TRUSTED® COMPUTING GROUP

SPECIFICATION

TCG Server Management Domain Firmware Profile Specification

Family “2.0”
Level 00 Revision 1.00
December 10, 2020

Contact: admin@trustedcomputinggroup.org

DISCLAIMERS, NOTICES, AND LICENSE TERMS

Acknowledgements

The writing of a specification, particularly a security specification, takes many hours for both development and review. The TCG would like to acknowledge the contribution of those individuals (listed below) and the companies who allowed them to volunteer their time to the development of this specification. Special thanks are due to Shiva Dasari, who served as Chair of the Server Working Group during the development of this specification. The TCG would also like to give special thanks to Tim Block who served as the editor of this specification.

Contributors

Bill Jacobs, Cisco Systems

Scott Phuong, Cisco Systems

Shiva Dasari, Hewlett Packard Enterprise

Ronald Aigner, Microsoft

Dick Wilkins, Phoenix

Dean Liberty, Advanced Micro Devices

Travis Gilbert, Dell

Scott Piper, Lenovo

Matt Areno, Intel

Dave Riss, Intel

Tim Block, IBM

Authors of the TCG PC Client Platform Firmware Profile Specification Family “2.0”, which was used as the basis for this document.

1 Contents

List of Figures

List of Tables

TPM Dependency and Requirements

1. A TPM used within the Management Domain claiming adherence to this specification SHALL be compliant with the TPM Library Specification; Family 2.0; Level 00; Revision 01.38 or later and the TCG PC Client Platform TPM Profile Specification for TPM 2.0 Version 1.04 Revision 37 February 3, 2020 or later.
2. TPM structures referenced in this Management Domain specification SHALL be compliant with the TCG PC Client Platform Firmware Profile Specification Family "2.0" Level 00 Revision 1.04 June 3, 2019 or later.

1 SCOPE

This document serves as implementation reference documentation for Management Domain firmware architectures. Specifically, this document defines:

1. Usage of Platform Configuration Registers (PCRs) in the transition through the management firmware stack in a management domain boot flow.
2. How Management Domain firmware, or a component thereof, contributes to the Root of Trust for Measurement (RTM) of the platform, specifically through extending digests (measurements) to a TPM based PCR and documentation of those measurements in a log (event log).
3. Behavior entering, during, and exiting power and initialization states.

1.1 Key Words

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this document normative statements are to be interpreted as described in RFC-2119, Key words for use in RFCs to Indicate Requirement Levels.

1.2 Statement Type

Please note a very important distinction between different sections of text throughout this document. There are two distinctive kinds of text: **informative comment** and **normative statements**. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, it can be considered a kind of normative statements.

EXAMPLE: Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind *informative comment* ...

This is the second paragraph of text of the kind *informative comment* ...

This is the nth paragraph of text of the kind *informative comment* ...

To understand the TCG specification the user must read the specification. (This use of MUST does not require any action).

End of informative comment

2 Introduction and Concepts

Start of informative comment

The Trusted Computing Group's architecture is platform-independent, intended to enhance trust in computing platforms. As such, the TPM Library Specification Family 2.0 is general in specifying both hardware and software requirements. The goal of the TCG member companies is to ensure compatibility among implementations within each type of computing architecture. It is anticipated that companion implementation documents will be created for each architecture.

This specification is based on the TPM Library Specification Family 2.0, Level 00 Revision 1.38. The reader is expected to be familiar with 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 TPM Library Specification will not be defined in this document. If there is a conflict in interpretation between this and the TPM Library Specification, the concept or functional description as defined in the TPM Library Specification takes precedence.

This specification also references other specifications as listed in Section 2.6 External Specifications. The reader is expected to be familiar with the concepts and terminology contained in each where relevant.

It is important to understand that there are two uses for measurements: Attestation and Sealed Storage.

1. Attestation is used to provide information about the platform's state to a challenger. However, PCR contents are difficult to interpret; therefore, attestation is typically more useful when the PCR contents are accompanied by a measurement log. While not trusted on its own, the measurement log contains a richer set of information than the PCR contents. The measurement log is not intended to be used to validate the PCR contents, rather, the PCR contents are used to provide the validation of the measurement log.

2. Sealed Storage uses only the PCR contents to determine its action. Sealed Storage operations make no use of the measurement log and are simpler but provide only a success or fail for the validation of the platform's configuration.

If the only use of PCRs was attestation, there would be little reason to have more than just a few PCRs because they are only used for the validation of the measurement log. Challengers could validate the log using the PCRs, then parse through the measurement log for the information they need. Sealed Storage, on the other hand, is best done when the types of measured objects are grouped together, with objects that have similar impact on the validation of the platform's trust grouped into the same PCR. This grouping was determined when arriving at the PCR mapping in this specification.

End of informative comment

2.1 Management Domain Specific Architecture

Start of informative comment

The concepts and descriptions of the Management Domain architecture are illustrated in Figure 1 and explained in the descriptions. While the diagram in Figure 1 implies physical connections, the connections and associations between components are actually logical.

In this document, the term '*Management Domain*' is used to describe the portion of a managed server that contains a service processor, aka Baseboard Management Controller (BMC), as well as its Roots of Trust, Trusted Building Block (TBB), and associated firmware and components. '*Management Domain*' excludes the Host Domain of the server, which can be defined as the processor, Roots of Trust and associated firmware and peripherals running the server's tenants' workloads.

End of informative comment

Figure 1 depicts the general architectural components of the Management Domain as used in this specification. The components and their relationships within the diagram are meant to provide a reference for discussion and are not meant to require or imply any particular design or implementation beyond what is stated in the normative text in this specification.

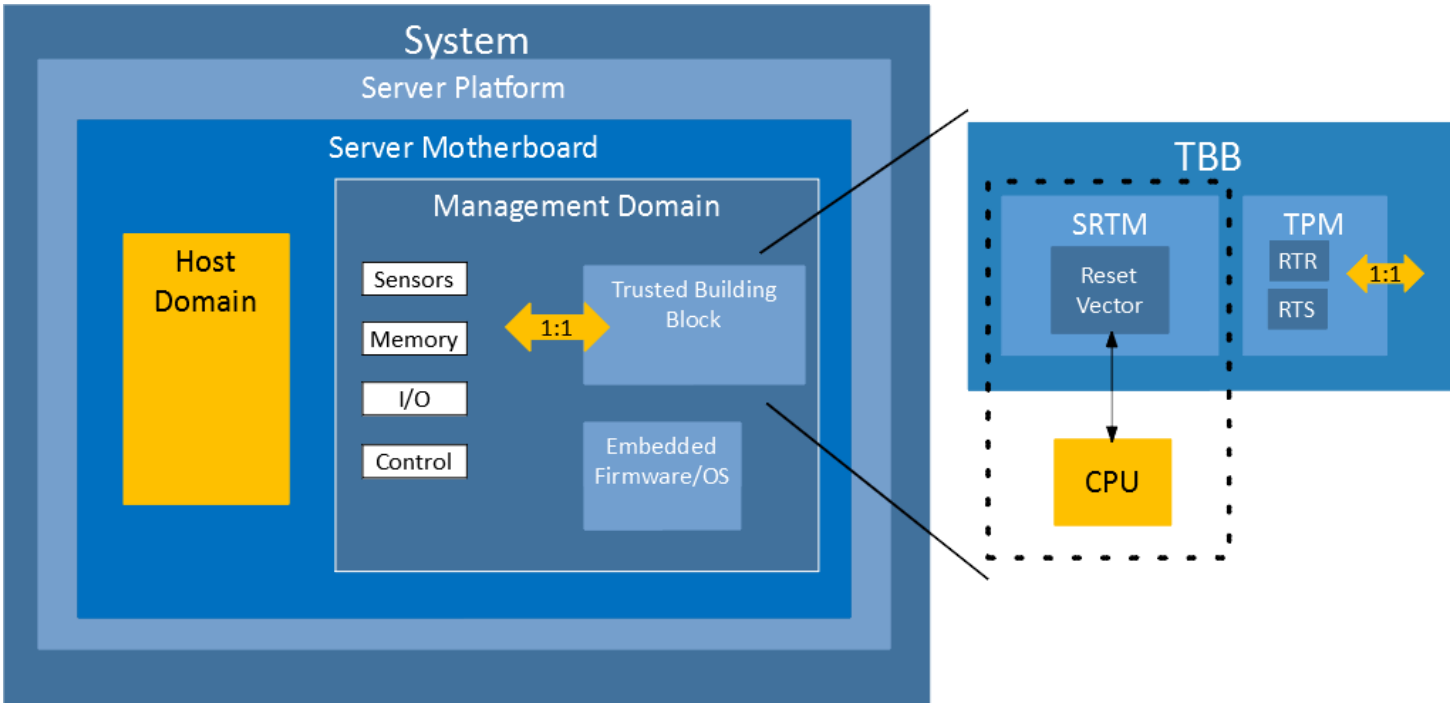


Figure 1 Management Domain Architectural Diagram

2.2 Management Domain Concepts

2.2.1 Management Domain TPM

The term *TPM* within this specification means the TPM attached to the Management Domain. This TPM provides protected capabilities for the Management Domain and complies with the TPM Library Specification Family 2.0. This specification does not address a TPM used for the Host domain or used in any capacity other than described in the text of this paragraph.

2.2.2 Trusted Building Block (TBB)

A TBB consists of hardware and/or software that establishes a Root of Trust for Measurement by providing an integrity measurement and provides connectivity between a Static Root of Trust for Measurement (SRTM), the TPM, the Management Domain, the Management Domain reset, and the optional TPM physical presence signal.

Since the SRTM and the TPM are the only implicitly trusted components of the Management Domain and since indication of physical presence requires a trusted mechanism to be enabled by the Management Domain owner, the indication of physical presence MUST be contained within the TBB.

The TBB provides functionality that permits an entity to trust in measurements that describe the current computing environment of the platform. An entity can assess those measurement results and compare them with reference values that represent a known trusted state of the platform. If there is a match between the measurement results and the expected values, the entity can trust computations within the platform to execute as expected.

The SRTM initiates the measurement of the state of the hardware and software environment in a platform. Three data components are involved in creation and evaluation of an integrity metric. The first component is the method used to gather the data (measurement algorithm). The second component is predicted values of measured data in a platform (measurement storage). The third component is the actual value of measured data in a platform (measurement evaluation). Any integrity challenger needs to know about all of these components in order to make a decision regarding the integrity of the platform.

While a server may have multiple Management Domains, any single Management Domain has a TBB consisting of the SRTM and the TPM. A Management Domain has only one TPM, one SRTM, and one TBB. The TBB has a one-to-one binding with the Management Domain of the server.

Figure 1 depicts the “one-to-one” logical relationships between the TBB, the TPM, and the Management Domain. The SRTM and TPM within the box labeled “TBB” are part of the TBB as well as the connections between the TPM and the Management Domain. **Note**: the input devices, output devices, non-bootstrap related execution cores, remaining portion of Management Domain firmware, and supporting hardware are not part of the TBB.

2.2.3 Roots of Trust

The terms Root of Trust for Measurement (RTM) and Code RTM (CRTM) within this specification refer to those entities that are associated with the Management Domain.

2.2.3.1 Root of Trust for Measurement (RTM)

The RTM is implicitly trusted. Trust in this component may be expressed in an optional Management Domain Platform Certificate. The RTM is the point from which all trust in the measurement process is predicated. The RTM includes a code component (the CRTM), the computing engine to run the code component, and the physical connections between the code and the computing engine.

2.2.3.2 Code Root of Trust for Measurement (CRTM)

The CRTM is the component of the RTM from which the platform begins execution of one of its trusted states. The transitive trust chain is rooted at this point.

2.2.4 Management Domain

The Management Domain is the entity that executes the server management OS, which executes, presents output from, and receives input for the various sensors and capabilities of the managed platform, whether remote or local. The Management Domain includes the Management Domain CPU, Management Domain RTM, and Management Domain TPM. The term “CPU” in this specification refers to the Management Domain CPU unless otherwise stated. If a Platform Credential or Security Target is produced for this Management Domain, it would provide an indication that this is a Management Domain entity.

