

TPM Main

Part 2 TPM Structures

Specification version 1.2
Level 2 Revision 85
13 February 2005
Published

Contact: tpmwg@trustedcomputinggroup.org

TCG PUBLISHED

Copyright © TCG 2003

TCG

Copyright © 2003 Trusted Computing Group, Incorporated.

Disclaimer

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

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

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

Contact the Trusted Computing Group at [website link] for information on specification licensing through membership agreements.

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

Copyright C 2005 Trusted Computing Group (TCG) (www.trustedcomputinggroup.org). All rights reserved.

The only official, normative version of a TCG Specification or related document is the English-language text adopted by TCG under its Bylaws and published on the TCG website, www.trustedcomputing.org. TCG does not guarantee the accuracy or completeness of any other version. Other language versions of Specifications and related documents are provided for convenience and are intended to be technically identical to the official English version, but they may contain translation and other errors.

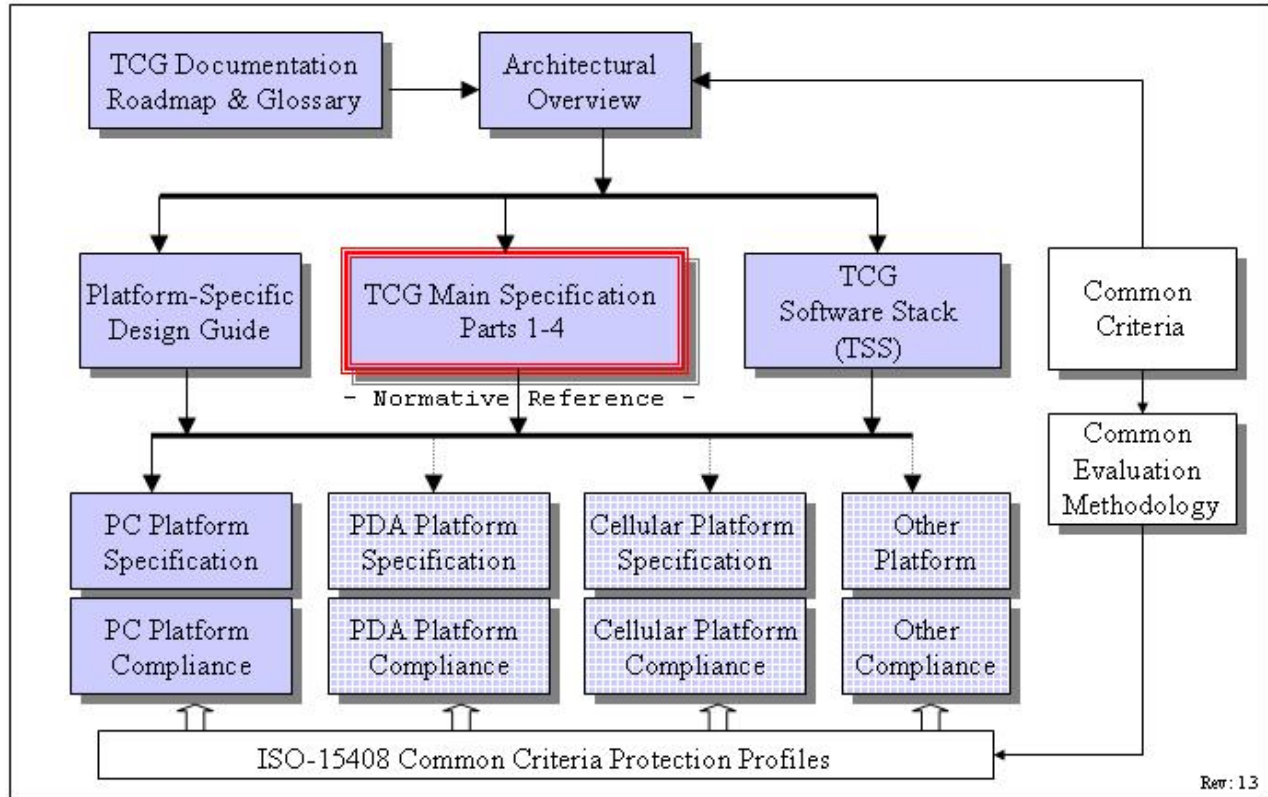
Translations may be provided by volunteers through the Trusted Computing Group's translation program (see www.trustedcomputinggroup.org/specifications).

Other legal notices and terms governing the publication of materials on the TCG website are found at www.trustedcomputinggroup.org/about/legal. TCG incorporates by reference the same notices and terms with respect to TCG-authorized translations of Specifications and related documents, whether published on the TCG website or at another online location.

Revision History

.10	Started: 1 April 2003. Last Updated: 04/30/01 by David Grawrock
52	Started 15 July 2003 by David Graworck
53	Started 5 Aug 2003 by David Grawrock
63	Started 2 October 2003 by David Grawrock All change histroy now listed in Part 1 (DP)

TCG Doc Roadmap – Main Spec



TCG Main Spec Roadmap

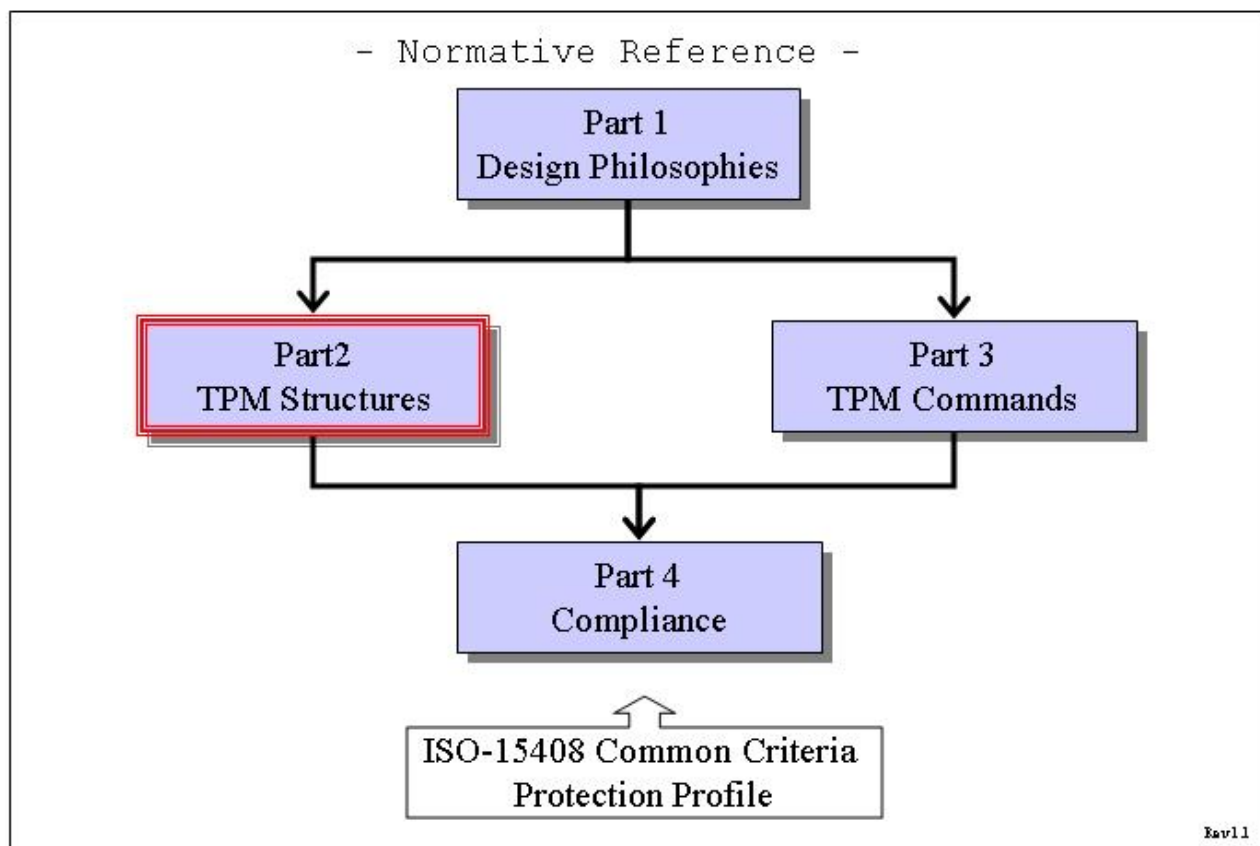


Table of Contents

1. Scope and Audience.....	1
1.1 Key words.....	1
1.2 Statement Type.....	1
2. Basic Definitions	2
2.1 Representation of Information.....	2
2.1.1 Endness of Structures.....	2
2.1.2 Byte Packing	2
2.1.3 Lengths	2
2.2 Defines	3
2.2.1 Basic data types.....	3
2.2.2 Boolean types	3
2.2.3 Helper redefinitions	3
2.2.4 Vendor specific.....	5
3. Structure Tags	6
3.1 TPM_STRUCTURE_TAG	7
4. Types	9
4.1 TPM_RESOURCE_TYPE	9
4.2 TPM_PAYLOAD_TYPE	10
4.3 TPM_ENTITY_TYPE.....	11
4.4 Handles	12
4.4.1 Reserved Key Handles	13
4.5 TPM_STARTUP_TYPE	14
4.6 TPM_STARTUP_EFFECTS	15
4.7 TPM_PROTOCOL_ID.....	16
4.8 TPM_ALGORITHM_ID.....	17
4.9 TPM_PHYSICAL_PRESENCE	18
4.10 TPM_MIGRATE_SCHEME.....	19
4.11 TPM_EK_TYPE.....	20
4.12 TPM_PLATFORM_SPECIFIC	21
5. Basic Structures.....	22
5.1 TPM_STRUCT_VER.....	22
5.2 TPM_VERSION_BYTE	23
5.3 TPM_VERSION.....	24
5.4 TPM_DIGEST	25
5.4.1 Creating a PCR composite hash.....	26

5.5	TPM_NONCE	27
5.6	TPM_AUTHDATA	28
5.7	TPM_KEY_HANDLE_LIST	29
5.8	TPM_KEY_USAGE values	30
5.8.1	Mandatory Key Usage Schemes	30
5.9	TPM_AUTH_DATA_USAGE values	32
5.10	TPM_KEY_FLAGS.....	33
5.11	TPM_CHANGEAUTH_VALIDATE	34
5.12	TPM_MIGRATIONKEYAUTH	35
5.13	TPM_COUNTER_VALUE	36
5.14	TPM_SIGN_INFO Structure.....	37
5.15	TPM_MSA_COMPOSITE	38
5.16	TPM_CMK_AUTH.....	39
5.17	TPM_CMK_DELEGATE values	40
5.18	TPM_SELECT_SIZE.....	41
5.19	TPM_CMK_MIGAUTH	42
5.20	TPM_CMK_SIGTICKET	43
5.21	TPM_CMK_MA_APPROVAL.....	44
6.	Command Tags	45
7.	Internal Data Held By TPM	46
7.1	TPM_PERMANENT_FLAGS	47
7.1.1	Flag Restrictions	50
7.2	TPM_STCLEAR_FLAGS	51
7.2.1	Flag Restrictions	53
7.3	TPM_STANY_FLAGS.....	54
7.3.1	Flag Restrictions	55
7.4	TPM_PERMANENT_DATA	56
7.4.1	Flag Restrictions	59
7.5	TPM_STCLEAR_DATA	60
7.5.1	Flag Restrictions	60
7.6	TPM_STANY_DATA	61
7.6.1	Flag Restrictions	62
8.	PCR Structures	63
8.1	TPM_PCR_SELECTION.....	64
8.2	TPM_PCR_COMPOSITE	66
8.3	TPM_PCR_INFO.....	67
8.4	TPM_PCR_INFO_LONG	68

8.5	TPM_PCR_INFO_SHORT	69
8.6	TPM_LOCALITY_SELECTION	70
8.7	PCR Attributes	71
8.8	TPM_PCR_ATTRIBUTES	72
8.8.1	Comparing command locality to PCR flags	73
8.9	Debug PCR register	74
8.10	Mapping PCR Structures	75
9.	Storage Structures	77
9.1	TPM_STORED_DATA	77
9.2	TPM_STORED_DATA12	78
9.3	TPM_SEALED_DATA	79
9.4	TPM_SYMMETRIC_KEY	80
9.5	TPM_BOUND_DATA	81
10.	TPM_KEY complex	82
10.1	TPM_KEY_PARMS	83
10.1.1	TPM_RSA_KEY_PARMS	84
10.1.2	TPM_SYMMETRIC_KEY_PARMS	84
10.2	TPM_KEY	85
10.3	TPM_KEY12	86
10.4	TPM_STORE_PUBKEY	87
10.5	TPM_PUBKEY	88
10.6	TPM_STORE_ASYMKEY	89
10.7	TPM_STORE_PRIVKEY	90
10.8	TPM_MIGRATE_ASYMKEY	91
10.9	TPM_KEY_CONTROL	92
11.	Signed Structures	93
11.1	TPM_CERTIFY_INFO Structure	93
11.2	TPM_CERTIFY_INFO2 Structure	94
11.3	TPM_QUOTE_INFO Structure	96
11.4	TPM_QUOTE_INFO2 Structure	97
12.	Identity Structures	98
12.1	TPM_EK_BLOB	98
12.2	TPM_EK_BLOB_ACTIVATE	99
12.3	TPM_EK_BLOB_AUTH	100
12.4	TPM_CHOSENID_HASH	101
12.5	TPM_IDENTITY_CONTENTS	102
12.6	TPM_IDENTITY_REQ	103

12.7	TPM_IDENTITY_PROOF	104
12.8	TPM_ASYM_CA_CONTENTS.....	105
12.9	TPM_SYM_CA_ATTESTATION	106
13.	Transport structures.....	107
13.1	TPM_TRANSPORT_PUBLIC	107
13.1.1	TPM_TRANSPORT_ATTRIBUTES Definitions.....	107
13.2	TPM_TRANSPORT_INTERNAL.....	108
13.3	TPM_TRANSPORT_LOG_IN structure	109
13.4	TPM_TRANSPORT_LOG_OUT structure	110
13.5	TPM_TRANSPORT_AUTH structure.....	111
14.	Audit Structures	112
14.1	TPM_AUDIT_EVENT_IN structure	112
14.2	TPM_AUDIT_EVENT_OUT structure	113
15.	Tick Structures	114
15.1	TPM_CURRENT_TICKS	114
16.	Return codes.....	115
17.	Ordinals.....	120
17.1	TSC Ordinals.....	129
18.	Context structures.....	130
18.1	TPM_CONTEXT_BLOB.....	130
18.2	TPM_CONTEXT_SENSITIVE.....	132
19.	NV storage structures	133
19.1	TPM_NV_INDEX.....	133
19.1.1	Required TPM_NV_INDEX values	134
19.1.2	Reserved Index values	135
19.2	TPM_NV_ATTRIBUTES	136
19.3	TPM_NV_DATA_PUBLIC	138
19.4	TPM_NV_DATA_SENSITIVE	139
19.5	Max NV Size.....	140
19.6	TPM_NV_DATA_AREA	141
20.	Delegate Structures	142
20.1	Structures and encryption	142
20.2	Delegate Definitions	143
20.2.1	Owner Permission Settings.....	144
20.2.2	Owner commands not delegated.....	145
20.2.3	Key Permission settings.....	146
20.2.4	Key commands not delegated	147

20.3	TPM_FAMILY_FLAGS	148
20.4	TPM_FAMILY_LABEL	149
20.5	TPM_FAMILY_TABLE_ENTRY	150
20.6	TPM_FAMILY_TABLE	151
20.7	TPM_DELEGATE_LABEL	152
20.8	TPM_DELEGATE_PUBLIC	153
20.9	TPM_DELEGATE_TABLE_ROW	154
20.10	TPM_DELEGATE_TABLE	155
20.11	TPM_DELEGATE_SENSITIVE.....	156
20.12	TPM_DELEGATE_OWNER_BLOB.....	157
20.13	TPM_DELEGATE_KEY_BLOB.....	158
20.14	TPM_FAMILY_OPERATION Values	159
21.	Capability areas	160
21.1	TPM_CAPABILITY_AREA for GetCapability	160
21.2	CAP_PROPERTY Subcap values for GetCapability	162
21.3	Bit ordering for structures.....	164
21.3.1	Deprecated GetCapability Responses.....	164
21.4	TPM_CAPABILITY_AREA Values for SetCapability	165
21.5	SubCap Values for SetCapability.....	166
21.6	TPM_CAP_VERSION_INFO	167
22.	DAA Structures	168
22.1	Size definitions	168
22.2	Constant definitions.....	168
22.3	TPM_DAA_ISSUER.....	169
22.4	TPM_DAA_TPM.....	170
22.5	TPM_DAA_CONTEXT	171
22.6	TPM_DAA_JOINDATA	172
22.7	TPM_STANY_DATA Additions	173
22.8	TPM_DAA_BLOB.....	174
22.9	TPM_DAA_SENSITIVE	175
23.	Redirection.....	176
23.1	TPM_REDIR_COMMAND	176
24.	Deprecated Structures.....	177
24.1	Persistent Flags.....	177
24.2	Volatile Flags.....	177
24.3	TPM persistent data	177
24.4	TPM volatile data.....	177

24.5	TPM SV data	178
24.6	TPM_SYM_MODE	178

1. Scope and Audience

The TPCA main specification is an industry specification that enables trust in computing platforms in general. The main specification is broken into parts to make the role of each document clear. A version of the specification (like 1.2) requires all parts to be a complete specification.

