TCG EFI Platform Specification

Version 1.20 Final
Revision 1.0
7 June 2006

For TPM Family 1.1 or 1.2

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

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

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

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

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

Copyright © 2004 - 2006 Trusted Computing Group, Inc.

Table of Contents

Corrections and Comments

Please send comments and corrections to techquestions@trustedcomputinggroup.org.

1. Introduction to this Document

Start of informative comment:

This document is about the processes that boot an EFI platform and boot an OS on that platform. Specifically, this specification contains the requirements for measuring boot events into TPM PCRs and adding boot event entries into the Event Log.

Software on the platform uses PCR values to seal secrets into blobs and then unseal those secrets if the PCR values are the same as when the secret was sealed.

Software on the platform uses PCR values together with the Event Log entries to reconstruct boot events, which requires a complete Event Log.

Software on other platforms uses the PCR values, together with the Event Log entries, for remote attestation of the EFI platform that holds the PCR values and Event Log.

Structure of this document

The scope of this specification is limited to what EFI can measure.

This specification contains nine other sections.

- Section 2 defines the meaning of three fundamental TCG concepts on an EFI platform: Static Locality, Static CRTM, and the static transitive chain of trust.
- Section 3 is an overview of the platform boot process and the OS boot process on an EFI platform and provides the model for PCR usage and for adding events to the Event Log. If the user of this specification reads this section first, the details in section 4 through 7 will be easier to understand.
- Section 4 contains the requirements for measuring PE/COFF image files.
- Section 5 is the detailed specification for measuring the platform boot process on an EFI platform.
- Section 6 is the detailed specification for measuring the OS boot process on an EFI platform.
- Section 7 is the detailed specification for adding entries to the Event Log during the OS boot process and the platform boot process.
- Section 8 is about using PCRs that are not used to measure the platform and OS boot processes.
- Section 9 is about maintenance of the firmware on an EFI platform that measures OS boot and platform boot events into PCRs
- Section 10 is about certification of an EFI platform that measures OS boot events and platform boot events into PCRs.

End of informative comment.

1.1 Conventions Used in this Document

Start of informative comment:

This section gives the data structure description and typographic conventions used in this document.

End of informative comment.

1.1.1 Data Structure Descriptions

All constants and data SHALL be represented as little-endian bit format, which requires the low-order bit on the far left of a constant or data item and the high-order bit on the far right. Exceptions to this, if any, will be explicit in this specification.

All strings SHALL be represented as an array of ASCII bytes with the left-most character placed in the lowest memory location.

In some memory layout descriptions, certain fields are marked reserved. Software must initialize such fields to zero, and ignore them when read. On an update operation, software must preserve any reserved field.

1.1.2 Typographic Conventions

The following typographic conventions are used in this document to illustrate programming concepts:

`Prototype`	This typeface indicates prototype code.
`Argument`	This typeface indicates arguments.
`Name`	This typeface indicates actual code or a programming construct.
register	This typeface indicates a processor register

1.2 External Specifications

References to external specifications:

Advanced Configuration and Power Interface (ACPI), Version 2.0, http://www.acpi.info

Extensible Firmware Interface Main Specification Version 1.10, http://www.intel.com/technology/efi

Microsoft Portable Executable and Common Object File Format Specification, Revision 6.0 or later, available at www.microsoft.com/whdc/ system/platform/firmware/PECOFF.mspx

System Abstraction Layer (SAL), http://www.intel.com/design/itanium/downloads/245359.htm

TCG TPM Main Specification Version 1.2, https://www.trustedcomputinggroup.org/specs/TPM/

TCG PC Client Implementation Specification for Conventional BIOS, Version 1.2, https://www.trustedcomputinggroup.org/specs/PCClient/

TCG EFI Protocol Specification, Version 1.2, https://www.trustedcomputinggroup.org/specs/PCClient/

TCG IPF Generic Server Specification, Version 1.0, Revision 0.8 dated March 23, 2005, https://www.trustedcomputinggroup.org/specs/Server/

TCG Itanium Architecture Server Specification, Version 1.0, https://www.trustedcomputinggroup.org/specs/Server/

TCG ACPI General Specification, Version 1.00, Revision 1.00 dated August 8, 2005, https://www.trustedcomputinggroup.org/specs/Server/

Unified Extensible Firmware Interface Specification (UEFI), Version 2.0, http://www.uefi.org

1.3 Abbreviations and Terminology

See section 1.3 of the TCG EFI Protocol for TCG Version 1.2, for definitions of abbreviations and terminology used in this specification.

2. Conceptual Overview

2.1 Static Locality

Start of informative comment:

For the description of the usages of Locality, refer to sections 3.1 and 3.2 of the *TCG PC Client Implementation Specification for Conventional BIOS, Version 1.2* and/or the *TCG Generic Server Specification, Version 1.0.*

The Normative text within this specification applies only to Locality 0, also known as the static locality.

End of informative comment.

2.2 Static Core Root of Trust for Measurement on an EFI Platform

Start of informative comment:

For the general description of the Static Core Root of Trust for Measurement (S-CRTM), refer to the *TCG PC Client Implementation Specification for Conventional BIOS, Version 1.2* and/or the *TCG Generic Server Specification, Version 1.0.*

The transitive trust chain on a TCG-aware EFI platform is rooted in the S-CRTM component.

On an EFI platform, the S-CRTM is platform firmware from system board motherboard ROM. The S-CRTM either measures the ensuring code on the platform that provides the EFI Boot Services in the System Table or provides the EFI Boot Services itself; in either case, measure this code into PCR [0].

End of informative comment.

2.3 Maintaining the Static Transitive Chain of Trust

Start of informative comment:

In order to maintain the Host Platform Transitive Chain of Trust that is rooted in the S-CRTM, prior to transferring control to another entity within Locality 0, an entity must measure the entity to which it will transfer control.

End of informative comment.

The code which provides the EFI boot services, including LoadImage ( ) and StartImage ( ), must be measured into PCR [0].

All successive EFI image loads are measured into PCR [0], PCR [2], or PCR [4]

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 3-1, Figure 3-2, Figure 3-3, and Figure 3-4.

- Figure 3-1 is an architecture drawing that shows the principle firmware and software components defined in the EFI Specification – EFI Boot Services, EFI Runtime Services, and the EFI OS Loader – and their relationship to the platform hardware and the OS software.
- Figure 3-2 shows the sequence of behaviors for the EFI components during the platform boot process and the OS boot process.
- Figure 3-3 maps, in general, the behavioral components on an EFI platform to the PCRs into which each type of behavioral component is measured
- Figure 3-4 shows an example platform and OS boot process on a platform with both BIOS and EFI firmware.

Together, these four diagrams form the boot event and PCR usage model upon which the requirements in sections 4 through 9 of this specification are based.

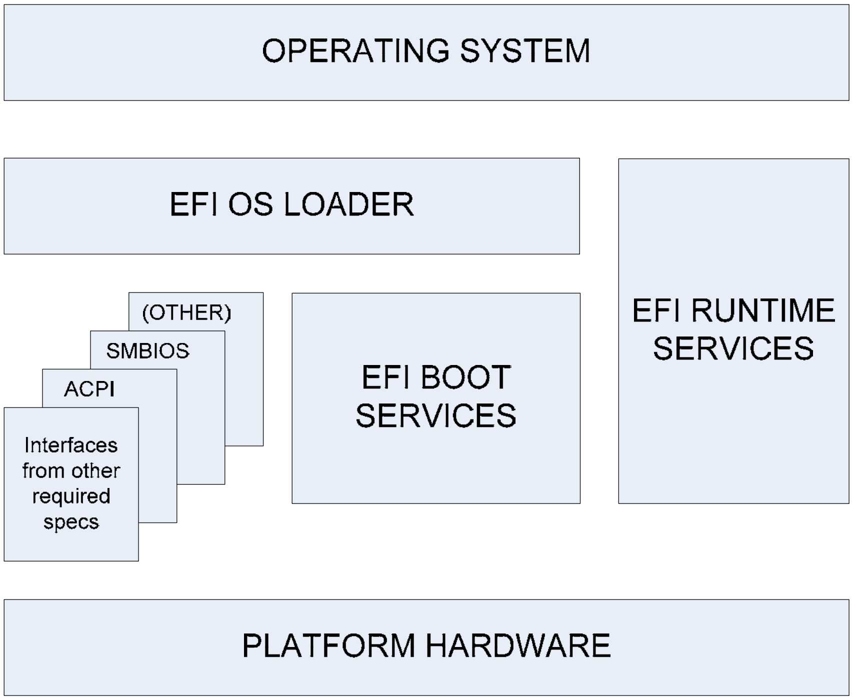
End of informative comment.

