TCG Physical Presence Interface Specification

Version 1.10 Final
Revision 1.00
10 June 2009
For TPM Family 1.2; Level 2

Disclaimers, Notices, and License Terms

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

This document is copyrighted by Trusted Computing Group (TCG), and no license, express or implied, is granted herein other than as follows: You may not copy or reproduce the document or distribute it to others without written permission from TCG, except that you may freely do so for the purposes of (a) examining or implementing TCG specifications or (b) developing, testing, or promoting information technology standards and best practices, so long as you distribute the document with these disclaimers, notices, and license terms.

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

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

Table of Contents

Change History

Revision	Date	Description
1.00	5 April, 2007	• Initial release of Version 1.00.
1.10	10 June, 2009	• Added new action 12: Deferred Physical Presence • Added “Submit TPM Operation Request to Pre-OS Environment 2”

Tables

Corrections and Comments

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

TPM Dependency and Requirements

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** TCG EFI Protocol Specification Version 1.20

1. Physical Presence Overview

Start of informative comment:

Physical Presence is a form of authorization to perform certain TPM functions. This authorization must come from the platform operator. These functions perform provisioning and de-provisioning operations such as allowing ownership and clearing owner when the current owner auth value is unknown.

There are two methods defined in the TPM specification for providing an indication of physical presence. The first and most obvious method is the hardware method. An example of an implementation of this method is button on the front of the platform wired to a pin on the TPM. Pressing the button causes the pin to change to the polarity determined by the TPM to set the TPM's internal physical presence flag. Using this hardware method, commands requiring the indication of physical presence can be executed at any time including full OS presence.

Providing a wire along with a button or switch to the outside of the platform is prohibited in some cases due to cost, form factor, usage or any number of reasons. For this reason, the TPM provides the other method of asserting physical presence called the "command method". One of the properties required of the indication of physical presence must be done by the operator who is typically the user physically at the platform. This requires the command method be restricted to available only while the platform state can provide this assurance. On a PC Client, this state is during the boot strapping of the platform prior to the availability of a network stack or untrusted software – specifically during the S-CRTM. These commands are therefore available only after TPM_Init and must be "closed" once the S-CRTM determines the user has not provided an indication of physical presence. This specification is targeted at platforms that implemented this "command method" of providing an indication of physical presence.

End of informative comment.

2. Physical Presence Interface

Start of informative comment:

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

This 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 BIOS -- a significant deterrent to the mass deployment of trusted platforms, especially across heterogeneous platforms.

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

In order to minimize the platform storage required to implement this Interface, the OS is restricted to requesting the execution of at most one TPM operation at a time. A TPM 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 TPM operations, all standard functionality involving physical presence can be carried out within one restart of the OS.

For definitions of the TPM commands that are associated with each TPM operation identified in the Interface, see Part 3 of the v1.2 TCG Main specification.

For more information on ACPI, see the materials available for download at http://acpi.info/. Refer to Section 9.15.1 of the ACPI 3.0 spec for information on the _DSM control method object. Refer to section 17.2.5 of the ACPI 3.0 spec for information about data types. For example, the "Integer" data type is a 64-bit little-endian unsigned integer and the "String" type is a null-terminated ASCII string with up to 200 characters.

Defined Uses:

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

1. Within the OS environment, the user requests a TPM operation that requires physical presence.
2. The OS informs the user of the platform-specific procedure that must take place to successfully execute the TPM operation.
3. The OS communicates the requested TPM operation to the BIOS through ACPI. The platform's ACPI handler stores the requested TPM operation in a location accessible to the pre-OS environment (e.g. CMOS, Flash ROM, TPM NVRAM, etc.).
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 BIOS reads the OS's TPM operation request from its stored location.

6. The BIOS confirms the physical presence of the user.

The BIOS prompts the user for confirmation to execute the requested TPM operation. Note that the BIOS may measure and execute a platform manufacturer pre-boot environment utility to prompt the user for confirmation.

If the user confirms the TPM operation, the BIOS executes the TPM commands that carry out the requested operation.

The BIOS clears the storage location containing the TPM operation request.

The BIOS communicates the response of the requested TPM operation back to the OS through ACPI. 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.

The OS loads, determines the response of the requested operation, and takes action as necessary.

Table 1 identifies the set of actions and prompts available to the BIOS or OS.

Security Implications:

The platform manufacturer should not assume that the OS imposes security constraints on what entity can request TPM operations through the Physical Presence Interface. There is always the possibility that a TPM operation was requested by malicious software without knowledge of the platform user.