This is Part 3 the structures that the TPM will use.

This document is an industry specification that enables trust in computing platforms in general.

1.1 Key words

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in the chapters 2-10 normative statements are to be interpreted as described in [RFC-2119].

1.2 Statement Type

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

For example:

Start of informative comment

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

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

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

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

End of informative comment

This is the first paragraph of one or more paragraphs (and/or sections) containing the text of the kind normative statements ...

To understand the TPM specification the user MUST read the specification. (This use of MUST indicates a keyword usage and requires an action).

35 **2. Basic Definitions**

36 **Start of informative comment**

37 The following structures and formats describe the interoperable areas of the specification.
38 There is no requirement that internal storage or memory representations of data must
39 follow these structures. These requirements are in place only during the movement of data
40 from a TPM to some other entity.

41 **End of informative comment**

42 **2.1 Representation of Information**

43 **2.1.1 Endness of Structures**

44 Each structure **MUST** use big endian bit ordering, which follows the Internet standard and
45 requires that the low-order bit appear to the far right of a word, buffer, wire format, or other
46 area and the high-order bit appear to the far left.

47 **2.1.2 Byte Packing**

48 All structures **MUST** be packed on a byte boundary.

49 **2.1.3 Lengths**

50 The “Byte” is the unit of length when the length of a parameter is specified.

51 **2.2 Defines**

52 **Start of informative comment**

53 These definitions are in use to make a consistent use of values throughout the structure
54 specifications.

55 **End of informative comment**

56 **2.2.1 Basic data types**

57 **Parameters**

Typedef	Name	Description
unsigned char	BYTE	Basic byte used to transmit all character fields.
unsigned char	BOOL	TRUE/FALSE field. TRUE = 0x01, FALSE = 0x00
unsigned short	UINT16	16-bit field. The definition in different architectures may need to specify 16 bits instead of the short definition
unsigned long	UINT32	32-bit field. The definition in different architectures may need to specify 32 bits instead of the long definition

58 **2.2.2 Boolean types**

Name	Value	Description
TRUE	0x01	Assertion
FALSE	0x00	Contradiction

59 **2.2.3 Helper redefinitions**

60 The following definitions are to make the definitions more explicit and easier to read.

61 **Parameters**

Typedef	Name	Description
BYTE	TPM_AUTH_DATA_USAGE	Indicates the conditions where it is required that authorization be presented.
BYTE	TPM_PAYLOAD_TYPE	The information as to what the payload is in an encrypted structure
BYTE	TPM_VERSION_BYTE	The version info breakdown
UINT16	TPM_PROTOCOL_ID	The protocol in use.
UINT16	TPM_STARTUP_TYPE	Indicates the start state.
UINT16	TPM_ENC_SCHEME	The definition of the encryption scheme.
UINT16	TPM_SIG_SCHEME	The definition of the signature scheme.
UINT16	TPM_MIGRATE_SCHEME	The definition of the migration scheme
UINT16	TPM_PHYSICAL_PRESENCE	Sets the state of the physical presence mechanism.
UINT16	TPM_ENTITY_TYPE	Indicates the types of entity that are supported by the TPM.
UINT16	TPM_KEY_USAGE	Indicates the permitted usage of the key.
UINT16	TPM_EK_TYPE	The type of asymmetric encrypted structure in use by the endorsement key
UINT16	TPM_STRUCTURE_TAG	The tag for the structure

Typedef	Name	Description
UINT16	TPM_PLATFORM_SPECIFIC	The platform specific spec to which the information relates to
UINT32	TPM_COMMAND_CODE	The command ordinal.
UINT32	TPM_CAPABILITY_AREA	Identifies a TPM capability area.
UINT32	TPM_KEY_FLAGS	Indicates information regarding a key.
UINT32	TPM_ALGORITHM_ID	Indicates the type of algorithm.
UINT32	TPM_MODIFIER_INDICATOR	The locality modifier
UINT32	TPM_ACTUAL_COUNT	The actual number of a counter.
UINT32	TPM_TRANSPORT_ATTRIBUTES	Attributes that define what options are in use for a transport session
UINT32	TPM_AUTHHANDLE	Handle to an authorization session
UINT32	TPM_DIRINDEX	Index to a DIR register
UINT32	TPM_KEY_HANDLE	The area where a key is held assigned by the TPM.
UINT32	TPM_PCRINDEX	Index to a PCR register
UINT32	TPM_RESULT	The return code from a function
UINT32	TPM_RESOURCE_TYPE	The types of resources that a TPM may have using internal resources
UINT32	TPM_KEY_CONTROL	Allows for controlling of the key when loaded and how to handle TPM_Startup issues
UINT32	TPM_NV_INDEX	The index into the NV storage area
UINT32	TPM_FAMILY_ID	The family ID. Families ID's are automatically assigned a sequence number by the TPM. A trusted process can set the FamilyID value in an individual row to NULL, which invalidates that row. The family ID resets to NULL on each change of TPM Owner.
UINT32	TPM_FAMILY_VERIFICATION	A value used as a label for the most recent verification of this family. Set to zero when not in use.
UINT32	TPM_STARTUP_EFFECTS	How the TPM handles var
UINT32	TPM_SYM_MODE	The mode of a symmetric encryption
UINT32	TPM_FAMILY_FLAGS	The family flags
UINT32	TPM_DELEGATE_INDEX	The index value for the delegate NV table
UINT32	TPM_CMK_DELEGATE	The restrictions placed on delegation of CMK commands
UINT32	TPM_COUNT_ID	The ID value of a monotonic counter
UINT32	TPM_REEDIT_COMMAND	A command to execute
UINT32	TPM_TRANSHANDLE	A transport session handle
UINT32	TPM_HANDLE	A generic handle could be key, transport etc.
UINT32	TPM_FAMILY_OPERATION	What operation is happening

62 **2.2.4 Vendor specific**

63 **Start of informative comment**

64 For all items that can specify an individual algorithm, protocol or item the specification
65 allows for vendor specific selections. The mechanism to specify a vendor specific mechanism
66 is to set the high bit of the identifier on.

67 **End of informative comment**

68 The following defines allow for the quick specification of a vendor specific item.

69 **Parameters**

Name	Value
TPM_Vendor_Specific32	0x00000400
TPM_Vendor_Specific8	0x80

70 **3. Structure Tags**71 **Start of informative comment**

72 There have been some indications that knowing what structure is in use would be valuable
73 information in each structure. This new tag will be in each new structure that the TPM
74 defines.

75 **End of informative comment**

76 **3.1 TPM_STRUCTURE_TAG**

77 **TPM_ResourceTypes**

Name	Value	Structure
TPM_TAG_CONTEXTBLOB	0x0001	TPM_CONTEXT_BLOB
TPM_TAG_CONTEXT_SENSITIVE	0x0002	TPM_CONTEXT_SENSITIVE
TPM_TAG_CONTEXTPOINTER	0x0003	TPM_CONTEXT_POINTER
TPM_TAG_CONTEXTLIST	0x0004	TPM_CONTEXT_LIST
TPM_TAG_SIGNINFO	0x0005	TPM_SIGN_INFO
TPM_TAG_PCR_INFO_LONG	0x0006	TPM_PCR_INFO_LONG
TPM_TAG_PERSISTENT_FLAGS	0x0007	TPM_PERMANENT_FLAGS
TPM_TAG_VOLATILE_FLAGS	0x0008	TPM_VOLATILE_FLAGS
TPM_TAG_PERSISTENT_DATA	0x0009	TPM_PERSISTENT_DATA
TPM_TAG_VOLATILE_DATA	0x000A	TPM_VOLATILE_DATA
TPM_TAG_SV_DATA	0x000B	TPM_SV_DATA
TPM_TAG_EK_BLOB	0x000C	TPM_EK_BLOB
TPM_TAG_EK_BLOB_AUTH	0x000D	TPM_EK_BLOB_AUTH
TPM_TAG_COUNTER_VALUE	0x000E	TPM_COUNTER_VALUE
TPM_TAG_TRANSPORT_INTERNAL	0x000F	TPM_TRANSPORT_INTERNAL
TPM_TAG_TRANSPORT_LOG_IN	0x0010	TPM_TRANSPORT_LOG_IN
TPM_TAG_TRANSPORT_LOG_OUT	0x0011	TPM_TRANSPORT_LOG_OUT
TPM_TAG_AUDIT_EVENT_IN	0x0012	TPM_AUDIT_EVENT_IN
TPM_TAG_AUDIT_EVENT_OUT	0x0013	TPM_AUDIT_EVENT_OUT
TPM_TAG_CURRENT_TICKS	0x0014	TPM_CURRENT_TICKS
TPM_TAG_KEY	0x0015	TPM_KEY
TPM_TAG_STORED_DATA12	0x0016	TPM_STORED_DATA12
TPM_TAG_NV_ATTRIBUTES	0x0017	TPM_NV_ATTRIBUTES
TPM_TAG_NV_DATA_PUBLIC	0x0018	TPM_NV_DATA_PUBLIC
TPM_TAG_NV_DATA_SENSITIVE	0x0019	TPM_NV_DATA_SENSITIVE
TPM_TAG_DELEGATIONS	0x001A	TPM_DELEGATIONS
TPM_TAG_DELEGATE_PUBLIC	0x001B	TPM_DELEGATE_PUBLIC
TPM_TAG_DELEGATE_TABLE_ROW	0x001C	TPM_DELEGATE_TABLE_ROW
TPM_TAG_TRANSPORT_AUTH	0x001D	TPM_TRANSPORT_AUTH
TPM_TAG_TRANSPORT_PUBLIC	0x001E	TPM_TRANSPORT_PUBLIC
TPM_TAG_PERMANENT_FLAGS	0x001F	TPM_PERMANENT_FLAGS
TPM_TAG_STCLEAR_FLAGS	0x0020	TPM_STCLEAR_FLAGS
TPM_TAG_STANY_FLAGS	0x0021	TPM_STANY_FLAGS
TPM_TAG_PERMANENT_DATA	0x0022	TPM_PERMANENT_DATA
TPM_TAG_STCLEAR_DATA	0x0023	TPM_STCLEAR_DATA
TPM_TAG_STANY_DATA	0x0024	TPM_STANY_DATA

Name	Value	Structure
TPM_TAG_FAMILY_TABLE_ENTRY	0X0025	TPM_FAMILY_TABLE_ENTRY
TPM_TAG_DELEGATE_SENSITIVE	0X0026	TPM_DELEGATE_SENSITIVE
TPM_TAG_DELG_KEY_BLOB	0X0027	TPM_DELG_KEY_BLOB
TPM_TAG_KEY12	0x0028	TPM_KEY12
TPM_TAG_CERTIFY_INFO2	0X0029	TPM_CERTIFY_INFO2
TPM_TAG_DELEGATE_OWNER_BLOB	0X002A	TPM_DELEGATE_OWNER_BLOB
TPM_TAG_EK_BLOB_ACTIVATE	0X002B	TPM_EK_BLOB_ACTIVATE
TPM_TAG_DAA_BLOB	0X002C	TPM_DAA_BLOB
TPM_TAG_DAA_CONTEXT	0X002D	TPM_DAA_CONTEXT
TPM_TAG_DAA_ENFORCE	0X002E	TPM_DAA_ENFORCE
TPM_TAG_DAA_ISSUER	0X002F	TPM_DAA_ISSUER
TPM_TAG_CAP_VERSION_INFO	0X0030	TPM_CAP_VERSION_INFO
TPM_TAG_DAA_SENSITIVE	0X0031	TPM_DAA_SENSITIVE
TPM_TAG_DAA_TPM	0X0032	TPM_DAA_TPM
TPM_TAG_CMK_MIGAUTH	0X0033	TPM_CMK_MIGAUTH
TPM_TAG_CMK_SIGTICKET	0X0034	TPM_CMK_SIGTICKET
TPM_TAG_CMK_MA_APPROVAL	0X0035	TPM_CMK_MA_APPROVAL
TPM_TAG_QUOTE_INFO2	0X0036	TPM_QUOTE_INFO2

78 **4. Types**

79 **4.1 TPM_RESOURCE_TYPE**

80 **TPM_ResourceTypes**

Name	Value	Description
TPM_RT_KEY	0x00000001	The handle is a key handle and is the result of a LoadKey type operation
TPM_RT_AUTH	0x00000002	The handle is an authorization handle. Auth handles come from TPM_OIAP, TPM_OSAP and TPM_DSAP
TPM_RT_HASH	0x00000003	Reserved for hashes
TPM_RT_TRANS	0x00000004	The handle is for a transport session. Transport handles come from TPM_EstablishTransport
TPM_RT_CONTEXT	0x00000005	Resource wrapped and held outside the TPM using the context save/restore commands
TPM_RT_COUNTER	0x00000006	Reserved for counters
TPM_RT_DELEGATE	0x00000007	The handle is for a delegate row. These are the internal rows held in NV storage by the TPM
TPM_RT_DAA_TPM	0x00000008	The value is a DAA TPM specific blob
TPM_RT_DAA_V0	0x00000009	The value is a DAA V0 parameter
TPM_RT_DAA_V1	0x0000000A	The value is a DAA V1 parameter

81 **4.2 TPM_PAYLOAD_TYPE**82 **Start of informative comment**

83 This structure specifies the type of payload in various messages.

84 **End of informative comment**85 **TPM_PAYLOAD_TYPE Values**

Value	Name	Comments
0x01	TPM_PT_ASYM	The entity is an asymmetric key
0x02	TPM_PT_BIND	The entity is bound data
0x03	TPM_PT_MIGRATE	The entity is a migration blob
0x04	TPM_PT_MAINT	The entity is a maintenance blob
0x05	TPM_PT_SEAL	The entity is sealed data
0x06	TPM_PT_MIGRATE_RESTRICTED	The entity is a restricted-migration asymmetric key
0x07	TPM_PT_MIGRATE_EXTERNAL	The entity is a external migratable key
0x08	TPM_PT_CMK_MIGRATE	The entity is a CMK migratable blob
0x09 – 0x7F		Reserved for future use by TPM
0x80 – 0xFF		Vendor specific payloads

86 **4.3 TPM_ENTITY_TYPE**

87 **Start of informative comment**

88 This specifies the types of entity that are supported by the TPM.

89 **End of informative comment**

90 **TPM_ENTITY_TYPE Values**

Value	Event Name	Key Handle	Comments
0x0001	TPM_ET_KEYHANDLE		The entity is a keyHandle or key
0x0002	TPM_ET_OWNER	0x40000001	The entity is the TPM Owner
0x0003	TPM_ET_DATA		The entity is some data
0x0004	TPM_ET_SRK	0x40000000	The entity is the SRK
0x0005	TPM_ET_KEY		The entity is a key or keyHandle
0x0006	TPM_ET_REVOKE	0x40000002	The entity is the RevokeTrust value
0x0007	TPM_ET_DEL_OWNER_BLOB		The entity is a delegate owner blob
0x0008	TPM_ET_DEL_ROW		The entity is a delegate row
0x0009	TPM_ET_DEL_KEY_BLOB		The entity is a delegate key blob
0x000A	TPM_ET_COUNTER		The entity is a counter
0x000B	TPM_ET_NV		The entity is a NV index
0x000C	TPM_ET_KEYAES		The entity is a keyhandle and authorization encryption is AES
0x000D	TPM_ET_KEYDES		The entity is a keyhandle and authorization encryption is 3DES
0x000E	TPM_ET_OWNERAES	0x40000001	The entity is TPM Owner and authorization encryption is AES
0x000F	TPM_ET_OWNERDES	0x40000001	The entity is TPM Owner and authorization encryption is 3DES
0x0010	TPM_ET_KEYXOR		The entity is a keyhandle and authorization encryption is XOR. This designation is held by stored data blobs. Normal use would be to indicate a key by use of TPM_ET_KEY
0x0040	TPM_ET_RESERVED_HANDLE		Reserved. This value avoids collisions with the handle MSB setting.