Start of informative comment:

Figure 3-1 is part of this, immediately below, is part of this Informative comment. This diagram illustrates the relationships of various components of an EFI-specification compliant system that accomplish platform boot and OS boot.

End of informative comment.

Figure 3-1 Architecture of an EFI Host Platform, Showing Principle Components for Platform Boot and OS Boot



Start of informative comment:

Figure 3-2, immediately below, is part of this Informative comment and shows the sequence of behaviors for the EFI components during the EFI platform boot process and the EFI OS boot process and, in general, the transitions where events are measured (labeled e0, e1, e2, and so on in the diagram).

In Figure 3-2, each one of the arrows with an "e" annotation represents a transition where a TCG Event is created; an un-annotated arrow does not represent a TCG event measurement. For example, the first measurement action, labeled "e0" is made by the platform initialization code.

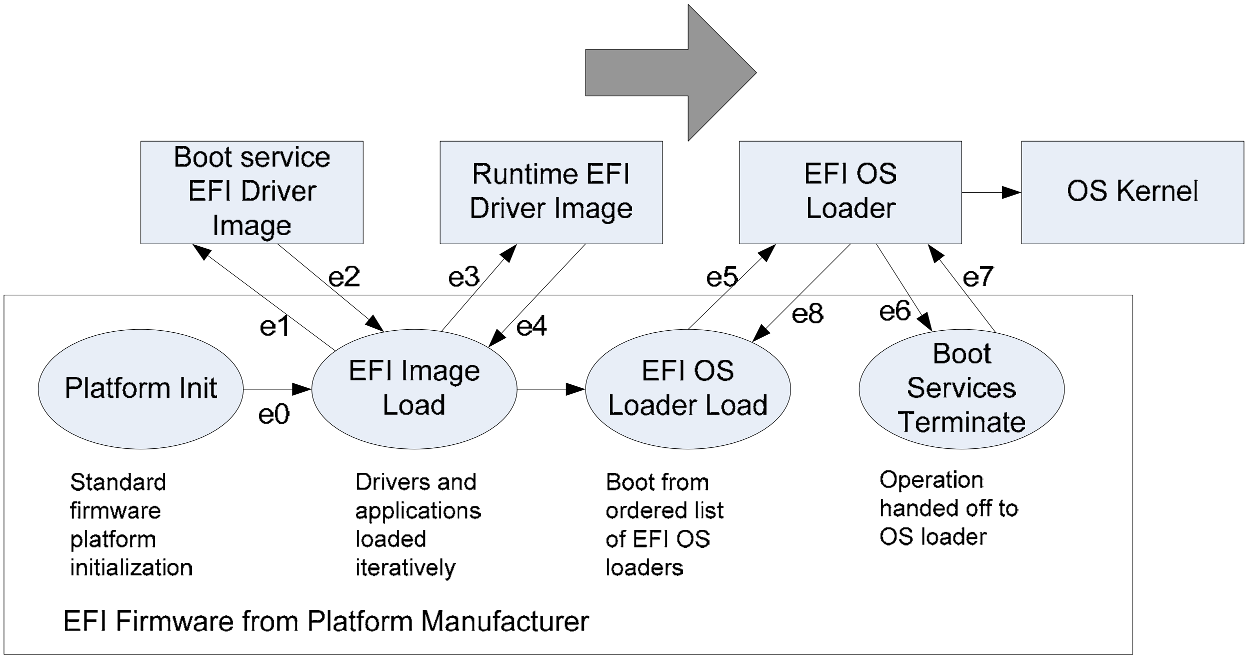
Event "e0" describes the behavior of platform firmware from system board ROM that measures code contained in the rest of the platform EFI firmware and that resides in the CRTM. This measuring agent may or may not contain the EFI Boot and Runtime Services. In the case of a CRTM that does not contain EFI Boot and Runtime Services, this measuring agent MUST then measure into PCR [0] the code that does contain EFI Boot and Runtime Services. This initial code, if in CRTM, must measure it's own version ID into PCR [0].

Event "e5" describes invocation of an EFI application in the boot order list. This is typically an operating system loader. The event "e5" will have associated with it an event that measures the PE/COFF image into PCR [4] and the contents of the boot variable, namely the EFI device path, into PCR [5]. For more information about this measurement action, see Section 7.4, Measuring OS Boot Events on an EFI Platform.

This Informative comment has given a couple examples of measurement actions associated with particular platform boot transition events, but makes no attempt to describe the measurement events associated with all the transitions shown in Figure 3-2. To see all the measurement events associated with all the transitions shown in Figure 3-2, see Sections 5, 6, and 7 of this specification.

End of informative comment.