2.2.5 Non-Management Domains

In the context of this specification, a Non-Management Domain is a self-contained execution environment within the system that is neither the Management Domain nor the Host Domain. Such platforms execute in an environment separate from the Management Domain components. The Management Domain MUST measure, record and report the true state of the Management Domain even in the presence of Non-Management Domain Platforms.

2.2.6 Management Domain and TPM Reset

Start of informative comment

A Management Domain reset is a hardware event that causes execution on the Management Domain's CPU to permanently end its current instruction sequence and begin executing within the CRTM. This means that the CPU loses its entire context and begins execution at its reset vector. A Management Domain reset causes all Management Domain components to revert to their default, reset state.

Several events can cause Management Domain reset, including those in the following non-exclusive list: initial Power-On; activation of a hardware reset line including activation of the __TPM_INIT signal; a new boot session initiated by the OS; or an unrecoverable fault condition detected by the CPU. The Management Domain has a consistent behavior, from a trust perspective, regardless of the cause of reset.

This section references only resets that apply to the Management Domain. Resets for the Host Platform and system are outside the scope of this specification.

Management Domain reset deals with establishing trust in the SRTM, not with other Management Domain components. Since all Management Domain components that are part of the transitive trust chain are measured, the action taken, or lack thereof, by these components to a Management Domain reset has no impact on the validity of the transitive trust chain. There may be an impact on the verifier's trust in the system but that is outside the scope of this specification.

The primary concern when establishing the transitive trust chain is that the reset of the Management Domain's CPU, which causes execution to begin within the SRTM, is "effectively simultaneous" with the Management Domain TPM's reset.

End of informative comment

2.2.6.1 Reset Types

Start of informative comment

A Hardware Management Domain reset occurs when a signal activates the reset signal of all Management Domain components. The signal may be a user-initiated or a software-initiated event triggered by a command to a hardware component that asserts the reset line.

A Cold Boot Management Domain reset occurs when transitioning the Management Domain to a Power-On state from a full Power-Off state in which no firmware or OS state is preserved on the Management Domain. This reset excludes returning from various power states that can occur after the Cold Boot Reset from an OS present state.

A Warm Boot Management Domain Reset occurs when a Management Domain Reset happens without a full Power-Off to Power-On cycle of the Management Domain power.

A __TPM_INIT occurs on a Cold or a Warm Boot. The SRTM issues the TPM2_Startup command, which tells the TPM whether to load saved state or not.

The normative text of this section applies to all types of Management Domain reset described above.

End of informative comment

2.2.6.2 Management Domain Reset

Upon any Management Domain reset, all execution cores within the Management Domain TBB MUST be reset and the TBB core(s) responsible for bootstrapping MUST begin execution within the SRTM.

2.2.6.3 TPM Reset

1. TPM Reset MUST NOT be executed without a Management Domain reset.
2. TPM Reset MUST be executed (i.e. __TPM_INIT is asserted) when the Management Domain is reset.

2.2.7 TPM Control Surface

2.2.7.1 TPM Visibility

Start of informative comment

For simplicity, Management Domain firmware options to hide the TPM from the Management Domain OS are not supported in this specification.

End of informative comment

If the TPM is physically present, the Management Domain firmware SHALL indicate the presence of the TPM device to the Management Domain OS thereby making the TPM visible to the OS. See Section 6 TPM Discoverability for more information.

2.2.7.2 TPM Discoverability

Start of informative comment

Mechanisms to control discoverability of the TPM by a Management Domain are platform manufacturer-specific. An example is a managed server administrative console configuration option to enable or disable the TPM.

End of informative comment

A TPM is discoverable by the management code stack on a Management Domain if it is physically present. A TPM is not discoverable on a Management Domain if it is not physically present on the platform or if the TPM is physically present but system components prevent the platform owner from taking any platform manufacturing defined action to enable the TPM. In the case that the TPM is not discoverable and the TPM interface is accessible, the Management Domain stack MUST disable the endorsement and storage hierarchies.

2.2.7.3 Enabling the TPM

Start of informative comment

The TPM 2.0 device starts up with all of its functionality fully enabled. The Management Domain firmware does not need to take any action, other than sending the TPM2_Startup and TPM2_SelfTest commands to make the TPM device ready for use.

End of informative comment

A TPM is enabled on a Management Domain if it is visible to the Management Domain OS and all available hierarchies are enabled. A TPM is disabled on a Management Domain if it is visible to the Operating System but all available hierarchies are not enabled.

2.2.8 Boot State Transition

The transition between Pre-OS and OS-Present states is denoted by recording of an EV_SEPARATOR event by the Pre-OS code as it exits the Pre-OS state.

2.2.9 Establishing the Chain of Trust

2.2.9.1 Binding Between an Endorsement Key, a TPM, and a Management Domain

The relationship between the Endorsement Key (EK), a TPM, and a Management Domain is described in the TPM Library Specification Family 2.0 Level 00 Revision 1.38, Part 1, Section 9.4.3.3 (RTR Binding to a Platform). A platform compliant with this specification SHALL ensure the TPM is reset at the same time as the Management Domain's TBB CPU.

2.2.9.2 Binding Methods

Start of informative comment

The method of binding the TPM to the Management Domain platform is an architectural and design decision made by the platform manufacturer and is not specified here. The two types of binding are: Physical and Logical. Physical binding relies on hardware techniques, while Logical binding relies on cryptographic techniques. The requirements for the strength of each binding are within the scope of a Protection Profile.

Example:

The TPM is a physical chip soldered to the Management Domain platform. Here the Endorsement Key is physically bound to the TPM (the EK is inside the TPM) and the TPM is physically bound to the Management Domain Platform by the solder connections.

End of informative comment

2.2.10 Locality

Start of informative comment

Hardware Proof of the source of a command

The TPM supports several authorization types to provide a trusted path between a user and a TPM's object (e.g. a TPM key or NV area). Common authorization methods are password, secret key, and public key. These methods rely on shared or provisioned secrets. Some technologies, however, require a trusted path between hardware components. Methods involving shared or provisioned secrets are not practical for these hardware components. Locality uses hardware addressing to enforce a trusted path between hardware components. By restricting the addresses at which some hardware components can address the TPM, locality provides proof of the source component's identity. This allows TPM object policies to apply to specific hardware components enforced by the hardware.

Note: the enforcement agent for the trusted path (i.e., the authorization of the source of the command) is not in the TPM, rather it is the hardware controlling the channel to the TPM. The TPM will take commands sent to it at any locality and act on those commands, giving that command access to that locality's objects.

The TPM adopted by this specification, that of the PC Client, supports five Localities numbered 0-4. However, in the context of this specification, the Management Domain firmware will only support Locality 0.

Note: the above description applies to hardware localities. There are also software localities used, for example, in virtualization, which are not associated with hardware addresses. These software localities are outside the scope of the Management Domain TPM discussed here.

Sharing of the TPM

In the Management Domain, no sharing of the TPM is anticipated.

End of informative comment

2.2.11 System and TPM Power States

Start of informative comment

For PCs, a power management standard called ACPI defines a set of power states that each have different performance and power requirements. Management Domains that comply with this specification do not support the same concept of generalized power states as defined by ACPI. Further the Management Domain power states are generally limited to OFF and ON.

End of informative comment

The TPM power states SHALL be OFF and ON.

2.3 Overview of the Measurement Process

Start of informative comment

This specification defines measurement as the process of hashing various components of the boot process, extending the results in the TPM and logging the component's measurement in the measurement log in the Management Domain's memory. The specific components, order of measurements, and storage locations are specified in normative text in subsequent sections.

As a TPM in a Management Domain is usually a discrete chip that runs at lower clock speeds than the CPU and is connected to the platform via a relatively slow bus, interaction with the TPM can be a platform bottleneck during the boot process. Thus, this specification recommends minimizing the amount of data sent to the TPM for measurement and, once the initial measurement is made, making use of the Management Domain's CPU to perform hash computations of large amounts of data.

End of informative comment

2.3.1 Usage and Optimization of Hash Functions

1. The Management Domain firmware MAY use the TPM's Hash interface command but SHOULD do so for as little data and as few times as possible.
2. The Management Domain firmware SHOULD use its own Management Domain CPU-based hashing function as early as possible in the boot cycle and SHOULD continue to use it for the remainder of the boot cycle.

2.4 Terminology

The following terms are used as defined below throughout the document. All other terms are defined in other TCG documents; e.g. TCG Glossary.

Start of informative comment

TPM Device Reset: the assertion of the __TPM_INIT hardware signal.

Management Domain Software: the source of TPM commands, which may be an operating system driver or an application within the Management Domain.

Management Domain Hardware: platform components including chipsets and associated microcode, and microprocessors and associated microcode, if any.

Platform: the management platform comprising the Management Domain hardware and Management Domain firmware and kernel on which any additional Management Domain applications can be run.

SRTM: code supplied by the platform manufacturer, as a subset of Management Domain firmware, that initializes and configures platform components and is the portion of Management Domain firmware that defines the initial trust boundary.

SRTM Environment: The Trusted Building Block consisting of the SRTM and the TPM in their initialized states following a Management Domain reset.

Read: a transaction where the calling entity requests and receives data from a specified register or buffer in the TPM.

Write: a transaction where the calling entity sends data to a register or buffer in the TPM.

NUL: indicates the ASCII null character, "\0". Used in this specification to indicate a null character.

End of informative comment

2.5 TCG Specification Dependency and Naming

2.5.1 Division of Documentation

Start of informative comment

The Management Domain Specifications are divided into two documents:

1. The TCG *PC Client Platform TPM Profile* (PTP) Specification (Family "2.0") discusses specific features of the physical TPM device used by PC Client platforms. This Management Domain Specification adopts the same physical TPM for reasons of TPM device manufacturer efficiency but utilizes a subset of the capabilities identified for the PC Client platforms. The PTP discusses the details of what interfaces and protocols are used to communicate with the TPM and the platform-specific set of requirements. The PTP includes the definitions of

the items identified in the TPM Library specification as “Platform Specific” such as the minimum number of PCRs required and NV Storage available. The target audience for the PTP is the TPM manufacturers but platform manufacturers should review it as well. **Note:** As there is no ‘server specific’ TPM 2.0 on the market, this specification relies upon the PTP for PC Client.

2. This specification, the TCG *Server Management Domain Firmware Profile,* specifies the requirements for the TPM as it is implemented on the Management Domain. Issues such as TPM, platform and firmware provisioning, usage of TPM to record measurements of platform code, PCR mapping, functional interfaces, and interfaces are discussed. The target audience for this document is managed server platform manufacturers.

The following additional TCG Specifications are referenced in this specification.

End of informative comment

2.5.2 TPM Library Specification

Start of informative comment

In this document, “TPM Library Specification” refers to the TCG TPM Library Specification, Family “2.0” as released. The TPM Library specification has four parts. Unless a specific part is referenced, use of the term refers to all parts of the TPM Library specification

End of informative comment

2.5.3 TCG Physical Presence Specification

Start of informative comment

This refers to the TCG PC Client Platform Physical Presence Interface Specification, Family “1.2” and “2.0”, Version 1.30, Revision 00.52.

End of informative comment

2.5.4 TCG Platform Reset Attack Mitigation Specification

Start of informative comment

This refers to the TCG PC Client Platform Reset Attack Mitigation Specification, Family “2.0” Version 1.10.

End of informative comment

2.6 External Specifications

Start of informative comment

This section lists references to external (non-TCG) specifications.

U.S. National Institute of Standards and Technology (NIST) Special Publication 800-147B BIOS Protection Guidelines available at: https://csrc.nist.gov/publications/detail/sp/800-147b/final

End of informative comment

2.7 Specification Conventions

Start of informative comment

TPM data and structures are Big Endian. All constants and data created and used by the Management Domain’s structures are presented in Little Endian format.