91 4.4 Handles

92 **Start of informative comment**

93 Handles provides pointers to TPM internal resources. Handles should provide the ability to
94 locate a value without collision.

95 Handles are 32 bit values. To enable ease of use in handles and to assist in internal use of
96 handles the TPM will use the following rules when creating the handle.

97 The most significant byte (MSB) of the handle contains the least significant byte (LSB) from
98 the entity type. The remaining three bytes of the handle contain whatever entropy the TPM
99 needs to provide collision avoidance.

100 **Examples**

101 For a handle pointing to a key loaded by LoadKey the TPM would return 0x05123456. 05 is
102 the LSB of the entity type of TPM_ET_KEY, the 123456 is a random value established by the
103 TPM.

104 For a handle pointing to a counter the TPM would return 0x0A234567. 0A is the LSB of the
105 entity type of TPM_ET_COUNTER and 234567 is a random value established by the TPM.

106 **Reserved handles**

107 Certain TPM entities have handles that point specifically to them, like the EK. These values
108 always use the MSB of 0x40. This is a reserved handle value and all special handles will use
109 the 0x40 prefix.

110 **Handle collisions**

111 The TPM provides good, but not foolproof protection against handle collisions. If system or
112 application software detects a collision that is problematic, the software should evict the
113 resource, and re-submit the command.

114 **End of informative comment**

- 115 1. The MSB of a handle MUST be the corresponding LSB of the TPM_ET_XXX representing
116 the entity loaded
- 117 2. The three LSB of the handle MUST contain the collision resistance values. The TPM
118 MUST provide protection against handle collision. The TPM MUST implement one of the
119 following:
 - 120 a. The three LSB of the handle MUST be generated randomly. The TPM MUST ensure
121 that no currently loaded entity of the same type has the same handle.
 - 122 i. Same type means that if two MSB defined in action 1 match, the three LSB MUST
123 be different.
 - 124 b. The three LSB of the handle MUST be generated from a monotonic counter. The
125 monotonic counter value MUST NOT reset on TPM startup, but may wrap over the
126 life of the TPM.

127 **4.4.1 Reserved Key Handles**

128 **Start of informative comment**

129 The reserved key handles. These values specify specific keys or specific actions for the TPM.

130 **End of informative comment**

131 1. All reserved key handles MUST start with 0x40.

132 **Key Handle Values**

Key Handle	Handle Name	Comments
0x40000000	TPM_KH_SRK	The handle points to the SRK
0x40000001	TPM_KH_OWNER	The handle points to the TPM Owner
0x40000002	TPM_KH_REVOKE	The handle points to the RevokeTrust value
0x40000003	TPM_KH_TRANSPORT	The handle points to the EstablishTransport static authorization
0x40000004	TPM_KH_OPERATOR	The handle points to the Operator auth
0x40000005	TPM_KH_ADMIN	The handle points to the delegation administration auth
0x40000006	TPM_KH_EK	The handle points to the PUBEK, only usable with TPM_OwnerReadInternalPub

133 **4.5 TPM_STARTUP_TYPE**134 **Start of informative comment**

135 To specify what type of startup is occurring.

136 **End of informative comment**137 **TPM_STARTUP_TYPE Values**

Value	Event Name	Comments
0x0001	TPM_ST_CLEAR	The TPM is starting up from a clean state
0x0002	TPM_ST_STATE	The TPM is starting up from a saved state
0x0003	TPM_ST_DEACTIVATED	The TPM is to startup and set the deactivated flag to TRUE

138 **4.6 TPM_STARTUP_EFFECTS**

139 **Start of Informative comment**

140 This structure lists for the various resources and sessions on a TPM the affect that
141 TPM_Startup has on the values.

142 The table makeup is still an open issue.

143 **End of informative comment**

144 **Types of Startup**

Bit position	Name	Description
31-8		No information and MUST be FALSE
7		TPM_Startup has no effect on auditDigest
6		auditDigest is set to NULL on TPM_Startup(ST_CLEAR) but not on other types of TPM_Startup
5		auditDigest is set to NULL on TPM_Startup(any)
4		TPM_RT_KEY resources are initialized by TPM_Startup(ST_ANY)
3		TPM_RT_AUTH resources are initialized by TPM_Startup(ST_STATE)
2		TPM_RT_HASH resources are initialized by TPM_Startup(ST_STATE)
1		TPM_RT_TRANS resources are initialized by TPM_Startup(ST_STATE)
0		TPM_RT_CONTEXT session (but not key) resources are initialized by TPM_Startup(ST_STATE)

145 **4.7 TPM_PROTOCOL_ID**146 **Start of informative comment**

147 This value identifies the protocol in use.

148 **End of informative comment**149 **TPM_PROTOCOL_ID Values**

Value	Event Name	Comments
0x0001	TPM_PID_OIAP	The OIAP protocol.
0x0002	TPM_PID_OSAP	The OSAP protocol.
0x0003	TPM_PID_ADIP	The ADIP protocol.
0x0004	TPM_PID_ADCP	The ADCP protocol.
0x0005	TPM_PID_OWNER	The protocol for taking ownership of a TPM.
0x0006	TPM_PID_DSAP	The DSAP protocol
0x0007	TPM_PID_TRANSPORT	The transport protocol

150 **4.8 TPM_ALGORITHM_ID**

151 **Start of informative comment**

152 This table defines the types of algorithms which may be supported by the TPM.

153 **End of informative comment**

154 **TPM_ALGORITHM_ID values**

Value	Name	Description
0x00000001	TPM_ALG_RSA	The RSA algorithm.
0x00000002	TPM_ALG_DES	The DES algorithm
0x00000003	TPM_ALG_3DES	The 3DES algorithm in EDE mode
0x00000004	TPM_ALG_SHA	The SHA1 algorithm
0x00000005	TPM_ALG_HMAC	The RFC 2104 HMAC algorithm
0x00000006	TPM_ALG_AES128	The AES algorithm, key size 128
0x00000007	TPM_ALG_MGF1	The XOR algorithm using MGF1 to create a string the size of the encrypted block
0x00000008	TPM_ALG_AES192	AES, key size 192
0x00000009	TPM_ALG_AES256	AES, key size 256
0x0000000A	TPM_ALG_XOR	XOR using the rolling nonces

155 **Description**

156 The TPM MUST support the algorithms TPM_ALG_RSA, TPM_ALG_SHA, TPM_ALG_HMAC,
157 TPM_ALG_MGF1

4.9 TPM_PHYSICAL_PRESENCE

Name	Value	Description
TPM_PHYSICAL_PRESENCE_HW_DISABLE	0x0200h	Sets the physicalPresenceHWEnable to FALSE
TPM_PHYSICAL_PRESENCE_CMD_DISABLE	0x0100h	Sets the physicalPresenceCMDEnable to FALSE
TPM_PHYSICAL_PRESENCE_LIFETIME_LOCK	0x0080h	Sets the physicalPresenceLifetimeLock to TRUE
TPM_PHYSICAL_PRESENCE_HW_ENABLE	0x0040h	Sets the physicalPresenceHWEnable to TRUE
TPM_PHYSICAL_PRESENCE_CMD_ENABLE	0x0020h	Sets the physicalPresenceCMDEnable to TRUE
TPM_PHYSICAL_PRESENCE_NOTPRESENT	0x0010h	Sets PhysicalPresence = FALSE
TPM_PHYSICAL_PRESENCE_PRESENT	0x0008h	Sets PhysicalPresence = TRUE
TPM_PHYSICAL_PRESENCE_LOCK	0x0004h	Sets PhysicalPresenceLock = TRUE

159 **4.10 TPM_MIGRATE_SCHEME**

160 **Start of informative comment**

161 The scheme indicates how the StartMigrate command should handle the migration of the
162 encrypted blob.

163 **End of informative comment**

164 **TPM_MIGRATE_SCHEME values**

Name	Value	Description
TPM_MS_MIGRATE	0x0001	A public key that can be used with all TPM migration commands other than 'ReWrap' mode.
TPM_MS_REWRAP	0x0002	A public key that can be used for the ReWrap mode of TPM_CreateMigrationBlob.
TPM_MS_MAINT	0x0003	A public key that can be used for the Maintenance commands
TPM_MS_RESTRICT_MIGRATE	0x0004	The key is to be migrated to a Migration Authority.
TPM_MS_RESTRICT_APPROVE _DOUBLE	0x0005	The key is to be migrated to an entity approved by a Migration Authority using double wrapping
TPM_MS_RESTRICT_MIGRATE_ EXTERNAL	0x0006	The key is to be migrated using an external entity

165 **4.11 TPM_EK_TYPE**166 **Start of informative comment**

167 This structure indicates what type of information that the EK is dealing with.

168 **End of informative comment**

Name	Value	Description
TPM_EK_TYPE_ACTIVATE	0x0001	The blob MUST be TPM_EK_BLOB_ACTIVATE
TPM_EK_TYPE_AUTH	0x0002	The blob MUST be TPM_EK_BLOB_AUTH

169 **4.12 TPM_PLATFORM_SPECIFIC**

170 **Start of informative comment**

171 This enumerated type indicates the platform specific spec that the information relates to.

172 **End of informative comment**

Name	Value	Description
TPM_PS_PC_11	0x0001	PC Specific version 1.1
TPM_PS_PC_12	0x0002	PC Specific version 1.2
TPM_PS_PDA_12	0x0003	PDA Specific version 1.2
TPM_PS_Server_12	0x0004	Server Specific version 1.2
TPM_PS_Mobile_12	0x0005	Mobil Specific version 1.2

173 **5. Basic Structures**174 **5.1 TPM_STRUCT_VER**175 **Start of informative comment**

176 This indicates the version of the structure.

177 Version 1.2 deprecates the use of this structure in all other structures. The structure is not
178 deprecated as many of the structures that contain this structure are not deprecated.179 The rationale behind keeping this structure and adding the new version structure is that in
180 version 1.1 this structure was in use for two purposes. The first was to indicate the
181 structure version, and in that mode the revMajor and revMinor were suppose to be set to 0.
182 The second use was in getCap and the structure would then return the correct revMajor
183 and revMinor. This use model caused problems in keeping track of when the revs were or
184 were not set and how software used the information. Version 1.2 went to structure tags.
185 Some structures did not change and the TPM_STRUCT_VER is still in use. To avoid the
186 problems from 1.1 this structure now is a fixed value and only remains for backwards
187 compatibility. Structure versioning comes from the tag on the structure and the getCap
188 response for TPM versioning uses TPM_VERSION.189 **End of informative comment**190 **Definition**191

```
typedef struct tdTPM_STRUCT_VER {
192     BYTE major;
193     BYTE minor;
194     BYTE revMajor;
195     BYTE revMinor;
196 } TPM_STRUCT_VER;
```

197 **Parameters**

Type	Name	Description
BYTE	Major	This SHALL indicate the major version of the structure. MUST be 0x01
BYTE	Minor	This SHALL indicate the minor version of the structure. MUST be 0x01
BYTE	revMajor	This MUST be 0x00
BYTE	revMinor	This MUST be 0x00

198 **Descriptions**

- 199 1. Provides the version of the structure
- 200 2. The TPM SHALL inspect all fields to determine if the TPM can properly interpret the
201 structure.
- 202 a. On error the TPM MUST return TPM_BAD_VERSION

203 **5.2 TPM_VERSION_BYTE**

204 **Start of Informative comment**

205 Allocating a byte for the version information is wasteful of space. The current allocation
206 does not provide sufficient resolution to indicate completely the version of the TPM. To allow
207 for backwards compatibility the size of the structure does not change from 1.1.

208 To enable minor version, or revision, numbers with 2-digit resolution, the byte representing
209 a version splits into two BDC encoded nibbles. The ordering of the low and high order
210 provides backwards compatibility with existing numbering.

211 An example of an implementation of this is; a version of 1.23 would have the value 2 in bit
212 positions 3-0 and the value 3 in bit positions 7-4.

213 **End of informative comment**

214 TPM_VERSION_BYTE is a byte. The byte is broken up according to the following rule

Bit position	Name	Description
7-4	leastSigVer	Least significant nibble of the minor version. MUST be values within the range of 0000-1001
3-0	mostSigVer	Most significant nibble of the minor version. MUST be values within the range of 0000-1001

215 **5.3 TPM_VERSION**216 **Start of informative comment**

217 This structure provides information relative the version of the TPM. This structure should
218 only be in use by TPM_GetCapability to provide the information relative to the TPM.

219 **End of informative comment**220 **Definition**

```
221 typedef struct tdTPM_VERSION {
222     TPM_VERSION_BYTE major;
223     TPM_VERSION_BYTE minor;
224     BYTE revMajor;
225     BYTE revMinor;
226 } TPM_VERSION;
```

227 **Parameters**

Type	Name	Description
TPM_VERSION_BYTE	Major	This SHALL indicate the major version of the TPM, mostSigVer MUST be 0x01, leastSigVer MUST be 0x00
TPM_VERSION_BYTE	Minor	This SHALL indicate the minor version of the TPM, mostSigVer MUST be 0x01 or 0x02, leastSigVer MUST be 0x00
BYTE	revMajor	This SHALL be the value of the TPM_PERMANENT_DATA -> revMajor
BYTE	revMinor	This SHALL be the value of the TPM_PERMANENT_DATA -> revMinor

228 **Descriptions**

- 229 1. The major and minor fields indicate the specification version the TPM was designed for
- 230 2. The revMajor and revMinor fields indicate the manufactures version of the TPM
- 231 a. Most challengers of the TPM MAY ignore the revMajor and revMinor fields

232 5.4 TPM_DIGEST

233 Start of informative comment

234 The digest value reports the result of a hash operation.

235 In version 1 the hash algorithm is SHA-1 with a resulting hash result being 20 bytes or 160
236 bits.

237 It is understood that algorithm agility is lost due to fixing the hash at 20 bytes and on SHA-
238 1. The reason for fixing is due to the internal use of the digest. It is the AuthData values, it
239 provides the secrets for the HMAC and the size of 20 bytes determines the values that can
240 be stored and encrypted. For this reason, the size is fixed and any changes to this value
241 require a new version of the specification.

242 End of informative comment

243 Definition

```
244 typedef struct tdTPM_DIGEST{
245     BYTE digest[digestSize];
246 } TPM_DIGEST;
```

247 Parameters

Type	Name	Description
BYTE	digest	This SHALL be the actual digest information

248 Description

249 The digestSize parameter MUST indicate the block size of the algorithm and MUST be 20 or
250 greater.

251 For all TPM v1 hash operations, the hash algorithm MUST be SHA-1 and the digestSize
252 parameter is therefore equal to 20.

253 Redefinitions

Typedef	Name	Description
TPM_DIGEST	TPM_CHOSENID_HASH	This SHALL be the digest of the chosen identityLabel and privacyCA for a new TPM identity.
TPM_DIGEST	TPM_COMPOSITE_HASH	This SHALL be the hash of a list of PCR indexes and PCR values that a key or data is bound to.
TPM_DIGEST	TPM_DIRVALUE	This SHALL be the value of a DIR register
TPM_DIGEST	TPM_HMAC	
TPM_DIGEST	TPM_PCRVALUE	The value inside of the PCR
TPM_DIGEST	TPM_AUDITDIGEST	This SHALL be the value of the current internal audit state
TPM_DIGEST	TPM_DAA_TPM_SEED	This SHALL be a random value generated by a TPM immediately after the EK is installed in that TPM, whenever an EK is installed in that TPM
TPM_DIGEST	TPM_DAA_CONTEXT_SEED	This SHALL be a random value

254 **5.4.1 Creating a PCR composite hash**

255 The definition specifies the operation necessary to create TPM_COMPOSITE_HASH.

256 **Action**

- 257 1. The hashing **MUST** be done using the SHA-1 algorithm.
- 258 2. The input must be a valid TPM_PCR_SELECTION structure.
- 259 3. The process creates a TPM_PCR_COMPOSITE structure from the TPM_PCR_SELECTION
260 structure and the PCR values to be hashed. If constructed by the TPM the values **MUST**
261 come from the current PCR registers indicated by the PCR indices in the
262 TPM_PCR_SELECTION structure.
- 263 4. The process then computes a SHA-1 digest of the TPM_PCR_COMPOSITE structure.
- 264 5. The output is the SHA-1 digest just computed.