Hence, it is imperative that the pre-OS environment verifies the physical presence of the user and confirms that this physically-present user in fact requested the execution of the TPM 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). 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.

There also exists the possibility that malicious software can attempt to launch a denial of service attack on the platform by repeatedly invoking the ACPI call to submit TPM operation requests. It is the responsibility of the platform manufacturer to ensure that such abuse will not cause irreparable damage to the platform (e.g. by burning out flash memory used to store the requests).

Storage Implications:

The minimum platform storage required to implement this Interface is as follows:

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

4 bits – Stores the most recent operation request acted upon by the BIOS

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

The data above assumes that no vendor-specific 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 operation responses consisting of 99 TPM fatal errors, 4 TPM non-fatal errors, and 2 additional Physical Presence Interface-specific responses (User Abort and BIOS Failure).

Additional Notes:

This Interface assumes that the BIOS 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 BIOS 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 BIOS may need an additional reboot cycle 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 a permanent activation or deactivation to take into effect (see main spec related to TPM_PERMANENT_FLAGS.deactivated and TPM_STCLEAR_FLAGS.deactivated for more information). The additional reboot may need to take place in the middle of a multi-command physical presence operation.

Affected operations are:

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
Operation 14:	Clear + Enable + Activate

When the BIOS is carrying out these operations that require an additional BIOS reboot cycle, the user should not be prompted again for confirmation. In this way, the only indication to the user of the additional reboot cycle is the reappearance of the BIOS 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 utilizing the hardware method of indicating physical presence do not need this interface. Platforms are allowed to implement both method in which case the user may choose the one most convenient. The utility prompting the user to provide the indication of physical presence should be aware of the method implemented on the platform and prompt the user accordingly. This information can be found by either querying the TPM’s capabilities (e.g., calling TPM_GetCapability) directly or referring to the platform’s credentials.

See note in Section 3, TPM Functions and Confirmation Dialogs in the informative section under the subsection titled ***Platforms implementing the Hardware method*** for a discussion on platforms that implement the hardware method of indicating physical presence.

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.

End of informative comment.

All PC Client Platforms MUST implement this interface.

2.1 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 BIOS MUST support the Physical Presence Interface as defined below.
2. A platform manufacturer that implements the Physical Presence Interface SHOULD continue to expose TPM management within the BIOS setup, to allow users to configure the TPM independent of an available OS.
3. ACPI is the communication mechanism between the OS and BIOS for requesting TPM operations and responding to these requests. Table 1 below defines the ACPI functions exposed by the BIOS, and the behavior that the OS can expect upon invoking these functions.
4. These ACPI functions reside in the _DSM control method object. The UUID function identifier to be used exclusively for the Physical Presence Interface is:

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

Below is the list of ACPI Physical Presence Functions:

2.1.1 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 specification, the return value MUST be “1.1” as a NULL terminated ASCII string

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

Example:

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

2.1.2 Submit TPM Operation Request to Pre-OS Environment

Command Status:

Deprecated, 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]

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 is deprecated but MUST be implemented.

This function allows the OS to submit a request for a TPM operation to be executed in the pre-OS environment (see Table 1 below for a list of available TPM 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 BIOS 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 TPM operation.

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

If 2 is returned, the BIOS 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 TPM 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 a TPM operation request of value 0 to clear any previous requests.

Start of informative comment:

Many as-built operating systems and hardware implementations have implemented the Submit TPM Operation Request to Pre-OS Environment function differently than this specification for the third 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 pass a buffer of an Integer instead of Package containing an Integer. The ACPI Machine Language Interpreter on those operating systems is aware the argument type is 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 which should be the first element of the Package. The result on operating systems which 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 correctly.

End of informative comment.

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.

2.1.3 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 = 1

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 TPM operation requested by the OS

Description:

0: None

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

Functional Behavior:

This function returns the pending TPM operation that was previously requested, 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 BIOS is unable to return meaningful information due to an internal failure. In this case, the second integer is undefined.

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

If 0 is returned as the second integer, no pending TPM operations exist. No previous request has been submitted.

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

Examples:

A return value of {0, 0} indicates that no pending TPM operations exist.

A return value of {1, 0} indicates that the function failed to retrieve the pending TPM operation.

2.1.4 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 TPM 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 BIOS for execution of a requested TPM 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.