The justification for this is that when software deals with the Management Domain itself (e.g., the Management Domain data, structures, etc.), it uses Little Endian—always. Software already deals with this bifurcation when communicating over a network. When getting packets from the network (e.g., FTP, HTTP, etc.) it deals with Big Endian, or “Network Order”; when it deals with data and structures from the Host Platform itself, it does so using Little Endian. Changing endianness within the context would be inconsistent.

End of informative comment

1. All constants and data are represented as Little Endian format unless otherwise explicitly stated.
2. All strings are represented as an array of ASCII bytes with the left-most character placed in the lowest memory location.
3. ASCII strings, unless otherwise indicated, are not NUL terminated.
4. Hexadecimal numbers are designated by ‘0x’ preceding the number or ‘h’ following the number. Decimal numbers do not have the preceding 0x.
5. In some memory layout descriptions, certain fields are marked reserved. Firmware MUST initialize such fields to zero, and ignore them when read.

3 Management Domain Roots of Trust Requirements

3.1 Locality Support Requirements

3.1.1 Pre-OS Environment

Start of informative comment

Per section 2.2.10 Locality, Management Domain firmware uses Locality 0 to access the TPM. The TPM can be set to only one locality at a time.

End of informative comment

Before sending a command to the TPM, the software or other platform components MUST set the TPM to operate at Locality 0 (typically the default TPM locality setting).

3.2 Static Root of Trust for Measurement (SRTM)

3.2.1 Introduction of Concepts

Start of informative comment

The SRTM environment provides the Root of Trust for the components of the Management Domain when they are initialized following a Management Domain reset. The goal of the SRTM is to anchor an unbroken chain of measurements for components that executed on the platform or security-relevant configuration data that may have influenced platform state. By assessing the chain of components, an entity may establish trust in the current state of the platform. The SRTM is the RTM for all components because the SRTM measurements occur during boot by default. The SRTM uses Locality 0 and the memory addresses associated with Locality 0.

End of informative comment

3.2.2 Initial TBB Control and Management Domain Reset

Management Domain reset MUST cause the Management Domain to begin execution within the SRTM.

3.2.3 Static Root of Trust for Measurement (SRTM)

The Static Root of Trust for Measurement (SRTM) MUST be that portion of the Management Domain's initialization code that serves as a Root of Trust for Measurement with the initial integrity measurement occurring as a result of Management Domain reset. See Section 2.2.2 Trusted Building Block (TBB).

Start of informative comment

Note: The trust in the SRTM environment is based on the SRTM. The trust in all SRTM measurements is based on the integrity of the SRTM component. The SRTM is static because the PCRs associated with it cannot be re-initialized without a Management Domain reset.

End of informative comment

3.2.4 Transfer of Control from SRTM

Prior to transferring control to another entity within the SRTM environment, an executing entity MUST measure the entity to which control will be transferred.

3.3 Integrity Collection and Reporting

Start of informative comment

The Static PCRs, those that cannot be reset without a __TPM_INIT, are divided into two primary sets. The first set is designated for the Management Domain's Pre-OS, or firmware, environment (PCR[0-7]) and the other designated for the Management Domain's OS-Present (PCR[8-15]). The Pre-OS PCRs provide the Management Domain's initial chain of trust starting from Management Domain reset. These establish a chain of trust from the SRTM through to the OS's loader.

The property of the Extend operation makes the set of the integrity measurements taken into each PCR order-dependent. If two measurements, A and B, are extended into a single PCR, the PCR's resulting content will be different if the Extends are performed A then B vs. B then A. For this reason, the order in which the measurements are extended is important because Management Domain applications may "seal" data to the Pre-OS PCRs. A seal operation only functions properly with strict adherence to the measurement sequence.

The Management Domain will utilize a single event log, common for both the Pre-OS and OS-Present environments, with the Pre-OS code passing event log control to the OS-Present code as OS-Present code begins execution. See the figures in Section 3.3.3 Boot Event and PCR Usage Model for Pre-OS and OS Present environments.

End of informative comment

3.3.1 Collection and Reporting of Measurements

1. Platform Firmware MUST perform integrity measurements of the Management Domain's Pre-OS environment per this specification.
2. Integrity measurements made by the Management Domain firmware that are extended MUST be extended into all allocated banks for a PCR. See Section 9 Event Logging.
3. Management Domain firmware MUST log all measurements made to the TPM in the TCG Event Log. See Section 9 Event Logging.
4. Integrity measurements made by the platform manufacturer for PCR[6] MAY occur at any time (Pre-OS through Post-Boot). See Section 3.3.4.7 PCR[6] – Management Domain Manufacturer Specific Measurements.
5. Independent of TPM state, Management Domain firmware MUST measure EV_SEPARATOR as defined in section 9.4.1 Event Types.

Start of informative comment

Note: EV_SEPARATOR delineates the point in the Management Domain boot where Pre-OS Firmware relinquishes control of the TPM measurement process. Measuring and extending EV_SEPARATOR enables an observer to know where Pre-OS control ended.

End of informative comment

3.3.2 Error Conditions

Start of informative comment

There are multiple conditions where a TPM may respond with an error code to a TPM2_Startup command. Some of these conditions include failure of the TPM or its code on its power on initialization test, presentation by Management Domain firmware of an incorrect Startup Type, or the TPM2_Startup command originating from an incorrect locality. Unlike TPM 1.2, there are multiple ways to handle these errors.

If the SRTM is unable to successfully initialize the TPM using the TPM2_Startup command, it is possible that later code could successfully issue the command and record false integrity measurements. To prevent this, the SRTM takes steps to prevent use of the TPM or the Management Domain if it is unable to successfully initialize the TPM. A special case is when the attempt to initialize the TPM results in the TPM being in failure mode; later components will be unable to record false integrity measurements because for most commands the TPM will continue to return TPM_RC_FAILURE.

Figure 2 shows the remediation steps the SRTM takes when initialization of the TPM fails during normal boot.

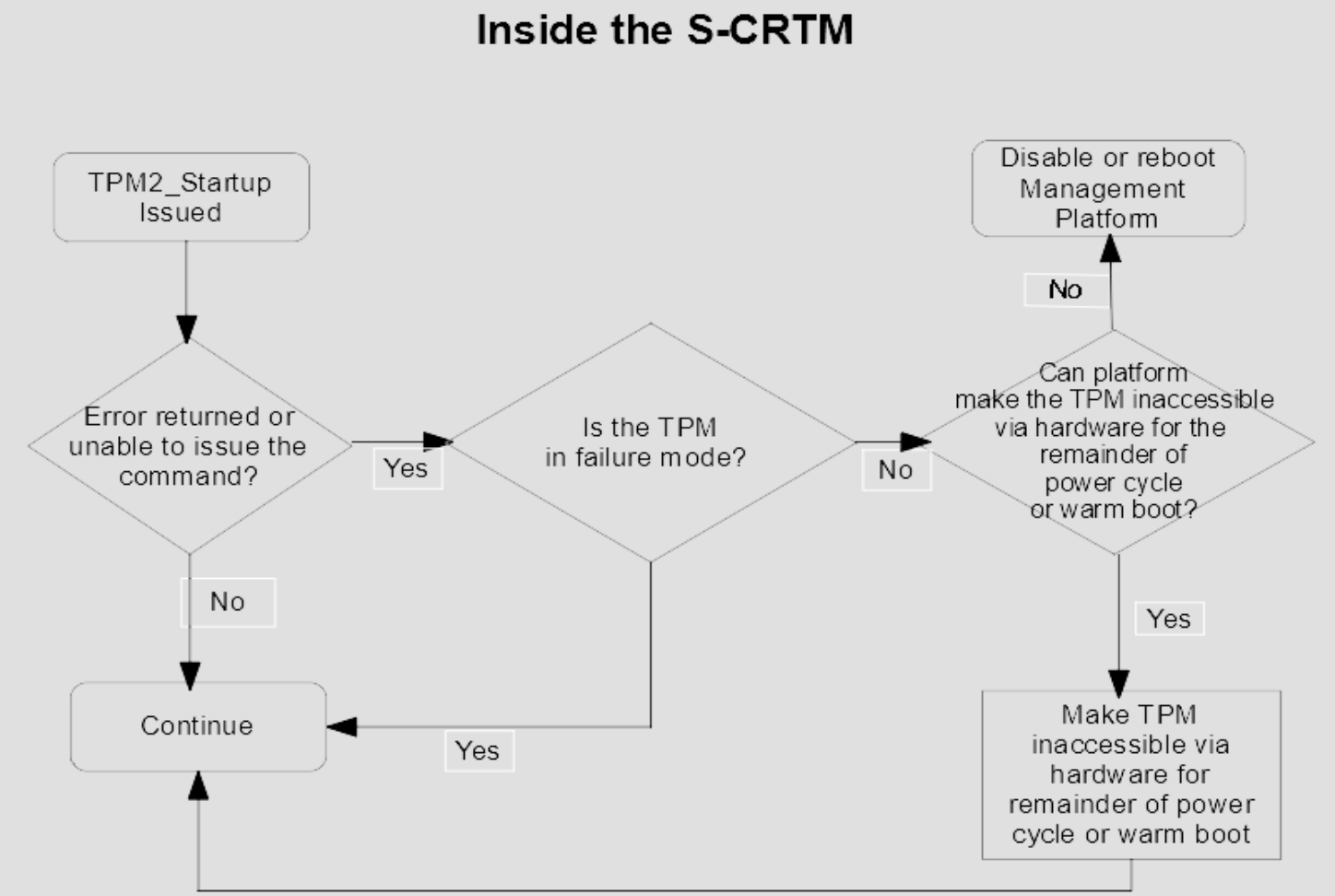


Figure 2 SRTM remediation steps when initializing the TPM

End of informative comment

3.3.2.1 Errors during TPM Initialization

1. If the TPM responds to a TPM2_Startup command with TPM_RC_INITIALIZE, Management Domain firmware MAY ignore the return code and continue to boot.
2. If the TPM responds to a TPM2_Startup command with TPM_RC_FAILURE, Management Domain firmware SHALL:
    - Reboot the Management Domain CPU and perform a TPM2_Startup (CLEAR), or
    - With the exception of sending the TPM2_HierarchyControl command, perform the actions defined in Section 6.1 for disabling the TPM.
3. If the TPM responds to a TPM2_Startup command with TPM_RC_VALUE or TPM_RC_LOCALITY, the Management Domain firmware SHALL:
    - Reboot the Management Domain CPU, perform a TPM2_Startup (CLEAR) and make all required measurements.

4. If the TPM responds to a TPM2_Startup command with any of the following error codes, TPM_RC_COMMAND_CODE, TPM_RC_UPGRADE, TPM_RC_BAD_TAG or TPM_RC_REBOOT, the TPM has attempted to perform a firmware field upgrade and encountered an error, e.g. power was removed during the upgrade. Management Domain firmware SHALL:
    - Restart the field upgrade process, OR
    - Notify the user and, with the exception of sending the TPM2_HierarchyControl command, perform the actions in Section 6.1 for disabling the TPM.

3.3.2.2 Errors recording Measurements

1. If a measurement of the SRTM or subsequent Management Domain firmware cannot be made due to a failure of the Management Domain's Firmware, the PCRs MUST be capped by extending the digest of the value 00000001h to each PCR[0-7].
2. The SRTM SHOULD log the error event to the event log.
3. An EV_SEPARATOR event per Section 9.4.1 Event Types SHOULD be recorded in the event log.
4. If each PCR[0-7] cannot be capped, the Management Domain SHOULD take any necessary action to notify the Management Domain's administrator or operator.
5. The Management Domain MUST transition into a "fail-safe" mode by performing one of these actions:
    a. Make the TPM interface inaccessible via hardware for the remainder of the power cycle or warm boot;
    b. Reboot the Management Domain CPU;
    c. Disable the Management Domain;
    d. Perform a vendor-specific action that is equivalent to one of the options above.

3.3.3 Boot Event and PCR Usage Model

Start of informative comment

The purpose of this section is to provide the model for PCR usage and for adding events to the Event Log. This entire section is an Informative comment, including Figure 3 and Figure 4.