Figure 3-2 Platform and EFI OS Booting Sequence

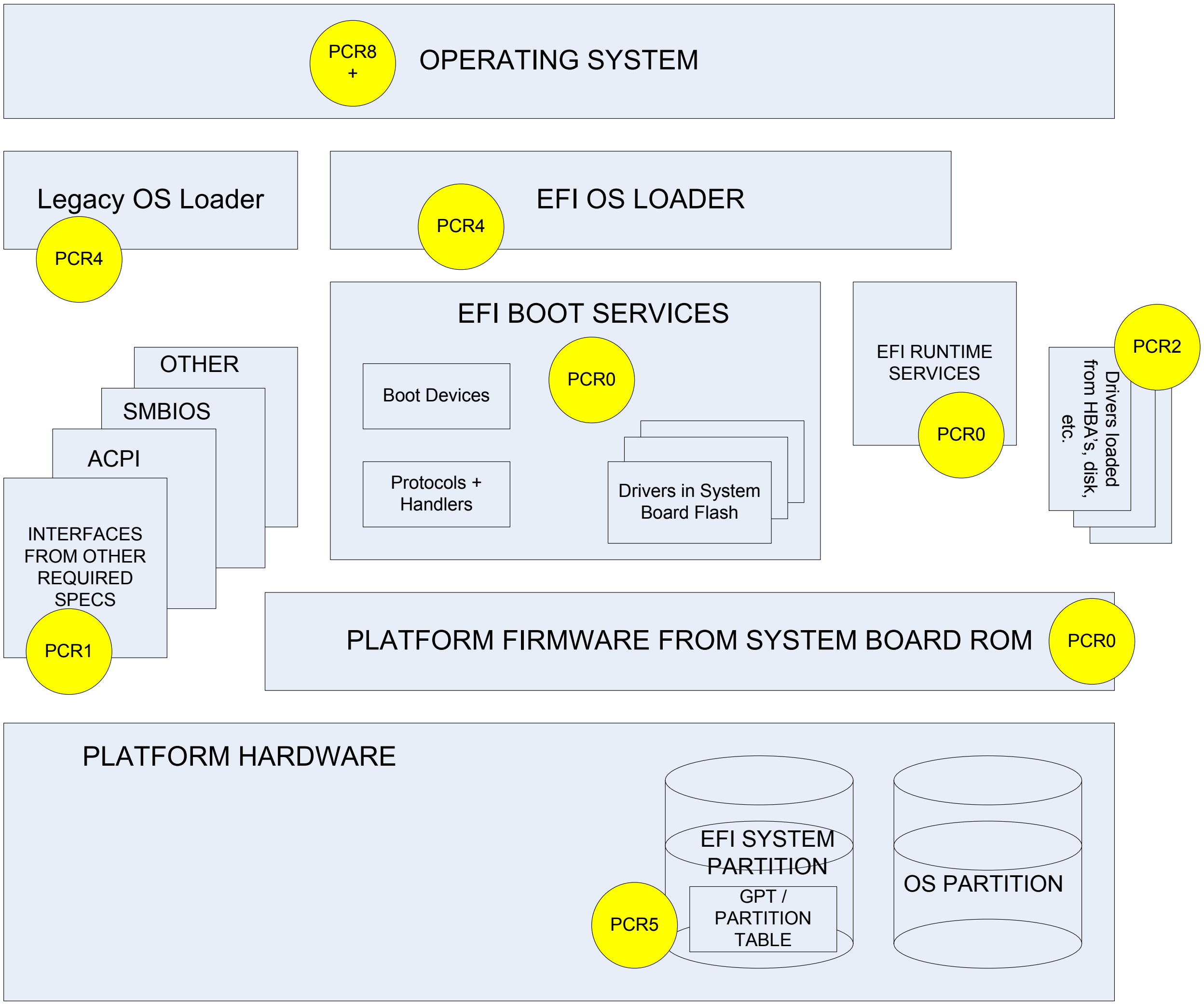

Boot service EFI Driver Image
Runtime EFI Driver Image
EFI OS Loader
OS Kernel
e0
e1
e2
e3
e4
e5
e6
e7
e8
Platform Init
EFI Image Load
EFI OS Loader Load
Boot Services Terminate
Standard firmware platform initialization
Drivers and applications loaded iteratively
Boot from ordered list of EFI OS loaders
Operation handed off to OS loader
EFI Firmware from Platform Manufacturer

Start of informative comment:

Figure 3-3, immediately below, shows with a picture the general scheme for PCR usage on an EFI platform.

End of informative comment.

Figure 3-3 Mapping of Measured Components to PCRs on an EFI Platform



3.1 Host Platforms with EFI and BIOS Firmware

Start of informative comment:

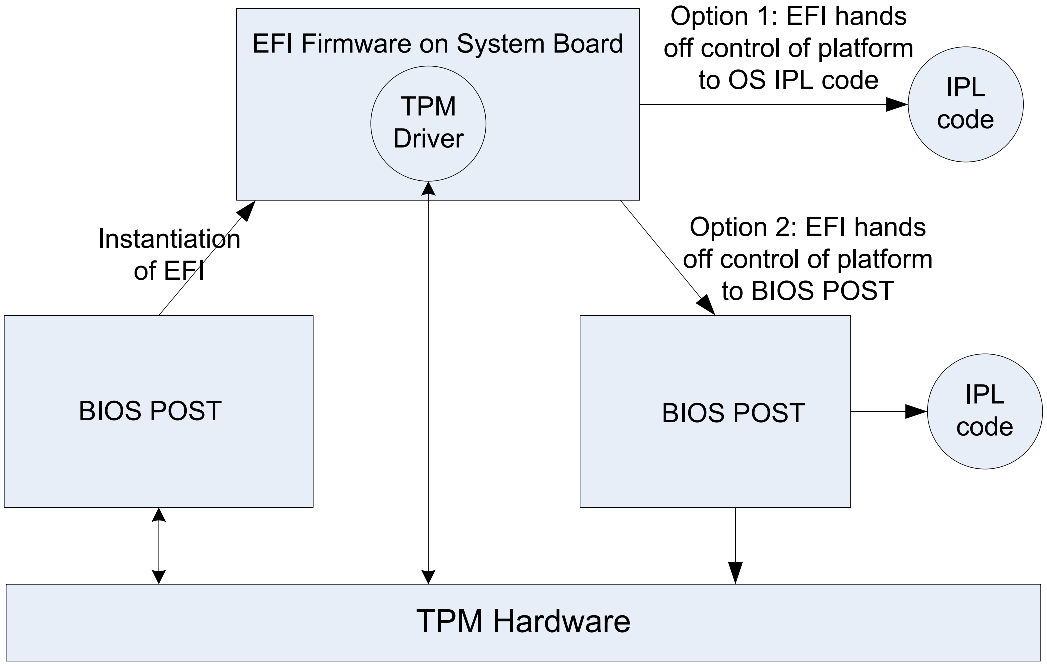
This subsection applies only to EFI layering on top of traditional BIOS.

The TCG EFI Platform Specification must include requirements for the platform architecture shown in Figure 3-4, immediately below. This drawing shows (reading from left-to-right):

- The Platform Boot Process is primarily done by the BIOS Boot Block, using memory-absent TPM driver code to access the TPM, and some variable part of BIOS POST, using memory-present TPM driver code to access the TPM.
- At some point, BIOS POST instantiates EFI and hands off control to the EFI Firmware on the system board to – primarily – boot the OS; the EFI Firmware uses its own TPM driver to access the TPM.
- The EFI Firmware may hand off control of the platform to the OS IPL code without giving control of the platform back to BIOS POST (shown as Option 1 in the diagram). Although this is most often the case, EFI may hand-off control of the platform back to BIOS POST to complete the OS Boot process.

End of informative comment.

Figure 3-4 Example Platform and OS Boot Process with BIOS and EFI


EFI Firmware on System Board
TPM Driver
Option 1: EFI hands off control of platform to OS IPL code
IPL code
Instantiation of EFI
Option 2: EFI hands off control of platform to BIOS POST
BIOS POST
BIOS POST
IPL code
TPM Hardware

4. Measuring PE/COFF Image Files

Start of informative comment:

PE/COFF images can be recorded into PCR's 0, 2, and 4 depending upon the platform deployment model. These images are measured in their memory-mapped store (ROM for execute-in-place and system memory for loaded images).

The measurement must be made prior to the application of any fix-ups or relocations.

An example of a PE/COFF image file is an EFI OS loader, which would be an OS subsystem type of IMAGE_SUBSYSTEM_EFI_APPLICATION, IMAGE_SUBSYSTEM_EFI_BOOT_SERVICE_DRIVER, and IMAGE_SUBSYSTEM_EFI_RUNTIME_DRIVER (as per PE/COFF specification section 3.4.2).

For EFI images, the data structure used in the TCG_PCR_EVENT.event field will include but is not limited to "where" the image came from, namely, its EFI device path, and also the address in memory for the shadowed PE image (see the definition of the EFI_IMAGE_LOAD_EVENT structure, in the Mandatory statements below).

The image in physical memory will ultimately have relocations applied (i.e., fix-ups). The event log will detail the load location of the image, so it will be possible to undo relocations by referring to the on-media image.

Table 4-1, in the Normative statements, identifies which PE/COFF image section types EFI measurement agents must measure and which types of PE/COFF image sections are ignorable. Figure 4-1, immediately below, is part of this Informative comment and is a flow chart that shows a process an EFI measurement agent can use to measure a PE/COFF image.

