TCG Specification

TPM 2.0 Mobile Command Response Buffer Interface

Family "2.0"

Level 00 Revision 12

16 December 2014

Contact: admin@trustedcomputinggroup.org

TCG Published

Copyright © TCG 2014

TCG

Copyright © 2014 Trusted Computing Group, Incorporated.

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.

Acknowledgements

TCG acknowledges the following contributors to this specification:

Alec Brusilovsky, Andreas Fuchs, Andrew Martin, Ariel Segall, Atul Shah, Carlin Covey, Carlton Northern, Cedric Colnot, David Wooten, Dean Liberty, Emily Ratliff, Gilles Peskine, Graeme Proudler, Hadi Nahari, Hervé Sibert, Ira McDonald, Janne Uusilehto, John Mersh, John Padgette, Joshua Schiffman, Kathleen McGill, Michael Chan, Michael Peck, Niall O'Donoghue, Paul England, Paul Waller, Rob Spiger, Ronald Aigner

CONTENTS

1 Scope and Audience

The Trusted Computing Group TPM 2.0 [1] specification defines a Trusted Platform Module (TPM).

This specification defines an interface between a TPM and software. This interface is the Command/Response Buffer Interface (CRB). The CRB design is intended to work with a large number of hardware implementation options. With this interface, it is possible to write a driver that can interact with a TPM whether it is implemented as a discrete component on a peripheral bus or in an execution mode in a Protected Environment.

Designers, developers and implementers of Trusted Computing technologies in mobile platforms are the target audience for this specification.

1.1 Key words

The upper-case key words “SHALL,” “SHOULD” and “MAY” in this document indicate normative statements and are to be interpreted as described in [3].

1.2 Statement Type

There are two distinctive kinds of text throughout this reference architecture: *informative comments* and *normative statements*.

Most of the content consists of informative comments that explain why the architecture is as defined, but which do not constrain implementation in any specific way.

Normative statements SHALL be obeyed if a standards compliant implementation is to be created. Normative statements are indicated by the presence in a statement of any of the upper-case key words listed in Section 1.1.

1.3 References

[1] Trusted Computing Group, Trusted Platform Module, Version 2.0, Parts 1-4, 2013

[2] ACPI General Specification, version 1.1

[3] IETF, RFC 2119, March 1997

1.4 Document structure

This document is divided and ordered into sections so that concepts are introduced in a logical sequence.

Section 1.5 contains a reference glossary defining the terms and definitions used throughout the specification.

Section 2 contains an introduction to the protocol.

Section 3 provides definitions for data structures used in the protocol.

Section 4describes possible implementations of the protocol.

1.5 Glossary

Glossary Term	Description
Buffer	Data structure used for transport to and from the TPM.
CLEAR	Bit with a value of zero (0), or the action of causing a bit to have the value of zero (0).
Command	The values sent to the TPM to indicate the operation to be performed.
Failure mode	Mode in which the TPM returns TPM_RC_FAILURE in response to all commands except TPM2_GetTestResult() or TPM2_GetCapability().
Octet	Eight bits of data.
Response	Values returned by the TPM when it completes processing of a command.
SET	Bit with value of one (1), or the action of causing a bit to have the value of one (1).
TPM Device Reset	Resetting the TPM internal state due to _TPM_Init.
Trusted Platform Module TPM	Implementation of the TPM Library Specification; Family 2.0; Level 00; Revision 103 or later.

1.6 Abbreviated Terms

Abbreviated Term	Description
ACPI	Advanced Configuration and Power Interface
CPU	Central Processing Unit
CRB	Command Response Buffer (interface)
FIFO	First-In First-Out. Refers to a pipeline access to the TPM device. Octets are written to the pipeline and are read in the order they have been written.
MSO	Most Significant Octet.
OS	Operating System
RSA	Rivest, Shamir and Adleman
TCG	Trusted Computing Group
TPM	Trusted Platform Module
TPM_	Prefix for an indication passed from the system interface of the TPM to a Potected Capability defined in the TPM Library Specification
TPM2_	Prefix for a command defined in the TPM Library Specification

2 Interface Overview

Software interacting with the TPM often directs commands through a TPM driver. The TPM driver performs the actual device interface access, which in the case of this specification, implies the manipulation of the Command and Response Buffer (CRB) interface.

The terms “software” and “TPM driver” are used interchangeably.

2.1 Introduction

The basic concept for the CRB interface is that the TPM’s command and response buffers are defined as linear ranges of memory. Software may put data in the TPM’s command buffer by writing to consecutive addresses in the processor’s address space. Similarly, the driver may read data from the TPM’s response buffer by reading from consecutive addresses in the processor’s linear address space.