- Figure 3 is an architectural drawing that shows the principal firmware components and their relationship to the platform hardware and the OS software.
- Figure 4 maps, in general, the behavioral components on a Management Domain to the PCRs into which each type of behavioral component is measured.

Together, these two diagrams form the boot event and PCR usage model upon which the requirements in subsequent sections of this specification are based.

Figure 3 illustrates the relationships of various components of a Management Domain that accomplish a fully booted Management Domain firmware.

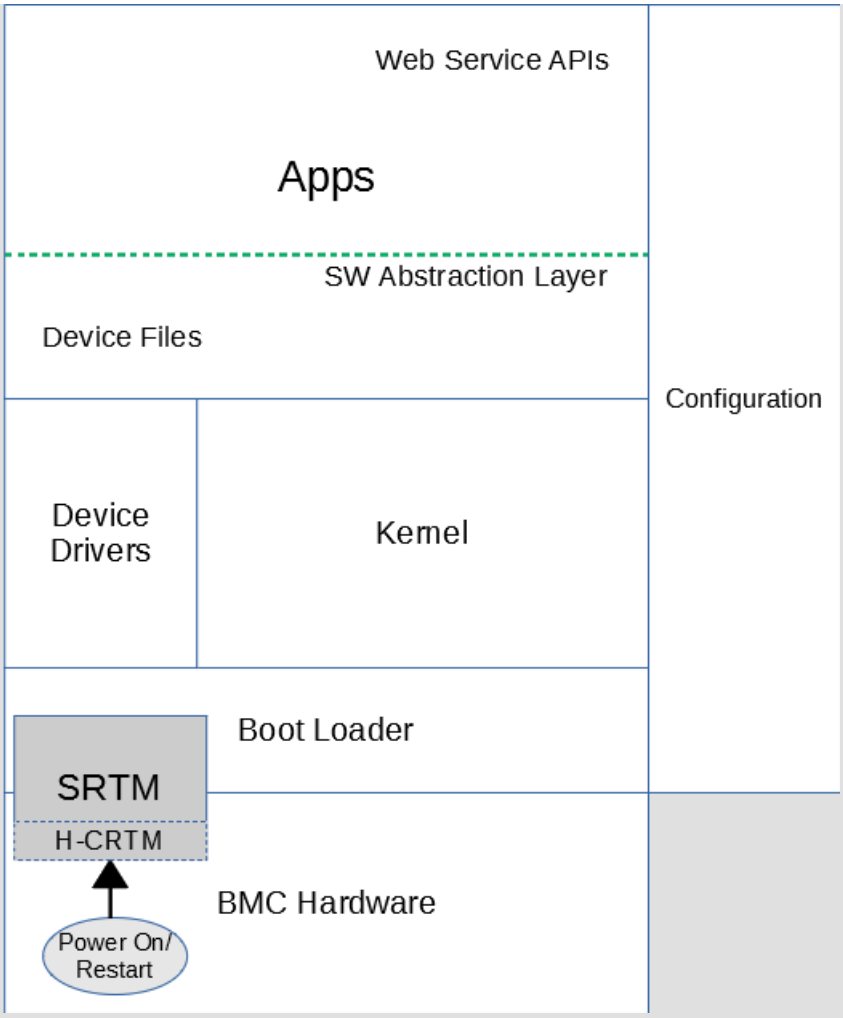


Figure 3 Management Domain Loader Architecture

Figure 4 illustrates the general scheme for PCR usage on a Management Domain.

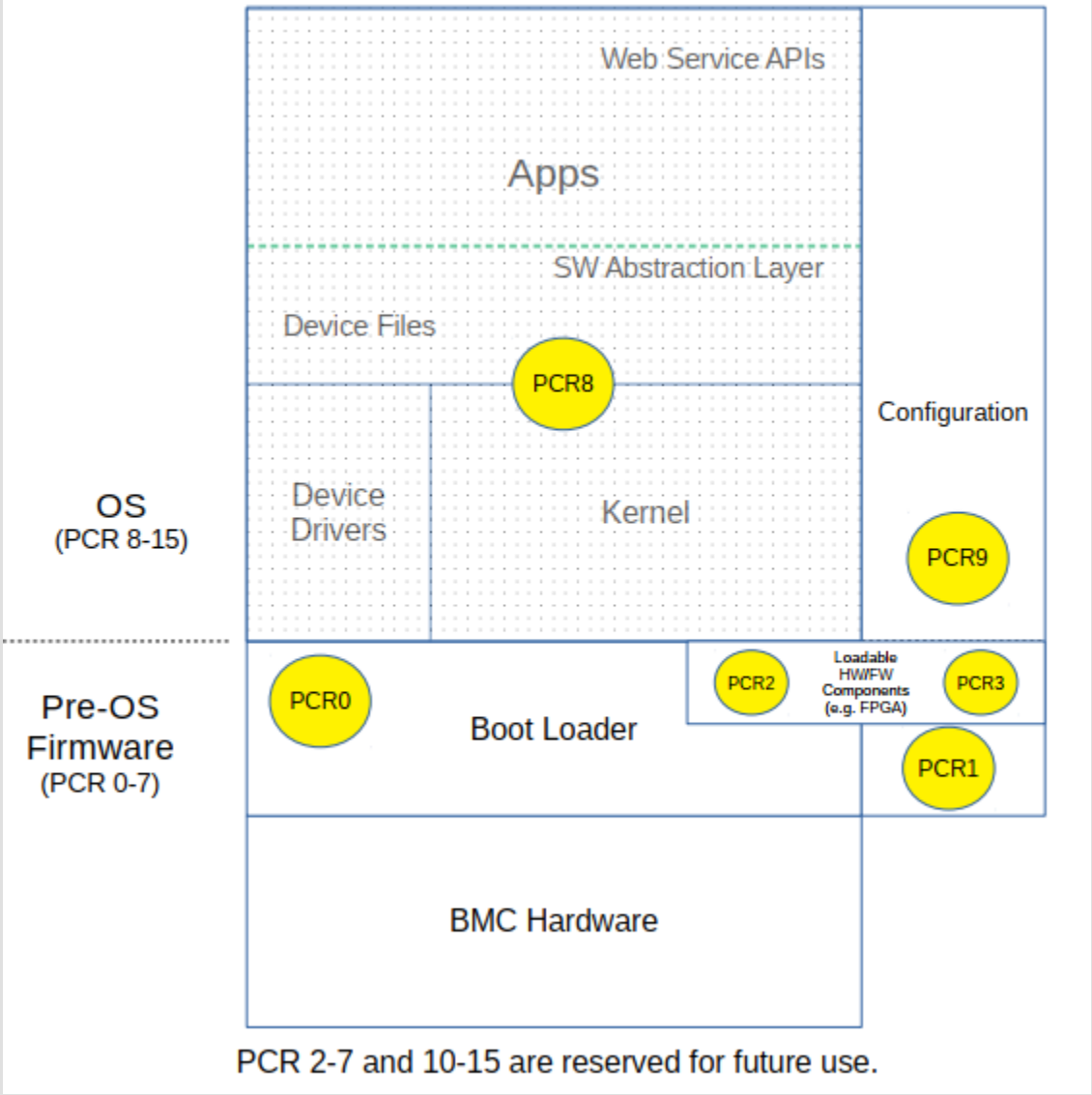


Figure 4 PCR Mapping of Management Domain Components

End of informative comment

3.3.4 PCR Usage

Start of informative comment

PCR[0-15] represent the Management Domain's static RTM and are associated with Locality 0. Therefore, all the PCRs associated with Locality 0 are resettable only upon Management Domain reset.

This section defines the PCR assignments used for boot time integrity metrics and the methodology for collecting the metrics. The first seven PCRs are defined for use within the Pre-OS environment. Throughout the platform boot process, a log of all executable code and relevant configuration data is created and extended into PCRs as described below.

Each time a PCR is extended, a log entry is made in the TCG event log. This allows a challenger to see how the final PCR digests were built.

While events extended to PCR[0-7] may be measured in any order, the selected order needs to be maintained across boot cycles.

End of informative comment

Table 1 PCR Usage

PCR Index	PCR Usage
0	SRTM & Boot Loader
1	Management Domain Configuration
2	Loadable devices
3	Loadable devices configuration
4	Reserved for future use
5	Reserved for future use
6	Vendor specific events
7	Reserved for future use
8	Defined for use by the Management Domain OS
9	Management Domain OS Configuration
10-15	Reserved for future use
16	Debug
17-23	Reserved for future use

Management Domain firmware compliant with this specification SHALL measure all normative items using a tagged Hash structure for each PCR bank enabled in the platform's TPM. See Section 9.2 TCG Defined Structures.

3.3.4.1 PCR[0] – SRTM

Start of informative comment

The SRTM may measure itself or may be measured by an H-CRTM event extended to PCR[0]. The SRTM must measure to PCR[0] any portion of the Boot Loader provided as part of the Management Domain. For example, a Primary Program Loader in ROM code could measure a first stage u-boot Secondary Program Loader (SPL) which could then measure u-boot as the second stage bootloader. Primarily executable code, stable version identifiers, and configuration data are measured.

All these components are under the control of the manufacturer or its agent.

If for any reason a measurement cannot be made to PCR[0], none of the other SRTM PCR values can be trusted and therefore are outside the chain of trust. It is therefore necessary to invalidate all Management Domain SRTM PCRs as described in Section 3.3.2 Error Conditions.

Because the measurement of early Management Domain firmware is typically done within a resource-constrained environment (for example, the SRTM) the event log corresponding to the Extend event may not be created at the time the PCR Extend is performed. It is acceptable to have the Management Domain firmware generate the corresponding event log entry, reconstructing it from information (for example, the value that was extended) passed to it from earlier code. If this procedure is used, the Management Domain firmware must be sure the entries within the event log are properly sequenced (for example, represent the same order as the sequence of TPM2_PCR_Extend operations).

Measurement of Non-Management Domains

Usage of the Event Type EV_POST_CODE

The intent of the event type EV_POST_CODE is to record measurements of components of the Management Domain's transitive trust chain. The imperative is to avoid omitting any portion of the transitive trust chain. If necessary, the event should be used as a catch-all for trust chain components provided by the platform manufacturer, even if the components are not technically Management Domain boot code for the manufacturer's specific implementation.

The normative text below recommends values for the event field for the event types defined in this specification. The reason for the recommendations is to promote consistency across implementations by different platform manufacturers. Per the discretion of the platform manufacturer, different values may be used that are relevant for their implementation.

Design Consideration and Distinctions Between PCR[0], PCR[2], and PCR[4]

PCR[0] typically represents a consistent view of the Management Domain between boot cycles regardless of user-selected options. This allows Attestation and Sealed Storage policies to be defined as those using the less changeable platform manufacturer-provided components of the transitive trust chain. This PCR contains the measurement of the code of the components provided by the Management Domain manufacturer. Therefore, the verifier or the entity performing the seal operation can choose to seal to only those components provided and updated by the Management Domain's manufacturer. The consistent boot cycle view of PCR[0] is also the reason embedded driver binaries are measured into PCR[0].

End of informative comment

Entity that MUST be logged in the Event Log if the TPM is enabled:

The Event Type EV_NO_ACTION Specification ID Event. **Note**: This event is not extended.

Entities that MUST be measured if the TPM is enabled:

1. The SRTM's version identifier, using the event type EV_S_CRTM_VERSION. See Section 9.4.1 Event Types.
2. A variety of entities listed below are measured using the event type EV_POST_CODE. This information MAY be measured as a single combined event or MAY be measured as multiple separate events. The event(s) MUST be recorded using the event type EV_POST_CODE. The entities measured include but are not limited to the following:
    - All Management Domain firmware that is executed by the Management Domain's CPU(s) and is part of the Management Domain's transitive trust chain.
3. Per Section 3.3.2 Error Conditions, if an error occurs, the digest of the value 00000001h MUST be extended in PCR[0-7] and an EV_SEPARATOR event SHOULD be recorded in the event log for each PCR. See Section 9.4.1 Event Types.
4. Per Section 7.1.1 Measuring OS Boot Events, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit. See Section 9.4.1 Event Types.

