TCG PC Client Platform

Physical Presence Interface Specification

Family "1.2" and "2.0"

Version 1.30 Revision 00.52

July 28, 2015

Contact: admin@trustedcomputinggroup.org

TCG Published

TCG

Disclaimers, Notices, and License Terms

Acknowledgements

The writing of a specification, particularly a security specification, takes many hours for both development and review. This specification is no exception with roughly 70 individuals involved in the process. 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 Amy Nelson, who served as the Chair of the PC Client Working Group, Carey Huscroft, who served as Chair of the Server Working Group, and Ronald Aigner, who was the editor during the development of this specification.

Contributors:

Ben Haidri, Absolute Software Corp

Pierre Chifflier, Agence Nationale de la Securite des Systemes d'Information (ANSSI)

Dean Liberty, AMD

Emily Ratliff, AMD

Gary Simpson, AMD

Gongyuan Zhuang, AMD

Frederick Otumfuor, American Megatrends, Inc.

Goulven Guiheux, AMOSSYS

Frédéric Remi, AMOSSYS

John Mersh, ARM Ltd.

Chris Berg, Atmel

Günter Fuchs, Atmel

James Hallman, Atmel

Mong Sim, Atmel

Todd Slack, Atmel

Ronnie Thomas, Atmel

Dietmar Wippig, BSI

Bill Jacobs, Cisco

Vaden Mohrmann, Dell, Inc.

Amy Nelson, Dell, Inc.

Johan Rahardjo, Dell, Inc.

Steffen Wagner, Fraunhofer AISEC

Benjamin Weggenmann, Fraunhofer AISEC

Andreas Fuchs, Fraunhofer Institute for Secure Information Technology (SIT)

Shiva Dasari, Hewlett-Packard

Thomas Ford, Hewlett-Packard
Russ Herell, Hewlett-Packard
Carey Huscroft, Hewlett-Packard
Tom Laffey, Hewlett-Packard
Lan Wang, Hewlett-Packard
Tim Block, IBM
Hubert Braunwarth, Infineon
Ga-Wai Chin, Infineon
Georg Rankl, Infineon
Johann Schoetz, Infineon
Nicholas Adams, Intel Corporation
Will Arthur, Intel Corporation
Alex Eydelberg, Intel Corporation
Hisaki Ohara, Intel Corporation
Lee Rosenbaum, Intel Corporation
David Song, Intel Corporation
Monty Wiseman, Intel Corporation
Vincent Zimmer, Intel Corporation
Robert Hart, Johns Hopkins University, Applied Physics Lab
James Hoff, Lenovo (United States) INC
Randy Springfield, Lenovo (United States) INC
Ronald Aigner, Microsoft
Paul England, Microsoft
Eugene Samsonov, Microsoft
Rob Spiger, Microsoft
Rahul Verma, Microsoft
David Wooten, Microsoft
Xin Liu, Nationz Technologies Inc.
Dan Morav, Nuvoton Technology
Oren Tanami, Nuvoton Technology
Dick Wilkins, Phoenix Technologies Ltd.
Darren Lasko, Qualcomm Incorporated
Fabien Arrivé, STMicroelectronics
Jean-Luc Blanc, STMicroelectronics
Olivier Collart, STMicroelectronics
Benoit Houyere, STMicroelectronics

Anne-Rose Gratadour, Thales Communications & Security
Tom Brostrom, United States Government
Eugene Myers, United States Government
Andrew Regenscheid, United States Government
Greg Kazmierczak, Wave Systems

Change History

Revision	Date	Description
1.00 / 1.00	5 April, 2007	• Initial release of Version 1.00.
1.10 / 1.00	10 June, 2009	• Added new action 12: Deferred Physical Presence • Added “Submit TPM Operation Request to Pre-OS Environment 2”
1.20 / 1.00	10 February 2011	• Added BIOS TPM Management Flags and corresponding operations • Added return code 3 to “Submit TPM Operation Request to Pre-OS Environment 2” • Added “Get User Confirmation Status for Operation” • Added Operations 21 and 22 that are analogous to 5 and 14 but perform enable and activate first.
1.30 / 0.31	15 September 2013	• Added function required for TPM 2.0
1.30 / 0.32	12 November 2013	• Consolidated multiple “Change hash algorithm operations” into one “Select Boot Log Format”
1.30 / 0.33	20 November 2013	• Adopted TCG document style
1.30 / 0.34	13 December 2013	• Add NoPPIProvision flag for TPM 2.0 and PPI operations to set/reset flag • Add PPI operation to enable/disable TPM 2.0
1.30 / 0.35	20 December 2014	• Formatting changes • Compliance section updated • Clarifications in non-normative text • Pseudo-code formatting updated
1.30 / 0.36	16 January 2014	• Update notes in table 4 • Add NoPPIChangeBootEventLog flag • Adjust SetBootEventLog
1.30 / 0.37	21 January 2014	• Add “Add Boot Event Log” and “Remove Boot Event Log” operations. • Removed “Set Boot Event Log”
1.30 / 0.38	26 March 2014	• Removed “Add Boot Event Log” and “Remove Boot Event Log” operations. • Add “Set Boot Event Log” operation, which takes a bitmask of algorithms • Propose new defaults for flags for server • Added timeout to confirmation request • Renamed NoPPIProvision to NoPPForProvision • Introduced NoPPForTurnOn NoPPForTurnOff • Renamed NoPPIClear to NoPPForClear • Renamed NoPPIChangeBootEventLog to NoPPForChangePCRs • Renamed NoPPIMaintenance to NoPPForMaintenance
1.30 / 0.39	6 June 2014	• Incorporated feedback from rev38 review

Revision	Date	Description
1.30 / 0.40		• Clarified hardware vs. command method • Clarification of use of disabled hierarchies • Expanded protocol sequence of command method • Reverted changes of TPM 1.2 flags • Consistent use of “Persistent Firmware TPM Management Flags” • Split table of PPI operations into table for TPM 1.2 and table for TPM 2.0 • Added user confirmation keys table for systems without keyboard
1.30 / 0.41	2 August 2014	• Incorporated feedback from June 2014 face to face meeting • Operation 14 is Enable + Clear • Fix table references • Clarify operations in Table 1 and 2 • Clarify correlation between operations • Added Section with different scenarios for recommended default values of the Persistent Firmware TPM Management Flags.
1.30 / 0.42	9 August 2014	• Match operation 23 and arguments of ACPI function to TCG log format proposal rev 9.
1.30 / 0.43	13 August 2014	• Address Will Arthur’s comments
1.30 / 0.44	17 October 2014	• Address feedback • Clarified operation 23 argument • Add operation 33 • Changed logic of Persistent Firmware TPM Management Flags for TPM 2.0 to positive logic
1.30 / 0.45	29 October 2014	• Change SHOULD to SHALL • Incorporate feedback from October 2014 face to face meeting • Remove Corrections and Comments section
1.30 / 0.46	7 November 2014	• Incorporate feedback from Shiva Disari
1.30 / 0.47	17 November 2014	• Incorporate feedback from Monty Wiseman o Fix description of SetPCRBanks o Fix description of operation 34
1.30 / 0.48	24 February 2015	• Incorporate comments from first review o Revised grouping of optional commands in Table 2
1.30 / 0.49	25 February 2015	• Feedback from February 2015 Face to Face meeting discussion. • Added operations for Storage management.
1.30 / 0.50	2 March 2015	• Use correct name for Storage spec • Add text for operations 96 to 101 to table 4 • Fix text for operations 25 – 32 in table 4
1.30 / 0.51	13 March 2015	• Minor change in confirmation text of operation 23
1.30 / 0.52	16 March 2015	• Corrections to text in table 4

Contents

List of Tables

List of Figures

1 Scope

This specification defines an interface between an operating system and the firmware to manage the configuration of a TPM and, if required, initiate TPM related operations. The specification gives suggestions on UI wording for interactions with users of a system, if UI interaction is required.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

A TPM in adherence to either:

1. The TPM Main Specification Level 2 Version 1.2

   or

2. The TPM library Specification, Family “2.0”

A PC Client platform in adherence to either:

1. The TCG PC Client Specific Implementation Specification for Conventional BIOS Version 1.20

   or

2. The TCG EFI Platform Specification Version 1.20 **and** the TCG EFI Protocol Specification Version 1.20

   or

3. The TCG PC Client Platform TPM Profile (PTP) Specification Version 2.0

Secure Storage Platforms supported by this specification should adhere to:

1. TCG Storage Feature Set: Block SID Authentication Specification

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
buffer

data structure used for transport to and from the TPM

3.2
CLEAR

bit with a value of zero (0), or the action of causing a bit to have a value of zero (0)

3.3
command

values sent to the TPM to indicate the operation to be performed

3.4
octet

eight bits of data

Note:

On most modern computers, this is the smallest addressable unit of data.

3.5
platform operator

The person interacting with the machine.

3.6
response

values returned by the TPM when it completes processing of a command

3.7
SET

bit with a value of one (1), or the action of causing a bit to have a value of one (1)

3.8
TPM Device Reset

resetting of TPM internal state due to _TPM_Init

3.9
Trusted Platform Module
TPM

implementation compliant with the *TPM Main Specification; Family 1.2* or *TPM Library Specification; Family 2.0; Level 00; Revision 99* or later

3.10
firmware

the code that is executed first when a PC platform is powered on; usually the BIOS or UEFI

4 Abbreviated Terms

For the purposes of this document, the following abbreviations apply.

Abbreviation	Description
TPM	Prefix for an indication passed from the system interface of the TPM to a Protected Capability defined in this specification
_DSM	Device Specific Method
ACPI	Advanced Configuration and Power Interface
AK	Accept Key
BIOS	Basic Input/Output System
CA	Certificate Authority
CRTM	Core Root of Trust for Measurement
D-RTM	dynamic RTM
EK	Endorsement Key
EPS	Endorsement Primary Seed
MSb	Most Significant bit
MSO	Most Significant Octet
NVRAM	Non-Volatile Random Access Memory
PCR	platform configuration register(s)
PM	Platform Manufacturer
POST	Power on Self-Test
PP	Physical Presence
PPI	Physical Presence Interface
RTM	Root of Trust for Measurement
S-RTM	static RTM
SHA	Secure Hash Algorithm
SHA1	secure hash algorithm with a 20 byte (160 bits) digest value
SHA256	secure hash algorithm with a 32 byte (256 bits) digest value
SHA384	secure hash algorithm with a 48 byte (384 bits) digest value
SHA512	secure hash algorithm with a 64 byte (512 bits) digest value
TCG	Trusted Computing Group
TPM	Trusted Platform Module
TPM2_	Prefix for a command defined in this specification
UEFI	Unified Extensible Firmware Interface
UI	User Interface

5 Compliance

To be compliant with this specification, a system providing a Physical Presence Interface for TPM 1.2 or TPM 2.0 SHALL implement the mandatory operations for the respective family of TPM devices. A compliant system MAY implement the optional operations for the respective family of TPM devices. A compliant system MUST implement the operations that it provides as described in this specification.

6 Conventions

Text in this specification on white background is normative. Informative text or notes are formatted with grey background.

6.1 Bit and Octet Numbering and Order

An integer value is considered to be an array of one or more octets. The octet at offset zero within the array is the most significant octet (MSO) of the integer. Bit number 0 of that integer is its least significant bit in the last octet in the array.

Example:

A 32-bit integer is an array of four octets; the MSO is at offset [0], and the most significant bit is bit number 31. Bit zero of this 32-bit integer is the least significant bit in the octet at offset [3] in the array.

Note:

Array indexing is zero-based.

Note:

This definition does not match the “network bit order” used in many IETF documents, such as RFC 4034. In those documents, the most significant bit of a datum has the lowest bit number. It is conventional practice to send that bit first when using a serial network protocol, and the bits are numbered in the order they are sent. This specification numbers bits according to their corresponding power of two within a datum. This numbering corresponds to the normal convention for bit numbering in hardware registers that hold integer values rather than fixed-point numbers.

The first listed member of a structure is at the lowest offset within the structure and the last listed member is at the highest offset within the structure.

For a character string (letters delimited by “”), the first character of the string contains the MSO.

6.2 Numbers

Numbers are decimal unless a different radix is indicated.

Unless the number appears in a table intended to be machine readable, the radix is a subscript following the digits of the number. Only radix values of 2 and 16 are used in this specification.

Radix 16 (hexadecimal) numbers have a space separator between groups of two hexadecimal digits.

Example:

40 FF 12 34_{16}

Radix 2 (binary) numbers use a space separator between groups of four binary digits.

Example:

$0100\ 1110\ 0001_2$

For numbers using a binary radix, the number of digits indicates the number of bits in the representation.

Examples:

20_{16} is a hexadecimal number that contains exactly 8 bits and has a decimal value of 32.

$10\ 0000_2$ is a binary number that contains exactly 6 bits and has a decimal value of 32.

$0\ 20_{16}$ is a hexadecimal number that contains exactly 12 bits and has a decimal value of 32.

A number in a machine-readable table may use the “0x” prefix to denote a base 16 number. In this format, the number of digits is not always indicative of the number of bits in the representation.