Software is restricted to writing/reading the bytes of the command/response buffer to consecutive addresses. Therefore, the data is presented to the TPM in a consumable form whether the command/response buffer is actual linear memory or a hardware FIFO.

The control flow for the CRB is a simple handshake through a memory-mapped control structure. This control structure is also defined such that it can either be in actual memory or be hardware registers on a discrete TPM.

Including the control structure, the three memory areas comprise the entirety of the CRB. There are no constraints on how those three memory areas are provided. They can all be in system RAM, or all be in device memory, or any combination. The control structure needs to occupy a unique address space but the command and response buffers may overlap (or be the same).

The interface definition provides extensions for power management of the TPM for those systems that need this capability. The power management protocol is also defined for simplicity so that the complexities of the device power management are largely hidden from the driver while maintaining a large degree of flexibility in the device’s implementation of power management (if any).

2.2 Basic Protocol

The TPM processes one command at a time. Software will place the command to be executed into the Command buffer and SET *Start* in the command structure. The TPM will detect that *Start* is SET and processes the command in the Command buffer. The TPM finishes command processing by putting its response in the Response buffer and clearing *Start*.

While *Start* is SET, the driver will not access the command or response buffer. While *Start* is CLEAR, the TPM will not access the command or response buffer.

An implementation may use polling or interrupts to signal when *Start* changes. If interrupts are used, the TPM would receive an interrupt when *Start* is SET and the driver would receive an interrupt when *Start* is CLEAR. The TPM may use interrupts and the driver may use polling, or vice versa.

Power management is added by including an Idle/Ready handshake. When the driver has nothing for the TPM to do, it SETS *goIdle* to indicate to the TPM that there is currently no work to be done. The TPM may power down. If it does, it will SET *Idle*. When the driver has a command for the TPM, it will SET *cmdReady* and wait for *Status* to CLEAR. When *Status* is CLEAR, the driver may start loading the Command buffer.

Details of the protocols are in the following sections.

2.3 Discoverability

TPM support for the CRB interface depends on the TPM implementation. For systems that support ACPI, a possible way to identify a TPM with the CRB interface is the Advanced Configuration and Power Interface (ACPI) table for TPM 2.0, as defined in the *TCG ACPI General Specification*, version 1.1.

The ACPI table contains a field to identify the communication interface for the TPM 2.0 devices. It also contains the address of the Control Area that further allows the discovery of command and response buffer addresses.

Other TPM implementations may publish the Control Area at a fixed address. Because the control area contains fields for the addresses of command and response buffer, these implementations only need to standardize the location of the Control Area.

3 Structures

3.1 Control Area

The Control Area structure contains status fields as well as physical addresses and sizes of the command and response buffer.

Field	Byte Length	Offset	Description
Request	4	00_{16}	Used to control power state transition.
Status	4	04_{16}	Used to indicate a status.
Cancel	4	08_{16}	Used to abort command processing.
Start	4	$0C_{16}$	Used to indicate that a command is available for processing.
Interrupt Control	8	10_{16}	Reserved.
Command Size	4	18_{16}	Size of the Command Buffer.
Command Address	8	$1C_{16}$	This field contains the physical address of the Command Buffer.
Response Size	4	24_{16}	Size of the Response Buffer.
Response Address	8	28_{16}	This field contains the physical address of the Response Buffer.

Table 1: CRB Control Area Layout

The CRB interface does not prescribe a specific access pattern to the fields of the Control Area. The fields have to be neither read nor written as a whole. Access may be smaller than the field size and can be random.

3.2 Request

This field can be used to negotiate power state transitions between software and the TPM. Software sets this field to indicate the request to transition into a different power state. The TPM will reflect the current state in the Status field. If necessary, software may trigger an interrupt to indicate the state change request to the TPM.

Bit	Name	Description
0	cmdReady	SET by software to transition TPM from Ready state into Idle state.
1	goIdle	SET by software to transition TPM from Idle state into Ready state.
31:2	Reserved	Reserved.

Table 2: Bit layout of CRB Request field

Software uses the Request field to indicate the requested behavior from the TPM. When software SETs the *goIdle* bit, it indicates that it finished accessing the Command and Response Buffer for now. The TPM may go into the “Idle” state. It may use this state to conserve power or perform other tasks. To signal the “Idle” state, the TPM will SET *tpmIdle* in *Status* and CLEAR *goIdle* in *Request*.

Software will SET *cmdReady* to indicate that that the TPM should prepare to receive the next command. To signal that it is ready to receive the next command (the “Ready” state), the TPM will CLEAR *tpmIdle* in *Status* and CLEAR *cmdReady* in *Request*.

Software should not use *Request* with TPM implementations that do not support a low power state (the "Idle" state).

The initial value of this field (when software has first access to the control area) is undetermined. The state of the TPM is reflected in *Status*. When software uses *Request*, it will write the whole field when SETting a bit in this field. It is good practice if the firmware sets this field to zero before making it visible to the TPM driver.