Entities that MAY be measured if the TPM is enabled:

1. The SRTM itself using event type EV_S_CRTM_CONTENTS. See Section 9.4.1 Event Types.
2. CPU microcode updates as part of a EV_POST_CODE event.

Entity that MUST be measured if the TPM is disabled:

Per Section 3.3.1, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and the matching EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit.

Method of measurement:

The SRTM performs PCR[0] measurements in the following order unless otherwise specified:

1. Measure the SRTM's version identifier.
2. Measure the code to which the S-CRTM is transferring control.
3. The remaining measurements MAY be performed in any order, except for the recording of an EV_SEPARATOR event by the Pre-OS code on exit, which MUST be last.
4. The Specification Event is not measured and it MAY be created at any time, but it MUST appear first in the event log. See Section 9.4.4.1 Specification ID Version Event. **Note**: This event is not extended.

Management Domain firmware may need to reconstruct events that could not be recorded in the event log due to the unavailability of memory. If it does so, it places this information into the event log and MUST NOT extend PCR[0] again with this reconstructed information, e.g. the Specification Version event.

3.3.4.2 PCR[1] – Management Domain Configuration

Start of informative comment

Information about the configuration of the Management Domain including hardware components and how they are configured is measured to PCR[1]. In general, the Management Domain firmware measures into PCR[1] the configuration data that is associated with the code that has been measured into PCR[0].

End of informative comment

Entities that MUST be measured if the TPM is enabled:

The following entities MUST always be measured. The platform manufacturer MUST NOT provide firmware configuration options that permit disabling these measurements:

1. If Platform Firmware loads a CPU microcode update, the microcode update MUST be measured in PCR[1] using event type EV_CPU_MICROCODE if the microcode update was not already measured in PCR[0] as part of a EV_POST_CODE event. See Section 9.4.1 Event Types.
2. If the Management Domain supports selection of optional PCR[1] measurements, the Management Domain firmware MUST measure which PCR[1] measurements are currently on or off using the event type EV_PLATFORM_CONFIG_FLAGS. The event data MUST not vary across boot cycles if the set of potential PCR[1] measurements measured does not vary. See the informative text for this section and Section 9.4.1 Event Types.
3. Per Section 3.3.2 Error Conditions, if an error occurs, the digest of the value 00000001h MUST be extended in PCR[0-7] and an EV_SEPARATOR event SHOULD be recorded in the event log for each PCR. See Section 9.4.1 Event Types.
4. Per Section 7.1.1 Measuring OS Boot Events, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit. See Section 9.4.1 Event Types.

Entities that MAY be measured if the TPM is enabled:

1. The following entities MAY be measured. Even if measurement of these is provided by Management Domain firmware, Management Domain firmware MAY allow the owner to disable individual measurements or all of the measurements.
   - Table of devices, using event type EV_TABLE_OF_DEVICES described in Section 9.4.1 Event Types.
   - This Event Type allows Management Domain firmware to measure devices attached to the Management Domain. Examples include PCI devices, onboard video adapters, etc. Because of the wide variance of Management Domain architectures, the actual format of the data providing this information is left to the Management Domain Manufacturer. It is left to the challenger to discover the format of this data. The challenger could, for example, reference the Management Domain Certificate, and then contact the Management Domain Manufacturer to obtain this information. Management Domain firmware MAY create multiple entries of this Event Type or MAY choose to encapsulate all the data into a single entry.
2. Security-relevant Management Domain NVRAM data SHOULD be measured.
3. Platform NVRAM data that contains security-relevant user configuration setup options in effect when the platform boots SHOULD be measured.
4. If Management Domain firmware has detected the platform case was opened or an analogous action occurred, the platform MAY record the EV_ACTION event "Chassis Intrusion". The event MAY be persistently recorded across Management Domain boots until the Management Domain firmware administrator clears the notification. See Section 9.4.1 Event Types.

Entity that MUST be measured if the TPM is disabled:

Per Section 3.3.1, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit.

Entities that MUST NOT be measured as part of the above measurements:

1. Values and registers that are automatically updated (e.g., clocks).
2. System-unique information such as asset, serial numbers, etc. This is to provide privacy or to enable keys to be sealed to a well-known platform configuration recorded in PCR1, irrespective of unique platform characteristics.
3. Sensitive data like passwords MUST be omitted from the NVRAM data because the data is placed unencrypted in the TCG event log.

Method of measurement:

Management Domain firmware performs these measurements as follows:

- The entities to be recorded in PCR[1] MAY be measured in any order deemed appropriate by the implementer, except for the recording of an EV_SEPARATOR event by the Pre-OS code on exit, which MUST be last.

3.3.4.3 PCR[2] - Loadable Devices

Start of informative comment

Loadable devices (devices which contain loadable firmware such as FPGAs) contained in the Management Domain are measured by Management Domain firmware to PCR[2]. It is notable that not all loadable devices can be measured by the Management Domain firmware, such as devices whose firmware contents are embedded within the device itself. While the device firmware is upgradeable, it may not be accessible to be measured. Such devices include CPLD, certain and even more advanced FPGAs such as secure FPGAs. Those

devices may be handled with vendor specific interchanges and corresponding configuration information being measured into PCR[3].

End of informative comment

Entities that MUST be measured if the TPM is enabled:

1. Hardware components with updateable code loaded (updated) by Management Domain firmware MUST be measured using event type EV_EVENT_TAG. See Section 9.4.1 Event Types.
2. Per Section 3.3.2 Error Conditions, if an error occurs, the digest of the value 00000001h MUST be extended in PCR[0-7] and an EV_SEPARATOR event SHOULD be recorded in the event log for each PCR. See Section 9.4.1 Event Types.
3. Per Section 7.1.1 Measuring OS Boot Events, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit. See Section 9.4.1 Event Types.

Entity that MUST be measured if the TPM is disabled:

Per Section 3.3.1, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit.

Method of measurement:

1. Management Domain firmware MUST measure the event EV_EVENT_TAG for each hardware device with updateable code prior to initializing the hardware component.
2. The visible portion of the hardware device code measured in PCR[2] by Management Domain firmware MAY be measured in any order deemed appropriate by the implementer. However, the order MUST be consistent across boot cycles. Management Domain firmware may measure all hardware device code and then initialize them all or MAY measure one and then initialize it before initializing the second, or MAY use some other variation per the discretion of the manufacturer.
3. Repeat steps 1 and 2 until all hardware device updateable code is measured and executed.
4. The hardware devices MAY measure their “hidden” code (if applicable), that is, any code not directly byte readable by the service processor.
5. If a hardware device measures “hidden” code, it MUST measure before executing it.
6. The recording of an EV_SEPARATOR event by the Pre-OS code on exit MUST be measured last.

3.3.4.4 PCR[3] - Loadable Devices Configuration

Start of informative comment

This section contains the PCR[3] measurement requirements for Management Domain loadable devices configuration.

In general, entities measure into PCR[3] data that is associated with code modules that are measured into PCR[2]. For example, on a Management Domain platform, this includes but is not limited to Management Domain firmware updateable code residing in flash based FPGAs. As noted earlier, not all loadable devices can be measured by the Management Domain firmware, such as devices whose firmware contents are embedded within the device itself. Those devices may be handled with vendor specific interchanges and corresponding configuration information being measured into PCR[3].

End of informative comment

Any pre-OS component that modifies a loadable device configuration MUST measure the new configuration into PCR[3] or cause a Management Domain Reset.

Entities that MUST be measured if the TPM is enabled:

1. If any configuration data exists for a loadable device, the loadable device MUST measure the configuration data specific to the device using event EV_EVENT_TAG. See Section 9.4.1 Event Types.
2. Per Section 3.3.2 Error Conditions, if an error occurs, the digest of the value 00000001h MUST be extended in PCR[0-7] and an EV_SEPARATOR event SHOULD be recorded in the event log for each PCR. See Section 9.4.1 Event Types.
3. Per Section 7.1.1 Measuring OS Boot Events, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit. See Section 9.4.1 Event Types.

Entity that MUST be measured if the TPM is disabled:

Per Section 3.3.1, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit.

Entities that MUST NOT be measured as part of the above measurements:

1. Values and registers that are automatically updated (e.g., clocks).
2. System-unique information such as asset, serial numbers, etc. This is to provide privacy.

Method of measurement:

Management Domain firmware performs these measurements in the following order unless otherwise specified:

1. Measure the event EV_EVENT_TAG. See Section 9.4.1 Event Types.
2. The remaining measurements MAY be performed in any order, except for the recording of an EV_SEPARATOR event by the Pre-OS code on exit, which MUST be last.

3.3.4.5 PCR[4] - Reserved For Future Use

PCR[4] is reserved for future use but is the recipient of error condition and separator events as specified below.

Entities that MUST be measured if the TPM is enabled:

1. Per Section 3.3.2 Error Conditions, if an error occurs, the digest of the value 00000001h MUST be extended in PCR[0-7] and an EV_SEPARATOR event SHOULD be recorded in the event log for each PCR. See Section 9.4.1 Event Types.
2. Per Section 7.1.1 Measuring OS Boot Events, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit. See Section 9.4.1 Event Types.

3.3.4.6 PCR[5] - Reserved For Future Use

PCR[5] is reserved for future use but is the recipient of error condition and separator events as specified below.

Entities that MUST be measured if the TPM is enabled:

1. Per Section 3.3.2 Error Conditions, if an error occurs, the digest of the value 00000001h MUST be extended in PCR[0-7] and an EV_SEPARATOR event SHOULD be recorded in the event log for each PCR. See Section 9.4.1 Event Types.
2. Per Section 7.1.1 Measuring OS Boot Events, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit. See Section 9.4.1 Event Types.

3.3.4.7 PCR[6] – Management Domain Manufacturer Specific Measurements

Start of informative comment

This PCR is reserved for use by the Management Domain manufacturer. This allows for Management Domain manufacturer-specific use of PCR[6]. The use of PCR[6] may not be common across different Management Domain manufacturers or even across different Management Domain models belonging to the same Management Domain manufacturer.

It is anticipated the TCG event log used during the Pre-OS environment may be transferred (pointer/size tuple) to and managed by TPM-capable Operating Systems. An EV_SEPARATOR event will be recorded by the Pre-OS firmware as the last record prior to transfer. No subsequent entries will typically be made to the TCG event log by the Pre-OS firmware. This specification does not define a mechanism for a platform manufacturer to add entries to an OS-managed TCG event log after the OS has started.

Operating Systems have their own TPM driver. This specification does not define a mechanism for a Management Domain manufacturer to send commands to the TPM after the firmware launches an Operating System. A platform manufacturer may have difficulty extending PCR[6] without interfering with OS management of the TPM. In this case, a platform manufacturer would need to work with their OS vendor to implement a mechanism to record measurements in the OS-present event log.

OS-present code should not use this PCR for sealing or attestation without knowing manufacturer-specific information about its usage

End of informative comment

1. The Management Domain manufacturer MAY define the purpose of PCR[6].
2. If the Management Domain manufacturer extends PCR[6] during the Pre-OS environment, it MUST record a corresponding log entry.
3. If the Management Domain manufacturer extends PCR[6] during the OS environment, it MUST record a corresponding log entry in the TCG event log by collaborating with the OS to place an event in the event log now managed by the OS.

Entities that MUST be measured if the TPM is enabled:

1. Per Section 7.1.1 Measuring OS Boot Events, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit. See Section 9.4.1 Event Types.
2. Per Section 3.3.2 Error Conditions, if an error occurs, the digest of the value 00000001h MUST be extended in PCR[0-7] and an EV_SEPARATOR event SHOULD be recorded in the event log for each PCR. See Section 9.4.1 Event Types.
3. Per Section 7.1.1 Measuring OS Boot Events, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit. See Section 9.4.1 Event Types.

Entity that MAY be measured if the TPM is enabled:

The Management Domain manufacturer MAY measure platform specific events using the event type EV_COMPACT_HASH. If measured, the Event Data field SHALL be a unique string.