Example:

0x20 is a hexadecimal number with a value of 32, and the number of bits is determined by the context.

7 TPM Management Overview

Physical Presence

Physical Presence is a form of authorization to perform certain TPM functions. This authorization normally comes from the platform operator. For TPM 1.2 devices the functions requiring physical presence are those that perform provisioning and re-provisioning operations such as allowing ownership and clearing ownership when the current owner authorization value is unknown. For TPM 2.0 devices physical presence is associated with operations that require platform authorization but are initiated by the OS. Usually, platform authorization can only be given by firmware, not by the OS.

There are two methods to provide physical presence authorization in a PC Client environment: the command method and the hardware method. These two methods are defined for TPM 1.2 devices in the TPM 1.2 Main Specification and for TPM 2.0 devices in the PC Client Platform TPM Profile Specification.

Hardware method to assert physical presence

An example of an implementation of the hardware method is a button on the front of the platform wired to a pin on the TPM. Pressing the button causes the pin to change the polarity and would cause the TPM to set its internal physical-presence flag. Using this hardware method, commands requiring the indication of physical presence could be executed at any time (in the pre-OS environment or from the OS environment) as long as the button were pressed and, for TPM 2.0, platform authorization is available. Implementation of the hardware method is outside the scope of this specification.

Physical Presence as discussed in the TPM Library specification for TPM 2.0, Part 1, refers to the hardware method. TPM 2.0 commands that should require a hardware assertion of physical presence have to be on a list of commands – the “PP command list”. TPM 2.0 commands can be added or removed from the list using the TPM2_PP_Commands command. For instance, if a system wants to assert clearing the TPM with the hardware method, firmware has to ensure that the TPM2_Clear command is added to the PP command list. Note that the TPM2_PP_Commands command requires physical presence itself.

Command method to assert physical presence

Providing a wire along with a button or switch to the outside of the platform is not feasible in some cases due to cost, form factor, usage, or other issues. For this reason, a second method of asserting physical presence called the “command method” is defined.

For the command method, firmware presents a user interface (UI) to explain the operation that has been requested. A physically present user confirms or rejects the operation by pressing a key on the keyboard. The command method then authorizes the TPM command to be sent to the TPM. No further check for physical presence is done in the TPM. The TPM performs the TPM operation if the user confirmed it through the UI.

One of the properties required of the indication of physical presence is, it must be done by the platform operator, typically the user, when physically present at the platform. This requires the command method to be restricted to only be available while the platform state can provide this assurance. On a PC Client, this state usually exists during the boot strapping of the platform prior to the availability of a network stack or untrusted software – specifically during the early stages of a Static Root of Trust for Measurement (S-RTM). These commands are therefore available only after TPM_Startup (for TPM 1.2) or TPM2_Startup (for TPM 2.0) and use of these commands must be disabled once the S-RTM determines the user has not provided an indication of physical presence. TPM 2.0 devices only require platform authorization to execute privileged commands. The platform authorization should only be present in a trusted environment, for instance in the firmware during boot. It is desirable to request user confirmation

for some of the operations requiring platform authorization, especially those that are initiated through PPI operations.

This specification is targeted at platforms implementing this "command method" of providing an indication of physical presence.

Enhancements added in version 1.2 of this specification permit the operator to set Persistent Firmware TPM Management Flags that allow certain actions to occur without an operator being physically present. The intent is to allow a Platform Manufacturer or an IT department to configure a platform once, so the operating system is able to perform TPM management actions without operator involvement in the future.

TPM Management Authorization

The TPM may be used in a variety of scenarios on a PC Client Platform to provide trusted computing. Depending on the scenario, different authorization schemes are appropriate to manage TPM actions like enabling, disabling, clearing the TPM or updating the TPM firmware.

This specification defines the two authorization mechanisms below to control the ability to perform TPM actions:

- Physical Presence Control – A platform operator physically present at the platform must press a key to authorize an action. A benefit of this authorization method is that software cannot manage the TPM without operator involvement and unevaluated software may be permitted to run without risk that the TPM state will change without operator approval. A drawback is an operator must be physically present at the computer to manage the TPM, inhibiting remote IT administration of the TPM state. This mechanism may be appropriate for a high security scenario where the platform operator has enough knowledge to participate in TPM management.
- Software Control – An OS or other software controlling the system manages the TPM without the need for a physically present operator to approve changes. A benefit is an operator does not need to be physically present to manage the TPM. A drawback is software may change the TPM state without operator involvement, so unevaluated software must be prevented from controlling the system. This mechanism may be appropriate for a managed desktop scenario where OS enforced access controls strictly regulate access to TPM management APIs and TPM management is performed remotely by an IT department.

Different TPM management actions have different consequences. For example, disabling the TPM can be reversed by enabling the TPM. However, clearing the TPM cannot be reversed. For this reason, this specification allows different aspects of TPM management to be configured to independently use one of the authorization mechanisms described above. The different aspects defined in this specification are Provisioning (and re-provisioning), Clearing, and Maintenance of the TPM.

Enabling TPM 1.2 Devices

If a TPM 1.2 device is in a disabled and deactivated state there exist multiple PPI operations to enable and active the TPM device. An end-user can enable the TPM device and then take ownership of the device. For example if the TPM 1.2 device is disabled and deactivated, the end user can invoke PPI operation 22 to enable, activate, and clear the TPM. After the TPM is cleared, the user can take ownership of the device.

Enabling TPM 2.0 Devices

A TPM 2.0 device has no notion of the TPM 1.2 semantic of disabled or deactivated states. A TPM 2.0 device may have some of its hierarchies disabled. To present similar PPI operations for TPM 1.2 and TPM 2.0 devices, some PPI operations to enable and activate a TPM are mapped to TPM operations that enable or disable hierarchies. This mimics a similar sematic.

To disable TPM 2.0 hierarchies, firmware can CLEAR ehEnable, shEnable, phEnable, and phEnableNV using the TPM2_HierarchyControl command. This specification provides operations to CLEAR ehEnable and shEnable but not phEnable nor phEnableNV. Because hierarchies are enabled on every boot, firmware has to perform the CLEAR on every boot. It has to keep track of the current enablement state of the TPM 2.0 device across boots.

Other mechanisms to deactivate or disable a TPM 2.0 device are outside of the scope of this specification.

Other TPM Management Authorization

Additional platform-manufacturer-specific authorization mechanisms not defined in this specification may be provided on platforms. Examples are firmware configuration utilities or remote system management tools that enable or disable the TPM with authorization controlled by a firmware password. These implementations are vendor specific.

Storage Management

Enhancements added in version 1.30 of this spec added operations that allow the management of secure storage devices. This specification defines the operation IDs and the associated Persistent Firmware Storage Management Flags. These operations do not change TPM state but manipulate the state of the secure storage device. This specification does not recommend default values for the Persistent Firmware Storage Management Flags. These are defined in the TCG Storage Feature Set: Block SID Authentication Specification.

8 Physical Presence Interface

The Physical Presence Interface utilizes the industry-standard Advanced Configuration and Power Interface (ACPI) to provide a communication mechanism between the OS and the firmware, enabling the OS and the firmware to cooperate to provide a simple and straightforward platform user experience for administering the TPM without sacrificing security.

For TPM 1.2 devices, the Physical Presence Interface was designed under the assumption that TPM commands requiring physical presence should only be executable in the pre-OS environment. Given this constraint, the Physical Presence Interface eases administration by minimizing the need to understand and configure a platform's firmware.

The implementation of the Physical Presence Interface specification is recommended for platforms that support the command method of asserting physical presence (as indicated by the physicalPresenceCMDEnable permanent flag of the TPM 1.2).

For TPM 2.0 devices, the Physical Presence Interface is designed for TPM commands requiring the platform authorization and which are initiated from an OS, e.g. commands that change the TPM state. A TPM 2.0 device may also implement a physical presence pin so that physical presence can be required in conjunction with platform authorization.

In order to minimize the platform storage required to implement this Interface, the OS is restricted to requesting the execution of at most one operation at a time. An operation is defined as one or more TPM commands that require physical presence authorization. By enumerating the most likely sequences of TPM commands and mapping them to unique operations, most standard functionality involving physical presence can be carried out within one restart of the OS. For detailed definitions of the TPM commands that are associated with each operation identified in the Interface, see Part 3 of the v1.2 TCG Main specification or Part 3 of the TPM Library Specification, Family "2.0".

Some operations permitted through this Physical Presence Interface do not change the state of the TPM, but instead change Persistent Firmware TPM Management Flags that control whether a physically present user must confirm future physical presence operations.

For more information on ACPI, see the materials available for download at http://uefi.org/acpi/specs. Refer to Section 9.14.1 of the ACPI 5.0 spec for information on the _DSM control method object. Refer to Section 19.2.5 of the ACPI 5.0 spec for information about data types. For example, the "Integer" data type is a 64-bit little-endian unsigned integer (with only the lower 32 bits having meaning) and the "String" type is a null-terminated ASCII string.

Defined Uses:

The primary use case of the Physical Presence Interface for command method is as follows:

1. Within the OS environment, the user requests an operation that requires physical presence or platform authorization. For instance, the user request to clear the TPM.
2. The OS informs the user via its user interface that a platform-specific procedure (e.g. reboot or shutdown and user confirmation) must take place to successfully execute the operation. For instance, a dialog box may tell the user to reboot to clear the TPM.
3. The OS communicates the requested operation to the firmware through ACPI. The platform's ACPI handler stores the requested operation in a location accessible to the pre-OS environment (such as, CMOS, Flash, TPM NV Index, etc.). The location does not have to be protected, as explained below (Security Implications). For instance, the OS uses the TPM device's _DSM method to invoke the PPI operation to request clearing the TPM.
4. The OS reboots or shuts down the platform; this is a user-visible event and transitions the platform to the CRTM-initiated pre-OS environment.
5. The firmware reads the OS's operation request from its stored location.

6. Depending on the operation and the state of Persistent Firmware TPM Management Flags, the firmware automatically approves the operation or asks a physically present user for confirmation. For instance, text is presented to the user to press a key to confirm clearing the TPM.
   Note: Firmware may have to measure and then execute a Platform Manufacturer (PM) pre-boot environment utility to prompt the user for confirmation.
7. If the operation is approved, depending on the type of operation, the firmware executes TPM commands that carry out the requested operation or change Persistent Firmware TPM Management Flags. For instance, the firmware executes the TPM_ForceClear or TPM2_Clear command.
8. The firmware clears the storage location containing the operation request.
9. The firmware stores the response of the last requested operation. When the OS loads it can query the response via an ACPI method. The response can be a success code if the operation was confirmed and executed, a failure code if the user failed to issue a confirmation, or a TPM command error.
10. The OS loads, determines the response of the requested operation, and takes action as necessary.

Table 1 and Table 2 identify the set of operations available to the OS.

Security Implications:

When Persistent Firmware TPM Management Flags or Persistent Firmware Storage Management Flags require a physically present user to approve an operation, it is imperative that the pre-OS environment verifies the physical presence of the user and confirms that the physically present user in fact approves the execution of an operation. This confirmation is achieved via a pre-OS dialog. The dialog should be implemented such that the user can understand at a high level the security implications of the operation and must actively choose to execute the operation (e.g. the default should not be to confirm the operation). Table 4 has recommended user confirmation text for each operation. If the request is rejected by the physically-present user, the pre-OS environment must clear the request so that the user is not prompted to confirm again on the next reboot. It is recommended to present the dialog only for a limited amount of time. This allows the machine to continue boot if the user is not present. An implementation may choose to use an infinite timeout.

Note: It is possible that malicious software repeatedly invokes the ACPI call to submit operation requests. It is advisable for the PM to prevent that such abuse will cause irreparable damage to the platform (e.g. by burning out flash memory used to store the requests).

The location(s) used by the firmware to store the pending operation (including any necessary platform reset) is not required to be trusted or secure. This is because if the operation is not performed, or performed incorrectly, the requested changes to the TPM state will not take place. This does not introduce a vulnerability to the user.

If a requested operation requires multiple platform resets, firmware stores the assertion of the indication of physical presence in a trusted location between resets.

Storage Implications for passing information between the OS and the firmware:

The minimum platform storage for operation requests is sufficient for the operations defined in this specification. Vendor specific operations may require additional storage space.

The minimum platform storage required to implement the information exchange between the OS and the firmware (and vice-versa) for this Interface for TPM 1.2 is as follows:

5 bits – Stores the pending operation request submitted by the OS

5 bits – Stores the most recent operation request acted upon by the firmware

7 bits – Stores the most recent operation response acted upon by the firmware

The data above assumes that no vendor-specific TPM 1.2 operations or error codes are implemented and there exist less than 128 possible operation responses. As of the time of this writing, there exist 105 possible TPM 1.2 operation responses consisting of 99 TPM fatal errors, 4 TPM non-fatal errors, and 2 additional Physical Presence Interface-specific responses (User Abort and firmware Failure).