265 5.5 TPM_NONCE

266 Start of informative comment

267 A nonce is a random value that provides protection from replay and other attacks. Many of
268 the commands and protocols in the specification require a nonce. This structure provides a
269 consistent view of what a nonce is.

270 End of informative comment

271 Definition

```
272 typedef struct tdTPM_NONCE{  
273     BYTE nonce[20];  
274 } TPM_NONCE;
```

275 Parameters

Type	Name	Description
BYTE	Nonce	This SHALL be the 20 bytes of random data. When created by the TPM the value MUST be the next 20 bytes from the RNG.

276 **5.6 TPM_AUTHDATA**277 **Start of informative comment**

278 The AuthData data is the information that is saved or passed to provide proof of ownership
279 of an entity. For version 1 this area is always 20 bytes.

280 **End of informative comment**281 **Definition**

```
282 typedef BYTE tdTPM_AUTHDATA[20];
```

283 **Descriptions**

284 When sending AuthData data to the TPM the TPM does not validate the decryption of the
285 data. It is the responsibility of the entity owner to validate that the AuthData data was
286 properly received by the TPM. This could be done by immediately attempting to open an
287 authorization session.

288 The owner of the data can select any value for the data

289 **Redefinitions**

Typedef	Name	Description
TPM_AUTHDATA	TPM_SECRET	A secret plaintext value used in the authorization process.
TPM_AUTHDATA	TPM_ENCAUTH	A ciphertext (encrypted) version of AuthData data. The encryption mechanism depends on the context.

290 5.7 TPM_KEY_HANDLE_LIST

291 **Start of informative comment**

292 TPM_KEY_HANDLE_LIST is a structure used to describe the handles of all keys currently
293 loaded into a TPM.

294 **End of informative comment**

295 **Definition**

```
296 typedef struct tdTPM_KEY_HANDLE_LIST {  
297     UINT16    loaded;  
298     [size_is(loaded)] TPM_KEY_HANDLE    handle[];  
299 } TPM_KEY_HANDLE_LIST;
```

300 **Parameters**

Type	Name	Description
UINT16	loaded	The number of keys currently loaded in the TPM.
UINT32	handle	An array of handles, one for each key currently loaded in the TPM

301 **Description**

302 The order in which keys are reported is manufacturer-specific.

303 **5.8 TPM_KEY_USAGE values**304 **Start of informative comment**

305 This table defines the types of keys that are possible.

306 Each key has a setting defining the encryption and signature scheme to use. The selection
307 of a key usage value limits the choices of encryption and signature schemes.308 **End of informative comment**

Name	Value	Description
TPM_KEY_SIGNING	0x0010	This SHALL indicate a signing key. The [private] key SHALL be used for signing operations, only. This means that it MUST be a leaf of the Protected Storage key hierarchy.
TPM_KEY_STORAGE	0x0011	This SHALL indicate a storage key. The key SHALL be used to wrap and unwrap other keys in the Protected Storage hierarchy
TPM_KEY_IDENTITY	0x0012	This SHALL indicate an identity key. The key SHALL be used for operations that require a TPM identity, only.
TPM_KEY_AUTHCHANGE	0x0013	This SHALL indicate an ephemeral key that is in use during the ChangeAuthAsym process, only.
TPM_KEY_BIND	0x0014	This SHALL indicate a key that can be used for TPM_Bind and TPM_Unbind operations only.
TPM_KEY_LEGACY	0x0015	This SHALL indicate a key that can perform signing and binding operations. The key MAY be used for both signing and binding operations. The TPM_KEY_LEGACY key type is to allow for use by applications where both signing and encryption operations occur with the same key. The use of this key type is not recommended
TPM_KEY_MIGRATE	0x0016	This SHALL indicate a key in use for TPM_MigrateKey

309 **5.8.1 Mandatory Key Usage Schemes**310 **Start of Informative comment**

311 For a given key usage type there are subset of valid encryption and signature schemes.

312 **End of informative comment**313 The key usage value for a key determines the encryption and / or signature schemes which
314 MUST be used with that key. The table below maps the schemes defined by this
315 specification to the defined key usage values.

Name	Allowed Encryption schemes	Allowed Signature Schemes
TPM_KEY_SIGNING	TPM_ES_NONE	TPM_SS_RSASSAPKCS1v15_SHA1 TPM_SS_RSASSAPKCS1V15_DER TPM_SS_RSASSAPKCSV15_INFO
TPM_KEY_STORAGE	TPM_ES_RSAESOAEP_SHA1_MGF1	TPM_SS_NONE
TPM_KEY_IDENTITY	TPM_ES_NONE	TPM_SS_RSASSAPKCS1v15_SHA1
TPM_KEY_AUTHCHANGE	TPM_ES_RSAESOAEP_SHA1_MGF1	TPM_SS_NONE
TPM_KEY_BIND	TPM_ES_RSAESOAEP_SHA1_MGF1 TPM_ES_RSAESPKCSV15	TPM_SS_NONE
TPM_KEY_LEGACY	TPM_ES_RSAESOAEP_SHA1_MGF1 TPM_ES_RSAESPKCSV15	TPM_SS_RSASSAPKCS1v15_SHA1 TPM_SS_RSASSAPKCS1V15_DER
TPM_KEY_MIGRATE	TPM_ES_RSAESOAEP_SHA1_MGF1	TPM_SS_NONE

316 Where manufacturer specific schemes are used, the strength must be at least that listed in
317 the above table for TPM_KEY_STORAGE, TPM_KEY_IDENTITY and
318 TPM_KEY_AUTHCHANGE key types.

319

320 The TPM MUST check that the encryption scheme defined for use with the key is a valid
321 scheme for the key type, as follows:

Key algorithm	Approved schemes	Scheme Value
TPM_ALG_RSA	TPM_ES_NONE	0x0001
	TPM_ES_RSAESPKCSv15	0x0002
	TPM_ES_RSAESOAEP_SHA1_MGF1	0x0003
TPM_ALG_AES or 3DES	TPM_ES_SYM_CNT	0x0004
TPM_ALG_AES or 3DES	TPM_ES_SYM_OFB	0x0005

322

323 The TPM MUST check that the signature scheme defined for use with the key is a valid
324 scheme for the key type, as follows:

Key algorithm	Approved schemes	Scheme Value
TPM_ALG_RSA	TPM_SS_NONE	0x0001
	TPM_SS_RSASSAPKCS1v15_SHA1	0x0002
	TPM_SS_RSASSAPKCS1v15_DER	0x0003
	TPM_SS_RSASSAPKCS1v15_INFO	0x0004

325 **5.9 TPM_AUTH_DATA_USAGE values**326 **Start of informative comment**

327 The indication to the TPM when authorization sessions for an entity are required. The only
 328 two options at this time are always or never. Future versions may allow for more complex
 329 decisions regarding AuthData checking.

330 **End of informative comment**

Name	Value	Description
TPM_AUTH_NEVER	0x00	This SHALL indicate that usage of the key without authorization is permitted.
TPM_AUTH_ALWAYS	0x01	This SHALL indicate that on each usage of the key the authorization MUST be performed.
TPM_AUTH_PRIV_USE_ONLY	0x03	This SHALL indicate that on commands that require the TPM to use the private portion of the key, the authorization MUST be performed. For commands that cause the TPM to read the public portion of the key, but not to use the private portion (e.g. TPM_GetPubKey), the authorization may be omitted.
		All other values are reserved for future use.

331 **5.10 TPM_KEY_FLAGS**

332 **Start of informative comment**

333 This table defines the meanings of the bits in a TPM_KEY_FLAGS structure, used in
334 TPM_STORE_ASYMKEY and TPM_CERTIFY_INFO.

335 **End of informative comment**

336 **TPM_KEY_FLAGS Values**

Name	Mask Value	Description
redirection	0x00000001	This mask value SHALL indicate the use of redirected output.
migratable	0x00000002	This mask value SHALL indicate that the key is migratable.
isVolatile	0x00000004	This mask value SHALL indicate that the key MUST be unloaded upon execution of the TPM_Startup(ST_Clear). This does not indicate that a nonvolatile key will remain loaded across TPM_Startup(ST_Clear) events.
pciIgnoredOnRead	0x00000008	When TRUE the TPM MUST NOT check digestAtRelease or localityAtRelease for commands that use the public portion of the key like TPM_GetPubKey When FALSE the TPM MUST check digestAtRelease and localityAtRelease for commands that use the public portion of the key
migrateAuthority	0x00000010	When set indicates that the key is under control of a migration authority. The TPM MUST only allow the creation of a key with this flag in TPM_CMK_CreateKey

337
338 The value of TPM_KEY_FLAGS MUST be decomposed into individual mask values. The
339 presence of a mask value SHALL have the effect described in the above table
340 On input, all undefined bits MUST be zero. The TPM MUST return an error if any undefined
341 bit is set. On output, the TPM MUST set all undefined bits to zero.

342 **5.11 TPM_CHANGEAUTH_VALIDATE**343 **Start of informative comment**

344 This structure provides an area that will stores the new AuthData data and the challenger's
345 nonce.

346 **End of informative comment**347 **Definition**

```
348 typedef struct tdTPM_CHANGEAUTH_VALIDATE {
349     TPM_SECRET newAuthSecret;
350     TPM_NONCE n1;
351 } TPM_CHANGEAUTH_VALIDATE;
```

352 **Parameters**

Type	Name	Description
TPM_SECRET	newAuthSecret	This SHALL be the new AuthData data for the target entity
TPM_NONCE	n1	This SHOULD be a nonce, to enable the caller to verify that the target TPM is on-line.

353 5.12 TPM_MIGRATIONKEYAUTH

354 Start of informative comment

355 This structure provides the proof that the associated public key has TPM Owner AuthData
356 to be a migration key.

357 End of informative comment

358 Definition

```
359 typedef struct tdTPM_MIGRATIONKEYAUTH{  
360     TPM_PUBKEY migrationKey;  
361     TPM_MIGRATE_SCHEME migrationScheme;  
362     TPM_DIGEST digest;  
363 } TPM_MIGRATIONKEYAUTH;
```

364 Parameters

Type	Name	Description
TPM_PUBKEY	migrationKey	This SHALL be the public key of the migration facility
TPM_MIGRATE_SCHEME	migrationScheme	This shall be the type of migration operation.
TPM_DIGEST	digest	This SHALL be the digest value of the concatenation of migration key, migration scheme and tpmProof

365 **5.13 TPM_COUNTER_VALUE**366 **Start of informative comment**

367 This structure returns the counter value. For interoperability, the value size should be 4
368 bytes.

369 **End of informative comment**370 **Definition**

```
371 typedef struct tdTPM_COUNTER_VALUE{
372     TPM_STRUCTURE_TAG  tag;
373     BYTE label[4];
374     TPM_ACTUAL_COUNT  counter;
375 } TPM_COUNTER_VALUE;
```

376 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_COUNTER_VALUE
BYTE	label	The label for the counter
TPM_ACTUAL_COUNT	counter	The 32-bit counter value.

377 5.14 TPM_SIGN_INFO Structure

378 Start of informative comment

379 This structure provides the mechanism for the TPM to quote the current values of a list of
380 PCRs.

381 This is an addition in 1.2 and must be added to all commands that produce a signature. It
382 will not be added to 1.1 commands that produce a signature.

383 End of informative comment

384 Definition

```
385 typedef struct tdTPM_SIGN_INFO {
386     TPM_STRUCTURE_TAG tag;
387     BYTE fixed[4];
388     TPM_NONCE replay;
389     UINT32 dataLen;
390     [size_is (dataLen)] BYTE* data;
391 } TPM_SIGN_INFO;
```

392 Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	Set to TPM_TAG_SIGNINFO
BYTE	fixed	The ASCII text that identifies what function was performing the signing operation
TPM_NONCE	replay	Nonce provided by caller to prevent replay attacks
UINT32	dataLen	The length of the data area
BYTE	data	The data that is being signed

393 **5.15 TPM_MSA_COMPOSITE**394 **Start of informative comment**

395 TPM_MSA_COMPOSITE contains an arbitrary number of digests of public keys belonging to
 396 Migration Authorities. An instance of TPM_MSA_COMPOSITE is incorporated into the
 397 migrationAuth value of a certified-migration-key (CMK), and any of the Migration
 398 Authorities specified in that instance is able to approve the migration of that certified-
 399 migration-key.

400 **End of informative comment**401 **Definition**

```
402 typedef struct tdTPM_MSA_COMPOSITE {
403     UINT32 MSAList;
404     TPM_DIGEST[] migAuthDigest[];
405 } TPM_MSA_COMPOSITE;
```

406 **Parameters**

Type	Name	Description
UINT32	MSAList	The number of migAuthDigests. MSAList MUST be one (1) or greater.
TPM_DIGEST[]	migAuthDigest[]	An arbitrary number of digests of public keys belonging to Migration Authorities.

407

408 TPMs MUST support TPM_MSA_COMPOSITE structures with MSAList of four (4) or less, and
 409 MAY support larger values of MSAList.

410 5.16 TPM_CMK_AUTH

411 **Start of informative comment**

412 The signed digest of TPM_CMK_AUTH is a ticket to prove that an entity with public key
413 “migrationAuthority” has approved the public key “destination Key” as a migration
414 destination for the key with public key “sourceKey”.

415 Normally the digest of TPM_CMK_AUTH is signed by the private key corresponding to
416 “migrationAuthority”.

417 To reduce data size, TPM_CMK_AUTH contains just the digests of “migrationAuthority”,
418 “destinationKey” and “sourceKey”.

419 **End of informative comment**

420 **Definition**

```
421 typedef struct tdTPM_CMK_AUTH{
422     TPM_DIGEST migrationAuthorityDigest;
423     TPM_DIGEST destinationKeyDigest;
424     TPM_DIGEST sourceKeyDigest;
425 } TPM_CMK_AUTH;
```

426 **Parameters**

Type	Name	Description
TPM_DIGEST	migrationAuthorityDigest	The digest of a public key belonging to a Migration Authority
TPM_DIGEST	destinationKeyDigest	The digest of a TPM_PUBKEY structure that is an approved destination key for the private key associated with “sourceKey”
TPM_DIGEST	sourceKeyDigest	The digest of a TPM_PUBKEY structure whose corresponding private key is approved by a Migration Authority to be migrated as a child to the destinationKey.

427 **5.17 TPM_CMK_DELEGATE values**428 **Start of informative comment**

429 The bits of TPM_CMK_DELEGATE are flags that determine how the TPM responds to
430 delegated requests to manipulate a certified-migration-key.

431 **End of informative comment**

Bit	Name	Description
31	TPM_CMK_DELEGATE_SIGNING	When set to 1, this bit SHALL indicate that a delegated command may manipulate a CMK of TPM_KEY_USAGE == TPM_KEY_SIGNING
30	TPM_CMK_DELEGATE_STORAGE	When set to 1, this bit SHALL indicate that a delegated command may manipulate a CMK of TPM_KEY_USAGE == TPM_KEY_STORAGE
29	TPM_CMK_DELEGATE_BIND	When set to 1, this bit SHALL indicate that a delegated command may manipulate a CMK of TPM_KEY_USAGE == TPM_KEY_BIND
28	TPM_CMK_DELEGATE_LEGACY	When set to 1, this bit SHALL indicate that a delegated command may manipulate a CMK of TPM_KEY_USAGE == TPM_KEY_LEGACY
27:0	reserved	MUST be 0

432 The default value of TPM_CMK_Delegate is zero (0)

433 5.18 TPM_SELECT_SIZE

434 Start of informative comment

435 This structure provides the indication for the version and sizeofSelect structure in
436 GetCapability

437 End of informative comment

438 Definition

```
439 typedef struct tdTPM_SELECT_SIZE {  
440     BYTE major;  
441     BYTE minor;  
442     UINT16 reqSize;  
443 } TPM_SELECT_SIZE;
```

444 Parameters

Type	Name	Description
BYTE	Major	This SHALL indicate the major version of the TPM. This MUST be 0x01
BYTE	Minor	This SHALL indicate the minor version of the TPM. This MAY be 0x01 or 0x02
UINT16	reqSize	This SHALL indicate the value for a sizeofSelect field in the TPM_SELECTION structure

445 **5.19 TPM_CMK_MIGAUTH**446 **Start of informative comment**

447 Structure to keep track of the CMK migration authorization

448 **End of informative comment**449 **Definition**

```

450 typedef struct tdTPM_CMK_MIGAUTH{
451     TPM_STRUCTURE_TAG tag;
452     TPM_DIGEST msaDigest;
453     TPM_DIGEST pubKeyDigest;
454 } TPM_CMK_MIGAUTH;

```

455 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	Set to TPM_TAG_CMK_MIGAUTH
TPM_DIGEST	msaDigest	The hash of a TPM_PUBKEY structure containing the migration authority public key and parameters.
TPM_DIGEST	pubKeyDigest	The hash of the associated public key

456 **5.20 TPM_CMK_SIGTICKET**

457 **Start of informative comment**

458 Structure to keep track of the CMK migration authorization

459 **End of informative comment**

460 **Definition**

```
461 typedef struct tdTPM_CMK_SIGTICKET{  
462     TPM_STRUCTURE_TAG tag;  
463     TPM_DIGEST verKeyDigest;  
464     TPM_DIGEST signedData;  
465 } TPM_CMK_SIGTICKET;
```

466 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	Set to TPM_TAG_CMK_SIGTICKET
TPM_DIGEST	verKeyDigest	The hash of a TPM_PUBKEY structure containing the public key and parameters of the key that can verify the ticket
TPM_DIGEST	signedData	The ticket data

467 **5.21 TPM_CMK_MA_APPROVAL**468 **Start of informative comment**

469 Structure to keep track of the CMK migration authorization

470 **End of informative comment**471 **Definition**

```

472 typedef struct tdTPM_CMK_MA_APPROVAL{
473     TPM_STRUCTURE_TAG tag;
474     TPM_DIGEST migrationAuthorityDigest;
475 } TPM_CMK_MA_APPROVAL;

```

476 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	Set to TPM_TAG_CMK_MA_APPROVAL
TPM_DIGEST	migrationAuthorityDigest	The hash of a TPM_MSA_COMPOSITE structure containing the hash of one or more migration authority public keys and parameters.

