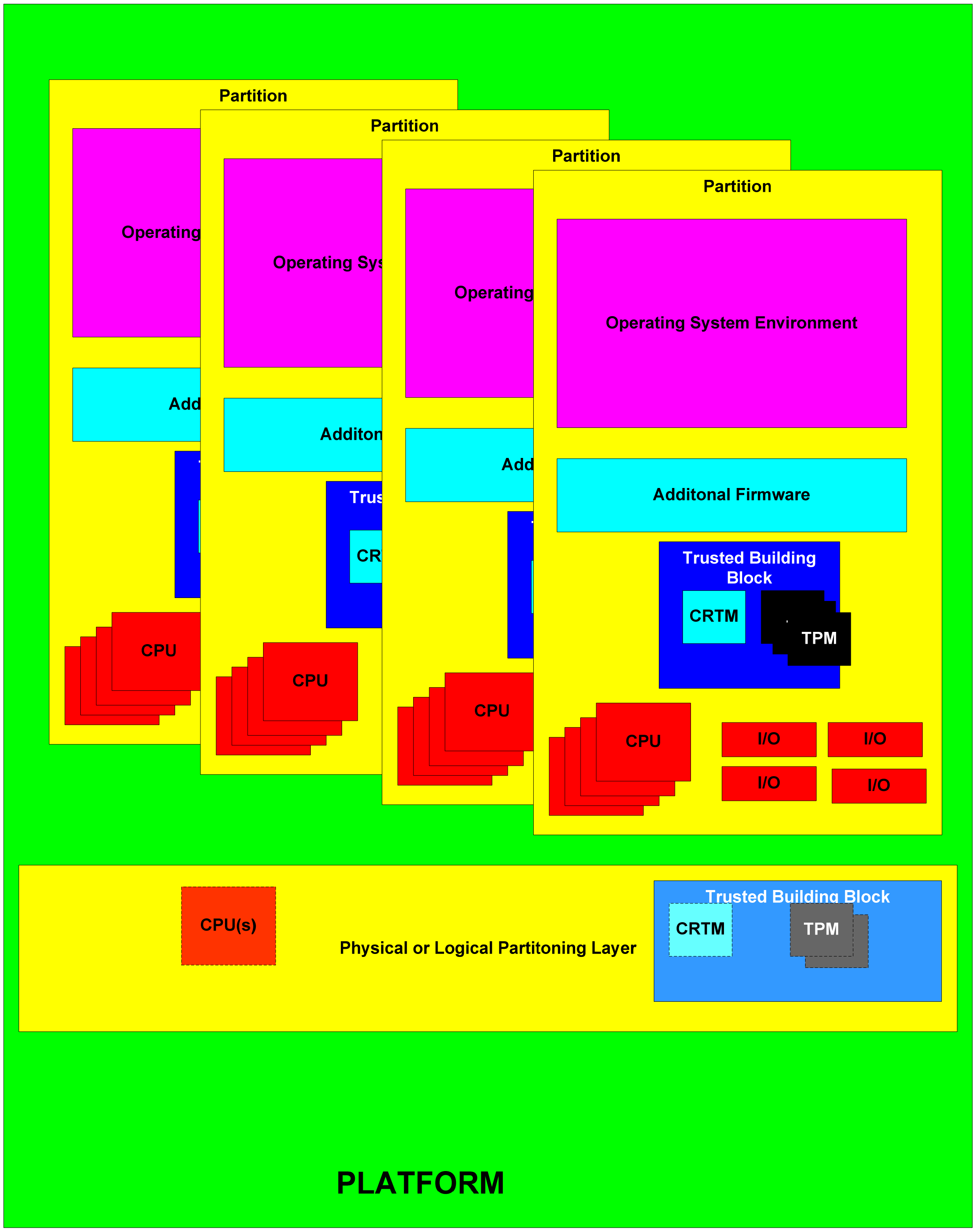


Figure 3b: Generic Server Platform (alternative view)

Start of Informative Comment

Figure 2b is a diagram that represents the view that the “partitioning layer” can be implemented as a separate computing engine that maintains its own TPM. Such a partitioning layer is equivalent to a physically or logically distinct machine. As such, the partitioning layer executes in a manner that is independent of the partitions, just as physical machines execute independently of each other. Because such a partitioning layer creates some or all of a partition, it is important that a partition in Figure 2b include the partitioning layer in the chain of trust.

End of informative comment

2.2 Server Concepts

2.2.1 Partitioning

Many server architectures may be partitioned (allowing for the concurrent execution of multiple partitions independent of each other). From a Trusted Computing view it is important to differentiate between the Platform and the partition. A partition is viewed as a representation of an isolated computing environment within a higher level combination of hardware and resources (the platform). Throughout this document the following definitions are used

Partition: the hardware and firmware environment, which provides the support for the execution of a single operating system image within a separated trust environment.

Partitioning Layer: the hardware, firmware, and/or software that divides a server’s resources into separate isolated partitions.

Platform: the complete package of physical hardware and firmware that a manufacturer produces for a single server product. This may be configured as one or more partitions.

In a non-partitioned environment there exists, for definition purposes, a single partition. Therefore, partition == Server == Platform. In a partitioned environment multiple partitions may exist. Within the domain of this document, a TPM can be bound to only one partition. However a single partition may have multiple TPM's bound to it, but in no case can a single TPM be bound to more than one partition.

The basic premise in the TCG partition abstraction is that each operating system environment has a unique RTR, RTM and RTS. The CRTM and Endorsement Key may be the same in an environment which supports multiple partitions, however each partition ends up with a unique set of measurement values and key storage capabilities through the bring up of the actual environment.

2.2.2 Core Root of Trust Measurement

Any server, which is to be a “TCG” conformant server, MUST contain an IMMUTABLE Core Root of Trust Measurement. The CRTM is a mandatory element of the TBB and MUST be under the sole control of the server manufacturer. The CRTM is the a priori trusted code that is part of the platform credential. In the Static RTM Model, this MUST be the very first piece of code executed on power on or upon reset of the server or complete physical hardware environment. In the Dynamic RTM model, the CRTM is the atomic sequence of processor instructions (operating as an atomic operation) which instantiate a “trusted” SW environment. The scope and size of the CRTM is left to the server manufacturer, however it MUST meet the immutability requirement and the TBB Protection Profile.