A TPM 2.0 device encodes the operation response in 12 bits. The minimum platform storage required to implement the information exchange between the OS and the firmware for this Interface and for TPM 2.0 is as follows:

5 bits – Stores the pending operation request submitted by the OS

5 bits – Stores the most recent operation request acted upon by the firmware

8 bits – Stores the optional argument to operation request

12 bits – Stores the most recent operation response acted upon by the firmware

Conditional Physically Present User Confirmation:

Several Persistent Firmware TPM Management Flags control whether a physically present user must confirm each physical presence operation. Refer to Annex A for a description of different scenarios and recommended defaults for the flags.

Persistent Firmware TPM Management Flag Storage

The Persistent Firmware TPM Management Flags control whether physical presence is required for TPM management actions. By design physical presence is required to change the flags to not require physical presence confirmation in the future. Care should be taken by Platform Manufacturers to prevent unintended modification of the flags. This can be achieved by storing the flags in trusted storage. For example, the flags should not be stored in NV RAM locations able to be manipulated by malicious software.

One mechanism to protect the Persistent Firmware TPM Management Flags is to store the flags in the TPM's NV RAM. To prevent entities other than the firmware from modifying the flags, write permission for the index may be configured to require physical presence for which the firmware controls assertion. For TPM 1.2 the "D" bit attribute should be set so the values persist even if the TPM is cleared. This specification has reserved index 0x50010000 for Platform Manufacturers if the TPM NV RAM used. The Platform Manufacturer should carefully provision appropriate default values for the Persistent Firmware TPM Management Flags.

Persistent Firmware Storage Management Flag Storage

Similarly to Persistent Firmware TPM Management Flags the Persistent Firmware Storage Management Flags should be protected by the Platform Manufacturer to prevent unintended modification of the flags.

Transitioning to the pre-OS environment from the OS – Restart versus Shutdown

Platform manufacturers may require the OS to restart, shutdown, or perform some other vendor specific action for the firmware to fulfill a requested operation during the next boot of the platform. However, to permit operations without any physical presence when the Persistent Firmware TPM Management Flags are set to TRUE, using restart instead of shutdown is necessary to permit remote management of a platform in absence of a physically present user to turn the platform back on.

Additional Notes for TPM 1.2:

This Interface assumes that the firmware is able to execute the TSC_PhysicalPresence command to configure physical presence assertion flags as well as the TPM commands that require physical presence authorization (e.g. TPM_PhysicalEnable).

For some TPM implementations, execution of these commands prior to issuing the TPM_ContinueSelfTest command causes an error of TPM_NEEDS_SELFTEST or TPM_DOING_SELFTEST to be returned. In these cases, the firmware should ensure that TPM_ContinueSelfTest is issued appropriately so that the self-test errors are not propagated to the OS.

After the user confirms the physical presence operation, the firmware may need one or more additional reboot cycles to carry out the user's request, as well as 1 bit of additional platform storage to track the extra boot cycle. The additional reboot is needed when executing the command TPM_PhysicalSetDeactivated because the TPM requires an additional boot cycle for an activation or deactivation to take effect (see main spec related to TPM_PERMANENT_FLAGS.deactivated and TPM_STCLEAR_FLAGS.deactivated for more information). Additional reboots may need to take place in the middle of some multi-command physical presence operations. TPM 1.2 operations that may need additional reboot cycle:

Operation 3:	Activate
Operation 4:	Deactivate
Operation 5:	Clear
Operation 6:	Enable + Activate
Operation 7:	Deactivate + Disable
Operation 10:	Enable + Activate + SetOwnerInstall_True
Operation 11:	SetOwnerInstall_False + Deactivate + Disable
Operation 12:	Deferred Physical Presence unownedFieldUpgrade
Operation 14:	Clear + Enable + Activate
Operation 21:	Enable + Activate + Clear
Operation 22:	Enable + Activate + Clear + Enable + Activate

When the firmware carries out operations that require an additional reboot cycle, the user should not be prompted again for confirmation. When this occurs, the only indication to the user of the additional reboot cycle is the reappearance of the firmware splash screen. For the OS user, there is still only one restart.

Method for performing TPM 1.2 field upgrade without using physical presence:

A field upgrade of an un-owned TPM may be accomplished without physical presence by taking ownership and performing the field upgrade. Therefore, the default for the NoPPIMaintenance flag is False.

Additional Notes for TPM 2.0:

The TPM 2.0 requires platform authorization for some of the TPM commands mentioned in this specification. This interface assumes that the firmware has access to the platform authorization value when the requested TPM commands are invoked.

After the user confirms the physical presence operation, the firmware may need one or more additional reboot cycles to carry out the user's request, as well as 1 bit of additional platform storage to track the extra boot cycle. When the firmware carries out operations that require an additional reboot cycle, the user should not be prompted again for confirmation. When this

occurs, the only indication to the user of the additional reboot cycle is the reappearance of the firmware splash screen. For the OS user, there is still only one restart.

Platform Implementation

This Interface is designed for platforms implementing the command method for indication of physical presence. Platforms are allowed to implement hardware and command methods in which case the user may choose the one most convenient. If a Platform Manufacturer provides a utility, the utility prompting the user to provide the indication of physical presence should be aware of the method(s) implemented on the platform and prompt the user accordingly. This information can be found by querying the capabilities of a TPM 1.2 (e.g. calling TPM_GetCapability and checking the PhysicalPresenceHwEnable and PhysicalPresenceCMDEnable flags). For a TPM 2.0 device the TPM2_GetCapability(TPM_CAP_PP_COMMANDS) command returns a list of commands currently requiring hardware method to assert physical presence.

Though this specification mentions the hardware method of asserting physical presence, it does not specify how an OS would interact with the hardware method. For example, TPM 1.2 management actions such as changing from deactivated to activated (which requires a platform reset to occur) with an OS present are not specified. In the absence of additional specifications, implementations using a hardware method may want to emulate the “command method” of asserting physical presence to accommodate the same OS interface for TPM management.

ACPI Command Status:

The command status field within each command indicates whether the command is mandatory, optional or deprecated.

Mandatory: If a platform implements this version of this specification it must implement the indicated command.

Optional: If a platform implements this version of this specification it may implement the indicated command.

Deprecated: This command may be removed from future versions of this specification. Software should not use it. The command is still present in the specification to support older software.

Deprecated and Mandatory: This command may be removed from future versions of this specification. Software should not use it. The command is still present in the specification to support older software. If a platform implements this version of the specification it must implement the indicated command.

8.1 ACPI Functions

1. If the Platform Manufacturer implements an interface between the OS and the pre-OS environment for supporting the execution of TPM operations requiring physical presence, the firmware MUST support the Physical Presence Interface as defined in this specification.
2. These ACPI functions reside in the _DSM control method object. The following UUID function identifier MUST be used exclusively for the Physical Presence Interface:

   `3DDDFAA6-361B-4eb4-A424-8D10089D1653`.
3. Functions start at index 1, since function 0 is the standard _DSM query function.

Below is the list of ACPI Physical Presence functions that MUST be implemented:

8.1.1 Get Physical Presence Interface Version

Command Status:

Mandatory

Input Arguments:

Arg0 (Buffer): UUID = 3DDDFAA6-361B-4eb4-A424-8D10089D1653

Arg1 (Integer): Revision ID = 1

Arg2 (Integer): Function Index = 1

Arg3 (Package): Arguments = Empty Package

Returns:

Type: String

Purpose: Supported Physical Presence Interface revision

Functional Behavior:

This function returns the version of the Physical Presence Interface supported by the platform.

For this version of this specification, the return value MUST be “1.3” as a NULL terminated ASCII string

Note that the Physical Presence Interface revision number does not correspond with the Revision ID parameter (Arg1).

Examples:

A return value of “1.0” indicates that the Interface is compatible with Physical Presence Interface specification v1.0.

A return value of “1.1” indicates that the Interface is compatible with Physical Presence Interface specification v1.1.

A return value of “1.2” indicates that the Interface is compatible with Physical Presence Interface specification v1.2.

A return value of “1.3” indicates that the Interface is compatible with Physical Presence Interface specification v1.3.

8.1.2 Submit TPM Operation Request to Pre-OS Environment

Command Status:

Deprecated and Mandatory

Input Arguments:

Arg0 (Buffer): UUID = 3DDDFAA6-361B-4eb4-A424-8D10089D1653

Arg1 (Integer): Revision ID = 1

Arg2 (Integer): Function Index = 2

Arg3 (Package): Arguments = Package --

Type: Integer

Purpose: Operation Value of the Request

Description: [see Table 1 and Table 2]

Returns:

Type: Integer

Purpose: Function Return Code

Description:

0: Success

1: Operation Value of the Request Not Supported

2: General Failure

Functional Behavior:

This function allows the OS to submit a request for an operation to be executed in the pre-OS environment (see Table 1 and Table 2 for a list of defined operations). This request is the only input from the OS to the pre-OS environment.

If 0 is returned, the requested operation can be read and acted upon by the firmware once the transition to the pre-OS environment takes place (e.g. after the platform has restarted). The OS expects that the pre-OS environment verifies physical presence and confirms that the physically-present user in fact requested the execution of the operation.

If 1 is returned, the firmware does not support the operation request. For example, the implementation of the operation may be optional or vendor-specific.

If 2 is returned, the firmware is otherwise unable to read and act upon the request. For example, platform-specific security protections may exist to prevent burnout of the storage location for the operation.

The OS may call this function or “Submit TPM Operation Request to Pre-OS Environment 2” multiple times before transitioning to the pre-OS environment. However, only the last submitted request is valid. The OS may submit an operation request of value 0 to clear any previous requests.

Many as-built operating systems and hardware implementations have implemented the Submit TPM Operation Request to Pre-OS Environment function differently than this specification with respect to the Arg3 argument. Due to the large number of deployed systems in the marketplace it is recommended Platform Manufacturers and operating system vendors consider compatibility with the following scenarios:

Some operating systems incorrectly pass a buffer of an Integer instead of a Package containing an Integer for Arg3. The ACPI Machine Language Interpreter on those operating systems treats the Arg3 argument type as a buffer when executing ACPI Machine Language instructions.

Some existing hardware platforms in the marketplace are dependent on this operating system mistake and explicitly convert the buffer of an Integer to an Integer using the ACPI Machine Language operation ToInteger. Using ToInteger would fail if the operating system actually passed a Package containing an Integer.

Other existing hardware platforms use the ACPI Machine Language Index operation. The original intent of these hardware platforms using Index was probably to extract the Integer that should be the first element of the Package. The result on operating systems that pass a buffer of an Integer is to extract the first byte of the Integer. Because ACPI Integers are defined as little-endian unsigned values and the number of defined operations is less than 256, this implementation works as expected.

Example:

A submitted operation value of 6 and a return value of 0 indicates that the platform has successfully received a request to enable and activate the TPM.

8.1.3 Get Pending TPM Operation Requested By the OS

Command Status:

Mandatory

Input Arguments:

Arg0 (Buffer): UUID = 3DDDFAA6-361B-4eb4-A424-8D10089D1653

Arg1 (Integer): Revision ID = 2

Arg2 (Integer): Function Index = 3

Arg3 (Package): Arguments = Empty Package

Returns:

Type: Package

Integer 1:

Purpose: Function Return code

Description:

0: Success

1: General Failure

Integer 2:

Purpose: Pending operation requested by the OS

Description:

0: None

>0: Operation Value of the Pending Request [see Table 1 and Table 2]

Integer 3:

Purpose: Optional argument to pending operation requested by the OS

Description:

0: None

>0: Argument Value of the Pending Request

Functional Behavior:

This function returns the pending operation that was previously requested since the last host platform boot, if any. This function is necessary to allow the OS to accurately determine platform state. One use case is to ensure that a previously submitted operation request is not overwritten.

If 1 is returned as the first integer, the firmware is unable to return meaningful information due to an internal failure. In this case, the second and third integers are undefined.

Assume in the following paragraphs that 0 is returned as the first integer.

If 0 is returned as the second integer, no pending operation exists. No previous request has been submitted.

If a value greater than 0 is returned as the second integer, a previously requested operation is pending (see Table 1 and Table 2 below for a list of defined operations). This request will be read and acted upon by the firmware once the transition to the pre-OS environment takes place.

If the function is invoked with a Revision ID of 1, no integer parameter beyond parameter two are expected in the response.

Examples:

A return value of {0, 0, 0} indicates that no operation is pending.

A return value of {1, 0, 0} indicates that the function failed to retrieve the pending operation. The meaning of the second Integer is undefined.

A return value of {0, 23, 1} indicates that operation 23 with argument 1 is pending.

8.1.4 Get Platform-Specific Action to Transition to Pre-OS Environment

Command Status:

Mandatory

Input Arguments:

Arg0 (Buffer): UUID = 3DDDFAA6-361B-4eb4-A424-8D10089D1653

Arg1 (Integer): Revision ID = 1

Arg2 (Integer): Function Index = 4

Arg3 (Package): Arguments = Empty Package

Returns:

Type: Integer

Purpose: Action that the OS should take to transition to the pre-OS environment for execution of a requested operation.