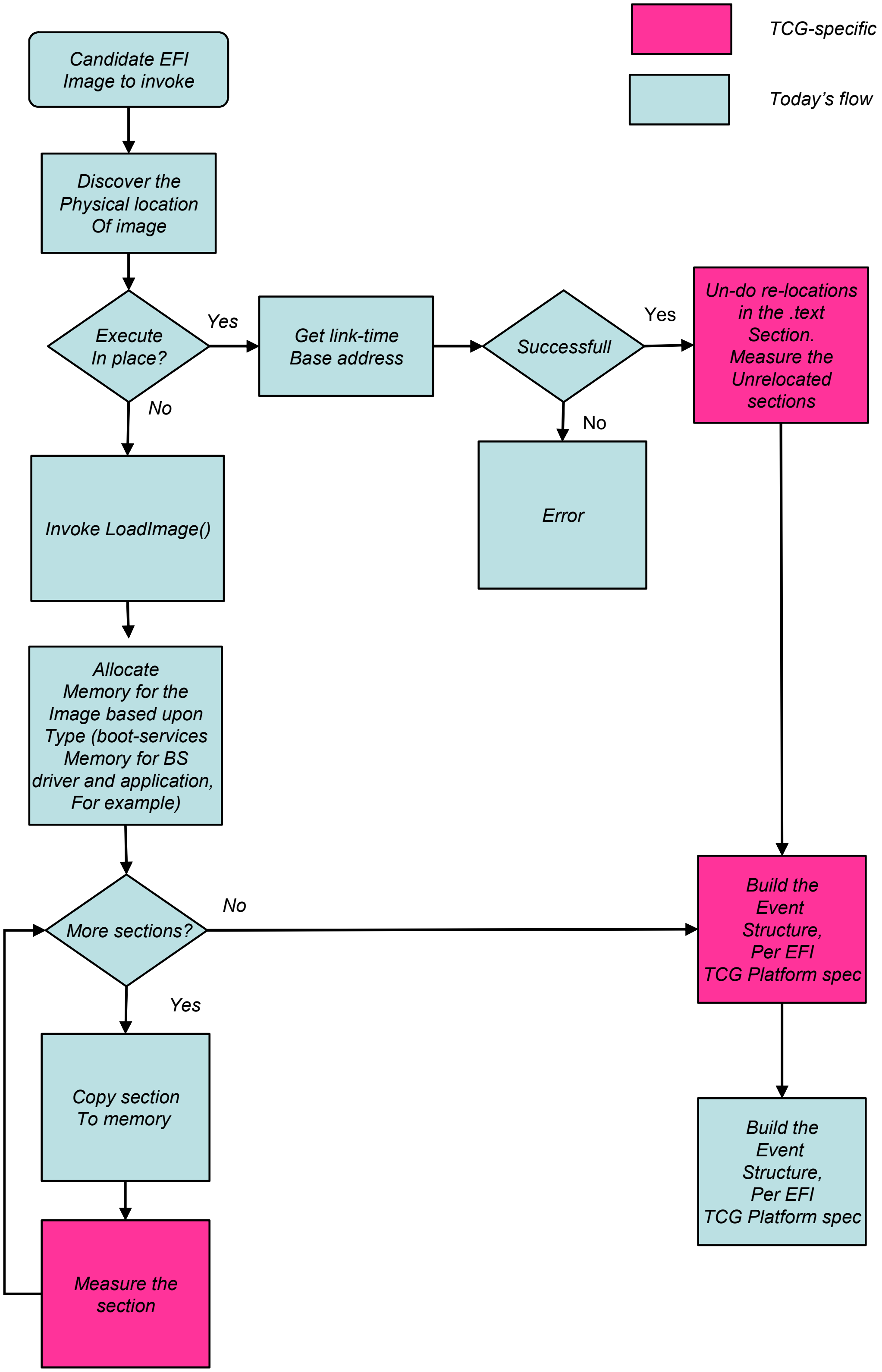
Most EFI images are copied into physical memory from a persistent store. As such the file and the section alignments may differ, and so on. For more information, see the Normative requirements for measuring a PE/COFF image that will be loaded into physical memory, later in this section of the specification.

There is the special case of an execute-in-place (XIP) PE/COFF image (that is, an image that has been re-based to execute directly from the memory-mapped Flash store). This may be relevant for different constructions of the platform initialization code. For more information, see the Normative statements for measuring a PE/COFF image that will execute in place, later in this section of the specification.

What the "measure before applying relocations" described below practically means is that the EFI implementation will perform "LoadImage ( )" actions (e.g., copying PE/COFF to memory, etc), measurement, and then relocation application, and finally, the EFI service "StartImage ( )." As such, EFI implementations of these services MUST punctuate their flow with this measurement action.

End of informative comment.

Figure 4-1 Informative Comment: Flowchart for Process of Measuring a PE/COFF Image


TCG-specific
Today's flow
Candidate EFI Image to invoke
Discover the Physical location Of image
Execute In place?
Yes
No
Get link-time Base address
Successfull
Yes
No
Un-do re-locations in the .text Section. Measure the Unrelocated sections
Error
Invoke LoadImage()
Allocate Memory for the Image based upon Type (boot-services Memory for BS driver and application, For example)
More sections?
No
Yes
Build the Event Structure, Per EFI TCG Platform spec
Copy section To memory
Build the Event Structure, Per EFI TCG Platform spec
Measure the section

Following are the Mandatory requirements for measuring PE/COFF images.

- When measuring a PE/COFF image, the TCG_PCR_EVENT structure Event field MUST contain the EFI_IMAGE_LOAD_EVENT structure defined below.

```
typedef struct {
     EFI_PHYSICAL_ADDRESS   ImageLocationInMemory; // PE/COFF image
     UINTN                  ImageLengthInMemory;
     UINTN                  ImageLinkTimeAddress;  //
     UINTN                  LengthOfDevicePath;    //
     EFI_DEVICE_PATH        DevicePath[1];         // See EFI spec for
                                                   // the encodings.
} EFI_IMAGE_LOAD_EVENT;
```

- To measure a PE/COFF image that will be loaded into physical memory, the EFI Firmware from the Platform Manufacturer MUST:
  1. Copy image from persistent media (that is, from flash or disk) into memory a section at a time.
  2. For PE/COFF header and all .text and .data sections, align these sections on the appropriate section alignment for the architecture, as stipulated in the SectionAlignment field of header. Pad inter-section regions (i.e., in case file alignment and section alignment are not the same) with 0's. The debug, security, and other sections are ignorable (see Table 4-1, below). The measurement should be applied prior to application of relocations by the image loader.
- To measure a PE/COFF image that will execute in place, the EFI Firmware from the Platform Manufacturer MUST:
  1. Discover the link-base address from the PE/COFF header
  2. Undo the relocations in the .text section
  3. Measure the un-relocated sections listed in Table 4-1

Table 4-1 Measurement Agent Behavior for Different PE/COFF Image File Section Types

Table type	Description	Behavior for measurement agent	Rationale
.arch	Alpha architecture information	Ignorable	This is an EFI-only binding, which to date includes EFI and IPF. Alpha does not yet have an EFI binding.
.bss	Uninitialized data	Measured	
.data	Initialized data	Measured	
.edata	Export tables	Ignorable	Not used in EFI
.idata	Import tables	Ignorable	Not used in EFI
.pdata	Exception information	Ignorable	Not used in EFI
.rdata	Read-only initialized data	Measured	
.reloc	Image relocations	Measured	
.rsrc	Resource directory	Ignorable	Not used in EFI
.text	Executable code	Measured	Measure prior to applying relocations, though
.tls	Thread-local storage	Ignorable	Not used in EFI
.xdata	Exception information	Ignorable	Not used in EFI
.debug	Debug information	Ignorable	Production builds typically ignore. See comment in PE/COFF spec section 6.1

5. Measuring Code that Boots an EFI Platform

Start of informative comment:

This section of the EFI Platform Specification contains PCR usage requirements for booting an EFI platform.

End of informative comment.

5.1 Measure Code into PCR [0]

Start of informative comment:

This section contains the PCR [0] measurement requirements for an EFI platform.

In general, measure the following types of code into PCR [0] (as shown in Figure 3-3, General Scheme for PCR Usage on an EFI Platform):

- Platform Firmware from system board ROM that either contains or measures the EFI Boot Services and EFI Run Time Services that are loaded from firmware
- Embedded Option ROMs wholly contained within the Platform Firmware on the system board ROM
- EFI Boot Services in the EFI System Table
- EFI Runtime Services in the EFI System Table

Measure firmware on the system board into PCR [0]. These are code modules known to be distributed by the Host Platform manufacturer. For an EFI platform, this includes but is not limited to measuring:

- Version of the base firmware that either contains or measures the EFI Boot Services (shown as the Platform Init code module of the EFI Boot Manager in Figure 3-1)

End of informative comment.

The following table lists all measurements that MUST be made into PCR [0] plus any optional measurements that MAY be made into PCR [0], along with the type of entry that MUST be made into the Event Log (for more information about types of entries in the Event Log, see Section 7.2, Event Types).