The trust in the server is based on this component and the platform credential. The trust in all measurements is based on the integrity of this component, and this component should

at some point in time have had TOTAL control over the server environment (both physical and logical) whose metrics are recorded in PCRs.

In order to meet the immutability requirement, the platform Manufacturer must control the update, modification, and maintenance of the entire CRTM, per the TBB Protection Profile requirements.

2.2.3 TPM

As stated in the introduction, the TPM within a server is a black box implementation that meets the criteria set forth in the TCG Main Specification. The mandatory ordinals specified within the Main Spec and this document MUST be implemented. Other TPM concepts such as Locality MAY be implemented in a manufacturer defined manner.

Start of informative comment:

As an example: While the Main specification defines 5 "localities", a Platform Manufacturer may decide to only implement Locality 0. Another Platform Manufacturer may implement other Localities. These differences would be communicated to end user applications via attributes within the various TCG credentials.

End of informative comment.

2.2.4 Resets

Resets are by their very nature undesirable in servers, given the uptime requirements. The incorporation of a TPM into a server does however put some fundamental requirements into this area.

2.2.4.1 TPM Init

A reset of the embodiment of the TPM within a partition requires that the partition itself be reset. A reset TPM has lost all of its "trust measurements" and therefore does not reflect the true state of the partition from a trust point. If more than one TPM is bound to a partition, then if one is Inited, then all must be. Upon TPM Init on ANY/ALL TPM(s) bound to a partition, if the partition implements the S-RTM model, the partition MUST start execution from its CRTM. TPM Init of one partition's TPM MUST not affect the security properties of other partitions.

2.2.4.2 Partition Reset

Reset of the partition itself MUST result in TPM Init occurring and execution of the partition MUST start from the CRTM, thereby establishing a new transitive chain of trust. This includes both soft and hard resets (power on resets).

2.2.5 Trusted Building Block (TBB)

Start of informative comment:

The combination of the CRTM, TPM function, connection of the CRTM to the system/platform, and the connection (or binding) of a TPM to the system/platform makes up the TBB. Trust in the connection of the CRTM to the TPM is transitive, and relies upon trust in the CRTM connection to the system/platform and in the TPM connection to the system/platform.

Server platforms may differ from other platforms in that TPM functionality may be physically distributed within the platform. However, the TBB encompasses the RTM (either static and/or dynamic), the TPM and the connections between them, these connections may be physical or they may be logical.

The RTM portion of the TBB may also be implemented as a Static RTM (S-RTM) or Dynamic RTM (D-RTM).

The TBB is an important construct within the trusted computing environment. The TBB, and the RTM in particular, represent the “a priori trusted function” that the platform credential represents, and really is not measured The scope of what exists in the TBB is a System manufacturer dependent definition. The TBB is certified through the Platform Certificate against the TBB Protection Profile. For some common server architectures additional specifications may be created to establish commonality in this area

End of informative comment.

The content and scope of the TBB MUST be under the complete control of the platform manufacturer, and MUST achieve the same immutability of the CRTM. The platform manufacturer MUST ensure that any function within the TBB does not create mechanisms by which the integrity of the TPM or communications with the TPM can be attacked.

2.2.5.1 Connection of TBB to the Server

The TBB is the combination of the CRTM, TPM, connection of the CRTM to the platform, and the connection of the TPM to the platform.

The TBB MUST be capable of placing either the entire platform or a single partition into a known trusted state, either from reset in the case of an S-RTM, or from a dynamic start-up event, in the case of a D-RTM.

2.2.5.2 TBB Startup

When using the S-RTM model, the RTM MUST have control of the TBB upon Platform Reset.

Upon start-up of a dynamic trusted execution environment using the D-RTM model, the RTM MUST have control of the TBB..

Start of Informative comment:

In one manifestation of the static environment, locality 0 must be asserted at the time the RTM gains control.

In one manifestation of the dynamic environment, locality 4 must be asserted at the time the RTM gains control.

End of informative comment:

A platform which supports both S-RTM and D-RTM contains two TBBs, by virtue of the fact that there are two distinct RTMs which share a common RTR.

2.2.5.3 Physical Presence

Start of informative comment:

Within the trusted computing environment, there exists a need to allow for certain operations to be performed irrespective of the knowledge of authorization data. These mechanisms are intended to insure that upper layer software cannot perform specific operations against the TPM. These “operator” types of operations require what has become commonly referred to as Physical Presence.

Physical Presence operations cover two classes, allow for re-setting of the device if all authorization data has been lost, and to control the availability of the TPM device to upper layers of software.

End of informative comment.

Platform Manufacturer MUST provide a mechanism that meets the intended goal of Physical Presence. The assertion of this privileged state MUST NOT create avenues of attack, and MUST be via mechanisms which can not be performed through upper level software. Assertion of physical presence may be through a “trusted process” or “trusted code” which meets the requirements of this specification, and the Protection Profile.

2.2.6 Server Bindings

A TPM MUST be bound to a partition in an N to 1 relationship, by this it is meant that a single TPM MUST be bound to one partition, however a partition MAY be bound to more than one TPM. Such a binding must guarantee that only that particular partition can use the TPM once the binding has been made. If more than one TPM is bound to a partition all measurements MUST be made to and available from all TPMs within the partition. The TPM Main Specification allows for either a physical or logical binding to occur, and either is acceptable from a Server specification viewpoint. The binding mechanisms are an element of the TBB.

Start of informative comment:

Bindings between an Endorsement Key, a TPM, and a Partition

The relationship between the Endorsement Key, a TPM, and a partition is described in Section 2.2 of the TCG Main Specification. In order to support partitionable servers in a reasonable manner, a single Endorsement Key may be associated with the entire hardware environment and utilized by all partitions within that environment. Each partition must have all other attributes of a TPM as unique (the uniqueness of the SRK and TPMProof assures that the AIKs associated with one partition are not accessible by another); this includes PCRs, counters etc. The base monotonic counter may also be shared as common.

End of informative comment.