Description:

0: None

1: Shutdown

2: Reboot

3: OS Vendor-specific

Functional Behavior:

This function allows the OS to determine the platform-specific action that should take place in order to transition to the firmware for execution of a requested operation. This function provides Platform Manufacturers the flexibility to vary how their platforms meet TCG's physical presence requirements, while minimizing the impact of these platform changes on OS applications.

If 0 is returned, no action is required. This return value MUST NOT be used for platforms implementing the command method of asserting physical presence.

If 1 is returned, the OS MUST shut down the machine to execute a pending operation requested by the OS. A physically-present user restarts the machine.

If 2 is returned, the OS MUST cause a warm reboot of the machine to execute a pending operation requested by the OS. This SHALL be used for platforms implementing the command method of physical presence.

If 3 is returned, an OS-specific action can take place. For example, instructions can be displayed for the physically-present user to consult platform documentation. The OS may specify additional requirements outside of this specification to determine the exact behavior for this return value.

8.1.5 Return TPM Operation Response to OS Environment

Command Status:

Mandatory

Input Arguments:

Arg0 (Buffer): UUID = 3DDDFAA6-361B-4eb4-A424-8D10089D1653

Arg1 (Integer): Revision ID = 1

Arg2 (Integer): Function Index = 5

Arg3 (Package): Arguments = Empty Package

Returns:

Type: Package

Integer 1:

Purpose: Function Return Code

Description:

0: Success

1: General Failure

Integer 2:

Purpose: Most recent operation request

Description:

0: None

>0: Operation value of the most recent request

[See Table 1 and Table 2]

Integer 3:

Purpose: Response to the most recent operation request

Description:

0: Success

0x00000001..0x00000FFF: Corresponding TPM error code

0xFFFFFFF0: User Abort or timeout of dialog

0xFFFFFFF1: firmware Failure

Functional Behavior:

This function allows the firmware to communicate the response to the most recent operation request it acted upon. The function returns both the operation number and the outcome of the firmware attempt to perform the operation.

The OS is advised to query the TPM directly to determine whether a request was indeed fulfilled and to use this function's return values only for troubleshooting failures and auditing purposes. The reason is that for some platforms, the return values may not be reliable. For example, if multiple OS's exist on the platform, the request-response values cannot be correlated to any particular OS. Furthermore, there is no guarantee that the correct response is available to the OS that originated the request; another OS may submit a new request, overwriting the response to the previous request.

Firmware should provide the same response for the whole boot cycle. For instance, if first the OS loader and later the OS queries the result of the last operation, both should receive the same response.

If 1 is returned as the first integer, the firmware is unable to return meaningful information due to an internal failure during this function call. In this case, the second and third integers are undefined.

Assume in the following paragraphs that 0 is returned as the first integer.

If 0 is returned as the second integer, no previously-requested operations exist and hence no response exists. In this case, the third integer is undefined.

If a value greater than 0 is returned as the second integer, that value is the most recent operation request seen by the firmware (see Table 1 and Table 2 below for a list of defined operations).

Assume in the following paragraphs that 0 is returned as the first integer and a value greater than 0 is returned as the second integer. The definitions below refer to the response (i.e. the third integer) to the operation request (i.e. the second integer).

If 0 is returned as the response, then the requested operation was confirmed and successfully executed in the pre-OS environment.

A response value from 0x00000001 to 0x00000FFF inclusive corresponds to the TPM error codes defined in Chapter 16 of the “TPM 1.2 Main Specification – Part 2 TPM Structures” document or Chapter 6.6 of the “TPM 2.0 Library Specification – Part 2 TPM Structures.”

If 0xFFFFFFF0 is returned as the response, the user failed to confirm the operation request. This might be due to the user rejecting the operation or the dialog timed out.

If 0xFFFFFFF1 is returned as the response, then a firmware failure prevented the successful execution of the requested operation (e.g. invalid operation request, firmware communication error with the TPM).

Examples:

A return value of {1, 0, 0} indicates that a firmware internal failure prevented the function from retrieving meaningful information. (This corresponds to a firmware error during this function call.)

A return value of {0, 0, 0} indicates that no previously requested operation exists and hence no response exists.

A return value of {0, 6, 0} indicates that the last operation request to enable and activate the TPM succeeded.

A return value of {0, 6, 0xFFFFFFF0} indicates that the last operation request to enable and activate the TPM was rejected by the physically present user in the pre-OS environment.

A return value of {0, 6, 0xFFFFFFF1} indicates that the last operation request to enable and activate the TPM failed to execute successfully due to an internal firmware error. (This corresponds to a previous firmware error performing the operation during the boot process.)

If the physical assertion through a pin is used, the refusal of physical presence assertion will manifest in a TPM error code.

8.1.6 Submit preferred user language

Command Status:

Deprecated and Mandatory

Input Arguments:

Arg0 (Buffer): UUID = 3DDDFAA6-361B-4eb4-A424-8D10089D1653

Arg1 (Integer): Revision ID = 1

Arg2 (Integer): Function Index = 6

Arg3 (Package): Arguments = Package --

Type: String

Purpose: Preferred language code

Description:

Language code that begins with the 2-character ISO-639-1 format

See http://www.loc.gov/standards/iso639-2/php/English_list.php

Returns:

Type: Integer

Purpose: Function Return Code

Description:

3: Not implemented

Functional Behavior:

It is mandatory this function be implemented as part of the ACPI interface, but this function is deprecated and MUST NOT perform any action. The function MUST return 3. (Refer to prior versions of this specification for more information about past usage of this function.)

8.1.7 Submit TPM Operation Request to Pre-OS Environment 2

Command Status:

Mandatory

Input Arguments:

Arg0 (Buffer): UUID = 3DDDFAA6-361B-4eb4-A424-8D10089D1653

Arg1 (Integer): Revision ID = 2

Arg2 (Integer): Function Index = 7

Arg3 (Package): Arguments = Package --

Integer 1:

Purpose: Operation Value of the Request

Description: [see Table 1 and Table 2]

Integer 2:

Purpose: Argument for Operation (optional)

Description:

For TPM 2.0 devices and operation 23: A bitmap of hashing algorithms of the PCR banks to enable in the TPM. See hashing algorithm bitmap definition in TCG Algorithm Registry for Family 2.0 revision 1.21.

Returns:

Type: Integer

Purpose: Function Return Code

Description:

0: Success

1: Not Implemented

2: General Failure

3: Operation blocked by current firmware settings

Functional Behavior:

This function allows the OS to submit a request for an operation to be executed in the pre-OS environment (see Table 1 and Table 2 for a list of defined operations). This request is the only input from the OS to the pre-OS environment.

If 0 is returned, the requested operation can be read and acted upon by the firmware once the transition to the pre-OS environment takes place (e.g. after the platform has restarted). The OS expects that the pre-OS environment verifies physical presence and confirms that the physically present user in fact requested the execution of the operation.

If 1 is returned, the firmware does not support the OS requesting the operation because it is not implemented and is never able to be requested by the OS regardless of firmware settings. For example, the implementation of the operation may be optional, reserved, “No operation”, or vendor-specific (e.g. implemented as an action that may only be performed using a Platform Manufacturer utility).

If 2 is returned, the firmware is otherwise unable to read and act upon the request. For example, platform specific security protections may exist to prevent burnout of the storage location for the operation.

If 3 is returned, the current firmware settings prohibit this action, but the platform may permit the operation if the platform owner configures the firmware settings differently.

The OS may call this function or “Submit TPM Operation Request to Pre-OS Environment” multiple times before transitioning to the pre-OS environment. However, only the last submitted request is valid. The OS may submit an operation request of value 0 to clear any previous requests.

If the OS calls this function with operation requests that do not match the version of the present TPM, the operation number MUST be remembered, the result MUST return success (return value 0), but no commands are sent to the TPM.

Examples:

A submitted operation value of 6 (Enable + Activate) and a return value of 0 indicate the platform has successfully received a request to enable and activate the TPM.

A submitted operation value of 18 (SetNoPPIClear_True) and a return value of 1 indicate the Platform Manufacturer did not implement the Persistent Firmware TPM Management Flag named NoPPIClear and did not implement the corresponding operation to set the value.

A submitted operation value of 5 (Clear) and a return value 3 indicates a current firmware setting currently blocks the OS from requesting to clear the TPM. However, a firmware administrator may configure the firmware settings to permit the OS to request to clear the TPM in the future.

8.1.8 Get User Confirmation Status for Operation

Command Status:

Mandatory

Input Arguments:

Arg0 (Buffer): UUID = 3DDDFAA6-361B-4eb4-A424-8D10089D1653

Arg1 (Integer): Revision ID = 1

Arg2 (Integer): Function Index = 8

Arg3 (Package): Arguments = Package --

Type: Integer

Purpose: Operation Value that may need user confirmation

Description: [see Table 1 and Table 2]

Returns:

Type: Integer

Purpose: Function Return Code

Description:

0: Not implemented

1: Firmware only

2: Blocked for OS by firmware configuration

3: Allowed and physically present user required

4: Allowed and physically present user not required

Functional Behavior:

This function allows the OS to determine whether it is allowed to request an operation to be executed in the pre-OS environment (see Table 1 and Table 2 for a list of defined operations) by the firmware. If the OS is not allowed by the firmware to perform the operation, the function returns whether the firmware administrator may reconfigure the firmware to allow the OS to request the operation.

If the firmware permits the OS to request an operation, this function allows the OS to determine whether a request would require confirmation from a physically present user or would be performed automatically because of the current settings for the Persistent Firmware TPM Management flags.

This function does not cause the operation to be requested.

Note: It is preferable for an OS to use this function to determine if an operation is supported instead of using “Submit TPM Operation Request to Pre-OS Environment 2” because this function does not write to NVRAM.

If 0 is returned, the firmware does not support this operation because it is not implemented. An example is if the Platform Manufacturer did not support the NoPPIClear flag and the OS queried for the operation SetNoPPIClear_True.

Note: For an implementation of version 1.2 or later of this interface, a return value of 0 means the operation value passed in Arg3 is not implemented. If an earlier version of this specification is implemented, a return value of 0 means this method is not implemented.

If 1 is returned, the firmware unconditionally does not permit the OS to request this action, but a firmware administrator may perform the equivalent action through a vendor specific mechanism like a firmware configuration utility.

If 2 is returned, the current firmware settings do not permit the OS to request this operation, but a firmware administrator may change the setting through a vendor specific mechanism like a firmware configuration utility to allow the OS to request the operation.

If 3 is returned, the firmware currently permits the OS to request this action and the operation would require physically present user approval

If 4 is returned, the firmware currently permits the OS to request this action and the current firmware settings permit execution of the operation without a physically present user approving the operation

To implement this function, the firmware may need to copy the persistent settings for the Persistent Firmware TPM Management Flags or Persistent Firmware Storage Management Flags when accessible during boot as the settings influence the return values for this function. The logic of this function will also need to implement the logic in the “When Physical Presence Confirmation is Required” column of Table 1 and Table 2.

Examples:

A submitted operation value of 18 (NoPPIClear_True) and a return value of 0 indicate the Platform Manufacturer did not implement the Persistent firmware TPM Management Flag named NoPPIClear and did not implement the corresponding operation to set the value.

A submitted operation value of 12 and a return value of 0 indicate the platform does not support Deferred Physical Presence for unowned field upgrade.

A submitted operation value of 5 and a return value of 2 indicate the firmware is currently configured to not allow an OS to request to clear the TPM using physical presence, but a firmware administrator may modify the firmware configuration using a vendor specific firmware configuration utility to permit the OS to request the operation.

A submitted operation value of 5 and a return value of 3 indicate the firmware is currently configured to allow an OS to request to clear the TPM, but the NoPPIClear flag is not implemented or has a value of FALSE so a physically present user would need to approve the operation.

A submitted operation value of 14 and a return value of 4 indicate the firmware is currently configured to allow an OS to request to clear, enable and activate the TPM, the NoPPIClear flag is set to TRUE, and the NoPPIProvision flag is set to TRUE so a physically present user is not needed to approve the operation.

8.2 Parameter Passing

With the exception of operation 13 defined in Table 1, all parameters between the OS and the firmware are passed using the standard ACPI defined parameter passing method. The values passed for operations, except 13, are small and are easily accommodated by the 4 arguments. However, operation 13 requires the passing of an additional value of a relatively large size: 20 bytes. As it is not possible to add another parameter, the method for passing this value from the OS to the firmware is not defined in this version of the specification. A later version of this specification may address, define and standardize the method for passing this value. Until then, any suitable method that is in agreement between the OS and firmware is permitted.

1. The method for passing the operator authentication value from the OS to the firmware is to be determined by convention between the OS and the firmware. Standardization and definition of the method will be addressed in a later revision of this specification.

9 Operation Definitions

Table 1 and Table 2 contain a list of the defined PPI operations for TPM 1.2 and TPM 2.0 respectively. There are optional and mandatory PPI operations and the following normative describes relations between different optional PPI operations.

Table 3 holds the definition of what ordered actions are performed by the firmware to complete each of the operations defined in this specification. The firmware may send commands to the TPM, performs platform resets and/or changes Persistent Firmware TPM Management Flags or Persistent Firmware Storage Management Flags. The operations defined in Table 3 could have been added to Table 1 and Table 2 respectively, but have been separated to maintain readability.