Table 5-1 Required and Optional PCR [0] Measurements

Measuring Agent	Measured Object	Associated Event Type	Required / Optional
Platform firmware from system board ROM that measures code contained in the rest of the platform EFI firmware and that resides in the CRTM. This measuring agent may or may not contain the EFI Boot and Runtime Services. In the case of a CRTM that does not contain EFI Boot and Runtime	It's own version ID	EV_S_CRTM_VERSION	Required

Measuring Agent	Measured Object	Associated Event Type	Required / Optional
Services, this measuring agent MUST then measure into PCR [0] the code that does contain EFI Boot and Runtime Services.			
Platform firmware from system board ROM that measures code which either contains or measures the EFI Boot Services and EFI Run Time Services (on an EFI platform, this may be in the CRTM,)	Platform firmware from system board ROM that either contains or measures the EFI Boot Services and EFI Run Time Services; this measured object is the CRTM code including embedded option ROMs	EV_S_CRTM_CONTENTS	Optional (optional if measured object in CRTM)
Platform firmware from system board ROM that measures code which either contains or measures the EFI Boot Services and EFI Run Time Services (on an EFI platform, this may be in the CRTM)	Platform firmware from system board ROM that either contains or measures the EFI Boot Services and EFI Run Time Services	EV_POST_CODE	Optional (optional if measured object in CRTM)
Platform firmware from system board ROM that measures code which either contains or measures the EFI Boot Services and EFI Run Time Services (on an EFI platform, this may be in the CRTM)	EFI driver embedded in system ROM by the manufacturer	EV_EFI_BOOT_SERVICES_DRIVER	Optional (optional if measured object in CRTM)
Platform firmware from system board ROM that measures code which either contains or measures the EFI Boot Services and EFI Run Time Services (on an EFI platform, this may be in the CRTM)	EFI driver embedded in system ROM by the manufacturer	EV_EFI_RUNTIME_SERVICES_DRIVER	Optional (optional if measured object in CRTM)
CRTM or code measured into PCR [0]	ACPI Static tables	EV_EFI_HANDOFF_TABLES	Required
CRTM or code measured into PCR [0]	Embedded X86 (32-bit and 64-bit) SMM code and the code	EV_S_CRTM_CONTENTS	Optional (optional if measured object in

Measuring Agent	Measured Object	Associated Event Type	Required / Optional
	that sets it up		CRTM)
CRTM or code measured into PCR [0]	BIS code (excluding the BIS certificate)	EV_S_CRTM_CONTENTS	Optional (optional if measured object in CRTM)

5.2 Measure Data into PCR [1]

Start of informative comment:

This section contains the PCR [1] measurement requirements for an EFI platform.

In general, measure into PCR [1] the data that is associated with the code that is measured into PCR [0].

For example, for an EFI server platform, this includes but is not limited to

- SAL system table
- SMBIOS Tables

End of informative comment.

- Platform configuration information that is automatically updated, such as clock registers, and system unique information, such as asset numbers or serial numbers, MUST NOT be measured into PCR [1], or any other PCR.
- The following table lists all measurements that MUST be made into PCR [1] plus any optional measurements that MAY be made into PCR [1], along with the type of entry that MUST be made into the Event Log (for more information about types of entries in the Event Log, see Section 7.2, Event Types).

Table 5-2 Required and Optional PCR [1] Measurements

Measuring Agent	**Measured Object**	**Associated Event Type**	**Required / Optional**
CRTM or code measured into PCR [0]	other tables	EV_EFI_HANDOFF_TABLES	Optional
CRTM or code measured into PCR [0]	EFI Variables that impact system configuration	EV_EFI_VARIABLE_CONFIG	Optional

5.3 Measure Code into PCR [2]

Start of informative comment:

This section contains the PCR [2] measurement requirements for an EFI platform.

In general, measure into PCR [2] firmware code modules added to the Host Platform by a Value Added Retailer (VAR), the platform owner, or some unknown entity. The source of these firmware code modules measured into PCR [2] is not the Host Platform manufacturer.

For an EFI platform, this includes but is not limited to

- EFI Drivers loaded from adapter cards
- EFI Applications loaded from adapter cards (e.g., setup utility for RAID controller)
- EFI Applications loaded from external storage media (system or peripheral Flash for example) via embedded option ROM contained within the EFI firmware on the system board.
- Drivers loaded from HBA's/disk and so on (as shown in Figure 3-3, General Scheme for PCR Usage on an EFI Platform

The measuring agent is always the platform firmware (that is, code measured into PCR [0], even in the case where a measuring agent not measured into PCR [0] "requests" the program load, such as an EFI Driver trying to load another EFI Driver or Application). The EFI Driver will use the EFI LoadImage service. Since measurement occurs between shadowing of PE sections and the application of relocations, only the LoadImage service can call the TCG EFI Protocol function TCG_HashLogExtendEvent at the appropriate time.

End of informative comment.

The following table lists all measurements that MUST be made into PCR [2] plus optional measurements that MAY be made into PCR [2], along with the type of entry that MUST be made into the Event Log (for more information about types of entries in the Event Log, see Section 7.2, Event Types).

Table 5-3 Required and Optional PCR [2] Measurements

Measuring Agent	**Measured Object**	**Associated Event Type**	**Required / Optional**
Platform firmware from system board ROM that either contains or measures EFI Boot Services and EFI Runtime Services	EFI Boot Services Drivers from adapter or loaded by driver in adapter	EV_EFI_BOOT_SERVICES_DRIVER	Required (aspect of any measuring agent that invokes LoadImage on a TCG platform)
Platform firmware from system board ROM that either contains or measures EFI Boot Services and EFI Runtime Services	EFI Boot Services applications from adapter or loaded by driver in adapter, or loaded from external storage via	EV_EFI_BOOT_SERVICES_APPLICATION	Required (aspect of any measuring agent that invokes LoadImage on a TCG platform)

Measuring Agent	Measured Object	Associated Event Type	Required / Optional
	option ROM loader		
Platform firmware from system board ROM that either contains or measures EFI Boot Services and EFI Runtime Services	EFI Runtime drivers from adapter or loaded by driver in adapter	EV_EFI_RUNTIME_SERVICES_DRIVER	Required (aspect of any measuring agent that invokes LoadImage on a TCG platform)

5.4 Measure Data into PCR [3]

Start of informative comment:

This section contains the PCR [3] measurement requirements for an EFI platform.

In general, entities measure into PCR [3] data that is associated with code modules that are measured into PCR [2]. For example, on an EFI platform, this includes but is not limited to EFI variables defined and managed by drivers. EFI Variables are an alternately volatile or non-volatile namespace of persistent values with name/value pairs.

End of informative comment.

The following table lists all measurements that MUST be made into PCR [3] plus any optional measurements that MAY be made into PCR [3], along with the type of entry that MUST be made into the Event Log (for more information about types of entries in the Event Log, see Section 7.2, Event Types).

Table 5-4 Required and Optional PCR [3] Measurements

Measuring Agent	Measured Object	Associated Event Type	Required / Optional
EFI Runtime or Boot service drivers measured into PCR[2]	EFI Variable (see Section 7.5, Measuring EFI Variables, for more information)	EV_EFI_VARIABLE_CONFIG	Optional

6. Measuring Code that Boots an OS

Start of informative comment:

This section of the EFI Platform Specification contains PCR usage requirements for the transition from the EFI Boot Manager to the OS Loader shown in Figure 3-2, above. Note that the term 'EFI OS Loader Load' is a label for a behavior, not necessarily a label for a code module.

Measure code that boots an operating system on an EFI platform into PCR [4] and measure data associated with that code into PCR [5].

End of informative comment.

6.1 Measure Code into PCR [4]

Start of informative comment:

This section contains the PCR [4] measurement requirements for an EFI platform.

In general, entities measure into PCR [4] code modules that could transition the platform from the pre-OS state to the OS-present state. Measure EFI applications into PCR [4]; for an EFI platform, EFI applications include but are not limited to:

- Pre-OS diagnostics
- EFI OS Loader

The measurement values into PCR [4] must be consistent and repeatable across EFI boots.

In the table below, 'measuring entity' means code in the CRTM or code measured into PCR [0] which provides image loading capability and invokes the measurement agent (the 'platform code').

The 'measured object' is a PE/COFF image; see Section 4, Measuring PE/COFF Image Files, for a definition of the methodology for measuring PE/COFF images on a TCG EFI platform.

The measuring agent is always the platform firmware (that is, code measured into PCR [0]). The measuring agent will typically measure the measurement object code as part of the LoadImage ( ) action. Since measurement occurs between shadowing of PE sections and the application of relocations, only the LoadImage service can call the TCG EFI Protocol function TCG_HashLogExtendEvent at the appropriate time. This contrasts with the Conventional BIOS Option ROMs that do not have access to an image load service; these code modules must invoke the measurement INT 1Ah call directly.