2.2.7 Dynamic Reconfiguration

Start of informative comment

In large server environments, it is common to add and remove hardware dynamically without causing a system reset. In some respects this is very similar to the insertion of a PC-Card device into a Laptop. However, different architectures reflect these asynchronous events in different manners. The platform manufacturer should provide mechanism by which the post-boot environment is notified that an asynchronous event occurred. Measurement of these events is the responsibility of the post-boot software environment. The post-boot environment should not alter the pre-boot measurements for these events, although this is not enforced other than by the convention that the pre-boot measurements not change.

End of informative comment.

2.2.8 Platform State

In a TCG Server supporting a Static CRTM, control and maintenance of each PCR is assigned to the Firmware or to the Operating System; Firmware PCRs and logs are read-only to the OS and vice versa (the read only attribute is a convention maintained by TSS, and not strictly enforced). System Firmware provides interfaces for OS-level software to obtain the log when needed ("at run-time"). Many server platforms provide Firmware that is "re-entrant" such that environmental events are handled by the firmware (e.g., IO card insertion or removal). The TSS will not modify the measurement logs assigned to the Firmware if the Firmware is re-entrant. For a TCG Server with once-entrant Firmware, the TSS will maintain the OS and Firmware measurement logs in the post-boot phase and the pre-boot PCRs will remain unchanging once the boot process ends

In a TCG Server supporting only a Dynamic CRTM, the state of the Firmware PCRs and logs is not defined. All PCRs and measurement logs must be managed by the Dynamic CRTM or its successors and delegates.

Restating, there are three cases of interest:

1. S-RTM, once-entrant Firmware: Firmware owns PCRs and logs in pre-boot phase; in the post-boot phase, the Firmware PCRs are treated as read-only. The OS PCRs and logs contain everything that happens after the boot transition (pre-boot to post-boot).
2. S-RTM, re-entrant Firmware: Firmware owns PCRs and logs during pre-boot and post-boot phases. The OS PCRs and logs are maintained by the OS. The TSS in the OS has a method to obtain the current PCRs and logs from the Firmware.
3. D-RTM: the OS always owns the OS PCRs and logs, and the Firmware PCRs are undefined. Until the D-RTM loads, the PCRs and logs are undefined; if the Dynamic RTM unloads, the PCRs and logs become undefined until the D-RTM is again loaded.

A high level view of the structure of the system environment might be represented by Figure 3:

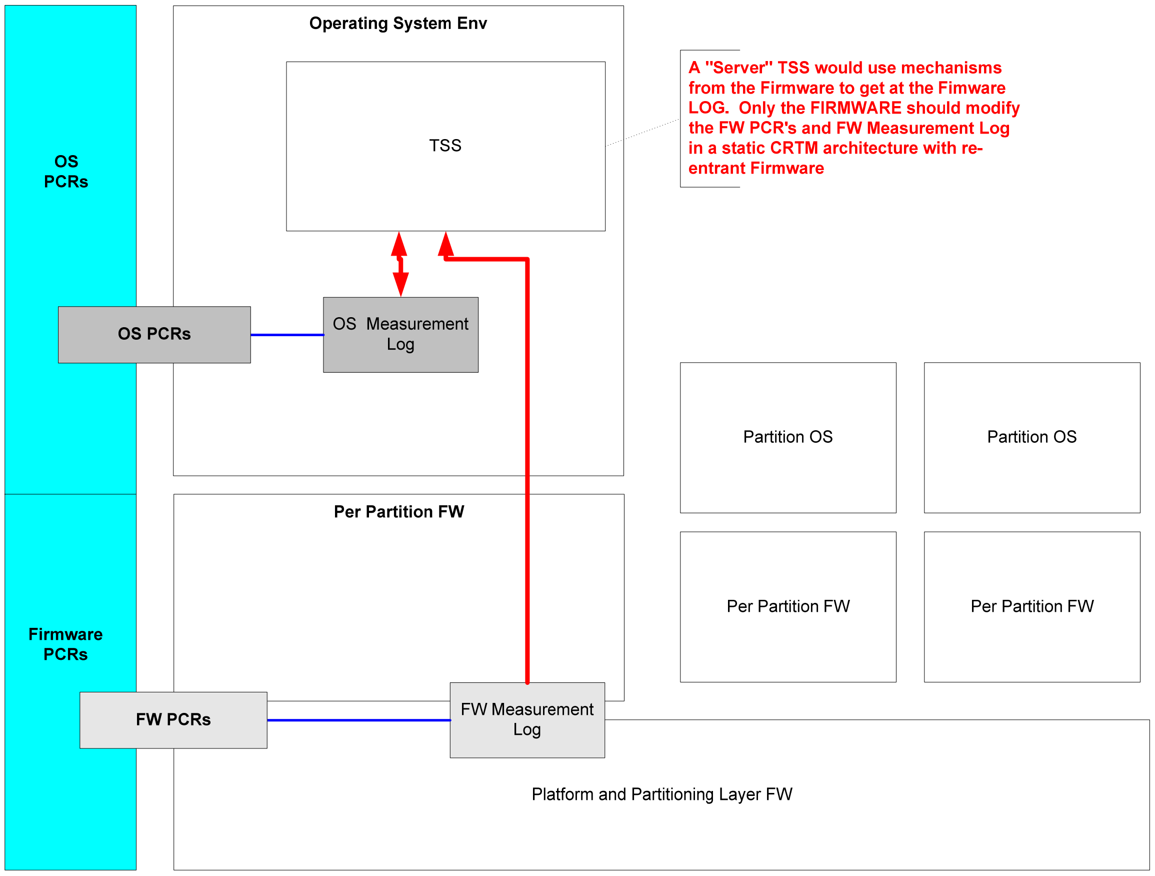

Operating System Env
A "Server" TSS would use mechanisms from the Firmware to get at the Fimware LOG. Only the FIRMWARE should modify the FW PCR's and FW Measurement Log in a static CRTM architecture with re-entrant Firmware
OS PCRs
TSS
OS PCRs
OS Measurement Log
Partition OS
Partition OS
Per Partition FW
Per Partition FW
Per Partition FW
Firmware PCRs
FW PCRs
FW Measurement Log
Platform and Partitioning Layer FW

Figure 4

3 Server TPM Requirements

3.1 Levels of requirements

Start of informative comment:

The Server specifications are written at two levels: a **Generic Server Specification**, and **Implementation Specific Specification**.