477 **6. Command Tags**

478 **Start of informative comment**

479 These tags indicate to the TPM the construction of the command either as input or as
480 output. The AUTH indicates that there are one or more AuthData values that follow the
481 command parameters.

482 **End of informative comment**

Tag	Name	Description
0x00C1	TPM_TAG_RQU_COMMAND	A command with no authentication.
0x00C2	TPM_TAG_RQU_AUTH1_COMMAND	An authenticated command with one authentication handle
0x00C3	TPM_TAG_RQU_AUTH2_COMMAND	An authenticated command with two authentication handles
0x00C4	TPM_TAG_RSP_COMMAND	A response from a command with no authentication
0x00C5	TPM_TAG_RSP_AUTH1_COMMAND	An authenticated response with one authentication handle
0x00C6	TPM_TAG_RSP_AUTH2_COMMAND	An authenticated response with two authentication handles

483 7. Internal Data Held By TPM

484 **Start of Informative comment**

485 There are many flags and data fields that the TPM must manage to maintain the current
486 state of the TPM. The areas under TPM control have different lifetimes. Some areas are
487 permanent, some reset upon TPM_Startup(ST_Clear) and some reset upon
488 TPM_Startup(ST_State).

489 Previously the data areas were not grouped exactly according to their reset capabilities. It
490 has become necessary to properly group the areas into the three classifications.

491 Each field has defined mechanisms to allow the control of the field. The mechanism may
492 require authorization or physical presence to properly authorize the management of the
493 field.

494 **End of informative comment**

495 7.1 TPM_PERMANENT_FLAGS

496 Start of Informative comment

497 These flags maintain state information for the TPM. The values are not affected by any
498 TPM_Startup command.

499 End of informative comment

```
500 typedef struct tdTPM_PERMANENT_FLAGS{
501     TPM_STRUCTURE_TAG tag;
502     BOOL disable;
503     BOOL ownership;
504     BOOL deactivated;
505     BOOL readPubek;
506     BOOL disableOwnerClear;
507     BOOL allowMaintenance;
508     BOOL physicalPresenceLifetimeLock;
509     BOOL physicalPresenceHwEnable;
510     BOOL physicalPresenceCmdEnable;
511     BOOL CEKPUSED;
512     BOOL TPMpost;
513     BOOL TPMpostLock;
514     BOOL FIPS;
515     BOOL operator;
516     BOOL enableRevokeEK;
517     BOOL nvLocked;
518     BOOL readSRKPub;
519     BOOL tpmEstablished;
520     BOOL maintenanceDone;
521 } TPM_PERMANENT_FLAGS;
```

522 Parameters

Type	Name	Description	Flag Name
TPM_STRUCTURE_TAG	tag	TPM_TAG_PERMANENT_FLAGS	
BOOL	disable	The state of the disable flag. The default state is TRUE	TPM_PF_DISABLE
BOOL	ownership	The ability to install an owner. The default state is TRUE.	TPM_PF_OWNERSHIP
BOOL	deactivated	The state of the inactive flag. The default state is TRUE.	TPM_PF_DEACTIVATED
BOOL	readPubek	The ability to read the PUBEK without owner AuthData. The default state is TRUE.	TPM_PF_READPUBEK
BOOL	disableOwnerClear	Whether the owner authorized clear commands are active. The default state is FALSE.	TPM_PF_DISABLEOWNERCLEAR
BOOL	allowMaintenance	Whether the TPM Owner may create a maintenance archive. The default state is TRUE.	TPM_PF_ALLOWMAINTENANCE
BOOL	physicalPresenceLifetimeLock	This bit can only be set to TRUE; it cannot be set to FALSE except during the manufacturing process. FALSE: The state of either physicalPresenceHwEnable or physicalPresenceCmdEnable MAY be changed. (DEFAULT) TRUE: The state of either physicalPresenceHwEnable or physicalPresenceCmdEnable MUST NOT be changed for	TPM_PF_PHYSICALPRESENCELIFETIMELOCK

Type	Name	Description	Flag Name
		the life of the TPM.	
BOOL	physicalPresenceHWEenable	FALSE: Disable the hardware signal indicating physical presence. (DEFAULT) TRUE: Enables the hardware signal indicating physical presence.	TPM_PF_PHYSICALPRESENCEHWEENABLE
BOOL	physicalPresenceCMDenable	FALSE: Disable the command indicating physical presence. (DEFAULT) TRUE: Enables the command indicating physical presence.	TPM_PF_PHYSICALPRESENCECMDENABLE
BOOL	CEKPUSED	TRUE: The PRIVEK and PUBEK were created using TPM_CreateEndorsementKeyPair. FALSE: The PRIVEK and PUBEK were created using a manufacturers process. NOTE: This flag has no default value as the key pair MUST be created by one or the other mechanism.	TPM_PF_CEKPUSED
BOOL	TPMpost	TRUE: the TPM MUST successfully complete TPM_SelfTestFull before permitting execution of any command except TPM_SelfTestFull and TPM_Startup The default state is FALSE	TPM_PF_TPMPPOST
BOOL	TPMpostLock	FALSE: The state of TPMpost MAY be changed. (DEFAULT) TRUE: The state of TPMpost MUST NOT be changed.	TPM_PF_TPMPPOSTLOCK
BOOL	FIPS	TRUE: This TPM operates in FIPS mode FALSE: This TPM does NOT operate in FIPS mode	TPM_PF_FIPS
BOOL	operator	TRUE: The operator AuthData value is valid FALSE: the operator AuthData value is not set (DEFAULT)	TPM_PF_OPERATOR
BOOL	enableRevokeEK	TRUE: The TPM_RevokeTrust command is active FALSE: the TPM RevokeTrust command is disabled	TPM_PF_ENABLEREVOKEEK
BOOL	nvLocked	TRUE: All NV area authorization checks are active FALSE: No NV area checks are performed, except for maxNVWrites. FALSE is the default value	TPM_PF_NV_LOCKED
BOOL	readSRKPub	TRUE: GetPubKey will return the SRK pub key FALSE: GetPubKey will not return the SRK pub key Default is FALSE	TPM_PF_READSRKPUB
BOOL	tpmEstablished	TRUE: TPM_HASH_START has been executed at some time FALSE: TPM_HASH_START has not been executed at any time Default is FALSE – resets using TPM_ResetEstablishmentBit	TPM_PF_TPMESTABLISHED
BOOL	maintenanceDone	TRUE: A maintenance archive has been created for the current SRK	TPM_PF_MAINTENANCEDONE

523 **Description**

524 These values are permanent in the TPM and MUST not change upon execution of
525 TPM_Startup(any) command.

526 **Actions**

- 527 1. If disable is TRUE the following commands will execute with their normal protections
- 528 a. The Avail Disabled column in the ordinal table indicates which commands can and
529 cannot execute

- 530 b. If the command is not available the TPM MUST return TPM_DISABLED upon any
531 attempt to execute the ordinal
- 532 c. TSC_PhysicalPresence can execute when the TPM is disabled
- 533 d. A disabled TPM never prevents the extend capabilities from operating. This is
534 necessary in order to ensure that the records of sequences of integrity metrics in a
535 TPM are always up-to-date. It is irrelevant whether an inactive TPM prevents the
536 extend capabilities from operating, because PCR values cannot be used until the
537 platform is rebooted, at which point existing PCR values are discarded
- 538 2. If ownership has the value of FALSE, then any attempt to install an owner fails with the
539 error value TPM_INSTALL_DISABLED.
- 540 3. If deactivated is TRUE
- 541 a. This flag does not directly cause capabilities to return the error code
542 TPM_DEACTIVATED.
- 543 b. TPM_Startup uses this flag to set the state of TPM_STCLEAR_FLAGS -> deactivated
544 when the TPM is booted in the state stType==TPM_ST_CLEAR. Only
545 TPM_STCLEAR_FLAGS -> deactivated determines whether capabilities will return the
546 error code TPM_DEACTIVATED.
- 547 c. A change in TPM_PERMANENT_FLAGS -> deactivated therefore has no effect on
548 whether capabilities will return the error code TPM_DEACTIVATED until the next
549 execution of TPM_Startup(ST_Clear)
- 550 4. If readPubek is TRUE then the TPM_ReadPubek will return the PUBEK, if FALSE the
551 command will return TPM_DISABLED_CMD.
- 552 5. If disableOwnerClear is TRUE then TPM_OwnerClear will return
553 TPM_CLEAR_DISABLED, if false the commands will execute.
- 554 6. The physicalPresenceHWEnable and physicalPresenceCMDEnable flags MUST mask
555 their respective signals before further processing. The hardware signal, if enabled by the
556 physicalPresenceHWEnable flag, MUST be logically ORed with the PhysicalPresence flag,
557 if enabled, to obtain the final physical presence value used to allow or disallow local
558 commands.

7.1.1 Flag Restrictions

Flag SubCap number 0x00000000 +	Set	Set restrictions	Actions from
+1 TPM_PF_DISABLE	Y	Owner authorization or physical presence Not available when TPM disabled	TPM_OwnerSetDisable TPM_PhysicalEnable TPM_PhysicalDisable
+2 TPM_PF_OWNERSHIP	Y	No authorization. No owner installed. Physical presence asserted Not available when TPM deactivated or disabled	TPM_SetOwnerInstall
+3 TPM_PF_DEACTIVATED	Y	No authorization, physical presence assertion Not available when TPM disabled	TPM_PhysicalSetDeactivated
+4 TPM_PF_READPUBEK	Y	Owner authorization Not available when TPM deactivated or disabled	TPM_DisableReadPubEK
+5 TPM_PF_DISABLEOWNERCLEAR	Y	Owner authorization. Can only set to TRUE, FALSE invalid value. After being set only ForceClear resets back to FALSE. Not available when TPM deactivated or disabled	TPM_DisableOwnerClear
+6 TPM_PF_ALLOWMAINTENANCE	Y	Owner authorization. Can only set to FALSE, TRUE invalid value. After being set only changing TPM owner resets back to TRUE Not available when TPM deactivated or disabled	TPM_KillMaintenanceFeature
+7 TPM_PF_PHYSICALPRESENCELIFETIME MELOCK	N		
+8 TPM_PF_PHYSICALPRESENCEHWE NABLE	N		
+9 TPM_PF_PHYSICALPRESENCECMDE NABLE	N		
+10 TPM_PF_CEKPUSED	N		
+11 TPM_PF_TPMPOST	N		
+12 TPM_PF_TPMPOSTLOCK	N		
+13 TPM_PF_FIPS	N		
+14 TPM_PF_OPERATOR	N		
+15 TPM_PF_ENBLEREVOKEEK	N		
+16 TPM_PF_NV_LOCKED	N		
+17 TPM_PF_READSRKPUB	Y	Owner Authorization Not available when TPM deactivated or disabled	TPM_SetCapability
+18 TPM_PF_RESETESTABLISHMENTBIT	Y	Owner Authorization and locality 3 or locality 4 Not available when TPM deactivated or disabled	TPM_ResetEstablishmentBit
+19 TPM_PF_MAINTENANCEDONE	N		

560 7.2 TPM_STCLEAR_FLAGS

561 Start of Informative comment

562 These flags maintain state that is reset on each TPM_Startup(ST_Clear) command. The
563 values are not affected by TPM_Startup(ST_State) commands.

564 End of informative comment

```
565 #define TPM_MAX_FAMILY 8
566
567 typedef struct tdTPM_STCLEAR_FLAGS{
568     TPM_STRUCTURE_TAG tag;
569     BOOL deactivated;
570     BOOL disableForceClear;
571     BOOL physicalPresence;
572     BOOL physicalPresenceLock;
573     BOOL bGlobalLock;
574 } TPM_STCLEAR_FLAGS;
```

575 Parameters

Type	Name	Description	Flag Name
TPM_STRUCTURE_TAG	tag	TPM_TAG_STCLEAR_FLAGS	
BOOL	deactivated	Prevents the operation of most capabilities. There is no default state. It is initialized by TPM_Startup to the same value as TPM_PERMANENT_FLAGS -> deactivated or a set value depending on the type of TPM_Startup. TPM_SetTempDeactivated sets it to TRUE.	TPM_SF_DEACTIVATED
BOOL	disableForceClear	Prevents the operation of TPM_ForceClear when TRUE. The default state is FALSE. TPM_DisableForceClear sets it to TRUE.	TPM_SF_DISABLEFORCECLEAR
BOOL	physicalPresence	Software indication whether an Owner is physically present. The default state is FALSE (Owner is not physically present)	TPM_SF_PHYSICALPRESENCE
BOOL	physicalPresenceLock	Indicates whether changes to the physicalPresence flag are permitted. TPM_Startup/ST_CLEAR sets PhysicalPresence to its default state of FALSE (allow changes to PhysicalPresence flag). The meaning of TRUE is: Do not allow further changes to PhysicalPresence flag. TSC_PhysicalPresence can change the state of physicalPresenceLock.	TPM_SF_PHYSICALPRESENCELOCK
BOOL	bGlobalLock	Set to FALSE on each TPM_Startup(ST_CLEAR). Set to TRUE when a write to NV_Index =0 is successful	TPM_SF_BGLOBALLOCK

576 Description

577 These values MUST reset upon execution of TPM_Startup(ST_Clear).

578 These values MUST NOT reset upon execution of TPM_Startup(ST_State) or
579 TPM_Startup(ST_Deactivated)

580 Actions

581 1. If deactivated is TRUE the following commands SHALL execute with their normal
582 protections

- 583 a. The Avail Deactivated column in the ordinal table indicates which commands can
584 and cannot execute
- 585 b. If the command is not available the TPM MUST return TPM_DEACTIVATED upon any
586 attempt to execute the ordinal
- 587 c. TSC_PhysicalPresence can execute when deactivated
- 588 d. TPM_Extend and TPM_SHA1CompleteExtend MAY execute with their normal
589 protections
- 590 2. If disableForceClear is TRUE then the TPM_ForceClear command returns
591 TPM_CLEAR_DISABLED, if FALSE then the command will execute.
- 592 3. If physicalPresence is TRUE and TPM_PERMANENT_FLAGS ->
593 physicalPresenceCMDEnable is TRUE, the TPM MAY assume that the Owner is
594 physically present.
- 595 4. If physicalPresenceLock is TRUE, TSC_PhysicalPresence MUST NOT change the
596 physicalPresence flag. If physicalPresenceLock is FALSE, TSC_PhysicalPresence will
597 operate.
- 598 a. Set physicalPresenceLock to TRUE at TPM manufacture.