Entity that MUST be measured if the TPM is disabled:

Per Section 3.3.1, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit.

Method of measurement:

The recording of an EV_SEPARATOR event by the Pre-OS code on exit MUST be the last Management Domain firmware initiated measurement in the PCR.

3.3.4.8 PCR[7] - Reserved For Future Use

PCR[7] is reserved for future use but is the recipient of error condition and separator events as specified below.

Entities that MUST be measured if the TPM is enabled:

1. Per Section 3.3.2 Error Conditions, if an error occurs, the digest of the value 00000001h MUST be extended in PCR[0-7] and an EV_SEPARATOR event SHOULD be recorded in the event log for each PCR. See Section 9.4.1 Event Types.
2. Per Section 7.1.1 Measuring OS Boot Events, the digest of the value FFFFFFFFh MUST be extended in PCR[0-7] and an EV_SEPARATOR event MUST be recorded in the event log for PCR[0-7] prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit. See Section 9.4.1 Event Types.

3.3.4.9 PCR[8] - Management Domain OS

Start of informative comment

PCR[8] is the target PCR for the Management Domain and should be used for kernel, device driver, and device files.

The Operating System defines the code measurement contents and method of measurement for PCR[8].

End of informative comment

3.3.4.10 PCR[9] - Management Domain OS Configuration

Start of informative comment

PCR[9] is the target PCR for the Management Domain and should be used for configuration related to kernel, device drivers, and device files.

The Operating System defines the configuration contents and method of measurement for PCR[9].

End of informative comment

3.3.4.11 PCR[16] – Debug

Start of informative comment

PCR[16] is resettable from any locality and is for use by any Management Domain entity. It is intended, by convention, to be used as a debug PCR. Components should use this PCR for debugging purposes only (e.g., software development of components utilizing the PCR features of the TPM, e.g., TPM2_Create). Applications targeted for user, final, or production environments should not use this PCR in their final release.

Because PCR[16] is not intended for use with sealing or attestation, measurements for it should not be recorded in the event log.

End of informative comment

Method of measurement:

1. Any component on the Management Domain MAY use and reset PCR[16] at any time.
2. On platforms targeted for user, final or production environments, if the firmware does extend PCR[16], it MUST NOT record the event in the event log.

4 Non-Volatile Storage

Start of informative comment

Non-Volatile Storage (NV RAM) is made available by the TPM for use by various processes on the Management Domain. A limited amount of resources is required to be provided by the TPM.

End of informative comment

4.1 NV RAM Size and Allocation

Start of informative comment

The PC Client Platform TPM Profile for TPM 2.0 specifies a minimum amount of NV Storage the TPM must provide. This value is based upon the anticipated usages of this area at the time the specifications were written.

End of informative comment

5 Maintenance

Start of informative comment

Maintenance in the context of this specification refers to the processes surrounding upgrade or replacement of the Management Domain firmware ROM / Flash. All requirements for TPM maintenance are manufacturer-defined per the TPM Library Specification for Family 2.0.

End of informative comment

Implementation of maintenance is optional. If it is implemented, it MUST be implemented as defined in this section.

5.1 Management Domain Firmware Recovery Mode

Start of informative comment

This is a failure-recovery mode of Management Domain firmware that is invoked by the SRTM typically when the firmware is found to be corrupt. The Management Domain may allow for configurable recovery through automated or manual procedures. The Management Domain firmware Recovery Mode may perform a minimal initialization of the system and then may attempt to boot a recovery program. The Management Domain firmware Recovery Mode may not have the capability to continue the SRTM measurement chain.

Some attack scenarios have been identified due to the “Firmware Recovery” feature implemented in many instances of Platform Firmware. A method to counter these attacks could implement:

1. Use of hardware to disable the TPM until the next __TPM_INIT; or,

2. Disabling the TPM by calling TPM2_Startup (CLEAR) followed by TPM2_HierarchyControl (EH Disable, SH Disable); or,

3. Unconditionally performing a Management Domain platform reset upon completion of recovery code.

End of informative comment

1. It MUST NOT be possible for a recovery mode to allow a different boot state as represented by the SRTM using the values in the PCR[0-7].
2. If Recovery requires a boot in recovery mode, the Management Domain firmware SHALL:
    - Initialize the TPM in a disabled state by sending a TPM2_Startup (CLEAR) followed by a TPM2_HierarchyControl (EH disable, SH disable) to disable each of the hierarchies.
    - Cap PCR[0-7] in all active PCR banks with the digest of the value FFFFFFFFh and record an EV_SEPARATOR Event in the event log. See Section 9.4.1 Event Types.

6 TPM Discoverability

Start of informative comment

A Management Domain TPM powers up with all hierarchies enabled and all functions available by default. There are cases where a Management Domain administrator might decide that they want to disable the TPM to ensure it is not usable.

This section describes the requirements for managed server OEMs when they provide the ability for a Management Domain administrator or Management Domain firmware to completely disable the TPM. The requirements relate to the visibility of the TPM to the Management Domain administrator interface.

End of informative comment

6.1 TPM Visibility to the Management Domain OS

Start of informative comment

The default state of the TPM makes it ready and available to the Management Domain OS.

End of informative comment

If the managed server OEM implements a capability to prevent the Management Domain firmware from discovering the TPM, or disabling the TPM, the firmware SHALL implement all of the following requirements:

1. When the Management Domain TPM is visible to the firmware and enabled:
    - The managed server OEM SHALL make all measurements in compliance with the requirements of this specification.
    - The Management Domain firmware MAY support the Physical Presence Interface as specified in the TCG PC Client Platform Physical Presence Interface Specification for TPM Family 2.0 revision 1.3 version 52 or later.
    - The Management Domain TPM enable/disable control MAY be present in Management Domain administrative console. If this control is implemented, it is the responsibility of the OEM to ensure integrity of this control.
    - All TPM Hierarchies MUST be enabled.
    - Management Domain firmware SHALL populate the TCG Event Log as described in Section 9.1.
2. When the Management Domain TPM is disabled:
    - The TPM SHALL not be usable by OS-present applications.
    - The Management Domain firmware SHALL disable the Storage and Endorsement Hierarchies by use of the command TPM2_Hierarchy_Control (ehDisable and shDisable).
    - The TPM Platform Hierarchy MAY be disabled by Management Domain firmware on every boot using a TPM2_HierarchyEnable command.
    - Management Domain firmware SHALL cap PCR[0-7] as described in Section 3.3.4 PCR Usage.
    - Management Domain firmware SHALL NOT create the TCG Event Log.

6.2 TPM Visibility to Management Domain OS through Administrative Console

Start of informative comment

Managed server OEMs may provide the control surface for the TPM through a Management Domain console.

End of informative comment

If the managed server OEM implements a capability to disable the TPM through an administrative console, the Management Domain firmware SHALL implement all of the requirements in Section 6.1 and the following additional requirement:

The TPM control surface as defined in Section 6.3 SHALL NOT be present in the managed server administrative console.

6.3 Platform Firmware Setup TPM Control Surface

Start of informative comment

Managed servers present a view of the TPM configuration to managed server administrators through their administrative consoles.

End of informative comment

1. Platform Firmware SHALL provide a mechanism to clear the TPM from the administrative console.
2. If the Platform supports changing the algorithm used for measuring events, Platform Firmware MAY support a mechanism to do so through the administrative console.

7 Management Domain State Transitions

7.1 Architecture and Definitions

Start of informative comment

A handoff to the Management Domain OS occurs after Management Domain firmware has completed its initialization and testing of the Management Domain hardware. As defined in Section 2.2.8 Boot State Transition, a specific event marks the transition from boot loader control to OS kernel control.

End of informative comment

7.1.1 Measuring OS Boot Events

Start of informative comment

An Event Log enables a challenger to determine the state of trust of the Management Domain and enables software on the Management Domain to reconstruct boot events. The Management Domain functions designed for computing hash values and extending PCRs should automatically log the extended events.

However, there are events that must be added to the Event Log that are not the result of a PCR extend operation.

The purpose of this section is to specify all the required events added to the Event Log for OS boot, including events that do not result from a PCR extend operation.

End of informative comment

The Management Domain firmware MUST measure the event EV_SEPARATOR into pre-OS PCRs [0-7] once for each platform boot cycle and the measurement of that event MUST happen when leaving the pre-OS environment and entering the OS-Present environment. The data within the event field of the EV_SEPARATOR event MUST be 32 bits (a double-word) of FFs (that is, FFFFFFFFh) unless an error condition occurs. See Section 3.3.2.2 Errors recording Measurements.

7.1.2 Passing Control of the TPM from Pre-OS to OS-Present Environments

Start of informative comment

Once Management Domain firmware has turned control over to an Operating System, the Post-Boot environment will load its own set of drivers and code to access the TPM. This could cause a potential conflict since there may be contention between the Pre-OS and OS-Present environments for use of and access of the TPM.

The Management Domain design needs to provide a clean way to pass access control from firmware to OS.

End of informative comment

Pre-OS and OS-Present environments MUST prevent contention between the two environments for use of and access to the TPM.

7.2 Power States, Transitions, and TPM Initialization

Start of informative comment

This specification supports two Management Domain power states; OFF and ON (PC Client's "S0 Working" equivalent).

End of informative comment

When transitioning from the OFF state to the ON state:

1. The Management Domain MUST issue a __TPM_INIT.
2. If the TPM interface is accessible, the SRTM MUST Issue a TPM2_Startup (CLEAR) command.

3. If the TPM interface is accessible and the TPM is enumerated, the SRTM MUST issue a TPM2_SelfTest command or ensure the TPM is configured for automatic self-test execution prior to the recording of an EV_SEPARATOR event by the Pre-OS code on exit.
4. If the TPM is visible to the OS and the TPM interface is accessible, the platform manufacturer MUST set platformAuth and MAY set platformPolicy to a high entropy value during execution of the SRTM such that later software is unable to change objects in the Platform Hierarchy or perform operations that require platform authorization.
5. If the TPM is enabled and visible, the SRTM MUST start the SRTM measurement chain by recording integrity measurements during the boot process per Section 3.3 Integrity Collection and Reporting.
6. Boot Managers SHOULD be prepared for the TPM's Self-Test actions associated with the TPM2_SelfTest command or configured automatic execution to be in progress when the Boot Manager is launched.

7.2.1 ON (S0-Working equivalent) to OFF

Start of informative comment

This transition is from the running OS to the OFF state. In contrast to power loss, this is an orderly shutdown.

The TCG Platform Reset Attack Mitigation Specification permits configuration of the platform so memory is erased during boot under certain conditions. One variable is whether the platform performed an orderly shutdown or unexpectedly lost power. If this transition is an orderly shutdown, the OS should be able to purge secrets from the Management Domain memory when appropriate.

End of informative comment

1. Management Domains implementing the TCG Platform Reset Attack Mitigation Specification MAY perform additional actions to avoid clearing memory on the next boot according to the TCG Platform Reset Attack Mitigation Specification.
2. Because TPM state is expected to be discarded after entering S5 (OFF), the OS present environment SHOULD issue a TPM2_Shutdown(CLEAR) command.
3. If no shutdown command is executed or fails, a TPM Reset with a TPM2_Startup (CLEAR) command MUST be issued on the following OFF state to ON state transition. This is the default behavior but is intended to identify the error handling path should a shutdown command fail to be executed.

8 TPM Interrupt Support

Start of informative comment

As usage of the TPM by faster embedded devices like Management Domains becomes more prevalent, methods of improving TPM performance are increasingly needed. Many discrete TPMs support interrupts as defined in the TCG *PC Client Platform TPM Profile* (PTP) Specification (Family "2.0"). As there is no programmatic method to determine TPM interrupt support, Management Domain firmware needs to be preconfigured based on the TPM implemented on the platform. This enhances performance.

End of informative comment

If the TPM supports interrupts, Management Domain firmware MAY enable them.