1. Generic Server Specification requirements

The Generic Server Specification applies to all implementations. As a result it describes the requirements at a high level to provide the flexibility for a variety of implementations. The Generic Specification describes features and requirements that all the various Specific Implementations must adhere to, but does not require optional features to be implemented. Such specific implementations may be proprietary, or may conform to one of the Implementation Specific Specifications (see below). This document is the Generic Server specification.

2. Implementation Specific Specification requirements.

Implementation Specific requirements focus on a more bounded set of requirements; for example particular processor architectures, or range of configurations, etc. Such Implementation Specific Specifications need only implement the mandatory parts of the Generic Server specification. However they may also more closely define some aspects where appropriate to enable interoperability within a more structured ecosystem of components and vendors.

Implementation Specific Specifications will be added for various classes of servers over time.

End of informative comment.

3.2 Types of servers and server configurations

Start of informative comment:

Servers come in many shapes and forms and are used in a variety of different ways to meet the application requirements. The servers usually form part of a larger system or configuration that involves internal and external peripherals, such as storage, networking, and other components.

The trusted computing environment may or may not extend to include the peripherals, network, and other components, but Server specifications include the necessary features to enable, where appropriate, the proper support for the wider trusted environment. However this Generic Server Specification does not describe the areas beyond the server TPM environment – these are covered in other TCG specifications.

End of informative comment.

This Generic Server specification MUST allow support for:

- Uni-processor (UP) servers.
- Multi-Processor (MP) servers.
- Partitioned server platforms running multiple software images.
- Servers based on the TCG PC Client specifications with version 1.1 or 1.2 TPMs.

Start of informative comment:

More details of these server models and configurations are described in the ***Context Section*** of this document.

End of informative comment.

3.3 Application environments

Start of informative comment:

One of the goals of the Server specification is to enable a consistent application environment across the range of different vendor and different server model platforms. Application software suppliers can expect to be able to deploy their applications across the range of servers (that claim compliance to the TCG server specifications) without implementation dependent customization to their software that handles the various TCG subsystems and components.

End of informative comment.

3.4 System software and firmware environments

Start of informative comment:

Generally the operating environment of different server platforms contain vendor and platform specific implementations to enable and communicate with the specific platform features. This Generic Server specification does not require any conformance in this area, however the implementation specific specifications may require conformance to certain Operating System and/or firmware implementations and may require work with the appropriate standards organizations in these areas.

End of informative comment.

3.5 Server scalability and redundancy

Start of informative comment:

Servers provide scalable features to address the demands of the different applications they support. Such scalability is usually offered in a number of different areas, such as CPU performance, memory & storage capacity, and network & I/O performance.

Different applications demand different mixes of these resources and TPM performance and capacity must be able to offer similar flexibility.

Many servers also offer redundancy for such resources as well – either as single system redundancy, or clustered redundancy.

One implication of redundancy, particularly for single system redundancy, is that failing components require seamless isolation and replacement while the rest of the system continues to operate normally.

Scalability also implies that server capacity can be expanded over time to meet increasing application demands. For example CPUs, memory, disks, and I/O cards can be added when needed, and in some servers, hot-added into an active system.

End of informative comment.

This Generic Server specification does not specify how TPM scalability and redundancy are implemented.

3.6 Server partitions and virtual environments

Start of informative comment:

Servers that are capable of supporting partitions for multiple virtual Operating System environments also need to provide the corresponding virtual TPM environments to be able to offer the TCG application environment consistently with non-partitioned server platforms.

End of informative comment.

This Generic Server specification does not specify how TPM functionality is implemented in partitioned or virtual server environments.

3.7 Server configuration and provisioning

Start of informative comment:

Server configuration and deployment software needs to incorporate the necessary TCG defined features to be able to properly configure the TCG TPM environment on each server. This configuration should be implemented in a standard way according to TCG specifications and guidelines to provide consistent interfaces and procedures in heterogeneous data center environments.

End of informative comment.

This Generic Server specification does not specify mechanisms or infrastructure for configuration or provisioning.

3.7.1 Physical Presence

The platform MUST provide an indicator of Physical Presence to the TPM consistent with the protection profile.

3.8 TPM interface

Start of informative comment:

Different server platforms may support different physical interface specifications and thus may or may not be able to support the LPC bus specification required for the PC client TPM interface. The actual physical implementation must be able to support the necessary functionality to be able to properly and securely deliver the commands and responses to the TPM.

End of informative comment.

3.8.1 Physical Interface

This Generic Server specification does not specify a physical TPM interface. However the security of the connection between TPM and CRTM (of which the physical interface is part) MUST meet the platform protection profile requirements.

3.8.2 Locality

Systems implementing a D-RTM model MUST provide the necessary Locality mechanisms on the physical TPM interface to allow the processor and the various software entities to communicate the appropriate Locality information to the TPM.

3.9 TPM command/response interface

Start of informative comment:

The TPM command/response interface provides the primary definition of the TPM functionality visible to the server.

End of informative comment.

Server systems implementing This Generic Server specification MUST use the command/response interface and data structures defined in the **TPM Main Specification – Part 3 Commands** and **TPM Main Specification Part 2 Structures** documents (V1.2) or **TCG Main Specification** (V1.1).

3.9.1 TPM_SetTempDeactivated Owner Authorization

Start of informative comment:

The TPM 1.x Specifications allows for the temporary disabling of a TPM using ordinal *TPM_SetTempDeactivated*. The 1.1 *TPM_SetTempDeactivated* command does not require any privilege to activate, and reactivating the TPM requires a reboot. Such behavior by unprivileged users is unacceptable in many server environments. The *TPM_SetTempDeactivated could be prevented from reaching the TPM by blocking code implemented in the TSS stack.*

Note: the 1.2 TPM Specification includes an authorization check prior to performing this *TPM_SetTempDeactivated* operation.

In the main TPM specification there is a requirement (in the discussion on Activating a TPM) that states the only way to reactivate a TPM is to power-cycle the system. In a server system in this context a “power-cycle” really means a “logical hard reset” during which TPM_Init is asserted and all TBB/PP required operations performed, even though power to the entire platform may, or may not, be disconnected and reconnected. Once TPM_Init has been asserted, any associated S-RTM partition must be reset, and any associated D-RTM partition must have any trusted environment sessions terminated.