If a platform implements the hardware method of asserting physical presence, the utility which prompts the user to indicate physical presence should display the same confirmation dialogs (see Table 5) and perform the same actions for each operation (see Table 3) for consistency with this specification. On server systems physical presence may be asserted using the hardware method only.

Notes for operations 23:

For systems with TPM 2.0 that implement operations 23 it may not be necessary to invoke the TPM operation TPM2_PCR_Allocate when switching PCR banks. When using the crypto agile format for TCG log, the BIOS may keep only one boot log independent of the selected PCR bank or banks.

Warning:

Changes to the currently active PCR banks will effect TPM objects bound to PCRs. It is recommended to allow removal of PCR banks from the list of active PCR banks only when the PCR banks are not used by users or applications outside the direct control of the OS. If that is not possible, the OS should appropriately inform the user about consequences of changing the PCR bank before requesting the operation.

1. In order to persist the OS requested operation to the next boot cycle, the firmware MUST allocate a location for storing the operation's corresponding value.
2. The operation value of 0 is reserved to indicate that no Physical Presence Interface operation has been requested.
3. Firmware MUST implement all operations marked as mandatory in the respective operations summary table.
4. Operations with values at or above 128 (0x80) are designated for vendor-specific usage.
5. For each operation implemented by the firmware, the firmware MUST perform the TPM commands listed in the third column of Table 3 in the order listed.

Note:

A reboot is required for some changes to take effect. This is noted with the keyword: <PLATFORM RESET>.

6. On systems with TPM 1.2, for operations 5 and 14 the platform manufacturer MAY perform the TPM commands Enable and Activate before performing the TPM_ForceClear operation.
7. On systems with TPM 2.0, for operation 5 the platform manufacturer MAY perform the TPM Enable actions before performing the TPM2_Clear operation.
8. The firmware is responsible for tracking and performing operations that require a PLATFORM RESET.

9. The location for tracking the pending PPI operation, including the tracking of necessary PLATFORM RESET operations, does not need to be a secure or trusted location.
10. If during an operation that requires a PLATFORM RESET, the firmware stores the assertion of an indication of physical presence across the PLATFORM RESET, that location MUST be trusted.
11. On systems with a TPM 1.2, if the firmware implements operations 15 and 16, the firmware MUST provide persistent storage for the NoPPIProvision flag.
12. On systems with a TPM 1.2, if the firmware implements operations 17 and 18, the firmware MUST provide persistent storage for the NoPPIClear flag.
13. On systems with a TPM 1.2, if the firmware implements operations 19 and 20, firmware MUST provide persistent storage for the NoPPIMaintenance flag.
14. On systems with a TPM 2.0, if the firmware implements operations 17 and 18, the firmware MUST provide persistent storage for the PPRequiredForClear flag.
15. On systems with TPM 2.0, firmware MUST NOT implement operation 27, 28, 29, and 30 if operation 1, 2, and 34 are not implemented.
16. On systems with TPM 2.0, if firmware implements operations 27, 28, 29, and 30, it MUST provide persistent storage for the PPRequiredForTurnOn and PPRequiredForTurnOff flags.
17. On systems with TPM 2.0, firmware MUST NOT implement operations 25 and 26 if operation 23 is not implemented.
18. On systems with TPM 2.0, if the firmware implements operations 25 and 26, it MUST provide persistent storage for the PPRequiredForChangePCRs flag.
19. On systems with TPM 2.0, firmware MUST only implement operation 23 if the TPM supports multiple PCR banks, i.e. implements operation TPM2_PCR_Allocate().
20. On systems with TPM 2.0, firmware MUST NOT implement operations 31 and 32 if operation 24 is not implemented.
21. On systems with TPM 2.0, if firmware implements operations 31 and 32, it MUST provide persistent storage for the PPRequiredForChangeEPS flag.
22. On systems with TPM 2.0, firmware MUST only implement operation 24 if the TPM implements operation TPM2_ChangeEPS().
23. On systems with TPM 2.0, firmware MAY implement operation 33 if the TPM supports multiple PCR banks, i.e. implements operation TPM2_PCR_Allocate().
24. If an operation in Table 1 or Table 2 is marked as optional with O1 or similar, all operations within that group have to be implemented. For instance, operations 1, 2, and 34 from Table 2 have to be implemented together.
25. On systems with TPM 2.0, if firmware implements operations 96 and 97, it MUST provide persistent storage for the PPRequiredForEnable_BlockSIDFunc and PPRequiredForDisable_BlockSIDFunc flags.

Table 1: Physical Presence Interface Operation Summary for TPM 1.2

Operation Value	Operation Name	What the operation may change		Mandatory or Optional	When Physical Presence Confirmation is Required	May need additional boot cycle
		TPM State	Persistent Firmware TPM Management Flags			
0	No Operation			M		
1	Enable	X		M	NoPPIProvision is FALSE	
2	Disable	X		M	NoPPIProvision is FALSE	
3	Activate	X		M	NoPPIProvision is FALSE	X
4	Deactivate	X		M	NoPPIProvision is FALSE	X
5	Clear	X		M	NoPPIClear is FALSE	X
6	Enable + Activate	X		M	NoPPIProvision is FALSE	X
7	Deactivate + Disable	X		M	NoPPIProvision is FALSE	X
8	SetOwnerInstall_True	X		M	NoPPIProvision is FALSE	
9	SetOwnerInstall_False	X		M	NoPPIProvision is FALSE	
10	Enable + Activate + SetOwnerInstall_True	X		M	NoPPIProvision is FALSE	X
11	SetOwnerInstall_False + Deactivate + Disable	X		M	NoPPIProvision is FALSE	X
12	Deferred Physical Presence-unownedFieldUpgrade	X		O	NoPPIMaintenance is FALSE	X
13	SetOperatorAuth	X		O	NoPPIProvision is FALSE	
14	Clear + Enable + Activate	X		M	NoPPIClear is FALSE OR NoPPIProvision is FALSE	X
15	SetNoPPIProvision_False		X	M		
16	SetNoPPIProvision_True		X	M	Always	
17	SetNoPPIClear_False		X	O1		
18	SetNoPPIClear_True		X	O1	Always	
19	SetNoPPIMaintenance_False		X	O2		
20	SetNoPPIMaintenance_True		X	O2	Always	
21	Enable + Activate + Clear	X		M	NoPPIClear is FALSE	X

Operation Value	Operation Name	What the operation may change		Mandatory or Optional	When Physical Presence Confirmation is Required	May need additional boot cycle
		TPM State	Persistent Firmware TPM Management Flags			
22	Enable + Activate + Clear + Enable + Activate	X		M	NoPPIClear is FALSE OR NoPPIProvision is FALSE	X
23 – 127	Reserved					
>=128	Vendor Specific	X	X	O		X

Table 2: Physical Presence Interface Operation Summary for TPM 2.0

Operation Value	Operation Name	What the operation may change		Mandatory or Optional	When Physical Presence Confirmation is Required	May need additional boot cycle
		TPM State	Persistent Firmware TPM Management Flags			
0	No Operation			M		
1	Enable	X		O1	PPRequiredForTurnOn is TRUE	
2	Disable	X		O1	PPRequiredForTurnOff is TRUE	
3, 4	Reserved					
5	Clear	X		M	PPRequiredForClear is TRUE	X
6 – 13	Reserved					
14	Enable + Clear	X		M	PPRequiredForClear is TRUE OR PPRequiredForTurnOn is TRUE	X
15, 16	Reserved					
17	SetPPRequiredForClear_True		X	O2		
18	SetPPRequiredForClear_False		X	O2	Always	
19, 20	Reserved					
21	Enable + Clear	X		M	PPRequiredForClear is TRUE OR PPRequiredForTurnOn is TRUE	X
22	Enable + Clear	X		M	PPRequiredForClear is TRUE OR PPRequiredForTurnOn is TRUE	X
23	SetPCRBanks	X		O	PPRequiredForChangePCRs is TRUE	X
24	ChangeEPS	X		O	PPRequiredForChangeEPS is TRUE	
25	SetPPRequiredForChangePCRs_False		X	O3		
26	SetPPRequiredForChangePCRs_True		X	O3	Always	
27	SetPPRequiredForTurnOn_False		X	O4		
28	SetPPRequiredForTurnOn_True		X	O4	Always	

Operation Value	Operation Name	What the operation may change		Mandatory or Optional	When Physical Presence Confirmation is Required	May need additional boot cycle
		TPM State	Persistent Firmware TPM Management Flags			
29	SetPPRequiredForTurnOff_False		X	O5		
30	SetPPRequiredForTurnOff_True		X	O5	Always	
31	SetPPRequiredForChangeEPS_False		X	O6		
32	SetPPRequiredForChangeEPS_True		X	O6	Always	
33	LogAllDigests			O		X
34	DisableEndorsementEnableStorageHierarchy		X	O1	PPRequiredForTurnOn is TRUE OR PPRequiredForTurnOff is TRUE	
35 – 95	Reserved (for TPM management)					
96	Enable_BlockSIDFunc			O7	PPRequiredForEnable_BlockSIDFunc is TRUE	X
97	Disable_BlockSIDFunc			O7	PPRequiredForDisable_BlockSIDFunc is TRUE	X
98	SetPPRequiredForEnable_BlockSIDFunc_True			O8	Always	
99	SetPPRequiredForEnable_BlockSIDFunc_False			O8		
100	SetPPRequiredForDisable_BlockSIDFunc_True			O9	Always	
101	SetPPRequiredForDisable_BlockSIDFunc_False			O9		
102 – 127	Reserved (for Storage management)					
>= 128	Vendor Specific	X	X	O		X

Table 3: Definition of Operation Actions

Operation Value	Operation Name	TPM commands sent by firmware and other firmware actions to perform the operation
0	No Operation	*No operation*
1	Enable	For TPM 1.2: TPM_PhysicalEnable For TPM 2.0: Firmware stores information that storage and endorsement hierarchy are enabled on each boot. TPM2_HierarchyControl() after TPM2_Startup()
2	Disable	For TPM 1.2: TPM_PhysicalDisable For TPM 2.0: Firmware stores information that disables storage and endorsement hierarchy on each boot. TPM2_HierarchyControl() after TPM2_Startup()
3	Activate	TPM_PhysicalSetDeactivated = FALSE <PLATFORM RESET>
4	Deactivate	TPM_PhysicalSetDeactivated = TRUE <PLATFORM RESET>
5	Clear	For TPM 1.2: TPM_ForceClear <PLATFORM RESET> For TPM 2.0: TPM2_ClearControl(NO) TPM2_Clear
6	Enable + Activate	TPM_PhysicalEnable TPM_PhysicalSetDeactivated = FALSE <PLATFORM RESET>
7	Deactivate + Disable	TPM_PhysicalSetDeactivated = TRUE TPM_PhysicalDisable <PLATFORM RESET>
8	SetOwnerInstall_True	TPM_SetOwnerInstall = TRUE
9	SetOwnerInstall_False	TPM_SetOwnerInstall = FALSE
10	Enable + Activate + SetOwnerInstall_True	TPM_PhysicalEnable TPM_PhysicalSetDeactivated = FALSE <PLATFORM RESET> TPM_SetOwnerInstall = TRUE
11	SetOwnerInstall_False + Deactivate + Disable	TPM_SetOwnerInstall = FALSE TPM_PhysicalSetDeactivated = TRUE TPM_PhysicalDisable <PLATFORM RESET> Note: The firmware MAY perform the PLATFORM RESET immediately after the TPM_PhysicalSetDeactivated = TRUE but that requires tracking the next command.

Operation Value	Operation Name	TPM commands sent by firmware and other firmware actions to perform the operation
12	Deferred Physical Presence-unownedFieldUpgrade	TPM_SetCapability -> setValue (Cap= TPM_SET_STCLEAR_FLAGS; SubCap= TPM_SD_DEFERREDPHYSICALPRESENCE ; value= unownedFieldUpgrade Implementer's note: The firmware calls the above SetCapability while the operator is asserting Physical Presence. Rather than a command to perform a specific function, this function sets an attribute within the TPM indicting that the Operator has asserted Physical Presence to authorize the TPM_FieldUpgrade function.
13	SetOperatorAuth	TPM_SetOperatorAuth, with operatorAuth prompted by the firmware
14	Clear + Enable + Activate	For TPM 1.2: TPM_ForceClear TPM_PhysicalEnable TPM_PhysicalSetDeactivated = FALSE <PLATFORM RESET> For TPM 2.0: TPM2_ClearControl(NO) TPM2_Clear
15	SetNoPPIProvision _False (Require physical presence confirmation for provisioning.)	This operation does not change TPM state. Clears the Persistent Firmware TPM Management Flag NoPPIProvision to FALSE.
16	SetNoPPIProvision _True (Do not require physical presence confirmation for provisioning.)	This operation does not change TPM state. Sets the Persistent Firmware TPM Management Flag NoPPIProvision to TRUE.
17	SetPPRequiredForClear_True SetNoPPIClear_False (Require physical presence for clear.)	This operation does not change TPM state. Sets the Persistent Firmware TPM Management Flag PPRequiredForClear to TRUE or Clears NoPPIClear to FALSE.
18	SetPPRequiredForClear_False SetNoPPIClear_True (Do not require physical presence confirmation for clear.)	This operation does not change TPM state. Clears the Persistent Firmware TPM Management Flag PPRequiredForClear to FALSE or Sets NoPPIClear to TRUE.
19	SetNoPPIMaintenance_False (Require physical presence confirmation for maintenance.)	This operation does not change TPM state. Clears the Persistent Firmware TPM Management Flag NoPPIMaintenance to FALSE.
20	SetNoPPIMaintenance_True (Do not require physical presence confirmation for maintenance.)	This operation does not change TPM state. Sets the Persistent Firmware TPM Management Flag NoPPIMaintenance to TRUE.