If 1 is returned, the OS must shut down the machine to execute . A physically-present user restarts the machine.

If 2 is returned, the OS must cause a warm reboot of the machine.

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.

Examples:

None

2.1.5 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 TPM operation request

Description:

0: None

>0: Operation value of the most recent request
[see Table 1]

Integer 3:

Purpose: Response to the most recent TPM operation request

Description:

0: Success

1..0x00000FFF: Corresponding TPM error code

0xFFFFFFF0: User Abort

0xFFFFFFF1: BIOS Failure

Functional Behavior:

This function allows the BIOS to communicate the response to the most recent TPM operation request it acted upon. The function returns both the most recent request and the response to that request.

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 response is available to the OS that originated the request; another OS may submit a new request, overwriting the response to the previous request.

If 1 is returned as the first integer, the BIOS 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 previously-requested TPM 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 TPM operation request seen by the BIOS (see Table 1 below for a list of available TPM 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 (e.g. the third integer) to the operation request (e.g. the second integer).

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

A response value from 1 to 0x00000FFF inclusive corresponds to the TPM error codes defined in Chapter 16 of the TPM Main Specification – Part 2 TPM Structures document.

If 0xFFFFFFF0 is returned as the response, the user failed to confirm the TPM operation request.

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

Examples

A return value of {1, 0, 0} indicates that a BIOS internal failure prevented the function from retrieving meaningful information.

A return value of {0, 0, 0} indicates that no previously-requested TPM operations exist and hence no response exists.

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

A return value of {0, 6, 0xFFFFFFF0} indicates that the last TPM 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 TPM operation request to enable and activate the TPM failed to execute successfully due to an internal BIOS error.

2.1.6 Submit preferred user language

Command Status:

Deprecated, Manditory

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/englangn.html

Returns:

Type: Integer

Purpose: Function Return Code

Description:

0: Success

1: Language Not Supported

2: General Failure

3: Not implemented

Functional Behavior:

This function allows the OS to communicate the user's preferred language to the BIOS. This function is deprecated for conventional BIOS'es. It is also deprecated for EFI BIOS'es because in EFI, the "Lang" variable which can be set by an EFI OS using the "SetVariable" function encapsulates the user's preferred language information in the 3-character ISO-639-2 format. Because of these reasons, BIOS SHOULD take no action and return code 3: Not implemented.

If this function is available, the OS queries the current user's preferred language and submits this information to the BIOS using the 2-character language code defined by the ISO-639-1 standard.

If 0 is returned, the language code can be read and acted upon by the BIOS 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 displays the confirmation dialog in the user's preferred language.

If 1 is returned, the BIOS does not support the language indicated. For example, the BIOS has not localized the confirmation dialog to that language.

If 2 is returned, the BIOS is otherwise unable to read and act upon the language code.

The OS may call this function multiple times before transitioning to the pre-OS environment. However, only the last submitted language code is valid.

In the case where the BIOS can read an operation request but is unable to obtain the user’s preferred language, the BIOS should prompt the user to select a preferred language. The BIOS may also use its default language to display the confirmation dialog.

Support for this function requires the BIOS to allocate additional platform storage. The minimum additional platform storage required depends on the number of languages supported by the BIOS. For example, if the BIOS is localized for 32 languages, 5 additional bits are required.

Example:

A submitted language code of “en” indicates that the user’s preferred language is English.

To distinguish between variations of a single language, Windows Vista and above include additional characters that appear after the 2-character ISO-639-1 string.

For example:

- Simplified Chinese is "zh-CHS"
- Traditional Chinese is "zh-CHT"
- U.S. English is "en-US"

Contact Microsoft for more information on Windows language strings and for testing of this function.

2.1.7 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 = 1

Arg2 (Integer): Function Index = 7

Arg3 (Package): Arguments = Package --

Type: Integer

Purpose: Operation Value of the Request

Description: [see Table 1]

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 a TPM operation to be executed in the pre-OS environment (see Table 1 below for a list of available TPM 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 BIOS 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 TPM operation.

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

If 2 is returned, the BIOS 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 TPM operation.

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 a TPM operation request of value 0 to clear any previous requests.

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.

2.2 Parameter Passing

Start of informative comment:

With the exception of Operation #13 defined in Table 1, all parameters between the OS and the BIOS 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 BIOS 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 BIOS is permitted.

End of informative comment.