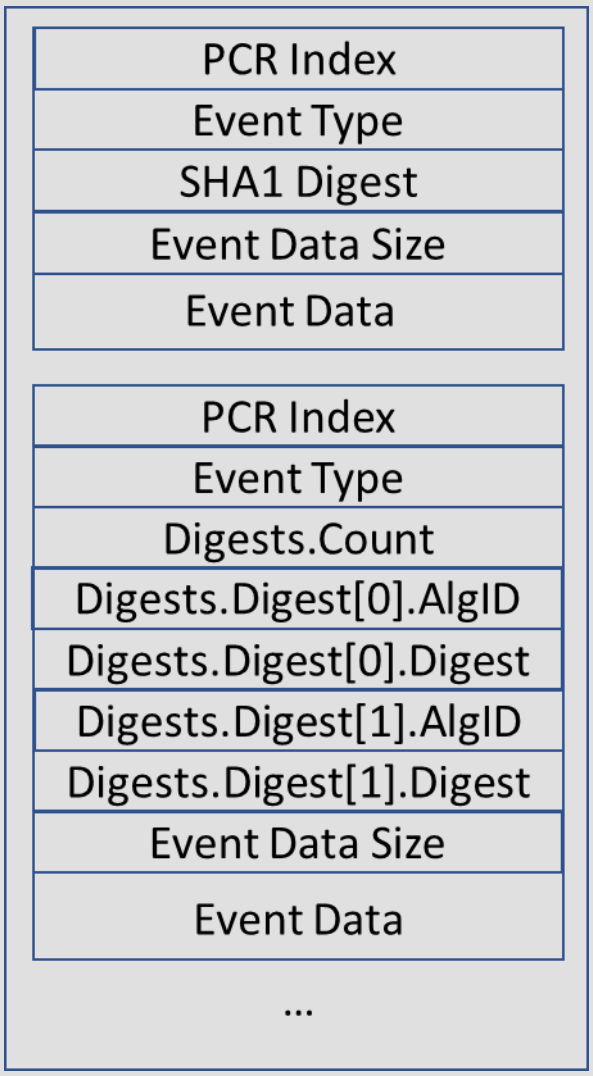
9 Event Logging

9.1 Introduction

Start of informative comment

The Event Log is the informational record of measurements made to PCRs by the Management Domain firmware and may include some informational events that are not PCR extensions. The informational events are used to convey valuable but unsubstantiated information to an evaluator of the log. Each measurement made, as well as the information events, are recorded in the event log as individual entries with specified fields. The first event in the event log is the informational event TCG_EfiSpecIdEvent. As parsers need a way to determine the format of the events in the log, the first event identifies the version of the log, which implies the format of the events, and the number and size of the recorded digests. The layout of the TCG_PCR_EVENT2 structure that is used for the rest of the event log depends on the information in the first event. For that reason, the first event is formatted as TCG_PCClientPCREvent structure and is independent of digest sizes. The structures are as defined in the TCG PC Client Platform Firmware Profile (PFP) Specification (Family “2.0”).

A single event in the event log, unless of informational type, always starts with a header that consists of the PCR index, an event type, the recorded digest(s), and a size of the event data. The PCR index identifies the PCR into which one or more digest of the event data has been extended. The event type identifies the type of event that is recorded in the event log entry. The digest(s) field contains the digest values of the hashed information that is extended into the PCR. **Note**: the digests’ values may occur in any order within an event. Finally, the event data size defines the size of additional event data. For the cryptographically agile log format, the digest consists of a TPML_DIGEST_VALUES (defined in the TPM2 Library Specification Part 2) structure. Figure 5 Depiction of Log Formatsshows the log format.



Crypto Agile Event Log with 2 Digests and one PCR

Figure 5 Depiction of Log Formats

Note: for the digests in the cryptographically agile log format, although the type names from the TPM 2.0 Library Specification are used (e.g. TPML_DIGEST_VALUES), the storage of Integers in the event log is little-endian. In this case, the Count and the AlgID fields, leveraged from the TPM 2.0 Library specification, are stored as little-endian. If Management Domain firmware utilizes the TPM2_PCR_Event command to hash the data, this command returns a TPML_DIGEST_VALUES structure encoded in big-endian. Management Domain firmware would need to modify the TPML_DIGEST_VALUES structure so that Count and AlgID fields are little-endian before adding the event to the event log. Likewise, if Management Domain firmware is preparing the TPML_DIGEST_VALUES structure to send to the TPM using a TPM2_PCR_Extend command, the values must be big-endian for the TPM structure. There may be other instances where this type of re-encoding is required. This specification does not produce an exhaustive list.

An event will be densely packed. That is, even though there can be multiple digests in an event, the algorithm ID of the next digest follows immediately after the last byte of the previous digest, without padding. Table 2 and Table 3 below illustrate an example using an EV_SEPARATOR event in the cryptographically agile log event format.

Table 2 Crypto Agile Event Log Event Example 1

Field Name	Offset	Size (in bytes)	Content
PCRIndex	0x00	4	2
eventType	0x04	4	0x00 00 00 04 (EV_SEPARATOR)
Digests	0x08		
Digests.Count	0x08	4	1
Digests.Digests	0x0C		
Digests.Digests[0].AlgID	0x0C	2	0x00 04 (SHA1)
Digests.Digests[0].Digest	0x0E	20	0x90, 0x69 0xca, 0x78, 0xe7, 0x45, 0x0a, 0x28, 0x51, 0x73, 0x43, 0x1b, 0x3e, 0x52, 0xc5, 0xc2, 0x52, 0x99, 0xe4, 0x73
EventSize	0x22	4	4
Event	0x26	4	0x00, 0x00, 0x00, 0x00

The encoding as a byte stream would look as follows. The start of the line describes the offset for the first byte in the line. (All values in hexadecimal.)

0000: 02 00 00 00 04 00 00 00 – 01 00 00 00 04 00 90 69

0010: ca 78 e7 45 0a 28 51 73 – 43 1b 3e 52 c5 c2 52 99
0020: e4 73 04 00 00 00 00 00 – 00 00

Table 3 is another example of the same separator event, but in this case using two PCR banks:

Table 3 Crypto Agile Event Log Event Example 2

Field Name	Offset	Size (in bytes)	Content
PCRIndex	0x00	4	2
eventType	0x04	4	0x00 00 00 04 (EV_SEPARATOR)
Digests	0x08		
Digests.Count	0x08	4	2
Digests.Digests	0x0C		
Digests.Digests[0].AlgID	0x0C	2	0x00 04 (SHA1)
Digests.Digests[0].Digest	0x0E	20	0x90, 0x69 0xca, 0x78, 0xe7, 0x45, 0x0a, 0x28, 0x51, 0x73, 0x43, 0x1b, 0x3e, 0x52, 0xc5, 0xc2, 0x52, 0x99, 0xe4, 0x73
Digests.Digests[1].AlgID	0x22	2	0x00 0b (SHA-256)
Digests.Digests[1].Digest	0x24	32	0xdf, 0x3f, 0x61, 0x98, 0x04, 0xa9, 0x2f, 0xdb, 0x40, 0x57, 0x19, 0x2d, 0xc4, 0x3d, 0xd7, 0x48, 0xea, 0x77, 0x8a, 0xdc, 0x52, 0xbc, 0x49, 0x8c, 0xe8, 0x05, 0x24, 0xc0, 0x14, 0xb8, 0x11, 0x19
EventSize	0x44	4	4
Event	0x48	4	0x00, 0x00, 0x00, 0x00

The byte stream in the example of Table 3 would look like this:

0000: 02 00 00 00 04 00 00 00 – 02 00 00 00 04 00 90 69

0010: ca 78 e7 45 0a 28 51 73 – 43 1b 3e 52 c5 c2 52 99

0020: e4 73 0b 00 df 3f 61 98 – 04 a9 2f db 40 57 19 2d

0030: c4 3d d7 48 ea 77 8a dc – 52 bc 49 8c e8 05 24 c0

0040: 14 b8 11 19 04 00 00 00 – 00 00 00 00

Note: the algorithm ID of the SHA-256 digest at offset 34 follows directly after the last meaningful byte of the SHA-1 digest. Also, the event data size field (EventSize) at offset 68 follows directly after the last meaningful byte of the SHA-256 digest.

Information events have an event type of value EV_NO_ACTION and PCR index of zero. The event data field of an EV_NO_ACTION event may contain different data. To distinguish between these different no action events, each EV_NO_ACTION event data starts with a 16 bytes Signature that defines the format of the event data. The TCG_EfiSpecIdEvent event in the crypto agile event log has a SHA1 log format digest field of 20 bytes of zero. All other EV_NO_ACTION events in a crypto agile log have digest entries for each recorded hashing algorithm and the digests are set to be all zeros.

Note: This Management Domain firmware specification does not provide support for Predictive Event Logs (logs that contain digests for all supported PCR banks, active and non-active).

End of Informative comment

1. Management Domain firmware SHALL create an event log containing all events measured by Management Domain firmware.
2. The first event log entry SHALL be a TCG_PCClientPCREvent structure.
3. The first log entry in the measurement log MUST start at the Measurement Log Start Address. Subsequent measurements MUST be contiguous.
4. Event log entries after the first entry SHALL be TCG_PCR_EVENT2 structures. See Section 9.2 TCG Defined Structures.
5. For each Hash algorithm enumerated in the TCG_PCClientPCREvent entry, there SHALL be a corresponding digest in all TCG_PCR_EVENT2 structures. **Note:** This includes EV_NO_ACTION events which do not extend the PCR.
6. Sizes for all Hash algorithms included in a TCG_PCR_EVENT2 structure SHALL be enumerated in the TCG_PCClientPCREvent structure.
7. There SHALL be a Hash algorithm in the TCG_PCClientPCREvent structure for all allocated PCR banks.
8. There SHALL NOT be padding between event log entries.
9. All integer structure members SHALL be marshaled in little endian format.

9.2 TCG Defined Structures

Start of informative comment

The three structures referenced in this section, TCG_PCClientPCREvent, TCG_PCR_EVENT2, and TCG_PCClientTaggedEvent are defined in the TCG PC Client Platform Firmware Profile (PFP) Specification (Family “2.0”). These are the only three Event structures supported in this Management Domain firmware specification.

End of informative comment

9.3 Measurement Event Entries and Log

Start of informative comment

The value within a PCR is used both for Sealed Storage and for attestation. When used for attestation, the raw hash value carries little meaning. Therefore, more meaningful structures are used that carry with them information. This information is contained within the structure TCG_PCR_EVENT2.event as referenced in Section 9.2 TCG Defined Structures above.

End of informative comment

1. The hash used MUST be the hash identified by the algorithmID field recorded by Management Domain firmware in the TCG_EfiSpecIdEvent (tdTCG_PCClientPCREvent) Structure. There MAY be more than one.
2. The procedure for forming a TCG_PCR_EVENT2 structure is:
    a. Set A to an instantiation of a TCG_PCR_EVENT2 structure.
    b. Set A.pcrIndex to the index of the PCR to be extended.
    c. Set A.eventType to the specified Event Type.
    d. Fill A.event with either the data to be measured or the description of data to be measured and set A.eventSize to the size of A.event.
    e. Set B to A.digests, a TPML_DIGEST_VALUES structure:
        i. Set B.count to the number of digests in A.digests
        ii. Create C, a TPMT_HA structure, = B.digests[index] where index starts at 0 and continues to count - 1:
            (1) Set C.hashAlg to algorithmID of B.digests[index].
            (2) Create C.digest according to A.eventType field as defined in Table 4
            (3) Repeat for each implemented hash algorithm
        iii. Create B.digests by appending each instance of C. Similar to the TPMT_HA structure used in TPM commands, the fields are densely packet, i.e., if the result of the hashing algorithm is 20 bytes, the C.digest field is only 20 bytes.