Operation Value	Operation Name	TPM commands sent by firmware and other firmware actions to perform the operation
21	Enable + Activate + Clear	For TPM 1.2: If the TPM is disabled then perform TPM_PhysicalEnable OR unconditionally perform TPM_PhysicalEnable If the TPM is deactivated, then perform TPM_PhysicalSetDeactivated = FALSE & <PLATFORM RESET> OR unconditionally perform TPM_PhysicalSetDeactivated = FALSE & <PLATFORM RESET> TPM_ForceClear <PLATFORM RESET> For TPM 2.0: TPM2_ClearControl(NO) TPM2_Clear
22	Enable + Activate + Clear + Enable + Activate	For TPM 1.2: If the TPM is disabled then perform TPM_PhysicalEnable OR unconditionally perform TPM_PhysicalEnable If the TPM is deactivated, then perform TPM_PhysicalSetDeactivated = FALSE & <PLATFORM RESET> OR unconditionally perform TPM_PhysicalSetDeactivated = FALSE & <PLATFORM RESET> TPM_ForceClear TPM_PhysicalEnable TPM_PhysicalSetDeactivated = FALSE <PLATFORM RESET> For TPM 2.0: TPM2_ClearControl(NO) TPM2_Clear
23	SetPCRBanks	The bitmap of hashing algorithms for the operation is extracted as extra parameter from the passed package. Firmware should check that all requested (set) hashing algorithms are supported with respective PCR banks. If not, an error is returned and no action is taken. If all hashing algorithms are supported, the list of active PCR banks is updated to match the requested algorithms. If a change of the PCR banks is required, firmware then executes: TPM2_PCR_Allocate(active algorithm IDs) <PLATFORM RESET> Firmware MUST create TCG logs for all active hashing algorithms after <PLATFORM RESET>. Firmware has to ensure that at least one PCR banks is active.

Operation Value	Operation Name	TPM commands sent by firmware and other firmware actions to perform the operation
24	ChangeEPS	TPM2_ChangeEPS <PLATFORM RESET>
25	SetPPRequiredForChangePCRs_False	This operation does not change TPM state. Clears the Persistent Firmware TPM Management Flag PPRequiredForChangePCRs to FALSE.
26	SetPPRequiredForChangePCRs_True	This operation does not change TPM state. Sets the Persistent Firmware TPM Management Flag PPRequiredForChangePCRs to TRUE.
27	SetPPRequiredForTurnOn_False	This operation does not change TPM state. Clears the Persistent Firmware TPM Management Flag PPRequiredForTurnOn to FALSE.
28	SetPPRequiredForTurnOn_True	This operation does not change TPM state. Sets the Persistent Firmware TPM Management Flag PPRequiredForTurnOn to TRUE.
29	SetPPRequiredForTurnOff_False	This operation does not change TPM state. Clears the Persistent Firmware TPM Management Flag PPRequiredForTurnOff to FALSE.
30	SetPPRequiredForTurnOff_True	This operation does not change TPM state. Sets the Persistent Firmware TPM Management Flag PPRequiredForTurnOff to TRUE.
31	SetPPRequiredForChangeEPS_False	This operation does not change TPM state. Clears the Persistent Firmware TPM Management Flag PPRequiredForChangeEPS to FALSE.
32	SetPPRequiredForChangeEPS_True	This operation does not change TPM state. Sets the Persistent Firmware TPM Management Flag PPRequiredForChangeEPS to TRUE.
33	LogAllDigests	This operation does not change TPM state. On next reboot (and only on next reboot), add digests for all supported hashing algorithms to log file.
34	DisableEndorsementEnableStorageHierarchy	Firmware stores information that enables storage hierarchy and disables endorsement hierarchy on each boot. TPM2_HierarchyControl() after TPM2_Startup()
35 – 95	Reserved (for TPM management)	Reserved, do not implement or use

Operation Value	Operation Name	TPM commands sent by firmware and other firmware actions to perform the operation
96	Enable_BlockSIDFunc	This operation does not change TPM state. Firmware send command as defined in TCG Storage Feature Set: Block SID Authentication Specification to block SID authentication in a TCG Storage device.
97	Disable_BlockSIDFunc	This operation does not change TPM state. Firmware send command as defined in TCG Storage Feature Set: Block SID Authentication Specification to allow SID authentication in a TCG Storage device.
98	SetPPRequiredForEnable_BlockSIDFunc_True	This operation does not change TPM state. Sets the Persistent Firmware Storage Management Flag PPRequiredForEnable_BlockSIDFunc to TRUE.
99	SetPPRequiredForEnable_BlockSIDFunc_False	This operation does not change TPM state. Clears the Persistent Firmware Storage Management Flag PPRequiredForEnable_BlockSIDFunc to FALSE.
100	SetPPRequiredForDisable_BlockSIDFunc_True	This operation does not change TPM state. Sets the Persistent Firmware Storage Management Flag PPRequiredForDisable_BlockSIDFunc to TRUE.
101	SetPPRequiredForDisable_BlockSIDFunc_False	This operation does not change TPM state. Clears the Persistent Firmware Storage Management Flag PPRequiredForDisable_BlockSIDFunc to FALSE.
102 – 127	Reserved (for Storage management)	Reserved, do not implement or use
>= 128	Vendor Specific	TPM commands mapping to vendor specific operation

10 Confirmation Dialogs and Keys

Confirmation Dialogs

This section is intended to provide design guidance for the firmware dialogs used to confirm that the physically present user in fact requested execution of the operation. Each operation request is mapped to the dialog presented to the user to accept or reject the operation.

In consideration of the limited space in the firmware to store text, the confirmations presented in this section attempt to maximize the potential for text reuse while minimizing impact on security and user understanding.

For consistency in user experience, it is recommended that Platform Manufacturers implement confirmation dialogs in a similar manner.

Platform Manufacturers should also take into consideration localization requirements for the dialog text. The firmware should provide a means of ensuring that the dialog text displayed to the user can be understood by that user (e.g. allow the user to change the language of the confirmation request, default to the user's preferred language, etc.).

While the text and confirmation keys in Table 4 have been carefully considered to meet a broad range of language and cultural environments, these may not convey the meaning of the action to all persons depending on their language, culture, education, and expertise with computing platforms. For this reason, the text provided is preferred but is not mandatory for all situations.

Confirmation Keys

For security purposes, note that the key used to confirm the Clear and the Deferred Physical Presence unownedFieldUpgrade operations differs from the confirmation key used for other operations.

The confirmation keys are selected for most keyboards and expected user experiences. While the keys selected do not conflict with a majority of firmware implementation, some conflicts may occur. The firmware provider may have used the confirmation keys in Table 4 for other purposes already familiar to the platform user. In this case, changing the meaning of the key would be confusing to the user. For this reason, the firmware is allowed to use other keys when such conflicts occur.

Timeouts

A firmware implementation may choose to present the user confirmation dialog only for a limited amount of time. If no user confirmation is presented during that time, the default response for the requested operation is assumed. Usually, the default response would be to deny the operation.

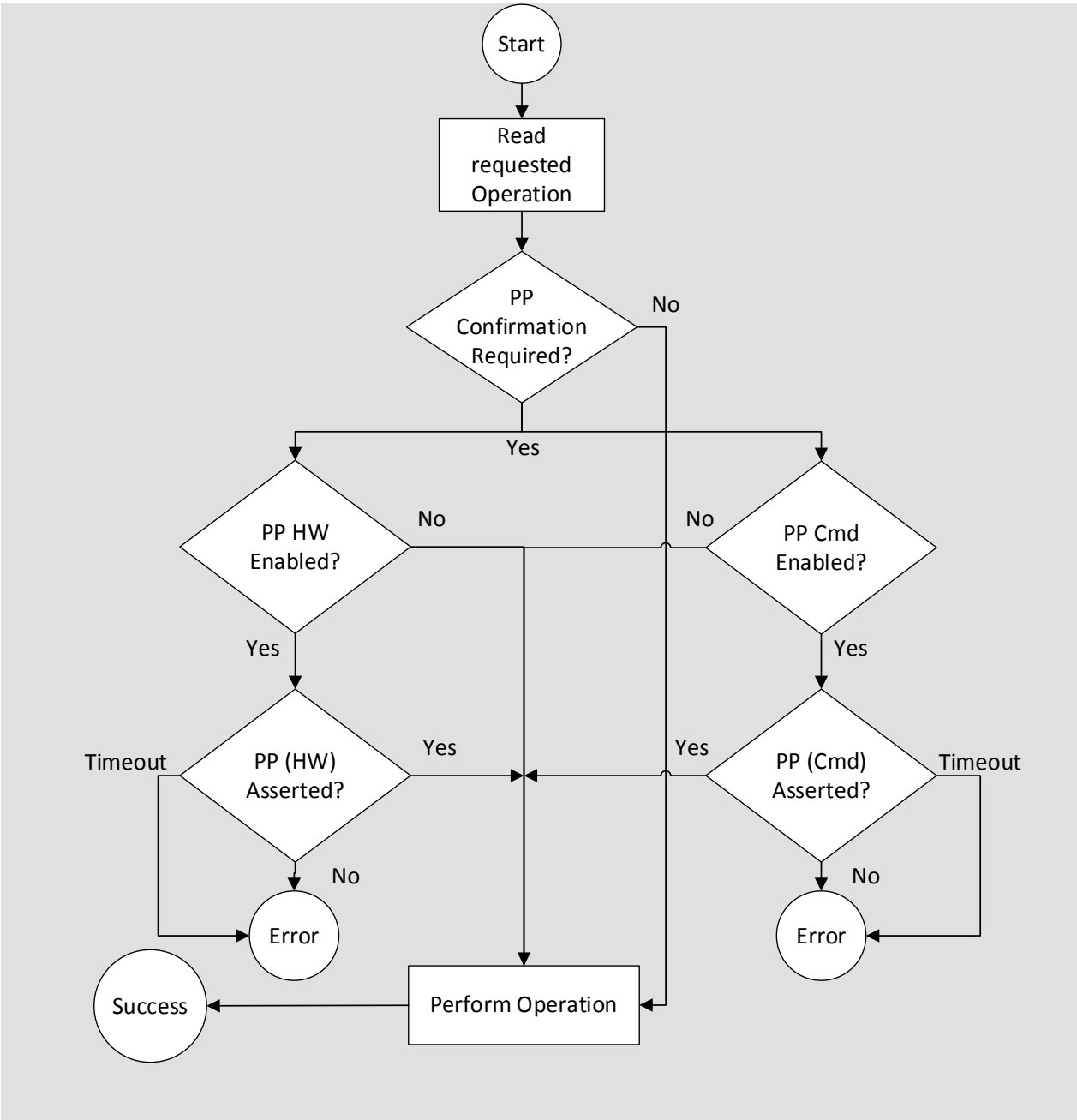

Figure 1: Flowchart for timeout behavior

1. For each operation to be performed through the Physical Presence Interface the firmware MUST ask for confirmation from a physically present operator per the “When Physical Presence Confirmation is Required” column in Table 1 and Table 2. For example, for operation 14, the firmware MUST ask for a physically present user to confirm the operation if the flag NoPPIClear is FALSE. However, if the flag is TRUE, then the firmware MUST NOT ask for confirmation.
2. The text in Table 4 is the preferred confirmation text wording to be displayed to the user. The firmware MAY provide alternate confirmation text wording that differs from Table 4 based upon the platform’s expected user experience and language; however, the meaning of each message MUST be retained. The text in Table 4 MAY be translated into languages other than the English text indicated in Table 4 provided the meaning of the text in Table 4 is conveyed in the translated text.