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

3. TPM Functions and Confirmation Dialogs

Start of informative comment:

This section is intended to provide design guidance for the BIOS dialogs used to confirm that the physically-present user in fact requested execution of the TPM operation. Each operation request is mapped to the dialog presented to the user to accept or reject the operation. Only the mandatory operations values (1-11, 14) are illustrated.

In consideration of the limited space in the BIOS 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 ease of reading, reusable text is shaded.

For security purposes, note that the key used to confirm the Clear operation differs from the confirmation key used for other operations.

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. Function 6 in the Physical Presence Interface allows the BIOS to determine the user’s preferred language from the OS. If this function is not implemented, the BIOS should provide an alternate 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.).

The location(s) used by the BIOS 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 TPM’s operations that are intended to be affected will not perform as the user expects. This does not introduce a vulnerability to the user.

For usability reasons it may be desirable to not prompt the user to provide more than one assertion of physical presence. However, as stated in the PC Client specification, the checking of the indication of physical presence does need to be a trusted operation. Therefore, if between Platform Resets, the BIOS stores the assertion of the indication of physical presence, that location must be trusted.

Platforms implementing the Hardware method

Platforms implementing the hardware method of asserting physical presence can get by without using this interface. On these platforms there is no requirement for OS or utilities to use this interface but requiring all PC Client to implement this interface provides a consistent usage model across all PC Client platforms. This allows an OS or its utility to present to the user a single method for managing the provisioning the platform without querying the platform’s capabilities.

While the text and confirmation keys in Table 1 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.

The confirmation keys are selected for most keyboards and expected user experiences. Further, while the keys selected do not conflict with a majority of BIOS implementation, some conflicts may occur. The BIOS provider may have used the confirmation keys in Table 1 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 BIOS is allowed to use other keys when such conflicts occur.

End of informative comment.

1. If a utility prompts the user to indicate physical presence using the hardware method, the utility SHOULD call these functions.
2. The text in Table 1 is the preferred confirmation text wording to be displayed to the user. The BIOS may provide alternate confirmation text wording that differs from Table 1 based upon the platform’s expected user experience and language; however, the meaning of each message MUST be retained. The text in Table 1 MAY be translated into languages other than the English text indicated in Table 1 provided the meaning of the text in Table 1 is conveyed in the translated text.
3. Table 1 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 SHOULD be used for most environments are provided in Table 2. All occurrences of confirmation keys must be consistent for each operation for the same platform, i.e., the BIOS for each platform MUST use the same key associated to each label. The text representing the key SHOULD exclude the angle bracket (i.e., the ‘<’ and ‘>’ characters) and SHOULD use the string representing the actual key indicated in Table 2 when presenting the text in Table 1.
4. The BIOS may provide alternate confirmation keys that differ from Table 2 based upon the platform’s keyboard configuration or expected user experience.
5. In order to persist, the requested TPM operation to the next boot cycle, the BIOS must allocate a location for storing the TPM operation’s corresponding value. Table 1 below maps each TPM operation to its associated values and TPM commands. The TPM commands for each operation must be executed in the order specified.
6. The operation value of 0 is reserved to indicate that no Physical Presence Interface operation has been requested. Operations with values at or above 128 (0x80) are designated for vendor-specific usage.
7. The BIOS MUST implement operations 0 to 11 and operation 14. The BIOS MAY implement all other operations.
8. Note that the order is significant for some of the TPM command sequences. Also a reboot required for some changes to take affect. This is noted with the keyword: <PLATFORM RESET>.
9. The BIOS is responsible for tracking and performing operations that require a PLATFORM RESET.
10. The location for tracking the pending operation, including the tracking of necessary PLATFORM RESET operations, does not need to be in secure or trusted location.

11. If during an operation that requires a Platform Reset, the BIOS stores the assertion of an indication of physical presence across the Platform Reset, that location must be trusted.

Operation Value	Operation Name	Related TPM Command(s) sent by BIOS	Confirmation Text
0	No Operation	*No operation*	None
1	Enable	TPM_PhysicalEnable	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	TPM_PhysicalDisable	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	TPM_PhysicalSetDeactivated = FALSE
		`<PLATFORM RESET>`
			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

Operation Value	Operation Name	Related TPM Command(s) sent by BIOS	Confirmation Text

4	Deactivate	TPM_PhysicalSetDeactivated = TRUE
		`<PLATFORM RESET>`
			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	TPM_ForceClear `<PLATFORM RESET>`	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