599 **7.2.1 Flag Restrictions**

Flag SubCap number 0x00000000 +	Set	Set restrictions	Actions from
+1 TPM_SF_DEACTIVATED	N		
+2 TPM_SF_DISABLEFORCECLEAR	Y	Not available when TPM deactivated or disabled	TPM_DisableForceClear
+3 TPM_SF_PHYSICALPRESENCE	N		
+4 TPM_SF_PHYSICALPRESENCELOCK	N		
+5 TPM_SF_BGLOBBLOCK	N		

600 **7.3 TPM_STANY_FLAGS**601 **Start of Informative comment**

602 These flags reset on any TPM_Startup command.

603 TOSPresent indicates the presence of a Trusted Operating System (TOS) that was
604 established using the TPM_HASH_START command in the TPM Interface.605 **End of informative comment**

```

606 typedef struct tdTPM_STANY_FLAGS{
607     TPM_STRUCTURE_TAG tag;
608     BOOL postInitialise;
609     TPM_MODIFIER_INDICATOR localityModifier;
610     BOOL transportExclusive;
611     BOOL TOSPresent;
612 } TPM_STANY_FLAGS;

```

613 **Parameters**

Type	Name	Description	Flag Name
TPM_STRUCTURE_TAG	tag	TPM_TAG_STANY_FLAGS	
BOOL	postInitialise	Prevents the operation of most capabilities. There is no default state. It is initialized by TPM_Init to TRUE. TPM_Startup sets it to FALSE.	TPM_AF_POSTINITIALISE
TPM_MODIFIER_INDICATOR	localityModifier	This SHALL indicate for each command the presence of a locality modifier for the command. It MUST be always ensured that the value during usage reflects the currently active locality.	TPM_AF_LOCALITYMODIFIER
BOOL	transportExclusive	Defaults to FALSE. TRUE when there is an exclusive transport session active. Execution of ANY command other than TPM_ExecuteTransport or TPM_ReleaseTransportSigned MUST invalidate the exclusive transport session.	TPM_AF_TRANSPORTEXCLUSIVE
BOOL	TOSPresent	Defaults to FALSE Set to TRUE on TPM_HASH_START set to FALSE using setCapability	TPM_AF_TOSPRESENT

614 **Description**

615 This structure MUST reset on TPM_Startup(any)

616 **Actions**

- 617 1. If postInitialise is TRUE, TPM_Startup SHALL execute as normal
 - 618 a. All other commands SHALL return TPM_INVALID_POSTINIT
- 619 2. localityModifier is set upon receipt of each command to the TPM. The localityModifier
620 MUST be cleared when the command execution response is read

621

7.3.1 Flag Restrictions

Flag SubCap number 0x00000000 +	Set	Set restrictions	Actions from
+1 TPM_AF_POSTINITIALISE	N		
+2 TPM_AF_LOCALITYMODIFIER	N		
+3 TPM_AF_TRANSPORTEXCLUSIVE	N		
+4 TPM_AF_TOSPRESENT	Y	Locality 3 or 4, can only set to FALSE Not available when TPM deactivated or disabled	TPM_SetCapability

622 **7.4 TPM_PERMANENT_DATA**623 **Start of Informative comment**

624 This is an informative structure and not normative. It is purely for convenience of writing
625 the spec.

626 This structure contains the data fields that are permanently held in the TPM and not
627 affected by TPM_Startup(any).

628 Many of these fields contain highly confidential and privacy sensitive material. The TPM
629 must maintain the protections around these fields.

630 **End of informative comment**631 **Definition**

```

632 #define TPM_MIN_COUNTERS 4 // the minimum number of counters is 4
633 #define TPM_DELEGATE_KEY TPM_KEY
634 #define TPM_NUM_PCR 16
635 #define TPM_MAX_NV_WRITE_NOOWNER 64
636
637 typedef struct tdTPM_PERMANENT_DATA{
638     TPM_STRUCTURE_TAG        tag;
639     BYTE                      revMajor;
640     BYTE                      revMinor;
641     TPM_NONCE                 tpmProof;
642     TPM_NONCE                 ekReset;
643     TPM_SECRET                ownerAuth;
644     TPM_SECRET                operatorAuth;
645     TPM_DIRVALUE              authDIR[1];
646     TPM_PUBKEY                manuMaintPub;
647     TPM_KEY                   endorsementKey;
648     TPM_KEY                   srk;
649     TPM_KEY                   contextKey;
650     TPM_KEY                   delegateKey;
651     TPM_COUNTER_VALUE         auditMonotonicCounter;
652     TPM_COUNTER_VALUE         monotonicCounter[TPM_MIN_COUNTERS];
653     TPM_PCR_ATTRIBUTES        pcrAttrib[TPM_NUM_PCR];
654     BYTE                      ordinalAuditStatus[];
655     BYTE*                     rngState;
656     TPM_FAMILY_TABLE          familyTable;
657     TPM_DELEGATE_TABLE        delegateTable;
658     UINT32                    maxNVBufSize;
659     UINT32                    lastFamilyID;
660     UINT32                    noOwnerNVWrite;
661     TPM_CMK_DELEGATE          restrictDelegate;
662     TPM_DAA_TPM_SEED          tpmDAASeed
663 }TPM_PERMANENT_DATA;
```

664 **Parameters**

Type	Name	Description	Flag Name
TPM_STRUCTURE_TAG	tag	TPM_TAG_PERMANENT_DATA	
BYTE	revMajor	This is the TPM major revision indicator. This SHALL be set by the TPME, only. The default value is manufacturer-specific.	TPM_PD_REVMAJOR
BYTE	revMinor	This is the TPM minor revision indicator. This SHALL be set by the TPME, only. The default value is manufacturer-specific.	TPM_PD_REVMINOR
TPM_NONCE	tpmProof	This is a random number that each TPM maintains to validate blobs in the SEAL and other processes. The default value is manufacturer-specific.	TPM_PD_TPMPROOF
TPM_SECRET	ownerAuth	This is the TPM-Owner's AuthData data. The default value is manufacturer-specific.	TPM_PD_OWNERAUTH
TPM_SECRET	operatorAuth	The value that allows the execution of the SetTempDisabled command	TPM_PD_OPERATORAUTH
TPM_PUBKEY	manuMaintPub	This is the manufacturer's public key to use in the maintenance operations. The default value is manufacturer-specific.	TPM_PD_MANUMAINTPUB
TPM_KEY	endorsementKey	This is the TPM's endorsement key pair.	TPM_PD_ENDORSEMENTKEY
TPM_KEY	srk	This is the TPM's StorageRootKey.	TPM_PD_SRK
TPM_KEY	delegateKey	This key encrypts delegate rows that are stored outside the TPM. The key MAY be symmetric or asymmetric. The key size for the algorithm SHOULD be equivalent to 128-bit AES key. The TPM MAY set this value once or allow for changes to this value. This key MUST NOT be the EK or SRK To save space this key MAY be the same key that performs context blob encryption. If an asymmetric algorithm is in use for this key the public portion of the key MUST never be revealed by the TPM. This value MUST be reset when the TPM Owner changes. The value MUST be invalidated with the actions of TPM_OwnerClear. The value MUST be set on TPM_TakeOwnership. The contextKey and delegateKey MAY be the same value.	TPM_PD_DELEGATEKEY
TPM_KEY	contextKey	This is the key in use to perform context saves. The key may be symmetric or asymmetric. The key size is predicated by the algorithm in use. This value MUST be reset when the TPM Owner changes. This key MUST NOT be a copy of the EK or SRK. The contextKey and delegateKey MAY be the same value.	TPM_PD_CONTEXTKEY
TPM_COUNTER_VALUE	auditMonotonicCounter	This SHALL be the audit monotonic counter for the TPM. This value starts at 0 and increments according to the rules of auditing	TPM_PD_AUDITMONOTONICCOUNTER
TPM_COUNTER_VALUE	monotonicCounter	This SHALL be the monotonic counters for the TPM. The individual counters start and increment according to the rules of monotonic counters.	TPM_PD_MONOTONICCOUNTER
TPM_PCR_ATTRIBUTES	pcrAttrib	The attributes for all of the PCR registers supported by the TPM.	TPM_PD_PCRRATTRIB
byte	ordinalAuditStatus	Table indicating which ordinals are being audited.	TPM_PD_ORDINALAUDITSTATUS

Type	Name	Description	Flag Name
TPM_DIRVALUE	authDIR	The array of TPM Owner authorized DIR. Points to the same location as the NV index value.	TPM_PD_AUTHDIR
BYTE*	rngState	State information describing the random number generator.	TPM_PD_RNGSTATE
TPM_FAMILY_TABLE	familyTable	The family table in use for delegations	TPM_PD_FAMILYTABLE
TPM_DELEGATE_TABLE	delegateTable	The delegate table	TPM_DELEGATETABLE
TPM_NONCE	ekReset	Nonce held by TPM to validate TPM_RevokeTrust. This value is set as the next 20 bytes from the TPM RNG when the EK is set using TPM_CreateRevocableEK	TPM_PD_EKRESET
UINT32	maxNVBufSize	The maximum size that can be specified in TPM_NV_DefineSpace. This is NOT related to the amount of current NV storage available. This value would be set by the TPM manufacturer and would take into account all of the variables in the specific TPM implementation. Variables could include TPM input buffer max size, transport session overhead, available memory and other factors. The minimum value of maxNVBufSize MUST be 512 and can be larger.	TPM_PD_MAXNVBUFSIZE
UINT32	lastFamilyID	A value that sets the high water mark for family ID's. Set to 0 during TPM manufacturing and never reset.	TPM_PD_LASTFAMILYID
UINT32	noOwnerNVWrite	The count of NV writes that have occurred when there is no TPM Owner. This value starts at 0 in manufacturing and after each TPM_OwnerClear. If the value exceeds 64 the TPM returns TPM_MAXNVWRITES to any command attempting to manipulate the NV storage. Commands that manipulate the NV store are: TPM_Delegate_Manage TPM_Delegate_LoadOwnerDelegation TPM_NV_DefineSpace TPM_NV_WriteValue	TPM_PD_NOOWNERNVWRITE
TPM_CMK_DELEGATE	restrictDelegate	The settings that allow for the delegation and use on CMK keys. Default value is FALSE.	TPM_PD_RESTRICTDELEGATE
TPM_DAA_TPM_SEED	tpmDAASeed	This SHALL be a random value generated after generation of the EK. tpmDAASeed does not change during TPM Owner changes If the EK is removed (RevokeTrust) then the TPM MUST invalidate the tpmDAASeed	TPM_PD_TPMDAASEED

665 **7.4.1 Flag Restrictions**

Flag SubCap number 0x00000000 +	Set	Set restrictions	Actions from
+1 TPM_PD_REVMAJOR	N		
+2 TPM_PD_REVMINOR	N		
+3 TPM_PD_TPMPROOF	N		
+4 TPM_PD_OWNERAUTH	N		
+5 TPM_PD_OPERATORAUTH	N		
+6 TPM_PD_MANUMAINTPUB	N		
+7 TPM_PD_ENDORSEMENTKEY	N		
+8 TPM_PD_SRK	N		
+9 TPM_PD_DELEGATEKEY	N		
+10 TPM_PD_CONTEXTKEY	N		
+11 TPM_PD_AUDITMONOTONICCOUNT ER	N		
+12 TPM_PD_MONOTONICCOUNTER	N		
+13 TPM_PD_PCRATTRIB	N		
+14 TPM_PD_ORDINALAUDITSTATUS	N		
+15 TPM_PD_AUTHDIR	N		
+16 TPM_PD_RNGSTATE	N		
+17 TPM_PD_FAMILYTABLE	N		
+18 TPM_DELEGATETABLE	N		
+19 TPM_PD_EKRESET	N		
+20 TPM_PD_MAXNVBUFSIZE	N		
+21 TPM_PD_LASTFAMILYID	N		
+22 TPM_PD_NOOWNERNVWRITE	N		
+23 TPM_PD_RESTRICTDELEGATE	Y	If TPM owner installed, owner authorization else physical presence assertion	TPM_CMK_SetRestrictions
+24 TPM_PD_TPMDAASEED	N		

666 **7.5 TPM_STCLEAR_DATA**667 **Start of Informative comment**

668 This is an informative structure and not normative. It is purely for convenience of writing
669 the spec.

670 Most of the data in this structure resets on TPM_Startup(ST_Clear). A TPM may implement
671 rules that provide longer-term persistence for the data. The TPM reflects how it handles the
672 data in various getcapability fields including startup effects.

673 **End of informative comment**674 **Definition**

```
675 typedef struct tdTPM_STCLEAR_DATA{
676     TPM_STRUCTURE_TAG    tag;
677     TPM_NONCE            contextNonceKey;
678     TPM_COUNT_ID        countID;
679     UINT32               ownerReference;
680     BOOL                 disableResetLock;
681 }TPM_STCLEAR_DATA;
```

682 **Parameters**

Type	Name	Description	Flag Name
TPM_STRUCTURE_TAG	tag	TPM_TAG_STCLEAR_DATA	
TPM_NONCE	contextNonceKey	This is the nonce in use to properly identify saved key context blobs This SHALL be set to null on each TPM_Startup (ST_Clear).	TPM_SD_CONTEXTNONCEKEY
TPM_COUNT_ID	countID	This is the handle for the current monotonic counter. This SHALL be set to NULL on each TPM_Startup(ST_Clear).	TPM_SD_COUNTID
UINT32	ownerReference	Points to where to obtain the owner secret in OIAP and OSAP commands. This allows a TSS to manage 1.1 applications on a 1.2 TPM where delegation is in operation. Default value is TPM_KH_OWNER.	TPM_SD_OWNERREFERENCE
BOOL	disableResetLock	Disables TPM_ResetLockValue upon authorization failure. The value remains TRUE for the timeout period. Default is FALSE. The value is in the STCLEAR_DATA structure as the implementation of this flag is TPM vendor specific.	TPM_SD_DISABLERESETLOCK

683 **7.5.1 Flag Restrictions**

Flag SubCap number 0x00000000 +	Set	Set restrictions	Actions from
+1 TPM_SD_CONTEXTNONCEKEY	N		
+2 TPM_SD_COUNTID	N		
+3 TPM_SD_OWNERREFERENCE	N		
+4 TPM_SD_DISABLERESETLOCK	N		

684 7.6 TPM_STANY_DATA

685 Start of Informative comment

686 This is an informative structure and not normative. It is purely for convenience of writing
687 the spec.

688 Most of the data in this structure resets on TPM_Startup(ST_State). A TPM may implement
689 rules that provide longer-term persistence for the data. The TPM reflects how it handles the
690 data in various getcapability fields including startup effects.

691 End of informative comment

692 Definition

```
693 #define TPM_MIN_SESSIONS 3
694 #define TPM_MIN_SESSION_LIST 16
695
696 typedef struct tdTPM_SESSION_DATA{
697 ... // vendor specific
698 } TPM_SESSION_DATA;
699
700 typedef struct tdTPM_STANY_DATA{
701     TPM_STRUCTURE_TAG    tag;
702     TPM_NONCE            contextNonceSession;
703     TPM_DIGEST           auditDigest ;
704     TPM_CURRENT_TICKS    currentTicks;
705     UINT32               contextCount;
706     UINT32               contextList[TPM_MIN_SESSION_LIST];
707     TPM_SESSION_DATA     sessions[TPM_MIN_SESSIONS];
708 }TPM_STANY_DATA;
```

709 Parameters of STANY_Data

Type	Name	Description	Flag Name
TPM_STRUCTURE_TAG	tag	TPM_TAG_STANY_DATA	
TPM_NONCE	contextNonceSession	This is the nonce in use to properly identify saved session context blobs. This MUST be set to null on each TPM_Startup (ST_Clear). The nonce MAY be set to null on TPM_Startup(any).	TPM_AD_CONTEXTNONCESESSION
TPM_DIGEST	auditDigest	This is the extended value that is the audit log. This SHALL be set to NULLS at the start of each audit session.	TPM_AD_AUDITDIGEST
TPM_CURRENT_TICKS	currentTicks	This is the current tick counter. This is reset to 0 according to the rules when the TPM can tick. See the section on the tick counter for details.	TPM_AD_CURRENTTICKS
UINT32	contextCount	This is the counter to avoid session context blob replay attacks. This MUST be set to 0 on each TPM_Startup (ST_Clear). The value MAY be set to 0 on TPM_Startup (any).	TPM_AD_CONTEXTCOUNT
UINT32	contextList	This is the list of outstanding session blobs. All elements of this array MUST be set to 0 on each	TPM_AD_CONTEXTLIST

Type	Name	Description	Flag Name
		TPM_Startup (ST_Clear). The values MAY be set to 0 on TPM_Startup (any). TPM_MIN_SESSION_LIST MUST be 16 or greater.	
TPM_SESSION_DATA	sessions	List of current sessions. Sessions can be OSAP, OIAP, DSAP and Transport	TPM_AD_SESSIONS

710 Descriptions

- 711 1. The group of contextNonceSession, contextCount, contextList MUST reset at the same
712 time.
- 713 2. The contextList MUST keep track of UINT32 values. There is NO requirement that the
714 actual memory be 32 bits
- 715 3. contextList MUST support a minimum of 16 entries, it MAY support more.
- 716 4. The TPM MAY restrict the absolute difference between contextList entries
- 717 a. For instance if the TPM enforced distance was 10
- 718 i. Entries 8 and 15 would be valid
- 719 ii. Entries 8 and 28 would be invalid
- 720 b. The minimum distance that the TPM MUST support is 2^{16} , the TPM MAY support
721 larger distances

722 7.6.1 Flag Restrictions

Flag SubCap number 0x00000000 +	Set	Set restrictions	Actions from
+1 TPM_AD_CONTEXTNONCESESSION	N		
+2 TPM_AD_AUDITDIGEST	N		
+3 TPM_AD_CURRENTTICKS	N		
+4 TPM_AD_CONTEXTCOUNT	N		
+5 TPM_AD_CONTEXTLIST	N		
+6 TPM_AD_SESSIONS	N		

723 8. PCR Structures

724 **Start of informative comment**

725 The PCR structures expose the information in PCR register, allow for selection of PCR
726 register or registers in the SEAL operation and define what information is held in the PCR
727 register.