Some platforms may be unable to trigger requests based on changes of *Request* and may require an additional notification. One possible implementation is the use of an ACPI method (see *TCG ACPI General Specification* version 1.1). For platforms without ACPI, an alternative notification such as interrupts could be used.

3.3 Status

This field is used to communicate TPM state to software. If *Status* is CLEAR, the TPM is in the "Ready" state and is able to receive a command.

Bit	Name	Description
0	Error	SET by the TPM in case of unrecoverable error.
1	tpmIdle	SET by the TPM if in low power state.
31:2	Reserved	Reserved.

Table 3: Bit layout of CRB Status field

tpmIdle is SET by the TPM to indicate to software that the TPM is in the "Idle" state. Software has to SET *cmdReady* in *Request* to initiate a transition of the TPM into the "Ready" state.

Error indicates an unrecoverable error condition in the TPM that cannot be communicated by the error return codes defined in the TPM Library Specification. No assumption about the state of the TPM should be made when *Error* is SET.

This field should reflect the state of the TPM when the Control Area is initialized, for instance by firmware before making it visible to the TPM driver.

3.4 Cancel

Software may use *Cancel* to request that TPM processing of the currently executing command be terminated.

Bit	Name	Description
0	Cancel	SET by software to request cancelation of the current command.
31:1	Reserved	Reserved.

Table 4: Bit layout of CRB Cancel field

When *Cancel* is SET by software and the TPM is executing a command, the TPM will stop executing the current command at the earliest convenient point, that is, when the TPM has the ability to check *Cancel*. For most commands, it is expected that the TPM will complete the command and provide a normal response. For commands that are long-running (for example, RSA key generation), the TPM may exit command execution and return a TPM response of TPM_RC_Cancelled.

Cancel has only meaning while *Start* is SET. Software should CLEAR *Cancel* when *Start* is CLEAR, for instance before SETTING *Start*, that is, before issuing a command.

Cancel should be CLEAR when the Control Area is initialized, for instance by firmware before making it visible to the TPM driver.

Some platforms may be unable to trigger cancelation of TPM commands based on changes of *Cancel* and may require an additional notification. One possible implementation is the use of an ACPI method (see *TCG ACPI General Specification* version 1.1). For platforms without ACPI, an alternative notification such as interrupts could be used.

3.5 Start

Software will SET this field to indicate to the TPM that a new command has been placed in the Command buffer. The TPM will CLEAR *Start* when it completes processing a command.

Bit	Name	Description
0	Start	SET by software to signal that Command Buffer contains a new command. CLEARED by the TPM when a response to the command is in the Response Buffer.
31:1	Reserved	Reserved.

Table 5: Bit layout of CRB Start field

Start is SET by software to signal to the TPM that it should start execution of the TPM command that is currently in the Command Buffer. The TPM changes into the “Command Execution” state.

The TPM will CLEAR *Start* when a response is ready in the Response Buffer. The response will contain the result of the command or a TPM error response. The TPM transitions from “Command Execution” to “Command Completion” (or “Ready” if “Command Completion” is not supported).

Some platforms may not be able to trigger TPM commands based on changes of *Start* and may require an additional notification. One possible implementation is the use of an ACPI method (see *TCG ACPI General Specification* version 1.1). For platforms without ACPI, an alternative notification such as interrupts could be used. Some platforms may actively transfer the current CPU into the TPM execution environment and return to the calling software when the response is in the Response Buffer.

Start should be CLEAR when the Control Area is initialized.

3.6 Interrupt Control

Platforms that support the generation of interrupts or notifications by the TPM may use this field to define the interface to subscribe to these interrupts.

3.7 Command Size

This is the size of the Command buffer in bytes.

The value of Command Size field should be chosen such that the largest command that can be issued for the specific TPM implementation fits into the command buffer.

3.8 Command Address

This is the physical address of the command buffer. Software will write the TPM command to be executed to this address.

Software should not write a command larger than “Command Size”.

Software should not write to this memory area unless *Start* is CLEAR. System hardware might prevent writing to this address if *Start* is SET.

3.9 Response Size

This is the size of the Response Buffer in bytes.

The value of Response Size field should be chosen such that the largest response that can be returned from the specific TPM implementation fits into the response buffer.

3.10 Response Address

This is the physical address of the Response Buffer. Software will read the response to the last TPM command from this address.

Software should not read a response larger than “Response Size”.

Software should only read a response after *Start* changed from SET to CLEAR and *Error* is CLEAR.

4 Examples of CRB Interface State Transitions

The following sample state-transition diagrams illustrate possible execution flows.