Operation Value	Operation Name	Related TPM Command(s) sent by BIOS	Confirmation Text
6	Enable + Activate	TPM_PhysicalEnable TPM_PhysicalSetDeactivated = FALSE <PLATFORM RESET>	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
7	Deactivate + Disable	TPM_PhysicalSetDeactivated = TRUE TPM_PhysicalDisable <PLATFORM RESET>	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	TPM_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

Operation Value	Operation Name	Related TPM Command(s) sent by BIOS	Confirmation Text
9	SetOwnerInstall_False	TPM_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	TPM_PhysicalEnable
		TPM_PhysicalSetDeactivated = FALSE
		`<PLATFORM RESET>`
		TPM_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

Operation Value	Operation Name	Related TPM Command(s) sent by BIOS	Confirmation Text
11	Deactivate + Disable + SetOwnerInstall_False	TPM_SetOwnerInstall = FALSE TPM_PhysicalSetDeactivated = TRUE TPM_PhysicalDisable <PLATFORM RESET> Note: The BIOS MAY perform the PLATFORM RESET immediately after the TPM_PhysicalSetDeactivated = TRUE but that requires tracking the next command.	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	TPM_SetCapability -> setValue (Cap= TPM_SET_STCLEAR_FLAGS; SubCap= TPM_SD_DEFERREDPHYSICALPRESENCE ; value= unownedFieldUpgrade Implementer's note: The BIOS 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.	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

Operation Value	Operation Name	Related TPM Command(s) sent by BIOS	Confirmation Text
13	SetOperatorAuth	TPM_SetOperatorAuth, with operatorAuth prompted by the BIOS	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	TPM_ForceClear
		TPM_PhysicalEnable
		TPM_PhysicalSetDeactivated = FALSE
		`<PLATFORM RESET>`
			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

15 – 127	Reserved	Reserved, do not implement or use	
>=128	Vendor Specific	TPM commands mapping to vendor specific operation	

Table 1: TPM functions for and Confirmations of Physical Presence Interface Operations

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	Funtion 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 2: User Confirmation Key Mappings

4. Physical Presence Interface Pseudocode

Start of informative comment:

The pseudocode below is intended to provide design guidance for the implementation of the Physical Presence Interface. As long as normative requirements are adhered to, the platform manufacturer is free to vary implementation details.

End of informative comment.

```
/******************************************************************************
This section of the pseudocode illustrates the behavior of the operating system
user mode components that initiate the Physical Presence Interface call into
the BIOS. 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)

              // see pseudocode in the method below for more details
              Optional_Register_OS_Component_To_Run_On_Next_Boot(OS_Component_Po
              st_Reboot())

              Execute_Platform_Specific_Transition()
       }
  }
```

OS_Component_Post_Reboot()

```
If (Get_Physical_Presence_Interface_Response() != SUCCESS) {

       Optional_Provide_Failure_Notice()
}
```

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

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()

```
If (OperationValue = Read_Physical_Presence_Interface_Request()) {

       RequestsConfirmed = False

       If (Physical_Presence_Asserted_In_CRTM()) {

              If (Optional_BIOS_Administrative_Password_Authenticated()) {
                     RequestConfirmed = Prompt_Confirmation_Of_Request()
              }
       }

       Response = UserAbort

       If (RequestsConfirmed) {

              // see pseudocode in the method below for more details
              Response =
              Execute_Physical_Presence_Interface_Request(OperationValue)
       }

       Clear_Physical_Presence_Interface_Request()
       Set_Physical_Presence_Interface_Response(Request, Response)

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



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

       Standard_Setup()
}

Lock_Physical_Presence()
```

Execute_Physical_Presence_Interface_Request(OperationValue)

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

       If ReturnValue != SUCCESS
              Return ReturnValue

}
```

```
/*******************************************************************************
This section of the pseudocode 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 platform manufacturer 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 1).

   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 platform manufacturer 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)

   The OS calls Function b) of the Physical Presence Interface to submit the TPM operation request to the BIOS (see Table 1).

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 Function e) of the Physical Presence Interface to read the return value of the request (see Table 1).

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) and returns this TPM operation value.

14. 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().

15. Optional_BIOS_Administrative_Password_Authenticated()

    The platform manufacturer 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.

16. 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 3 in the next section for guidance on the dialog text.

17. Clear_Physical_Presence_Interface_Request()

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

18. Set_Physical_Presence_Interface_Response(Request, Response)

    Writes the response so that it can be retrieved by calling Function e) of the Physical Presence Interface (see Table 1).

19. 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.