This model purposely precludes pre-OS agents from loading and invoking code outside the EFI LoadImage Service. Only the PCR [4] application, such as the EFI OS Loader, can use a different code load and measurement methodology, but the target PCR for this action will be in the range PCR [8] and above and outside the scope of this specification.

End of informative comment.

The measuring entity MUST measure normalized code for all EFI applications into PCR [4].

The code modules measured into PCR [4] MUST be measured in the same order every boot, in absence of a system configuration change.

The following table lists all measurements that MUST be made into PCR [4] plus any optional measurements that MAY be made into PCR [4], along with the type of entry that MUST be made into the Event Log (for more information about types of entries in the Event Log, see Section 7.2, Event Types).

Table 6-1 Required and Optional PCR [4] Measurements

Measuring Agent	**Measured Object**	**Associated Event Type**	**Required / Optional**
EFI Firmware from Platform Manufacturer (code measured into PCR [0]	EFI application (e.g., EFI OS Loader)	EV_EFI_BOOT_SERVICES_APPLICATION	Required
EFI Firmware from Platform Manufacturer (code measured into PCR [0])	EFI application (e.g., HBA system configuration utility, such as RAID setup)	EV_EFI_BOOT_SERVICES_APPLICATION	Required
EFI Firmware from Platform Manufacturer (code measured into PCR [0])	EFI application (e.g., OS loader spawning a separate pre-OS application, such as EfiChkDsk.efi or Diskpart.efi)	EV_EFI_BOOT_SERVICES_APPLICATION	Required

6.2 Measure Data into PCR [5]

Start of informative comment:

This section contains the PCR [5] measurement requirements for an EFI platform.

In general, entities measure into PCR [5] data associated with the code modules measured into PCR [4]. For an EFI platform, this includes but is not limited to

- The Device Path to the code
- The GPT/Partition Table (as shown in Figure 3-3, General Scheme for PCR Usage on an EFI Platform)

PCR [5] contains Boot Policy measurements. In order to record the alternate boot behaviors of an EFI system, the EFI platform firmware will measure the EFI Boot#### variables and BootOrder Variable into PCR [5]. These boot variables are just device paths. A description of the device paths and variables can be found in the EFI Specification.

Additional variables that may impact system behavior beyond the boot options listed above (the source of these additional variables may be the EFI Specification or private variables) may be optionally measured by the EFI boot application. These loader variables shall be measured into PCR [5]. These variables can include, but are not limited to, language code, and so on.

End of informative comment.

The following table lists all measurements that MUST be made into PCR [5] plus any optional measurements that MAY be made into PCR [5], along with the type of entry that MUST be made into the Event Log (for more information about types of entries in the Event Log, see Section 7.2, Event Types).

Table 6-2 Required and Optional PCR [5] Measurements

Measuring Entity	Measured Entity	Associated Event Type	Required / Optional
EFI Firmware from Platform Manufacturer (code measured into PCR [0])	EFI Boot#### variables and BootOrder variable	EV_EFI_VARIABLE_EVENT	Required
EFI Firmware from Platform Manufacturer (code measured into PCR [0])	GPT Table	EV_EFI_GPT_EVENT	Required
EFI application measured into PCR[4] (e.g., OS loader)	EFI variables, either defined in the EFI spec or private, that typically do not change from boot-to-boot and contain system configuration information.	EV_EFI_VARIABLE_EVENT	Optional

7. Event Logging on an EFI Platform

Start of informative comment:

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 useful 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 re-constructing the events that triggered the measurements into the PCRs.

Section 3.1.3 of the TCG EFI Protocol Specification defines the structure that an entry in the Event Log must have, but a copy of that structure is in Section 7.1, for the convenience of the reader.

Table 7-1, in Section 7-2, Event Types, lists all the Event Types for an EFI platform.

End of informative comment.

7.1 Event Log Entry Structure Definition

This section shows a copy of the Event Log entry data structure specified in the TCG EFI Protocol Specification. The data elements must be in big endian format and byte-aligned/packed data structures.

```
typedef struct {
  TCG_PCRINDEX   PCRIndex;  //PCRIndex event extended to
  TCG_EVENTTYPE  EventType;//See Table 7-1, below
  TCG_DIGEST     Digest;    //Value extended into PCRIndex
  UINT32         EventSize;//Size of the event data
  UINT8          Event[1];  //The event data
} TCG_PCR_EVENT;            //Structure to be added to the
                            //Event Log
```

7.2 Event Types

Start of informative comment:

One element in an Event Log entry is TCG_PCR_EVENT.EventType (see section 7.1). Table 7-1, below, 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 an EFI platform.

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

This specification does not speak to measurement of optional tagged events, as defined in the TCG v1.2 PC Client Implementation Specification for Conventional BIOS, section 10.4.2. A composite platform that supports EFI and conventional BIOS may have such entries in the Event Log.

This specification shares the EV_POST_CODE, EV_SEPARATOR, EV_S_CRTM_VERSION and EV_S_CRTM_CONTENTS event types.

End of informative comment.

The value associated with an EFI platform event type must not overlap with any of the event type values specified for other TCG platform architecture specifications including, but not limited to, the *TCG v1.2 PC Client Implementation Specification for Conventional BIOS* and the *TCG IPF*

Architecture Server Specification. The value associated with an EFI platform event type MUST be in the range between 0x80000000 and 0x800000FF, inclusive.

For all EFI platform-specific events, the TCG_PCR_EVENT.EventType field in the Event Log entry must be one of the values listed in Table 7-1.

Table 7-1 EFI Platform Event Types

Label	**Value**	**Description**

EV_POST_CODE	0x01	TCG_PCR_EVENT.PCRIndex = 0
		TCG_PCR_EVENT.digest = SHA-1 Hash of the version string of the CRTM
		TCG_PCR_EVENT.Event[1] = EFI_PLATFORM_FIRMWARE_BLOB structure

EV_SEPARATOR	0x04	TCG_PCR_EVENT.PCRIndex = Any of PCR [0-7]
		TCG_PCR_EVENT.digest = SHA-1 Hash of 0x00000000
		TCG_PCR_EVENT.Event[1] = 0x00000000
		For more information about this event type, see Section 7.4

EV_S_CRTM_VERSION	0x08	TCG_PCR_EVENT.PCRIndex = 0 TCG_PCR_EVENT.digest = SHA-1 Hash of all the code (PE/COFF .text sections) contained in the Host Platform system board firmware. TCG_PCR_EVENT.Event[1] = The version string of the CRTM For more information about this event type, see Section 5.1

EV_S_CRTM_CONTENTS	0x07	TCG_PCR_EVENT.PCRIndex = 0
		TCG_PCR_EVENT.digest = SHA-1 Hash of the version string of the CRTM
		TCG_PCR_EVENT.Event[1] = EFI_PLATFORM_FIRMWARE_BLOB structure
		For more information about this event type, see Section 5.1

EV_EFI_EVENT_BASE	0x80000000	Base value for all EFI platform event type values below

EV_EFI_VARIABLE_DRIVER_CONFIG	EV_EFI_EVENT_BASE + 0x1	TCG_PCR_EVENT.PCRIndex = 1, 3 or 5
		TCG_PCR_EVENT.digest = SHA-1 (Variable data, GUID, Unicode string)

Label	Value	Description
		TCG_PCR_EVENT.Event[1] = EFI_VARIABLE_DATA

EV_EFI_VARIABLE_BOOT	EV_EFI_EVENT_BASE + 0x2	TCG_PCR_EVENT.PCRIndex = 5
		TCG_PCR_EVENT.digest = SHA-1 Hash of the EFI Boot#### variables and the BootOrder variable
		TCG_PCR_EVENT.Event[1] = EFI_VARIABLE_DATA structure

EV_EFI_BOOT_SERVICES_APPLICATION	EV_EFI_EVENT_BASE + 0x3	TCG_PCR_EVENT.PCRIndex = 2, 4
		TCG_PCR_EVENT.digest = SHA-1 Hash of the normalized code from the loaded EFI Boot Services application
		TCG_PCR_EVENT.Event[1] = EFI_IMAGE_LOAD_EVENT structure