3. Table 5 contains labels for the following user confirmation keys: reject key (labeled as <RK>), accept key (labeled as <AK>), and cautionary accept key (labeled as <CAK>). The key associated with each label represents a user action indicated by the associated text. The actual key used for each labeled key is to be determined by the platform's expected user experience. Typical and example keys that SHALL be used for most environments are provided in Table 5. All occurrences of confirmation keys MUST be consistent for each operation for the same platform, i.e. the firmware for each platform MUST use the same key associated to each label. The text representing the key SHALL exclude the angle bracket (i.e. the '<' and '>' characters) and SHALL use the string representing the actual key indicated in Table 5 when presenting the text in Table 4.
4. The firmware may provide alternate confirmation keys that differ from Table 5 based upon the platform's keyboard configuration, expected user experience, or if no keyboard is present.
5. When changing the Endorsement Primary Seed (EPS) using operation 25, all keys in the endorsement hierarchy become invalid and unusable. This implies that the certificates for keys in the endorsement hierarchy now certify invalid keys. A TPM 2.0 will generate a new Endorsement Key (EK) after the EPS has been changed. This new EK could be certified by the TPM vendor if the vendor can validate that the key has been generated from one of its TPMs. Keys derived from the EK will also become unusable and will have to be recreated.

Table 4: Confirmations of Physical Presence Interface Operations

Op Value	Operation Name	Confirmation Text
0	No Operation	None
1	Enable	A configuration change was requested to enable this computer's TPM (Trusted Platform Module) Press <AK> to enable the TPM Press <RK> to reject this change request and continue
2	Disable	A configuration change was requested to disable this computer's TPM (Trusted Platform Module) WARNING: Doing so might prevent security applications that rely on the TPM from functioning as expected Press <AK> to disable the TPM Press <RK> to reject this change request and continue
3	Activate	A configuration change was requested to activate this computer's TPM (Trusted Platform Module) Press <AK> to activate the TPM Press <RK> to reject this change request and continue
4	Deactivate	A configuration change was requested to deactivate this computer's TPM (Trusted Platform Module) WARNING: Doing so might prevent security applications that rely on the TPM from functioning as expected Press <AK> to deactivate the TPM Press <RK> to reject this change request and continue
5	Clear	A configuration change was requested to clear this computer's TPM (Trusted Platform Module) WARNING: Clearing erases information stored on the TPM. You will lose all created keys and access to data encrypted by these keys. Press <CAK> to clear the TPM Press <RK> to reject this change request and continue
6	Enable + Activate	A configuration change was requested to enable and activate this computer's TPM (Trusted Platform Module) NOTE: This action will turn on the TPM Press <AK> to enable and activate the TPM Press <RK> to reject this change request and continue

Op Value	Operation Name	Confirmation Text
7	Deactivate + Disable	A configuration change was requested to deactivate and disable this computer's TPM (Trusted Platform Module) NOTE: This action will turn off the TPM WARNING: Doing so might prevent security applications that rely on the TPM from functioning as expected Press <AK> to deactivate and disable the TPM Press <RK> to reject this change request and continue
8	SetOwnerInstall_True	A configuration change was requested to allow a user to take ownership of this computer's TPM (Trusted Platform Module) Press <AK> to allow a user to take ownership of the TPM Press <RK> to reject this change request and continue
9	SetOwnerInstall_False	A configuration change was requested to disallow a user to take ownership of this computer's TPM (Trusted Platform Module) Press <AK> to disallow a user to take ownership of the TPM Press <RK> to reject this change request and continue
10	Enable + Activate + SetOwnerInstall_True	A configuration change was requested to enable, activate, and allow a user to take ownership of this computer's TPM (Trusted Platform Module) NOTE: This action will turn on the TPM Press <AK> to enable, activate, and allow a user to take ownership of the TPM Press <RK> to reject this change request and continue
11	SetOwnerInstall_False + Deactivate + Disable	A configuration change was requested to deactivate, disable, and disallow a user to take ownership of this computer's TPM (Trusted Platform Module) NOTE: This action will turn off the TPM WARNING: Doing so might prevent security applications that rely on the TPM from functioning as expected Press <AK> to deactivate, disable, and disallow a user to take ownership of the TPM Press <RK> to reject this change request and continue
12	Deferred Physical Presence-unownedFieldUpgrade	A configuration change was requested to allow changes to the TPM's (Trusted Platform Module's) firmware WARNING: Allowing changes to the TPM's firmware may affect the operation of the TPM and may erase information stored on the TPM. You may lose all created keys and access to data encrypted by these keys Press <CAK> to Allow field upgrade of the TPM Press <RK> to reject this change request and continue

Op Value	Operation Name	Confirmation Text
13	SetOperatorAuth	A configuration change was requested to allow the creation of an operator authentication value that permits the temporary deactivation of this computer's TPM (Trusted Platform Module) Press <AK> to allow the creation of the operator authentication value of the TPM Press <RK> to reject this change request and continue
14	Clear + Enable + Activate	For TPM 1.2: A configuration change was requested to clear, enable, and activate this computer's TPM (Trusted Platform Module) NOTE: This action will clear and turn on the TPM WARNING: Clearing erases information stored on the TPM. You will lose all created keys and access to data encrypted by these keys. Take ownership as soon as possible after this step. Press <CAK> to clear, enable, and activate the TPM Press <RK> to reject this change request and continue For TPM 2.0: A configuration change was requested to clear this computer's TPM (Trusted Platform Module) WARNING: Clearing erases information stored on the TPM. You will lose all created keys and access to data encrypted by these keys. Press <CAK> to clear the TPM Press <RK> to reject this change request and continue
15	SetNoPPIProvision_False	Implementers Note: Physical presence confirmation is not necessary. The firmware MAY display a message during boot to the user but MUST not ask the user for confirmation.
16	SetNoPPIProvision_True	A configuration change was requested to allow the Operating System to provision the computer's TPM (Trusted Platform Module) without asking for user confirmation in the future. Press <AK> to approve future Operating System requests to provision the TPM Press <RK> to reject this change request and continue
17	SetNoPPIClear_False or SetPPRequiredForClear_True	Implementers Note: Physical presence confirmation is not necessary. The firmware MAY display a message during boot to the user but MUST not ask the user for confirmation.

Op Value	Operation Name	Confirmation Text
18	SetNoPPIClear_True or SetPPRequiredForClear_False	A configuration change was requested to allow the Operating System to clear the computer's TPM (Trusted Platform Module) without asking for user confirmation in the future. NOTE: This action does not clear the TPM, but by approving this configuration change, future actions to clear the TPM will not require user confirmation. WARNING: Clearing erases information stored on the TPM. You will lose all created keys and access to data encrypted by these keys. Press <CAK> to approve future Operating System requests to clear the TPM Press <RK> to reject this change request and continue
19	SetNoPPIMaintenance_False	Implementers Note: Physical presence confirmation is not necessary. The firmware MAY display a message during boot to the user but MUST not ask the user for confirmation.
20	SetNoPPIMaintenance_True	A configuration change was requested to allow the Operating System to maintain the computer's TPM (Trusted Platform Module) without asking for user confirmation in the future. WARNING: Allowing future changes to the TPM's firmware may affect the operation of the TPM and may erase information stored on the TPM. You may lose all created keys and access to data encrypted by these keys Press <CAK> to approve future Operating System requests to maintain the TPM Press <RK> to reject this change request and continue.
21	Enable + Activate + Clear	A configuration change was requested to enable, activate and clear this computer's TPM (Trusted Platform Module) WARNING: Clearing erases information stored on the TPM. You will lose all created keys and access to data encrypted by these keys. Press <CAK> to clear the TPM Press <RK> to reject this change request and continue

Op Value	Operation Name	Confirmation Text
22	Enable + Activate + Clear + Enable + Activate	For TPM 1.2: A configuration change was requested to enable, activate, clear, enable, and activate this computer's TPM (Trusted Platform Module) NOTE: This action will clear and turn on the TPM WARNING: Clearing erases information stored on the TPM. You will lose all created keys and access to data encrypted by these keys. Take ownership as soon as possible after this step. Press <CAK> to clear, enable, and activate the TPM Press <RK> to reject this change request and continue For TPM 2.0: A configuration change was requested to clear this computer's TPM (Trusted Platform Module) WARNING: Clearing erases information stored on the TPM. You will lose all created keys and access to data encrypted by these keys. Press <CAK> to clear the TPM Press <RK> to reject this change request and continue
23	SetPCRBanks	A configuration change was requested to change the boot measurements to use <NEW> PCR bank(s). WARNING: Changing the PCR bank(s) of the boot measurements may prevent the Operating System from properly processing the measurements. Please check if your Operating System supports the new PCR bank(s). WARNING: Secrets in the TPM that are bound to the boot state of your machine may become unusable. Press <CAK> to change the PCR bank(s). Press <RK> to keep the PCR bank(s) currently in use. Note: Where <NEW> is the list of PCR bank(s) that would be active after the change is accepted.
24	ChangeEPS	A configuration change was requested to clear this computer's TPM (Trusted Platform Module) and change its identity. WARNING: Clearing erases information stored on the TPM. You will lose all created keys and access to data encrypted with these keys. WARNING: Changing the identity of the TPM may require additional steps to establish trust into the new identity. Press <CAK> to clear the TPM and change its identity. Press <RK> to reject this change request and continue.

Op Value	Operation Name	Confirmation Text
25	SetPPRequiredForChangePCRs_False	A configuration change was requested to allow the Operating System to change the format of boot measurement log without asking for user confirmation in the future. WARNING: Allowing future changes to format of the boot measurement log may affect the Operating System. Press <CAK> to approve future Operating System requests to change the boot log format. Press <RK> to reject this change request and continue
26	SetPPRequiredForChangePCRs_True	Implementers Note: Physical presence confirmation is not necessary. The firmware MAY display a message during boot to the user but MUST not ask the user for confirmation.
27	SetPPRequiredForTurnOn_False	A configuration change was requested to allow the Operating System to turn on the computer's TPM (Trusted Platform Module) without asking for user confirmation in the future. Press <AK> to approve future Operating System requests to provision the TPM Press <RK> to reject this change request and continue
28	SetPPRequiredForTurnOn_True	Implementers Note: Physical presence confirmation is not necessary. The firmware MAY display a message during boot to the user but MUST not ask the user for confirmation.
29	SetPPRequiredForTurnOff_False	A configuration change was requested to allow the Operating System to turn off the computer's TPM (Trusted Platform Module) without asking for user confirmation in the future. Press <AK> to approve future Operating System requests to provision the TPM Press <RK> to reject this change request and continue
30	SetPPRequiredForTurnOff_True	Implementers Note: Physical presence confirmation is not necessary. The firmware MAY display a message during boot to the user but MUST not ask the user for confirmation.

Op Value	Operation Name	Confirmation Text
31	SetPPRequiredForChangeEPS_False	A configuration change was requested to allow the Operating System to maintain the computer's TPM (Trusted Platform Module) without asking for user confirmation in the future. WARNING: Allowing future changes to the TPM's firmware may affect the operation of the TPM and may erase information stored on the TPM. You may lose all created keys and access to data encrypted by these keys Press <CAK> to approve future Operating System requests to maintain the TPM Press <RK> to reject this change request and continue
32	SetPPRequiredForChangeEPS_True	Implementers Note: Physical presence confirmation is not necessary. The firmware MAY display a message during boot to the user but MUST not ask the user for confirmation.
33	LogAllDigests	Implementers Note: Physical presence confirmation is not necessary. The firmware MAY display a message during boot to the user but MUST not ask the user for confirmation.
34	DisableEndorsementEnableStorageHierarchy	A configuration change was requested to disable access to secrets stored in the endorsement hierarchy and enable access to secrets stored in the storage hierarchy of this computer's TPM (Trusted Platform Module) WARNING: Doing so might prevent security applications that rely on the TPM from functioning as expected Press <AK> to change the configuration of the TPM. Press <RK> to reject this change request and continue
35 – 95	Reserved (for TPM Management)	
96	Enable_BlockSIDFunc	A configuration change was requested to issue a Block SID authentication command on subsequent boots.
97	Disable_BlockSIDFunc	A configuration change was requested to disable issuing a Block SID authentication command on subsequent boots.
98	SetPPRequiredForEnable_BlockSIDFunc_True	Implementers Note: Physical presence confirmation is not necessary. The firmware MAY display a message during boot to the user but MUST not ask the user for confirmation.
99	SetPPRequiredForEnable_BlockSIDFunc_False	A configuration change was requested to allow the Operating System to enable blocking SID authentication without asking for user confirmation in the future. Press <AK> to approve future Operating System requests to block SID authentication Press <RK> to reject this change request and continue

Op Value	Operation Name	Confirmation Text
100	SetPPRequiredForDisable_BlockSIDFunc_True	Implementers Note: Physical presence confirmation is not necessary. The firmware MAY display a message during boot to the user but MUST not ask the user for confirmation.
101	SetPPRequiredForDisable_BlockSIDFunc_False	A configuration change was requested to allow the Operating System to disable blocking SID authentication without asking for user confirmation in the future. Press <AK> to approve future Operating System requests to disable blocking SID authentication Press <RK> to reject this change request and continue
102 – 127	Reserved (for Storage Management)	
>= 128	Vendor Specific	

Table 5: User Confirmation Key Mappings

User Confirmation Keys	Key Meaning	Example Actual Key	Text to Represent the Actual Key
<RK>	Reject Key – user rejects the action	ESC key	[ESC]
<AK>	Accept Key – user accepts the action where the action does not cause a potential loss of data	Function F10 key	[F10]
<CAK>	Caution Accept Key – user accepts the action using a different key than the Accept Key to mitigate the potential of accepting an action that may cause the loss of data.	Function F12 key	[F12]