End of informative comment.

A server using a 1.1 TPM SHOULD prevent a *TPM_SetTempDeactivated* command from reaching the TPM by implementing the necessary blocking software.

3.10 Root of Trust for Measurement

This Generic Server specification requires that either a Static Root of Trust for Measurement (S-RTM), or Dynamic Root of Trust for Measurement (D-RTM), or both, MUST be implemented by the server platform.

3.11 Monotonic Counters

Start of informative comment:

The Monotonic Counter in the TCG Main Specification (17.1) only requires 1 million sessions or not requiring roll-over for 7 years. These are conservative numbers for a Server-based TPM and server manufactures may want to specify higher requirements.

End of informative comment.

4 TPM Commentary

Start of informative comment:

This chapter discussed differences on how the PC (TPM) Specification is to be interpreted on Servers. These differences are based on TCG specification 1.2v62 and 1.2v75.

***End of informative comment*.**

4.1 TPM_Maintenance

***Start of informative comment*:**

The TPM_Maintenance mode described in section 13 of the TPM specification is recommended for servers. In addition, the restrictions in the TPM specification stating “it must only work on systems of the same model and type” are left to the system vendor's definition of what “same model and type” actually are.
***End of informative comment*.**

4.2 Platform Binding

Start of informative comment:

Server TPMs may include a variety of components of which some are removable while others are not. All components, particularly those which are removable, must be bound to the platform in a manner acceptable to the PP through either physical or cryptographic binding.

End of informative comment.

4.3 Multiple Systems and TPMs

Start of informative comment:

The use of TPMs in multiple systems and resulting impacts with the TPM specification are not discussed in this revision of the Server Specification. In the context of this comment multiple systems should be considered clusters and grids which are primarily software constructs.
***End of informative comment*.**

5 PCR Guidelines and Event Logging

5.1 Introduction

Start of informative comment:

This section defines the PCR assignments used for boot time integrity metrics and the methodology for collecting the metrics. During the system firmware boot process a log of all executable code is created and extended into PCRs as described below. As future TPMs become available with more PCRs, a future revision of this specification will need to define the usage of any additional pre-boot PCRs.

Each time a PCR is extended a log entry is made in a TCG Event Log. This allows a challenger to see how the final PCR digests were built. This specification allows the use of internal structure in the TCG Event Log. For example, a TCG Event Log could be created with internal separation of the log entries for each PCR. The TSS layer provides the abstraction for log retrieval. The location and PCR mapping of the Event Log are communicated in an architecture dependent fashion.

Use of the post-boot PCRs will be lightly discussed in the document, since the usage model for these PCRs is platform and operating system specific. The PCR guidelines in this chapter do not address all of the requirements for all server types, particularly partitioned servers. The appropriate Implementation Specific Specification should be consulted.

End of informative comment.

5.2 Locality

Start of informative comment:

There are 5 Localities defined in the version 1.2 client TPM specification: Localities 0 to 4 and the Locality Legacy, which is equivalent to Locality 0.

Localities are typically used when a platform/Operating System combination supports a dynamic root of trust model (RTM). The dynamic root of trust model relies on the various localities being bound to particular levels of trust, from normal trust up to hardware level trust. Localities other than zero are typically not needed for the static root of trust model, where the chain of trust is created from execution of the CRTM and extends until the operating system gains control of the platform.

The server platform may support localities; however, the usage model of localities, if supported, is platform and operating system dependent. In addition, the definition of how a system asserts a locality to the TPM is outside the scope of this document. As in the client space, a higher locality may infer a more trusted environment.

End of informative comment.

The server platform hardware MUST support the SRTM or DRTM trust model. The server platform hardware may support both trust models.

The server platform hardware is not required to support localities, except as required by the trust model.

A platform/Operating System combination MUST use PCR 17-22 in a manner conformant to the attributes assigned to those PCRs.

Start of informative comment:

If the TPM can be configured to allow all localities to access this PCR range (17-22), then the mapping of locality to PCR is implementation dependent.

PCR Register 23 use is not covered in this specification.

This document will not describe the methodology to communicate the usage of pre-boot PCRs.

End of informative comment.

5.3 Pre-Boot PCR Usage

Start of informative comment:

The goal is to use the pre-boot PCRs in the same way as the client. This will ease the burden placed on those entities that are attempting to interpret these PCR values. However the definition of the PCR assignment will be broadened appropriately to cover all the server architectures.

This section describes the basic PCRs that are common between TPM 1.1b and TPM 1.2 main specifications.

.End of informative comment.

A platform that supports both the static and dynamic RTM models MUST use the PCRs as described in the following sections, even though the dynamic RTM model does not require this complete usage model. A platform that supports only the dynamic RTM MUST use the PCRs only as annotated for dynamic RTM

5.3.1 PCR[0] – CRTM, system firmware, and system extensions

The base requirement is the server MUST measure the following into PCR[0]:

- the CRTM's version identifier

The CRTM version identifier should be a globally unique value, which a platform manufacturer can use to identify the type of platform, and the type and version of CRTM.

Start of informative comment:

The interpretation of the CRTM version identifier string is outside of the scope of this specification. For example, extending the hash of the CRTM is acceptable.

End of informative comment.

The scope of the CRTM is up to the platform manufacturer to define, however it MUST meet the requirements of the TBB.

A server may measure multiple version IDs into PCR[0].

Start of informative comment:

Multiple version IDs may be useful to track the revisions of multiple firmware modules.

End of informative comment.

If the CRTM contains the following types of firmware, then they are not required to be measured and extended into PCR[0]:

- All firmware physically bound to the system which will be executed on the main system processor, including system firmware, embedded firmware device drivers, and any firmware that may run only in special modes of processor execution.

5.3.2 PCR[1] – system configuration

The following types of information (unless already covered in the CRTM) MUST be logged into PCR[1], if present in the system:

- Processor microcode upgrades

- Each processor's system identity

The processor system identity MUST provide a unique value for each processor in the system at boot time.

Start of informative comment:

The processor system identity is platform and architecture dependent. It is recommended that processor serial numbers NOT be logged to meet the processor system identity requirement.

End of informative comment.