EV_EFI_BOOT_SERVICES_DRIVER	EV_EFI_EVENT_BASE + 0x4	TCG_PCR_EVENT.PCRIndex = 0, 2
		TCG_PCR_EVENT.digest = SHA-1 Hash of the normalized code from the loaded EFI Boot Services driver
		TCG_PCR_EVENT.Event[1] = EFI_IMAGE_LOAD_EVENT structure

EV_EFI_RUNTIME_SERVICES_DRIVER	EV_EFI_EVENT_BASE + 0x5	TCG_PCR_EVENT.PCRIndex = 0, 2
		TCG_PCR_EVENT.digest = SHA-1 Hash of the normalized code from the loaded EFI Runtime Services driver
		TCG_PCR_EVENT.Event[1] = EFI_IMAGE_LOAD_EVENT structure

EV_EFI_GPT_EVENT	EV_EFI_EVENT_BASE + 0x6	TCG_PCR_EVENT.PCRIndex = 5
		TCG_PCR_EVENT.digest = SHA-1 Hash of the GPT Table
		TCG_PCR_EVENT.Event[1] = EFI_GPT_DATA structure

EV_EFI_ACTION	EV_EFI_EVENT_BASE + 0x7	TCG_PCR_EVENT.PCRIndex = Depends on the specific string value in the Event [1] field (see Table 7-2). TCG_PCR_EVENT.digest = SHA-1 Hash of Event [1] field TCG_PCR_EVENT.Event [1] = See Table 7.2 for the specific string values that can be used in this field for an EFI platform

EV_EFI_PLATFORM_FIRMWARE_BLOB	EV_EFI_EVENT_BASE + 0x8	TCG_PCR_EVENT.PCRIndex = >1 (non-PE/COFF image load)
		TCG_PCR_EVENT.digest = SHA-1 Hash of all the code (PE/COFF .text sections or other).
		TCG_PCR_EVENT.Event[1] = EFI_PLATFORM_FIRMWARE_BLOB structure
		This allows for non-PE/COFF images

Label	Value	Description
		in PCR [2] or PCR [4]. PCR [0] already has EV_POST_CODE for this type of code.

EV_EFI_HANDOFF_TABLES	EV_EFI_EVENT_BASE + 0x9	TCG_PCR_EVENT.PCRIndex = 1
		TCG_PCR_EVENT.digest = SHA-1 Hash of the system configuration tables which are referenced by entries in EFI_HANDOFF_TABLE_POINTERS
		TCG_PCR_EVENT.Event[1] = EFI_HANDOFF_TABLE_POINTERS

7.3 Event Type EV_EFI_ACTION Strings

Start of informative comment:

The EV_EFI_ACTION event type defined in Table 7-1, above, extends into a specific PCR the measurement of a specific string value that indicates a specific event occurred during the platform or OS boot process. See the EV_EFI_ACTION event type definition in Table 7-1 for the format of the entry that is also added to the Event Log when such an event occurs.

NOTE: The opening and closing quote characters shown in the String Value column of Table 7-2 must not be included in the TCG_PCR_EVENT [1] field and must not be included in the input to the measurement function.

The reason this specification measures OS boot event strings into PCR [5] instead of PCR [4] is that the path or control flow information is typically measured into PCR [5] (for example, EFI Boot Variables) and the identity of code modules is measured into PCR [4]. In Table 7-2 below, for example, the string “Returned EFI NOT SUCCESS” helps to interpret the path of code measurements in the Event Log; for example, the next OS Loader to try after the previous one failed. Each OS Loader code image is extended into PCR [4], but the strings extended into PCR [5], along with the Event Log entry, will say why an additional EFI application was invoked.

End of informative comment.

Table 7-2, below, specifies all the specific string values that can be used for EV_EFI_ACTION on an EFI platform.

Table 7-2 Event Type EV_EFI_ACTION Strings

String Value	Meaning of that String Value	PCR to Extend Hash of String Value Into
“Calling EFI Application from Boot Option”	For now, see Section 7.4	PCR [5]
“Returning from EFI Application from Boot Option”	For now, see Section 7.4	PCR [4]
“Exit Boot Services Invocation“	For now, see Section 7.4	PCR [5]
“Exit Boot Services Returned with Failure”	For now, see Section 7.4	PCR [5]
“Exit Boot Services Returned with Success”	For now, see Section 7.4	PCR [5]

7.4 Measuring OS Boot Events on an EFI Platform

Start of informative comment:

An Event Log enables a challenger to determine the state of trust of the EFI platform and enables software on the EFI platform to reconstruct boot events. The Operating System handoff code needs to fill the Event Log with information about the boot devices used to get to the Operating System. The EFI platform 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.

Note: If an attempt is made to boot a non-EFI OS Loader (e.g., an Int 19h invocation into a Conventional BIOS boot), continue with measurement as defined in the *TCG v1.2 PC Client Implementation Specification for Conventional BIOS*.

End of informative comment.

- The EFI firmware MUST measure the event EV_SEPARATOR into PCR [0-7] once for each EFI platform boot cycle and the measurement of that event MUST draw the line between leaving the pre-Boot environment and entering the post-Boot environment. The data within the event field of the EV_SEPARATOR event must be a 32-bit (double-word) of 0's (that is, 0x00000000).
- The EFI OS Loader Load code MUST measure, into PCR [4], every attempt to load and execute an EFI OS Loader (an EFI application).
- A boot device has an EFI application. The boot variable describes the location of the application. The EFI firmware launches the application. If the application returns back to EFI firmware, this affects the transitive trust chain and MUST be measured.
- Sequence of measuring OS boot events MUST proceed as specified below.

1. Upon selecting a boot device, the EFI firmware measures the event type EV_EFI_ACTION "Calling EFI Application from Boot Option" into PCR [5].
2. Measure EV_SEPARATOR into PCR [0], PCR [1], PCR [2], PCR [3], PCR [4], PCR [5], PCR [6], and PCR [7]. This occurs only once in the flow.
3. Measure GPT with event type equal to EV_EFI_GPT_EVENT with the data structure EFI_GPT_DATA into PCR [5].
4. Measure the selected EFI application code PE/COFF image described by boot variable into PCR [4] using event type = EV_EFI_BOOT_SERVICES_APPLICATION.
5. Measure the boot policy data, if applicable, into PCR [5] using event type = EV_EFI_VARIABLE_BOOT for the respective, active boot variable and BootOrder.
6. Execute EFI application from boot variable.
7. (make 6a) In this optional step, if the executing code from step 6 loads additional applications or drivers prior to successive steps (ie., return or exit boot services), the loaded applications must be measured into PCR[4]. If the load is a driver, it is measured into PCR[4].
8. Measure event type = EV_EFI_ACTION "Returning from EFI Application from Boot Option" into PCR [4].
9. If need to select a next boot device, the EFI firmware MUST jump to step 4.
10. If ExitBootServices ( ) is invoked, then the event type = EV_EFI_ACTION must be measured. The return value of ExitBootServices ( ) must be reflected in a measured event, into PCR [5], of either

“Exit Boot Services Returned with Failure” or “Exit Boot Services Returned with Success”, depending upon the return code from the ExitBootServices ( ) call.

7.5 Measuring Industry-Standard Tables and Data Structures

Start of informative comment:

An EFI platform may support several industry-standard tables and data structures. These include, but are not limited to, ACPI, SMBIOS, and so on.

From the EFI Specification, EFI_CONFIGURATION_TABLE must be:

```
typedef struct {
      EFI_GUID          VendorGuid;
      VOID              *VendorTable;
} EFI_CONFIGURATION_TABLE;
```

End of informative comment.

Event EV_EFI_HANDOFF_TABLES must be used to describe the measurement of industry-standard tables and data structure regions. The event structure must be:

```
typedef struct {
     UINTN                       NumberOfTables;
     EFI_CONFIGURATION_TABLE  TableEntry[1];
} EFI_HANDOFF_TABLE_POINTERS;
```

7.6 EFI_PLATFORM_FIRMWARE_BLOB Structure Definition

- When the CRTM measures the platform firmware from system board ROM that materializes EFI Boot Services and EFI Run Time Services, the CRTM MUST measure the code contained in the Host Platform system board firmware.
- The CRTM MUST also add an entry of event type = EV_S_CRTM_CONTENTS to the Event Log (see Table 7-1). Beyond the CRTM contents, other system board code must also be measured using event type EV_POST_CODE. All of the “code” events must use the EFI_PLATFORM_FORMWARE_BLOB event structure.
- Below is the definition of the EFI_PLATFORM_FIRMWARE_BLOB structure that the CRTM MUST put into the Event Log entry TCG_PCR_EVENT.event[1] field for events EV_POST_CODE and EV_S_CRTM_CONTENTS.

```
typedef struct {
     EFI_PHYSICAL_ADDRESS   BlobBase;
     UINTN                  BlobLength;
} EFI_PLATFORM_FIRMWARE_BLOB;
```

7.7 Measuring EFI Variables

Start of informative comment:

This section defines the event data structures associated with the measurement of EFI variables.