    Note for steps d and e: The description used in Table 4 is not to be taken literally for all event types. Where the event field will contain data or structures, this statement is to be taken literally and the event field is to be filled in. However, when measuring code, it is not practical to place the entire code area into the event field just to take the hash of it. Also, it is not expected for the event field to contain the entire code area in the event log for these event types. For precise contents of this field, refer to the description of the event type.
3. Extend A.pcrIndex using A.digests as the value for each implemented PCR bank. **Note:** All integers should be encoded in Big Endian, before being sent to the TPM.
4. Append A to the event log. **Note**: All integers should be encoded in Little Endian before appending to the event log.
5. The sequence of the Measurement Log entries MUST be the same as when the events were extended into the TPM PCRs.

9.4 Event Descriptions

Start of informative comment

Each event recorded in the Event Log is tagged as a particular event type. This section specifies all the event type tags that must or may be added to the Event Log on a Management Domain platform compliant to this specification.

The value within a PCR is used for sealed storage, attestation, and re-construction of the boot flow. The raw hash value in a PCR is sufficient for sealed storage, but not for attestation or replay.

Event Log entries add value to the raw hash values in PCRs for attestation as well as for reconstructing the events that triggered the measurements into the PCRs.

End of informative comment

9.4.1 Event Types

Start of informative comment

One element in an Event Log entry is TCG_PCR_EVENT2.eventType. Table 4 is normative and specifies all the event types that can be used in an Event Log entry for the measurement events specified in this document for a Management Domain platform.

The values associated with these Management Domain specific platform event types must not overlap with the event type values already defined for other TCG platform architecture specifications.

Management Domain platforms may also use events originally from the PC Client Implementation Specification for Conventional BIOS, including but not limited to, EV_POST_CODE, EV_SEPARATOR, EV_S_CRTM_VERSION, EV_S_CRTM_CONTENTS and EV_NO_ACTION. This specification does not address measurement of optional tagged events, as defined in the TCG v1.21 PC Client Specification for Conventional BIOS, section 10.4.2.

End of informative comment

The event types in Table 4 are defined for the field TCG_PCR_EVENT2.eventType. Any event type value not defined in this table SHALL NOT be used by Management Domain Platform Firmware and is reserved for application and OS usage.

Table 4 Events

Value	Label	Description
0x00000001	EV_POST_CODE This event MUST extend the PCR	Used for PCR[0] only to record POST code or manufacturer-controlled embedded ROMs as a binary image. The digest field contains the tagged hash of the code or data to be measured (e.g., POST portion of Platform Firmware) for each PCR bank The event field SHOULD NOT contain the actual code or data, but MAY contain informative information about the POST code. For POST code, the event data SHOULD be “POST CODE”. For embedded ROMs, the event data SHOULD be “Embedded Driver”. See Section 3.3.4.1
0x00000003	EV_NO_ACTION This event MUST NOT extend any PCR	The fields: pcrIndex and digests MUST contain the value 0 for active PCR banks. The event field contains informative data that was not extended into any PCR.

Value	Label	Description
0x00000004	EV_SEPARATOR This event MUST extend the PCRs 0 through 7 inclusive.	Used for PCRs[0, 1, 2, 3, 4, 5, 6 and 7]. Per Section 3.3.2, used to indicate an error occurred recording the CRTM, POST BIOS, or Embedded ROMs. For this situation, the digest field MUST contain the tagged digest of the value 00000001h. The event field is not defined in this specification in order to allow platform manufacturers to include an indication of what error occurred or other useful troubleshooting information. Per Section 3.3.4, used to delimit actions taken during the pre-OS and OS environments. For this situation, the digests field MUST contain the tagged hash of the event data for each PCR bank. The event data size MUST be 4. The event field MUST contain the hex value 00000000h or FFFFFFFFh.
0x00000005	EV_ACTION This event MUST extend the PCR	Used for PCRs [1, 2, 3, 4, 5, and 6]. The digests field contains the tagged hash of the event field for each PCR bank. A specific action measured as a string defined in Section 9.4.3.
0x00000006	EV_EVENT_TAG	Used for PCRs defined for OS and application usage. Defined for use by Management Domain Operating System or Software. The event field contains the structure defined in Section 9.4.2. The digests field MUST contain the tagged hash of the event data for each bank. **Note**: The event data is not defined in this specification in order to enable Management Domain OS or software to define the Event Type format used to report events of their choosing.
0x00000007	EV_S_CRTM_CONTENTS This event MUST extend the PCR	Used for PCR[0] only. The digests field contains the tagged hash of the SRTM for each PCR bank. The event field SHOULD NOT contain the actual SRTM code but MAY contain informative information about the SRTM code. See Section 3.3.4.1.
0x00000008	EV_S_CRTM_VERSION This event MUST extend the PCR	Used for PCR[0] only. The digests field contains the tagged hash of the event field for each PCR bank. The event field contains the version string of the SRTM. See Section 3.3.4.1.
0x00000009	EV_CPU_MICROCODE This event MUST extend the PCR	Used for PCR[1] only. The digests field contains the tagged hash of the microcode patch applied for each PCR bank. The event field contains a descriptor of the microcode patch. See Section 3.3.4.2.
0x0000000A	EV_PLATFORM_CONFIG_FLAGS This event MUST extend the PCR	Used in PCR[1] only. The digests field contains the tagged hash of the event field for each PCR bank. The event field contents are manufacturer implementation-specific but MUST indicate whether each optional PCR[1] measurement is measured or not for the set of measurements that can be toggled by the platform owner. See Section 3.3.4.2.

Value	Label	Description
0x0000000B	EV_TABLE_OF_DEVICES This event MUST extend the PCR	Used in PCR[1] only. The digests field contains the tagged hash of the event field for each PCR bank. The event field contents are manufacturer implementation-specific. The event field contains the Platform Manufacturer-provided Table of Devices or other Platform Manufacturer-defined information. The Platform Manufacturer defines the content and format of the Table of Devices. The Management Domain Certificate may provide a reference to the meaning of these structures and data. See Section 3.3.4.2.
0x0000000C	EV_COMPACT_HASH This event MUST extend the PCR	May be used for any PCRs except 0, 1, 2, or 3. The event field MAY be informative or MAY be hashed to generate the digests field, depending on the component recording the event. This event is typically used by Management Domain firmware vendors to measure events into PCR[6]. See Section 3.3.4.7

9.4.2 Tagged Event Log Structure

Start of informative comment

This Event Type is defined in this specification for use by Management Domain OS and software. Only the Event Type and its structure are specified: measurements are not specified. This structure is defined in Section 9.2 TCG Defined Structures above.

End of informative comment

1. Tagged Event Data MUST be measured and logged using the TCG_PCR_EVENT2 structure.
2. The TCG_PCR_EVENT2.eventType SHALL be EV_EVENT_TAG.
3. The TCG_PCR_EVENT2.event SHALL contain one or more of the TCG_PCClientTaggedEvent structures.
4. TCG_PCR_EVENT2.eventSize contains the size of all of the TCG_PCClientTaggedEvent structures stored in the TCG_PCR_EVENT2.event field.

9.4.3 EV_ACTION Event Types

For each EV_ACTION event produced by the platform, Management Domain firmware MUST create the event indicated below in the following actions strings. The strings below are enclosed in quotes for clarity; the actual log entries SHALL NOT include the quote characters, nor a NUL terminator. They SHALL be logged according to the following:

- TCG_PCR_EVENT2.eventType = 05h (the value for the EV_ACTION event type from Table 4 in Section 9.4.1)
- TCG_PCR_EVENT2.digests = the tagged hash (for each PCR bank) of the Event[] field
- TCG_PCR_EVENT2.event[] = ASCII string from Table 5:

Table 5 EV_ACTION Event types

Action Index[1]	String	Purpose and Comments	PCR
0	"User Password Entered"	User has entered the correct user password at Management Domain firmware user interface. See note in Section 3.3.4.2.	1
1	"Administrator Password Entered"	User has entered the correct administrator password at Platform Firmware user interface. See note in Section 3.3.4.2.	1
2	"Password Failure"	The password typed at Platform Firmware user interface did not match the stored password after a specified number of retries. See Section 3.3.4.2.	1

9.4.4 EV_NO_ACTION Event Types

Start of informative comment

An EV_NO_ACTION event will not result in a digest being extended into a PCR, but has value to an evaluator of the log. For EV_NO_ACTION events other than the EFI Specification ID event the log will contain EV_NO_ACTION events in the TCG_PCR_EVENT2 format, so a parser does not have to switch between event formats. To eliminate further conditional treatment of the TCG_PCR_EVENT2 structure in the parser, the TCG_PCR_EVENT2 event for EV_NO_ACTION events will contain a digest entry for each algorithm that is used in other TCG_PCR_EVENT2 events, but the digest values will be all zeros. For example, if only the SHA256 PCR bank is used, the TCG_PCR_EVENT2.digests.count value is 1, TCG_PCR_EVENT2.digests.digests[0].algID is 0xB , and TCG_PCR_EVENT2.digests.digests[0].digest is 32 bytes of zeros. In the normative text the short notation of TCG_PCR_EVENT2.digests = 0 is used to express this.

End of informative comment

1. All EV_NO_ACTION events SHALL set TCG_PCR_EVENT2.pcrIndex = 0, unless otherwise specified.
2. All EV_NO_ACTION events SHALL set TCG_PCR_EVENT2.eventType = 03h (the value for the EV_NO_ACTION event type from Table 4 in Section 9.4.1).
3. All EV_NO_ACTION events SHALL set TCG_PCR_EVENT2.digests to all 0x00's for each allocated Hash algorithm. See Section 9.2 regarding theTCG_PCR_EVENT2 Structure.
4. All EV_NO_ACTION events SHALL set TCG_PCR_EVENT2.event to one of the events described in sections 9.4.4.1 and 9.4.4.2.
5. For each EV_NO_ACTION event produced by the platform, Platform Firmware MUST create and record the event indicated in this section.
6. EV_NO_ACTION events MUST NOT extend the PCR, but they MUST be recorded in the event log.

9.4.4.1 Specification ID Version Event

Start of informative comment

[1] This is only used for reference within this document

The first event in the event log is the Specification ID version. This event is not extended to a PCR. This event provides information to a consumer of the event log about what version of the specification is implemented by firmware. Because the TCG_PCR_EVENT2 structure contains digest size information which cannot be interpreted by the consumer, the TCG_EfiSpecIdEvent{} structure will be the event field of a TCG_PCClientPCREvent{} (see Section 9.2) structure with a 20 byte digest field of all zeros. It is an EV_NO_ACTION event type.

End of informative comment

9.4.4.2 Vendor Specific Events

Start of informative comment

Management Domain OEM's may define their own events for PCR[6]. Some events may use the event type EV_NO_ACTION. The events will be present in the event log, but won't extend a PCR. Management Domain OEMs may also define events that extend a PCR.

End of informative comment

Platform specific events that extend the PCR MUST use the event type EV_COMPACT_HASH. Each event that extends PCR[6] SHALL be logged with the following structure, where event data is a unique string pre-pended with the platform OEM name:

- TCG_ PCR_EVENT2.pcrIndex = 6
- TCG_ PCR_EVENT2.eventType = 0Ch (the value for the EV_COMPACT_HASH event type from Table 4 in Section 9.4.1)
- TCG_ PCR_EVENT2.digests = digest that was extended to the PCR (for all PCR banks)
- TCG_ PCR_EVENT2.event[] = "<Vendor Name> <Unique identifier>"

10 Platform Hierarchy (Physical Presence)

Start of informative comment

TPM 2.0 augments the concept of Physical Presence with the Platform Hierarchy authorization. Because the platform hierarchy is the point of control for the state of the TPM, it is important that the platform hierarchy be properly protected.

It is recommended that the Platform Hierarchy be locked at some point in the Management Domain boot process to ensure that the TPM cannot be exploited. Ideally this should occur before any non-trusted code is allowed to run.

End of informative comment

1. Management Domain firmware SHOULD protect access to the TPM's Platform Hierarchy and prevent access to the platform hierarchy by non-manufacturer-controlled components.
2. Management Domain firmware SHOULD:
    - Disable the Platform Hierarchy, or
    - If the Platform Hierarchy is enabled, Management Domain firmware MUST change platformAuth to a random value or a secret value prior to executing non-trusted code.