The following information may be logged:

- System firmware internal data structures stored in non-volatile memory
- System firmware data structures to be passed to the post-boot environment
- Per-processor data structures

Start of informative comment:

The order of these measurements is implementation dependent, as in the client specification.

End of informative comment.

If the platform is not able to log the processor microcode upgrades into PCR[1], then the processor microcode upgrades MUST be part of the CRTM of the platform.

Start of informative comment:

For example, some platforms may apply processor microcode upgrades through a side-band interface, and the main system processor cannot access the microcode upgrades for logging. In this case, the subsystem driving the side-band interface is warranted by the platform manufacturer to be part of the CRTM of the platform.

End of informative comment.

5.3.3 PCR[2] – firmware driver code

The following types of information, if utilized on the platform architecture, should be logged into PCR[2]:

- Firmware resident on an IO card, that are intended to execute on the main system processor.
- Firmware device drivers embedded in the system, but able to be updated by an entity other than the system manufacturer.

The server platform MUST log only pre-boot measurements into PCR[2].

If the platform architecture does not utilize these types of device drivers, the platform may utilize PCR[2] in a platform architecture defined fashion. However this new use of PCR[2] MUST be documented as specified in the platform architecture document, so that the consumer of PCR[2] has visibility to that fact.

5.3.4 PCR[3] – firmware driver Configuration and Data

The following types of information, if utilized on the platform architecture, should be logged into PCR[3]:

- Firmware device driver configuration information, for example settings that affect the boot process, such as boot device scanning, SCSI multi-initiator settings, etc.

The server platform MUST log only pre-boot measurements into PCR[3].

Any pre-boot application that modifies the firmware device driver configuration MUST measure the new configuration into PCR[3] or cause a system reset.

If the platform architecture does not utilize firmware device drivers, the platform is free to utilize PCR[3] in a platform architecture defined fashion. However this new use of PCR[3] must be documented as specified in the platform architecture document, so that the consumer of PCR[3] has visibility to that fact.

5.3.5 PCR[4] – Initial Program Load (IPL) Code

Start of informative comment:

PCR[4] is the transition PCR, measuring the IPL code that begins the transition of the platform from the pre-boot state to the post-boot state. Examples include bootp, grub, mbr, and universal boot.

End of informative comment.

The following types of information MUST be logged into PCR[4]:

- Each IPL that is attempted and executed.
- Additional code that is loaded by the IPL prior to handoff to the loaded code.

Start of informative comment:

There may be platform architectures that utilize specialized platform subsystems to load the operating system directly. These subsystems play the role of IPL code on more traditional platforms. In some instances this code may not be measurable on the main processor.

End of informative comment.

IPL code that cannot be measured, MUST be part of the TBB of the server platform.

5.3.6 PCR[5] – IPL Configuration and Data

Start of informative comment:
The IPL Code may have configuration or other data that is relevant to the trusted properties of the Platform. An example of this is IPL code that allows the selection of alternate boot devices or boot paths. In this example, the boot device selection information would be logged to this PCR by the IPL code.
End of informative comment.

The following types of information MUST be logged into PCR[5]:

- All relevant IPL configuration data, such as boot device or boot path settings
- Static data contained within the IPL Code

5.3.7 PCR[6] – State Transition

Start of informative comment:
Events recorded to this PCR are events related to State Transitions.
End of informative comment.
The following types of information MUST be logged into PCR[6]:

- All State Transitions relevant to the trust state of the platform
- Power management events which occur without operating system visibility and are controlled by software that is not part of the platform TBB

Start of informative comment:

Power management events may be performed by the service processor on a server platform. The service process may be part of the TBB of the platform, and as such would be exempt from logging power management events.

Server platforms will have a great variety of platform/vendor specific states, affecting the trust state of the platform that will be logged. PCR[6] will be used for that purpose, but the specifics of the logging will be covered by architecture specific specifications.
End of informative comment.

5.3.8 PCR[7] – Reserved

Reserved for future pre-boot use.

5.3.9 PCR[8] – PCR[15] Reserved

Reserved for OEM use, if the TPM provides at least 24 PCRs.

5.3.10 Unusual Pre-boot Actions

Start of informative comment:
There are actions that may be performed outside of the normal pre-boot time boot sequence that affect platform security state, for example: hot plug operation, firmware update, etc..
End of informative comment.
The trusted server platform MUST perform one of the following, when performing pre-boot actions that affect platform security state:

- Measure the change into the appropriate pre-boot PCR
- Cause a platform reset.

5.3.11 Post-Boot Actions and PCR Usage

Although this specification does not cover details of post-boot PCR usage, there are several principles that MUST be followed regarding post-boot activity:

- If a platform has the capability to affect changes to platform security state through operations such as processor microcode upgrades, without Operating system involvement, that capability MUST be logged in pre-boot PCRs.
- If the Operating System has the capability to affect changes to platform security state through operations such as upgrading processor microcode, it is the Operating system responsibility to measure either the module responsible for that capability, or each state change in a post-boot PCR.

If the platform has the capability to perform post-boot modifications to the platform security state, it may not be possible for the event to be logged. However, one of the following MUST be true:

1. The sub-system performing the run-time changes MUST be part of the CRTM as defined by the platform manufacturer.
2. Each modification event must be measured by run-time software.

Start of informative comment:
Some examples of post-boot modifications, include update of processor microcode, update of system firmware, and update of field programmable ASICs.
End of informative comment.

5.4 Event Logging

5.4.1 Measurement Event Log

Start of informative comment:
TCG_HashLogExtendEvent Events are logged using the TCG_SERVER_PCR_EVENT structure which is defined and documented in this specification. These structures are stored as unstructured arrays within one or more data areas whose location is communicated using an architecture dependent method. None of the pre-Boot entities, are required to interpret this data. Once the Post-Boot State controls the platform, the Post-Boot OS is expected to read this data and transfer it to its own event log.

The TCG_SERVER_PCR_EVENTstructure is considered to be server platform-specific. It is the responsibility of the platform's software (e.g., a TSS) to understand this structure and convert it to any relevant format. Because this structure contains cryptographic information (i.e, hash values) it is important to understand what can and cannot be converted or altered.
End of informative comment.

The instantiation of the event log MUST be an array of TCG_SERVER_PCR_EVENT structures as defined below.