4.1 CRB Interface State Transition with all states

An overly simplified possible state transition might look like Figure 1: it does not cover all of the possible state transitions, but walks through sending a TPM command with a TPM implementation that supports the “Idle”, “Ready”, “Command Reception”, “Command Execution”, and “Command Completion” states.

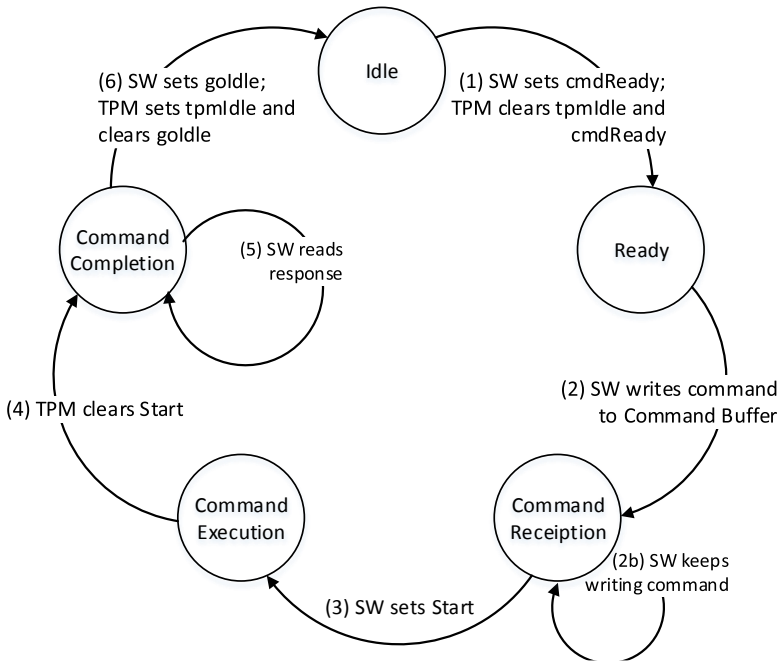

Figure 1: simplified CRB state transition with “Idle” state

Assuming the TPM is in the “Idle” state, step (1) transitions the TPM from “Idle” to “Ready”.

When the TPM is in the “Ready” state it can receive a command, which is done in steps (2) and (2b).

When software finishes copying the command into the Command Buffer, it will SET *Start* (step (3)).

The TPM will start execution of the command, which completes when the TPM CLEARs *Start* in step (4).

Now software can read the response in step (5).

When software is done with the response, it can signal the TPM that it may go into “Idle” state again in step (6).

4.2 CRB Interface State Transition with “Ready” and “Command Execution” states

The following state transition diagram (Figure 2) represents the interaction for normal TPM command execution if “Idle”, “Command Reception”, and “Command Completion” are not implemented by the TPM.

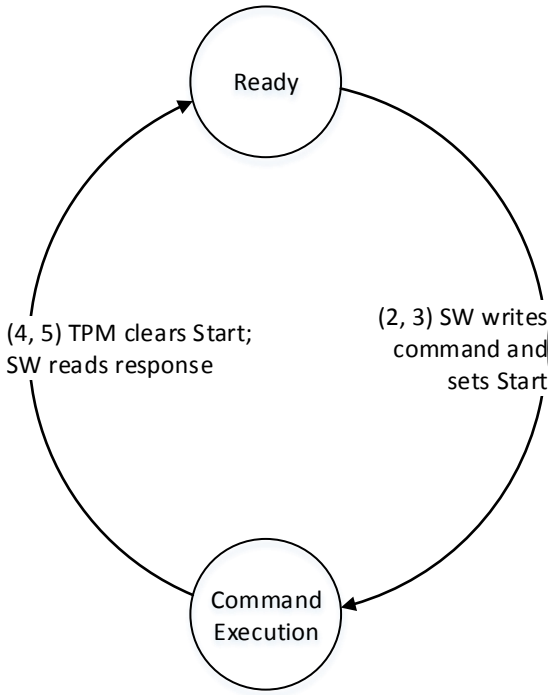

Figure 2: simplified CRB state transition without "Idle" state

If a TPM does not support the "Idle" state, steps (1) and (6) can be eliminated from the state transition diagram in Figure 1.

If the TPM furthermore cannot distinguish between the "Ready" state and "Command Reception" or "Command Completion", steps (2) and (3), and steps (4) and (5) can be combined respectively.

When the TPM is in the "Ready" state, that is, *Start* is CLEAR, software may write a command to the Command Buffer or read a response from the Response Buffer.

When software SETs *Start,* the TPM starts executing the command in the Command Buffer. The TPM will CLEAR *Start* after it has placed the response into the Response Buffer.

If Command Buffer and Response Buffer point to the same memory, software has to manage the "Command Reception" and "Command Completion" states to know when the buffer contains a response or a command.