The only EFI variables requiring measurement are those that effect boot policy (that is, where to get the OS Loader). Other EFI variables, such as language, or private variables (that is, GUIDs not defined in the EFI Specification) are measured into the appropriate PCR, depending upon usage (that is, into PCR [1], PCR [3], or PCR [5]); for more information, see sections 5 and 6 of this specification.

End of informative comment.

For Event types = EV_EFI_VARIABLE_CONFIG and EV_EFI_VARIABLE_BOOT, the event log entries share a common data structure, namely EFI_VARIABLE_DATA.. EV_EFI_VARIABLE_CONFIG can be used to designate the measurement of ANY EFI variable, with the exception of the boot variables listed below.

```
typedef struct {
  EFI_GUID     VariableName;
  UINTN        UnicodeNameLength;
  UINTN        VariableDataLength;
  CHAR16       UnicodeName[1];
  INT8         VariableData[1];   // Driver or platform-specific data
} EFI_VARIABLE_DATA;
```

For the boot variable measurement, the data to be recorded are precisely described in the EFI specification. Specifically, there is event type EV_EFI_VARIABLE_BOOT. The EFI firmware must measure BootOrder and EFI Boot#### variables. The event structure for this measurement shared EFI_VARIABLE_DATA.

EFI, unlike conventional BIOS, does not need active partition table flags to dictate which OS loader to choose. The OS loader choice is mediated by the EFI boot options in variables. But the disk partition topology is still important to reflect the system configuration. This configuration information is contained in an EFI_GPT_DATA structure. The event EV_EFI_GPT_EVENT designates the measurement of this on-disk geometry, and the event log data structure is described below.

```
typedef struct {
  EFI_PARTITION_TABLE_HEADER EfiPartitionHeader;
  UINTN                      NumberOfPartitions;
  EFI_PARTITION_ENTRY        Partitions [1];
} EFI_GPT_DATA;
```

7.8 ACPI Table Usage

Start of informative comment:

A system's firmware uses an ACPI table to identify the system's TCG capabilities to the Post-Boot environment. The information in this ACPI table is not guaranteed to be valid until the Host Platform transitions from pre-boot state to post-boot state.

End of informative comment.

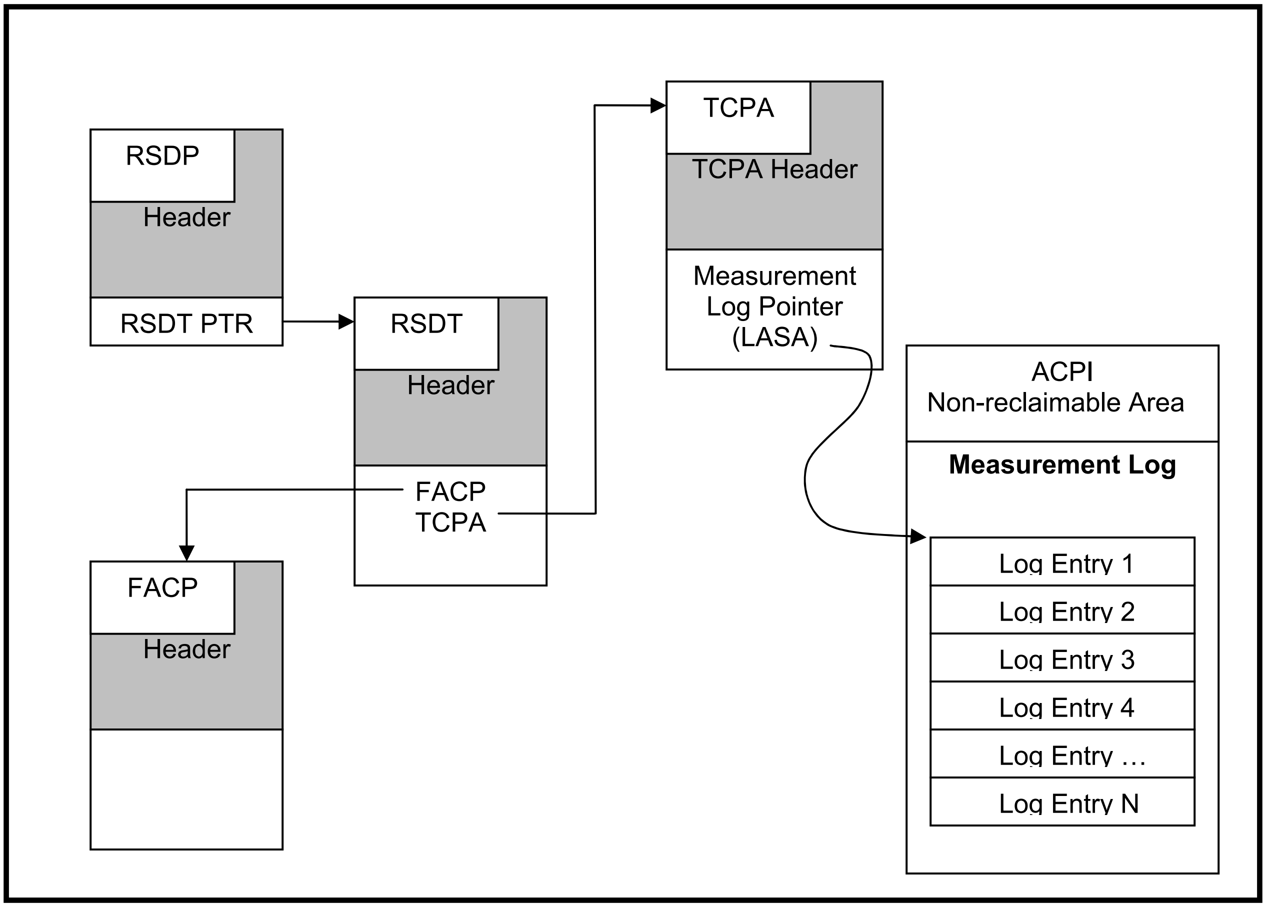


Figure 1 ACPI Table

The ACPI table indicated above as “TCPA” is defined in the TCG ACPI Specification.

8. Other PCR Usages on an EFI Platform

Start of informative comment:

This section contains the requirements for measuring events into PCR [6] and PCR [7].

End of informative comment.

8.1 Power State Transitions, TPM Initialization, and PCR [6]

Start of informative comment:

Measurement agents MUST measure resume from S4 and start up from S5 events into PCR [6]. For more information, see Section 8.3, Definitions and Conditions during Power States, of the *TCG v1.2 PC Client Implementation Specification for BIOS* and/or the comparable section in the *TCG IPF Generic Server Specification*.

End of informative comment.

8.2 Manufacturer-specific Uses of PCR [7]

Start of informative comment:

The Host Platform manufacturer may use PCR [7] for manufacturer-specific applications that need exclusive use of a PCR associated with Locality 0. For more information, see the *TCG v1.2 PC Client Implementation Specification for BIOS* and/or the *TCG IPF Generic Server Specification*.

End of informative comment.

- The Host Platform Manufacturer MAY define the purpose of this PCR.
- User applications MUST NOT use this PCR for sealing or attestation.

9. EFI Firmware Upgrade

Start of informative comment:

EFI firmware upgrades must meet the firmware upgrade requirements of the platform. For example, see the TCG Generic Server Specification.

End of informative comment.

10. TCG Certificates and Verification on an EFI Platform

Start of informative comment:

The Unified EFI Specification describes a methodology for providing credentials in a PE/COFF image.

End of informative comment.