5.4.2 Platform Independent Event Log Structure

Platform independent events MUST be logged using the events identified in the TCG Main Specification.

Start of informative comment:
Examples of these are Validation Certificates. These are logged using the EV_CODE_CERT event type.
End of informative comment.

5.4.3 Server Platform Specific Event Log

For the events described in this section, the EventType SHALL be EV_SERVER_PLATFORM_SPECIFIC and the event field within the TCG_SERVER_PCR_EVENT structure SHALL be the PlatformSpecificEventLogStruct as defined in a subsequent section, Platform Specific Event Log Structure.

5.4.3.1 PCR Event Structure

Start of informative comment:

This structure provides information about an individual PCR extend event.

End of informative comment.

Definition:

```
typedef struct TCG_ServerPCREventStruct{
  UINT32    pcrIndex;
  UINT32    eventType;    // EV_PLATFORM_SPECIFIC ; For backward compatibility ???
  UINT8     digest[20];
  UINT32    eventDataSize;
  UINT32    event;
  } TCG_ServerPCREventStruct;
```

5.4.3.2 Platform Specific Event Log Structure

For the events described in this subsection, the field TCG_ServerPCREventStruct.eventType SHALL be EV_PLATFORM_SPECIFIC.

The field TCG_ServerPCREventStruct.event SHALL be a PlatformSpecificEventLogStruct structure.

The server platform events shall be logged according to the following structure.

```
typedef struct TCG_ServerPlatformSpecificEventLogStruct{
    UINT32     EventID;          // Tag from arch. Specific Spec.
    UINT32     EventDataSize; // Size of EventData
    UINT8      EventData[];      // EventData
        } TCG_ServerPlatformSpecificEventLogStruct;
```

5.4.3.3 Server Platform Specific Event Tags

This covers items such as non-volatile data, card resident firmware driver execution etc. Most of these event tags need to be covered in the architecture specific specifications. Architecture specific event tags are in the range of 0x0080 – 0x00FF.

5.4.4 EV_ACTION Event Types

The following actions strings are defined. The strings below are enclosed in quotes for clarity; the actual log entries SHALL not include the quote characters. They SHALL be logged using the following:

TCG_ServerPCREventStruct.EventType = EV_ACTION
TCG_ServerPCREventStruct.Event[] = ASCII string of the following:

String	Purpose and Comments	PCR
"Starting Pre-OS App"	System firmware is starting a code image. If no additional strings posted in log that means that the software which was started did not return control to system firmware.	4
"Returned Pre-OS App"	System firmware Received control back from prior Pre-OS App invocation. If the called code is not TCG-aware it may have loaded additional unmeasured code. However there is a log entry showing entry to (and measurement of) untrusted code.	4
"Booting s"	System firmware is loading an IPL module from a boot path. This is used, if system firmware can identify an OS boot loader is being loaded. The value 's' is the boot path that describes the boot device.	4
"User Password Entered"	User has entered the correct user password.	4
"Administrator Password Entered"	User has entered the correct administrator password.	4
"Password Failure"	The typed password did not match the stored password after a specified number of retries.	4
"Boot Sequence User Intervention"	User altered the boot sequence	
"Chassis Intrusion"	The chassis, or subsystem bay on a larger server, was opened.	1
"Non Fatal Error"	A non-fatal System Firmware error (e.g. keyboard failure) was encountered. This is any error that does not prevent the system from continuing the boot process	1
"<IO Card Driver Specific non-IPL String>"	An IO Card Driver vendor specific string for non-Boot/IPL events.	3
"<IO Card Driver Specific IPL String>"	An IO Card Driver vendor specific string for Boot/IPL events.	5

6 Glossary

6.1 Definition of terms used in this Glossary

The purpose of the definitions in this Glossary is to provide a taxonomy that describes and allows us to talk about the structure of the computing environment.

6.2 Fundamental Definitions

The following set of definitions describe fundamental atomic components that comprise a computing machine.

The concepts described by these terms are not intended to be directly instantiated, rather they are used to develop and define by inheritance (and aggregation) a set of more complex terms which describe the structure of a computing environment.

6.2.1 Hardware

The physical instantiation of a computing environment.

6.2.2 Software

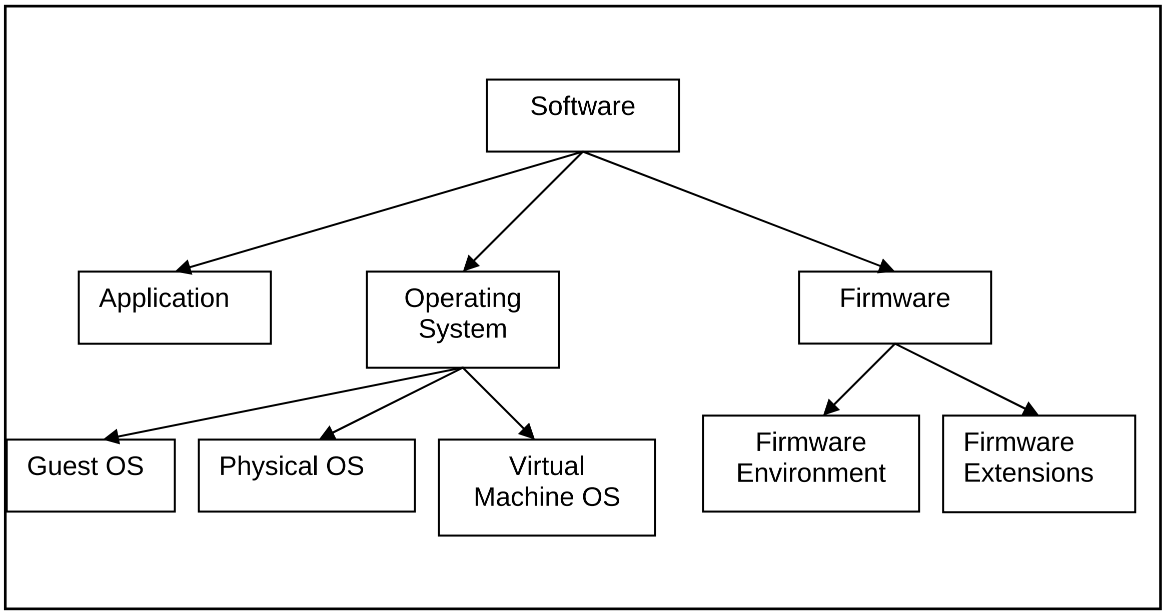
The programs of machine language instructions and data that are interpreted by the hardware[1].