728 These structures are in use during the wrapping of keys and sealing of blobs.

729 **End of informative comment**

730 **8.1 TPM_PCR_SELECTION**731 **Start of informative comment**

732 This structure provides a standard method of specifying a list of PCR registers.

733 **Design points**734 1. The user needs to be able to specify the null set of PCR. The mask in pcrSelect indicates
735 if a PCR is active or not. Having the mask be a null value that specifies no selected PCR is
736 valid.737 2. The TPM must support a sizeofSelect that indicates the minimum number of PCR on the
738 platform. For a 1.2 PC TPM with 24 PCR this value would be 3.739 3. The TPM may support additional PCR over the platform minimum. When supporting
740 additional PCR the TPM must support a sizeofSelect that can indicate the use of an
741 individual PCR.742 4. The TPM may support sizeofSelect that reflects PCR use other than the maximum. For
743 instance, a PC TPM that supported 48 PCR would require support for a sizeofSelect of 6
744 and a sizeofSelect of 3 (for the 24 required PCR). The TPM could support sizes of 4 and 5.745 5. It is desirable for the TPM to support fixed size structures. Nothing in these rules
746 prevents a TPM from only supporting a known set of sizeofSelect structures.747 **Odd bit ordering**748 To the new reader the ordering of the PCR may seem strange. It is. However, the original
749 TPM vendors all interpreted the 1.0 specification to indicate the ordering as it is. The
750 scheme works and is understandable, so to avoid any backwards compatibility no change to
751 the ordering occurs in 1.2. The TPM vendor's interpretation of the 1.0 specification is the
752 start to the comment that there are no ambiguities in the specification just context sensitive
753 interpretations.754 **End of informative comment**755 **Definition**756

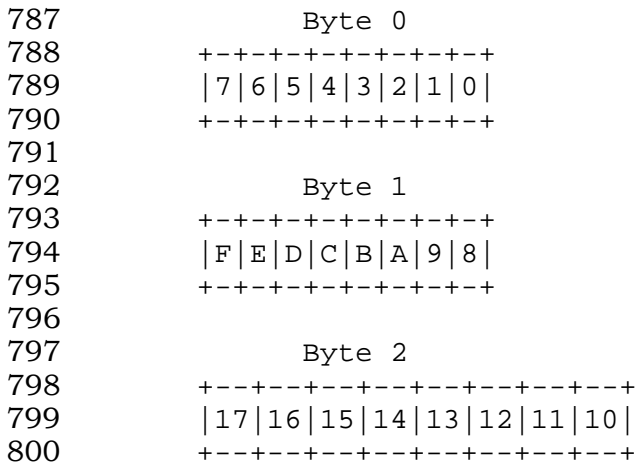
```
typedef struct tdTPM_PCR_SELECTION {  
757     UINT16 sizeofSelect;  
758     [size_is(sizeofSelect)] BYTE pcrSelect[];  
759 }
```

 TPM_PCR_SELECTION;760 **Parameters**

Type	Name	Description
UINT16	sizeofSelect	The size in bytes of the pcrSelect structure
BYTE []	pcrSelect	This SHALL be a bit map that indicates if a PCR is active or not

761 **Description**762 1. PCR selection occurs modulo 8. The minimum granularity for a PCR selection is 8. The
763 specification of registers MUST occur in banks of 8.

- 764 2. pcrSelect is a contiguous bit map that shows which PCR are selected. Each byte
765 represents 8 PCR. Byte 0 indicates PCR 0-7, byte 1 8-15 and so on. For each byte, the
766 individual bits represent a corresponding PCR. Refer to the figures below for the
767 mapping of an individual bit to a PCR within a byte. All pcrSelect bytes follow the same
768 mapping.
- 769 a. If the TPM supported 48 PCR to select PCR 0 and 47, the sizeofSelect would be 6 and
770 only two bits would be set to a 1. The remaining portion of pcrSelect would be NULL
- 771 3. When an individual bit is 1 the indicated PCR is selected. If 0 the PCR is not selected.
- 772 a. To select PCR 0, pcrSelect would be 00000001
- 773 b. To select PCR 7, pcrSelect would be 10000000
- 774 c. To select PCR 7 and 0, pcrSelect would be 10000001
- 775 4. If TPM_PCR_SELECTION.pcrSelect is all 0's
- 776 a. The process MUST set TPM_COMPOSITE_HASH to be all 0's.
- 777 5. Else
- 778 a. The process creates a TPM_PCR_COMPOSITE structure from the
779 TPM_PCR_SELECTION structure and the PCR values to be hashed. If constructed by
780 the TPM the values MUST come from the current PCR registers indicated by the PCR
781 indices in the TPM_PCR_SELECTION structure.
- 782 6. The TPM MUST support a sizeofSelect value that reflects the minimum number of PCR
783 as specified in the platform specific specification
- 784 7. The TPM MAY return an error if the sizeofSelect is a value greater than one that
785 represents the number of PCR on the TPM
- 786 8. The TPM MUST return an error if sizeofSelect is 0



801 **8.2 TPM_PCR_COMPOSITE**802 **Start of informative comment**

803 The composite structure provides the index and value of the PCR register to be used when
804 creating the value that SEALS an entity to the composite.

805 **End of informative comment**806 **Definition**

```
807 typedef struct tdTPM_PCR_COMPOSITE {
808     TPM_PCR_SELECTION select;
809     UINT32 valueSize;
810     [size_is(valueSize)] TPM_PCRVALUE pcrValue[];
811 } TPM_PCR_COMPOSITE;
```

812 **Parameters**

Type	Name	Description
TPM_PCR_SELECTION	select	This SHALL be the indication of which PCR values are active
UINT32	valueSize	This SHALL be the size of the pcrValue field
TPM_PCRVALUE	pcrValue[]	This SHALL be an array of TPM_PCRVALUE structures. The values come in the order specified by the select parameter and are concatenated into a single blob

813 8.3 TPM_PCR_INFO

814 Start of informative comment

815 The TPM_PCR_INFO structure contains the information related to the wrapping of a key or
816 the sealing of data, to a set of PCRs.

817 End of informative comment

818 Definition

```
819 typedef struct tdTPM_PCR_INFO{
820     TPM_PCR_SELECTION pcrSelection;
821     TPM_COMPOSITE_HASH digestAtRelease;
822     TPM_COMPOSITE_HASH digestAtCreation;
823 } TPM_PCR_INFO;
```

824 Parameters

Type	Name	Description
TPM_PCR_SELECTION	pcrSelection	This SHALL be the selection of PCRs to which the data or key is bound.
TPM_COMPOSITE_HASH	digestAtRelease	This SHALL be the digest of the PCR indices and PCR values to verify when revealing Sealed Data or using a key that was wrapped to PCRs.
TPM_COMPOSITE_HASH	digestAtCreation	This SHALL be the composite digest value of the PCR values, at the time when the sealing is performed.

825 **8.4 TPM_PCR_INFO_LONG**826 **Start of informative comment**

827 The TPM_PCR_INFO structure contains the information related to the wrapping of a key or
828 the sealing of data, to a set of PCRs.

829 The LONG version includes information necessary to properly define the configuration that
830 creates the blob using the PCR selection.

831 **End of informative comment**832 **Definition**

```
833 typedef struct tdTPM_PCR_INFO_LONG{
834     TPM_STRUCTURE_TAG tag;
835     TPM_LOCALITY_SELECTION localityAtCreation;
836     TPM_LOCALITY_SELECTION localityAtRelease;
837     TPM_PCR_SELECTION creationPCRSelection;
838     TPM_PCR_SELECTION releasePCRSelection;
839     TPM_COMPOSITE_HASH digestAtCreation;
840     TPM_COMPOSITE_HASH digestAtRelease;
841 } TPM_PCR_INFO_LONG;
```

842 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_PCR_INFO_LONG
TPM_LOCALITY_SELECTION	localityAtCreation	This SHALL be the locality modifier when the blob is created
TPM_LOCALITY_SELECTION	localityAtRelease	This SHALL be the locality modifier required to reveal Sealed Data or using a key that was wrapped to PCRs This value MUST not be zero (0).
TPM_PCR_SELECTION	creationPCRSelection	This SHALL be the selection of PCRs active when the blob is created
TPM_PCR_SELECTION	releasePCRSelection	This SHALL be the selection of PCRs to which the data or key is bound.
TPM_COMPOSITE_HASH	digestAtCreation	This SHALL be the composite digest value of the PCR values, when the blob is created
TPM_COMPOSITE_HASH	digestAtRelease	This SHALL be the digest of the PCR indices and PCR values to verify when revealing Sealed Data or using a key that was wrapped to PCRs.

843 8.5 TPM_PCR_INFO_SHORT

844 Start of informative comment

845 This structure is for defining a digest at release when the only information that is necessary
846 is the release configuration.

847 This structure does not have a tag to keep the structure short. Software and the TPM need
848 to evaluate the structures where the INFO_SHORT structure resides to avoid miss
849 identifying the INFO_SHORT structure.

850 End of informative comment

851 Definition

```
852 typedef struct tdTPM_PCR_INFO_SHORT{
853     TPM_PCR_SELECTION pcrSelection;
854     TPM_LOCALITY_SELECTION localityAtRelease;
855     TPM_COMPOSITE_HASH digestAtRelease;
856 } TPM_PCR_INFO_SHORT;
```

857 Parameters

Type	Name	Description
TPM_PCR_SELECTION	pcrSelection	This SHALL be the selection of PCRs that specifies the digestAtRelease
TPM_LOCALITY_SELECTION	localityAtRelease	This SHALL be the locality modifier required to release the information
TPM_COMPOSITE_HASH	digestAtRelease	This SHALL be the digest of the PCR indices and PCR values to verify when revealing auth data

858 **8.6 TPM_LOCALITY_SELECTION**859 **Start of informative comment**

860 When used with localityAtCreation only one bit is set and it corresponds to the locality of
861 the command creating the structure.

862 When used with localityAtRelease the bits indicate which localities CAN perform the release.

863 TPM_LOC_TWO would indicate that only locality 2 can perform the release

864 TPM_LOC_ONE || TPM_LOC_TWO would indicate that localities 1 or 2 could perform the
865 release

866 TPM_LOC_FOUR || TPM_LOC_THREE would indicate that localities 3 or 4 could perform
867 the release.

868 **End of informative comment**869 **Definition**

870 #define TPM_LOCALITY_SELECTION BYTE

871

Bit	Name	Description
7:5	Reserved	Must be 0
4	TPM_LOC_FOUR	Locality 4
3	TPM_LOC_THREE	Locality 3
2	TPM_LOC_TWO	Locality 2
1	TPM_LOC_ONE	Locality 1
0	TPM_LOC_ZERO	Locality 0. This is the same as the legacy interface.

872

873 The TPM MUST treat a value of 0 as an error. The default value is 0x1F which indicates that
874 localities 0-4 have been selected.

875 **8.7 PCR Attributes**

876 **Start of informative comment**

877 The PCR registers will have attributes associated with the PCR register. These attributes
878 allow for the PCR registers to be differentiated between other PCR registers.

879 This specification defines the generic meaning of the attributes. For a specific platform the
880 actual setting of the attribute is a platform specific issue.

881 The attributes are values that are set during the manufacturing process of the TPM and
882 platform and are not field settable or changeable values.

883 To accommodate debugging PCR[15] for all platforms will have a certain set of attributes.
884 The setting of these attributes is to allow for easy debugging. This means that values in
885 PCR[15] provide no security information. It is anticipated that PCR[15] would be set by a
886 developer during their development cycle. Developers are responsible for ensuring that a
887 conflict between two programs does not invalidate the settings they are interested in.

888 The attributes are pcrReset, pcrResetLocal, pcrExtendLocal. Attributes can be set in any
889 combination that is appropriate for the platform.

890 The pcrReset attribute allows the PCR to be reset at times other than TPM_STARTUP.

891 The pcrResetLocal attribute allows the PCR to be reset at times other than TPM_STARTUP.
892 The reset is legal when the mapping of the command locality to PCR flags results in accept.
893 See 8.8.1 for details.

894 The pcrExtendLocal attribute modifies the PCR such that the PCR can only be Extended
895 when the mapping of the command locality to PCR flags results in accept. See 8.8.1 for
896 details.

897 **End of informative comment**

- 898 1. The PCR attributes MUST be set during manufacturing.
- 899 2. For a specific PCR register, the PCR attributes MUST match the requirements of the
900 TCG platform specific specification that describes the platform.

901 **8.8 TPM_PCR_ATTRIBUTES**902 **Informative comment :**

903 These attributes are available on a per PCR basis.

904 The TPM is not required to maintain this structure internally to the TPM.

905 When a challenger evaluates a PCR an understanding of this structure is vital to the proper
906 understanding of the platform configuration. As this structure is static for all platforms of
907 the same type the structure does not need to be reported with each quote.908 **End of informative comment**909 **Definition**

```

910 typedef struct tdTPM_PCR_ATTRIBUTES{
911     BOOL pcrReset;
912     TPM_LOCALITY_SELECTION pcrExtendLocal;
913     TPM_LOCALITY_SELECTION pcrResetLocal;
914 } TPM_PCR_ATTRIBUTES;

```

915 **Types of Persistent Data**

Type	Name	Description
BOOL	pcrReset	A value of TRUE SHALL indicate that the PCR register can be reset using the TPM_PCR_RESET command. If pcrReset is: FALSE- Default value of the PCR MUST be 0x00..00 Reset on TPM_Startup(ST_Clear) only Saved by TPM_SaveState Can not be reset by TPM_PCR_Reset TRUE – Default value of the PCR MUST be 0xFF..FF. Reset on TPM_Startup(any) MUST not be part of any state stored by TPM_SaveState Can be reset by TPM_PCR_Reset When reset as part of HASH_START the starting value MUST be 0x00..00
TPM_LOCALITY_SELECTION	pcrResetLocal	An indication of which localities can reset the PCR
TPM_LOCALITY_SELECTION	pcrExtendLocal	An indication of which localities can perform extends on the PCR.

916 **8.8.1 Comparing command locality to PCR flags**

917 **Start of informative comment**

918 This is an informative section to show the details of how to check locality against the
919 locality modifier received with a command. The operation works for any of reset, extend or
920 use but for example this will use read.

921 Map L1 to TPM_STANY_FLAGS -> localityModifier

922 Map P1 to TPM_PERMANENT_DATA -> pcrAttrib->[selectedPCR].pcrExtendLocal

923 If, for the value L1, the corresponding bit is set in the bit map P1

924 return accept

925 else return reject

926 **End of informative comment**

927 **8.9 Debug PCR register**928 **Start of informative comment**

929 There is a need to define a PCR that allows for debugging. The attributes of the debug
930 register are such that it is easy to reset but the register provides no measurement value
931 that can not be spoofed. Production applications should not use the debug PCR for any
932 SEAL or other operations. The anticipation is that the debug PCR is set and used by
933 application developers during the application development cycle. Developers are responsible
934 for ensuring that a conflict between two programs does not invalidate the settings they are
935 interested in.

936 The specific register that is the debug PCR MUST be set by the platform specific
937 specification.

938 **End of informative comment**

939 The attributes for the debug PCR SHALL be the following:

```
940     pcrReset = TRUE;  
941     pcrResetLocal = 0x1f;  
942     pcrExtendLocal = 0x1f;  
943     pcrUseLocal = 0x1f;
```

944

945 These settings are to create a PCR register that developers can use to reset at any time
946 during their development cycle.