Table 6: User Confirmation keys for Systems without keyboard

User Confirmation Keys	Key Meaning	Example Actual Key	Text to Represent the Actual Key
<RK>	Reject Key – user rejects the action	Volume Down	Volume Down
<AK>	Accept Key – user accepts the action where the action does not cause a potential loss of data	Volume Up	Volume Up
<CAK>	Caution Accept Key – user accepts the action using a different key than the Accept Key to mitigate the potential of accepting an action that may cause the loss of data.	Volume Up	Volume Up

11 Physical Presence Interface Pseudo Code

The pseudo code below is intended to provide design guidance for the implementation of the Physical Presence Interface for TPM management. This does not cover operations for Storage management. As long as normative requirements are adhered to, the PM is free to vary implementation details.

```
/*****************************************************************************
This section of the pseudo code illustrates the behavior of the operating system user
mode components that initiate the Physical Presence Interface call into the firmware.
Note that this section is for understanding only and does not dictate actual OS
behavior.
*****************************************************************************/

OS_Admin_Tool()

If (User_Desires_Functionality_That_Requires_Physical_Presence()) {

       Display_Platform_Specific_Dialog_Documentation()

       If (Software_Physical_Presence_Supported()) {

              Write_Physical_Presence_Interface_Request(OperationValue, Argument)

              // see pseudocode in the method below for more details
              Optional_Register_OS_Component_To_Run_On_Next_Boot(OS_Component_Post_Rebo
ot())

              Execute_Platform_Specific_Transition()
       }
  }


OS_Component_Post_Reboot()

If (Get_Physical_Presence_Interface_Response()!= SUCCESS) {

       Optional_Provide_Failure_Notice()
}
```

```
/*****************************************************************************
This section of the pseudo code illustrates the behavior of the pre-OS environment
including the CRTM and portion of the firmware that acts upon the Physical Presence
Interface requests. Implementation details may differ as the CRTM can be either the
boot block or the entire firmware.
*****************************************************************************/

Main()

CRTM()
Post_ BIOS()
Boot_Into_OS()


CRTM()

Reset_Vector()

If (Physical_Presence_Asserted()) {
       Turn_On_Physical_Presence_Flag()
} Else {
       Lock_Physical_Presence()
}


Post_BIOS()

Argument = 0

If (OperationValue = Read_Physical_Presence_Interface_Request()) {

       RequestConfirmed = False

       If (OperationValue = 1 Or 3) {
              If (NoPPIProvision = True Or PPRequiredForTurnOn = False) {
                     RequestConfirmed = True
              }
       } Else If (OperationValue = 2, 4, 6 Or 7) {
              If (NoPPIProvision = True Or PPRequiredForTurnOff = False) {
                     RequestConfirmed = True
              }
       } Else If (OperationValue = 5) {
              If (NoPPIClear = True Or PPRequiredForClear = False) {
                     RequestConfirmed = True
              }
       } Else If (OperationValue = 8, 9, 10, 11 Or 13) {
              If (NoPPIProvision = True) {
                     RequestConfirmed = True
              }
       } Else If (OperationValue = 12) {
              If (NoPPIMaintenance = True) {
                     RequestConfirmed = True
              }
       } Else If (OperationValue = 14, 21 Or 22) {
              If ((PPRequiredForTurnOn = False And PPRequiredForClear = False) Or
                  (NoPPIClear = True)) {
                     RequestConfirmed = True
              }
       } Else If (OperationValue = 23) {
              Argument = Read_Physical_Presence_Interface_Argument();
              If (PPRequiredForChangePCRs = False) {
                     RequestConfirmed = True
              }
       } Else If (OperationValue = 24) {
              If (PPRequiredForChangeEPS = False) {
                     RequestConfirmed = True
              }
```

```
      } Else If (OperationValue = 15, 17, 19, 25, 27, 29, 31 Or 33) {
            RequestConfirmed = True
      } Else If (OperationValue = 34) {
            If (IsStorageHierarchyDisabled() And PPRequiredForTurnOn = False) {
                  RequestConfirmed = True
            }
            If (IsEndorsementHierarchyEnabled() And PPRequiredForTurnOff = False) {
                  RequestConfirmed = True
            }
      }
      If (RequestConfirmed = False) {
            If (Physical_Presence_Asserted_In_CRTM()) {
                  If (Optional_BIOS_Administrative_Password_Authenticated()){
                        RequestConfirmed = Prompt_Confirmation_Of_Request() within
Timeout Period
                  }
            }
            Response = UserAbort
      }

      If (RequestConfirmed) {
             // see pseudocode in the method below for more details
            Response = Execute_Physical_Presence_Interface_Request(
                                                            OperationValue, Argument)
      }
      Clear_Physical_Presence_Interface_Request()
       Set_Physical_Presence_Interface_Response(Request, Response)

      If (RequestsConfirmed) {
            // a reboot is necessary to activate the operation
             ResetVector()
      }



} Else If (Request_To_Enter_Startup_Setup() within Timeout Period) {

      Standard_Setup()
}

Lock_Physical_Presence()


Execute_Physical_Presence_Interface_Request(OperationValue, Argumet)

For Each TPM_Command in OperationCommands.Get(OperationValue)  {
      ReturnValue = Execute(TPM_Command, Argument)

      If ReturnValue != SUCCESS
            Return ReturnValue

}


/*****************************************************************************
This section of the pseudo code describes functions used in previous sections.
*****************************************************************************/

1.  User_Desires_Functionality_That_Requires_Physical_Presence()

   If true, the user logged in to the OS has chosen to perform an action that requires
   physical presence authorization.  For example, the user has chosen to force clear
   the TPM.

2.  Display_Platform_Specific_Dialog_Documentation()
```

The OS displays platform-specific documentation to inform the user of the procedure that must take place to successfully execute the desired functionality. The platform-specific documentation is multi-lingual for users of different locales. The text may be customized by the PM during OS pre-install time.

The dialog that displays the platform-specific documentation also contains a button to carry out the platform's transition action as specified through ACPI (see Table 5).

If the specified ACPI functions are not implemented, the default dialog alerts the user to consult the documentation that shipped with their platform to execute the desired functionality.

3. Software_Physical_Presence_Supported()

If true, the PM has implemented the software/command method of asserting physical presence. This method reads the TPM permanent flag structure's physicalPresenceCMDEnable flag.

4. Write_Physical_Presence_Interface_Request(OperationValue, Argument)

The OS calls the "Submit TPM Operation Request to Pre-OS Environment 2" Function of the Physical Presence Interface to submit the operation request to the BIOS (see Table 1 and Table 2) and passes in the Argument.

5. Optional_Register_OS_Component_To_Run_On_Next_Boot(OS_Component_Post_Reboot())

The OS optionally registers a component to run post-reboot in order to perform clean up based on the return value. See the OS_Component_Post_Reboot() method in the pseudocode for more details

6. Execute_Platform_Specific_Transition()

The OS executes the action requested by the user in the platform-specific dialog (see Display_Platform_Specific_Dialog_Documentation()). For example, the action may be to shut down the platform.

7. Get_Physical_Presence_Interface_Response()

The OS calls the "Return TPM Operation Response to OS Environment" function of the Physical Presence Interface to read the return value of the request (see Table 1 and Table 2).

8. Optional_Provide_Failure_Notice()

Provides notification of the failure (e.g. write failure to event log).

9. Reset_Vector()

The vector within the CRTM that is first executed upon platform reboot.

10. Physical_Presence_Asserted()

If true, the CRTM has sensed the physical presence assertion of the user. For example, the user has pressed the startup button or inserted a USB dongle. The details of the implementation are vendor-specific.

11. Turn_On_Physical_Presence_Flag()

Sets the PhysicalPresence permanent flag to true by executing TSC_PhysicalPresence(TPM_PHYSICAL_PRESENCE = TPM_PHYSICAL_PRESENCE_PRESENT). At this time, the physicalPresenceLock has already been reset to false due to execution of TPM_Startup(STCLEAR).

12. Lock_Physical_Presence()

Sets the PhysicalPresenceLock = True and PhysicalPresence – False using the command TSC_PhysicalPresence(TPM_PHYSICAL_PRESENCE = TPM_PHYSICAL_LOCK)

13. **Read_Physical_Presence_Interface_Request()**

    Reads from the storage location modified by Write_Physical_Presence_Interface_Request(OperationValue, Argument) and returns this operation value.

14. **Read_Physical_Presence_Interface_Argument()**

    Reads from the storage location modified by Write_Physical_Presence_Interface_Request(OperationValue, Argument) and returns the operation argument.

15. **Physical_Presence_Asserted_In_CRTM()**

    If true, physical presence was asserted in the CRTM. This method reads the TPM permanent flag structure's physicalPresence flag or optionally, a manufacturer-specific flag that was set when physical presence was sensed via Physical_Presence_Asserted().

16. **Optional_BIOS_Administrative_Password_Authenticated()**

    The PM may optionally decide to implement the ability to authenticate with a BIOS administrative password before a TPM command that requires physical presence can be executed. If true is returned, the BIOS administrative password has been authenticated.

17. **Prompt_Confirmation_Of_Request()**

    This is the dialog displayed to the user that prompts for confirmation of the Physical Presence Interface request. The return value is True or False depending on whether the user confirms or rejects the request, respectively. The confirmation should be implemented such that the user can understand at a high level the security implications of the operation and must actively choose to execute the operation (e.g. the default should not be to confirm the operation).

    See Table 5 for guidance on the dialog text.

18. **Clear_Physical_Presence_Interface_Request()**

    Clears the storage area modified by Write_Physical_Presence_Interface_Request(OperationValue, Argument), for example by setting it to a value of 0.

19. **Set_Physical_Presence_Interface_Response(Request, Response)**

    Writes the response so that it can be retrieved by calling the "Return TPM Operation Response to OS Environment" function of the Physical Presence Interface.

20. **Request_To_Enter_Startup_Setup()**

    There must be a way to enter standard setup of the BIOS when no Physical Presence Interface requests are present. In this pseudo code, standard setup cannot be entered if a request exists on the Physical Presence Interface. The BIOS implementation may choose a different method.

12 Scenarios with recommended default for Persistent Firmware TPM Management Flags

This section is informative.

The flags may be set independently and their values may be changed through the use of physical presence operations or other vendor specific configuration utilities. Table 7 contains a list of the flags for a TPM 1.2. A firmware implementation may choose to implement separate flags to enable / activate and disable / deactivate a TPM 1.2.

Table 7: Persistent Firmware TPM Management Flags for TPM 1.2

Flag Description	Operations permitted without user confirmation when set	Recommended Default
NoPPIProvision	Enable Disable Activate Deactivate SetOwnerInstall_True SetOwnerInstall_False SetOperatorAuth	True
NoPPIClear	TPM_ForceClear	False
NoPPIMaintenance	Deferred Physical Presence-unownedFieldUpgrade	False

12.1 Persistent Firmware TPM Management Flags for TPM 2.0

The Persistent Firmware TPM Management Flags for TPM 2.0 may be used on systems with TPM 1.2. This specification does not provide a mapping between these flags and the operations they should be applied to.

Different scenarios ask for different settings of the flags. The following scenarios describe different requirements and the respective recommended default values.

Table 8: Scenario 1: PC Client system for end customer. A customer who owns his device and wants to exercise control over it will want to be prompted for changes to the content of the TPM, but is ok with the TPM being enabled automatically.

Table 8: Scenario 1

Flag Description	Operations permitted without user confirmation when set	Recommended Default
PPRequiredForTurnOn	Enable	False
PPRequiredForTurnOff	Disable	True
PPRequiredForClear	TPM2_Clear	True
PPRequiredForChangeEPS	TPM2_ChangeEPS	True
PPRequiredForChangePCRs	SetPCRBanks	False

Table 9: Scenario 2: A PC managed by the IT department of the company. The company wants to remotely manage the PC and the TPM. If possible, the interaction should be minimal and the TPM should always be enabled.

Table 9: Scenario 2

Flag Description	Operations permitted without user confirmation when set	Recommended Default
PPRequiredForTurnOn	Enable	False
PPRequiredForTurnOff	Disable	True
PPRequiredForClear	TPM2_Clear	False
PPRequiredForChangeEPS	TPM2_ChangeEPS	False
PPRequiredForChangePCRs	SetPCRBanks	False

Table 10: Scenario 3: A server machine managed by a remote administrator. The administrator wants to minimize interaction with the machine. Firmware should always execute all TPM operations. Operations requiring physical presence are asserted to a PP pin on the TPM. The pin is controlled by a board management controller that is in turn controlled by the remote administrator.

Table 10: Scenario 3

Flag Description	Operations permitted without user confirmation when set	Recommended Default
PPRequiredForTurnOn	Enable	False
PPRequiredForTurnOff	Disable	False
PPRequiredForClear	TPM2_Clear	False
PPRequiredForChangeEPS	TPM2_ChangeEPS	False
PPRequiredForChangePCRs	SetPCRBanks	False