6.2.3 Detailed description of the Software Environment

The following diagram shows a simple inheritance diagram of Software which serves as a basis for many of the definitions in this glossary:

[1] Operating Systems (second edition); H. M. Deitel p24.

Software Object Inheritance



The terms used in the diagram above will be defined further in the glossary items below, and will be used to further refine additional terms in the glossary.

6.3 Glossary Server-Specific Terms

This glossary is intended as a reference to the current document, and is not intended to be read sequentially. For that reason, the terms are placed in alphabetical order. This may cause some terms to be used before the import of their definitions are completely clear.

6.3.1 Embedded Firmware Device Driver

A firmware device driver that is part of the firmware image in the System ROM which is a Firmware Environment. Depending on the ROM update strategy, an embedded firmware device driver may or may not reside in the S-CRTM.

6.3.2 Firmware

A specific type of software that is supplied by the hardware component manufacturer for purposes that include, but are not limited to, bootstrapping or management of the hardware component. While components of the firmware may be provided by entities other than the hardware component manufacturer, the function and purpose of firmware is specifically to manage and operate the hardware and it is not intended to be used as a general purpose application environment. Use and other management functions (e.g., updating, loading additional modules, etc.) of this software is considered to be controlled by either the manufacturer or administrator of the hardware.

Much like the terms hardware and software, the term firmware is general. In simple environments it may be sufficient to refer to a computing component as firmware, however, in more complex implementations, firmware is the collection of the firmware environment and the firmware extensions.

Firmware may be software that is loaded and executed by a service processor or may be software that is loaded and executed by the “main” CPU. Firmware is distinguished from microcode.

6.3.3 Firmware Environment

This is a more complex implementation of firmware that may allow static or dynamic loading of software components or modules. While this more complex firmware may contain software to manage the hardware and software resources (e.g., memory management, device abstraction, etc.) and would normally be consistent with the definition of an Operating System, for the purpose of this definition, firmware is not an Operating System.

6.3.4 Firmware Extension

Software provided by any entity that executes along with or under the control of firmware whose purpose includes bootstrapping or management of the hardware component. Firmware extensions are not considered applications. A firmware extension runs under the context of the firmware environment.

> Example: In a PC Client, the BIOS is considered firmware. While Option ROMs are provided by the adapter (hardware component) manufacturer and the BIOS loads and executes the Option ROM, the BIOS is firmware and may be considered a firmware

environment (because it loads Option ROMs), it is not considered an Operating System. Further, the Option ROM is considered a firmware extension, not an application

6.3.5 Firmware Device Driver

Code image that is run on the system processor in the partition pre-boot environment to communicate with and control an I/O device. If the code is not from an immutable source, it must be measured before use. This is a Firmware Extension and is unlikely to be part of the S-CRTM but may be part of the chain of trust.

6.3.6 Locality

A modifier to the signaling of a TPM command that indicates to the TPM the source of the command. Note: this does not modify the TPM command itself, rather, it modifies the signaling mechanism for sending the command to the TPM. It is an indication of the privilege level of a particular resource. Assertion of locality may be different across different implementations.

6.3.7 Partition

A collection of unique hardware resources that is managed by a single instance of either a Physical or Virtual Machine Monitor. The set of hardware components may be defined by the platform manufacturer and unalterable (as in the case of a PC Client) or may be dynamically altered by firmware (as in the case of a reconfigurable server). Partitions are isolated such that the operations on or by one partition do not affect the TCG trust properties of other partitions. E.g., a Partition reset for one partition does not cause any change to the chain of trust (i.e., the TPM) of another partition.

On simpler platforms (e.g., clients) there may be no distinction between partition and platform, therefore a one-to-one relationship between them exists. In this case the term platform rather than partition shall be used.

In environments where partitions are “created”, they may be created by either a service processor, VMM or other mechanism

6.3.8 Partition Pre-boot

The time from instantiation of a partition firmware environment by the platform firmware until control is handed off and the boot environment is exited. On computing systems which do not support multiple partitions, the Partition Pre-boot and Platform Pre-boot environments are the same.

6.3.9 Partition Reset

The beginning of the initialization and configuration of the partition from its initial state. Partition Reset causes control to pass to the Static Core Root of Trust for Measurement for the Partition in the S-RTM model.

6.3.10 Platform

A computing machine that contains the physical resources needed to instantiate and run one or more partitions.

6.3.11 Platform Pre-boot

The time from first instruction fetch that occurs via the reset vector until firmware either instantiates the Partition Pre-boot environment or hands off control and exits the boot environment.

6.3.12 Platform Reset

That action which occurs when the entire platform enters the state that occurs on an initial power-on sequence. The state must be indistinguishable to software or external entities from that which occurs with an actual power-on. A Platform Reset will also cause a Partition Reset for all partitions within the platform. A platform reset terminates execution of all existing partition instances. Platform Reset causes the Platform Pre-boot environment to be entered.

6.3.13 Main CPU (Central Processing Unit) or Processor

The hardware component that executes the operating system and other system software for the purpose of hosting user application and user application instructions. The Main CPU is distinct from a Service Processor (q.v.) While the use appears to be singular, a plural use is implied; i.e., the Main CPU may be composed of one or more CPUs functioning together either for performance, redundancy, or other purposes. This is the computing engine that implements the Instruction Set Architecture.

6.3.14 Service Processor

An environment with distinct hardware and software from the main CPU, typically executing firmware for the purpose of configuring and managing and diagnostics for the platform. The service processor may contain an operating system and may even execute applications, but it does not execute an operating system for the purpose of hosting user applications.

6.3.15 Post-boot

Any time after the OS has been loaded and is running and the platform has exited the boot environment.

6.3.16 Server

A modifier that indicates a resource is used by multiple concurrent users or external client entities. For example, a Server Platform is a platform that is used by multiple concurrent users or external client platforms.

6.3.17 System

In other specifications this term has been used to variously mean Platform or Partition. However because it is an overloaded term, it should not be used in any SWG spec because of its ambiguity.