947 The debug PCR does NOT need to be saved during TPM_SaveState

948 8.10 Mapping PCR Structures

949 **Start of informative comment**

950 When moving information from one PCR structure type to another, i.e. TPM_PCR_INFO to
951 TPM_PCR_INFO_SHORT, the mapping between fields could be ambiguous. This section
952 describes how the various fields map and what the TPM must do when adding or losing
953 information.

954 **End of informative comment**

- 955 1. Set IN to TPM_PCR_INFO
- 956 2. Set IL is TPM_PCR_INFO_LONG
- 957 3. Set IS is TPM_PCR_INFO_SHORT
- 958 4. To set IS from IN
 - 959 a. Set IS -> pcrSelection to IN -> pcrSelection
 - 960 b. Set IS -> digestAtRelease to IN -> digestAtRelease
 - 961 c. Set IS -> localityAtRelease to 0x1F to indicate all localities are valid
 - 962 d. Ignore IN -> digestAtCreation
- 963 5. To set IS from IL
 - 964 a. Set IS -> pcrSelection to IL -> releasePCRSelection
 - 965 b. Set IS -> localityAtRelease to IL -> localityAtRelease
 - 966 c. Set IS -> digestAtRelease to IL -> digestAtRelease
 - 967 d. Ignore all other IL values
- 968 6. To set IL from IN
 - 969 a. Set IL -> localityAtCreation to 0x1F
 - 970 b. Set IL -> localityAtRelease to 0x1F
 - 971 c. Set IL -> creationPCRSelection to IN -> pcrSelection
 - 972 d. Set IL -> releasePCRSelection to IN -> pcrSelection
 - 973 e. Set IL -> digestAtRelease to IN -> digestAtRelease
 - 974 f. Set IL -> digestAtRelease to IN -> digestAtRelease
- 975 7. To set IL from IS
 - 976 a. Set IL -> localityAtCreation to 0x1F
 - 977 b. Set IL -> localityAtRelease to IS localityAtRelease
 - 978 c. Set IL -> creationPCRSelection to NULL
 - 979 d. Set IL -> releasePCRSelection to IS -> pcrSelection
 - 980 e. Set IL -> digestAtCreation to NULL
 - 981 f. Set IL -> digestAtRelease to IS -> digestAtRelease
- 982 8. To set IN from IS

- 983 a. Set IN -> pcrSelection to IS -> pcrSelection
- 984 b. Set IN -> digestAtRelease to IS -> digestAtRelease
- 985 c. Set IN -> digestAtCreation to NULL
- 986 9. To set IN from IL
- 987 a. Set IN -> pcrSelection to IL -> releasePCRSelection
- 988 b. Set IN -> digestAtRelease to IL -> digestAtRelease
- 989 c. If IL -> creationPCRSelection and IL -> localityAtCreation both match IL ->
- 990 releasePCRSelection and IL -> localityAtRelease
- 991 i. Set IN -> digestAtCreation to IL -> digestAtCreation
- 992 d. Else
- 993 i. Set IN -> digestAtCreation to NULL

994 9. Storage Structures

995 9.1 TPM_STORED_DATA

996 Start of informative comment

997 The definition of this structure is necessary to ensure the enforcement of security
998 properties.

999 This structure is in use by the TPM_Seal and TPM_Unseal commands to identify the PCR
1000 index and values that must be present to properly unseal the data.

1001 This structure only provides 1.1 data store and uses PCR_INFO

1002 End of informative comment

1003 Definition

```
1004 typedef struct tdTPM_STORED_DATA {
1005     TPM_STRUCTURE_VER ver;
1006     UINT32 sealInfoSize;
1007     [size_is(sealInfoSize)] BYTE* sealInfo;
1008     UINT32 encDataSize;
1009     [size_is(encDataSize)] BYTE* encData;
1010 } TPM_STORED_DATA;
```

1011 Parameters

Type	Name	Description
TPM_STRUCTURE_VER	ver	This MUST be 1.1.0.0
UINT32	sealInfoSize	Size of the sealInfo parameter
BYTE*	sealInfo	This SHALL be a structure of type TPM_PCR_INFO or a 0 length array if the data is not bound to PCRs.
UINT32	encDataSize	This SHALL be the size of the encData parameter
BYTE*	encData	This shall be an encrypted TPM_SEALED_DATA structure containing the confidential part of the data.

1012 Descriptions

1013 1. This structure is created during the TPM_Seal process. The confidential data is
1014 encrypted using a non-migratable key. When the TPM_Unseal decrypts this structure
1015 the TPM_Unseal uses the public information in the structure to validate the current
1016 configuration and release the decrypted data

1017 2. When sealInfoSize is not 0 sealInfo MUST be TPM_PCR_INFO

1018 **9.2 TPM_STORED_DATA12**1019 **Start of informative comment**

1020 The definition of this structure is necessary to ensure the enforcement of security
1021 properties.

1022 This structure is in use by the TPM_Seal and TPM_Unseal commands to identify the PCR
1023 index and values that must be present to properly unseal the data.

1024 **End of informative comment**1025 **Definition**

```
1026 typedef struct tdTPM_STORED_DATA12 {
1027     TPM_STRUCTURE_TAG tag;
1028     TPM_ENTITY_TYPE et;
1029     UINT32 sealInfoSize;
1030     [size_is(sealInfoSize)] BYTE* sealInfo;
1031     UINT32 encDataSize;
1032     [size_is(encDataSize)] BYTE* encData;
1033 } TPM_STORED_DATA12;
```

1034 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_STORED_DATA12
TPM_ENTITY_TYPE	et	The type of blob
UINT32	sealInfoSize	Size of the sealInfo parameter
BYTE*	sealInfo	This SHALL be a structure of type TPM_PCR_INFO_LONG
UINT32	encDataSize	This SHALL be the size of the encData parameter
BYTE*	encData	This shall be an encrypted TPM_SEALED_DATA structure containing the confidential part of the data.

1035 **Descriptions**

1036 1. This structure is created during the TPM_Seal process. The confidential data is
1037 encrypted using a non-migratable key. When the TPM_Unseal decrypts this structure
1038 the TPM_Unseal uses the public information in the structure to validate the current
1039 configuration and release the decrypted data.

1040 2. If sealInfoSize is not 0 then sealInfo MUST be TPM_PCR_INFO_LONG

1041 **9.3 TPM_SEALED_DATA**

1042 **Start of informative comment**

1043 This structure contains confidential information related to sealed data, including the data
1044 itself.

1045 **End of informative comment**

1046 **Definition**

```
1047 typedef struct tdTPM_SEALED_DATA {
1048     TPM_PAYLOAD_TYPE payload;
1049     TPM_SECRET authData;
1050     TPM_NONCE tpmProof;
1051     TPM_DIGEST storedDigest;
1052     UINT32 dataSize;
1053     [size_is(dataSize)] BYTE* data;
1054 } TPM_SEALED_DATA;
```

1055 **Parameters**

Type	Name	Description
TPM_PAYLOAD_TYPE	payload	This SHALL indicate the payload type of TPM_PT_SEAL
TPM_SECRET	authData	This SHALL be the AuthData data for this value
TPM_NONCE	tpmProof	This SHALL be a copy of TPM_PERMANENT_DATA -> tpmProof
TPM_DIGEST	storedDigest	This SHALL be a digest of the TPM_STORED_DATA structure, excluding the fields TPM_STORED_DATA -> encDataSize and TPM_STORED_DATA -> encData.
UINT32	dataSize	This SHALL be the size of the data parameter
BYTE*	data	This SHALL be the data to be sealed

1056 **Description**

- 1057 1. To tie the TPM_STORED_DATA structure to the TPM_SEALED_DATA structure this
- 1058 structure contains a digest of the containing TPM_STORED_DATA structure.
- 1059 2. The digest calculation does not include the encDataSize and encData parameters.

1060 **9.4 TPM_SYMMETRIC_KEY**1061 **Start of informative comment**

1062 This structure describes a symmetric key, used during the process “Collating a Request for
1063 a Trusted Platform Module Identity”.

1064 **End of informative comment**1065 **Definition**

```
1066 typedef struct tdTPM_SYMMETRIC_KEY {
1067     TPM_ALGORITHM_ID algId;
1068     TPM_ENC_SCHEME encScheme;
1069     UINT16 size;
1070     [size_is(size)] BYTE* data;
1071 } TPM_SYMMETRIC_KEY;
```

1072 **Parameters**

Type	Name	Description
TPM_ALGORITHM_ID	algId	This SHALL be the algorithm identifier of the symmetric key.
TPM_ENC_SCHEME	encScheme	This SHALL fully identify the manner in which the key will be used for encryption operations.
UINT16	size	This SHALL be the size of the data parameter in bytes
BYTE*	data	This SHALL be the symmetric key data

1073 **9.5 TPM_BOUND_DATA**

1074 **Start of informative comment**

1075 This structure is defined because it is used by a TPM_UnBind command in a consistency
1076 check.

1077 The intent of TCG is to promote “best practice” heuristics for the use of keys: a signing key
1078 shouldn’t be used for storage, and so on. These heuristics are used because of the potential
1079 threats that arise when the same key is used in different ways. The heuristics minimize the
1080 number of ways in which a given key can be used.

1081 One such heuristic is that a key of type TPM_KEY_BIND, and no other type of key, should
1082 always be used to create the blob that is unwrapped by TPM_UnBind. Binding is not a TPM
1083 function, so the only choice is to perform a check for the correct payload type when a blob
1084 is unwrapped by a key of type TPM_KEY_BIND. This requires the blob to have internal
1085 structure.

1086 Even though payloadData has variable size, TPM_BOUND_DATA deliberately does not
1087 include the size of payloadData. This is to maximise the size of payloadData that can be
1088 encrypted when TPM_BOUND_DATA is encrypted in a single block. When using TPM-
1089 UnBind to obtain payloadData, the size of payloadData is deduced as a natural result of the
1090 (RSA) decryption process.

1091 **End of informative comment**

1092 **Definition**

```
1093 typedef struct tdTPM_BOUND_DATA {
1094     TPM_STRUCT_VER ver;
1095     TPM_PAYLOAD_TYPE payload;
1096     BYTE[] payloadData;
1097 } TPM_BOUND_DATA;
```

1098 **Parameters**

Type	Name	Description
TPM_STRUCT_VER	ver	This MUST be 1.1.0.0
TPM_PAYLOAD_TYPE	payload	This SHALL be the value TPM_PT_BIND
BYTE[]	payloadData	The bound data

1099 **Descriptions**

- 1100 1. This structure MUST be used for creating data when (wrapping with a key of type
1101 TPM_KEY_BIND) or (wrapping using the encryption algorithm
1102 TPM_ES_RSAESOAEP_SHA1_M). If it is not, the TPM_UnBind command will fail.

1103 10. TPM_KEY complex**1104 Start of informative comment**

1105 The TPA_KEY complex is where all of the information regarding keys is kept. These
1106 structures combine to fully define and protect the information regarding an asymmetric key.

1107 This version of the specification only fully defines RSA keys, however the design is such that
1108 in the future when other asymmetric algorithms are available the general structure will not
1109 change.

1110 One overriding design goal is for a 2048 bit RSA key to be able to properly protect another
1111 2048 bit RSA key. This stems from the fact that the SRK is a 2048 bit key and all identities
1112 are 2048 bit keys. A goal is to have these keys only require one decryption when loading an
1113 identity into the TPM. The structures as defined meet this goal.

1114 Every TPM_KEY is allowed only one encryption scheme or one signature scheme (or one of
1115 each in the case of legacy keys) throughout its lifetime. Note however that more than one
1116 scheme could be used with externally generated keys, by introducing the same key in
1117 multiple blobs.

1118 End of informative comment:

1119 10.1 TPM_KEY_PARMS

1120 Start of informative comment

1121 This provides a standard mechanism to define the parameters used to generate a key pair,
1122 and to store the parts of a key shared between the public and private key parts.

1123 End of informative comment

1124 Definition

```
1125 typedef struct tdTPM_KEY_PARMS {
1126     TPM_ALGORITHM_ID algorithmID;
1127     TPM_ENC_SCHEME encScheme;
1128     TPM_SIG_SCHEME sigScheme;
1129     UINT32 parmSize;
1130     [size_is(parmSize)] BYTE* parms;
1131 } TPM_KEY_PARMS;
```

1132 Parameters

Type	Name	Description
TPM_ALGORITHM_ID	algorithmID	This SHALL be the key algorithm in use
TPM_ENC_SCHEME	encScheme	This SHALL be the encryption scheme that the key uses to encrypt information
TPM_SIG_SCHEME	sigScheme	This SHALL be the signature scheme that the key uses to perform digital signatures
UINT32	parmSize	This SHALL be the size of the parms field in bytes
BYTE[]	parms	This SHALL be the parameter information dependant upon the key algorithm.

1133 Descriptions

1134 The contents of the ‘parms’ field will vary depending upon algorithmId:

Algorithm Id	PARMS Contents
TPM_ALG_RSA	A structure of type TPM_RSA_KEY_PARMS
TPM_ALG_DES	A structure of type TPM_SYMMETRIC_KEY_PARMS
TPM_ALG_3DES	A structure of type TPM_SYMMETRIC_KEY_PARMS
TPM_ALG_SHA	No content
TPM_ALG_HMAC	No content
TPM_ALG_AES	A structure of type TPM_SYMMETRIC_KEY_PARMS
TPM_ALG_MGF1	No content

1135 **10.1.1 TPM_RSA_KEY_PARMS**1136 **Start of informative comment**

1137 This structure describes the parameters of an RSA key.

1138 **End of informative comment**1139 **Definition**

```

1140 typedef struct tdTPM_RSA_KEY_PARMS {
1141     UINT32 keyLength;
1142     UINT32 numPrimes;
1143     UINT32 exponentSize;
1144     BYTE[] exponent;
1145 } TPM_RSA_KEY_PARMS;

```

1146 **Parameters**

Type	Name	Description
UINT32	keyLength	This specifies the size of the RSA key in bits
UINT32	numPrimes	This specifies the number of prime factors used by this RSA key.
UINT32	exponentSize	This SHALL be the size of the exponent. If the key is using the default exponent then the exponentSize MUST be 0.
BYTE[]	exponent	The public exponent of this key

1147 **10.1.2 TPM_SYMMETRIC_KEY_PARMS**1148 **Start of informative comment**

1149 This structure describes the parameters for symmetric algorithms

1150 **End of informative comment**1151 **Definition**

```

1152 typedef struct tdTPM_SYMMETRIC_KEY_PARMS {
1153     UINT32 keyLength;
1154     UINT32 blockSize;
1155     UINT32 ivSize;
1156     [size_is(ivSize)] BYTE IV;
1157 } TPM_SYMMETRIC_KEY_PARMS;

```

1158 **Parameters**

Type	Name	Description
UINT32	keyLength	This SHALL indicate the length of the key in bits
UINT32	blockSize	This SHALL indicate the block size of the algorithm
UINT32	ivSize	This SHALL indicate the size of the IV
BYTE[]	IV	The initialization vector

1159 **10.2 TPM_KEY**

1160 **Start of informative comment**

1161 The TPM_KEY structure provides a mechanism to transport the entire asymmetric key pair.
1162 The private portion of the key is always encrypted.

1163 The reason for using a size and pointer for the PCR info structure is save space when the
1164 key is not bound to a PCR. The only time the information for the PCR is kept with the key is
1165 when the key needs PCR info.

1166 The 1.2 version has a change in the PCRInfo area. For 1.2 the structure uses the
1167 TPM_PCR_INFO_LONG structure to properly define the PCR registers in use.

1168 **End of informative comment:**

1169 **Definition**

```
1170 typedef struct tdTPM_KEY{
1171     TPM_STRUCT_VER ver;
1172     TPM_KEY_USAGE keyUsage;
1173     TPM_KEY_FLAGS keyFlags;
1174     TPM_AUTH_DATA_USAGE authDataUsage;
1175     TPM_KEY_PARMS algorithmParms;
1176     UINT32 PCRInfoSize;
1177     BYTE* PCRInfo;
1178     TPM_STORE_PUBKEY pubKey;
1179     UINT32 encDataSize;
1180     [size_is(encDataSize)] BYTE* encData;
1181 } TPM_KEY;
```

1182 **Parameters**

Type	Name	Description
TPM_STRUCT_VER	ver	This MUST be 1.1.0.0
TPM_KEY_USAGE	keyUsage	This SHALL be the TPM key usage that determines the operations permitted with this key
TPM_KEY_FLAGS	keyFlags	This SHALL be the indication of migration, redirection etc.
TPM_AUTH_DATA_USAGE	authDataUsage	This SHALL Indicate the conditions where it is required that authorization be presented.
TPM_KEY_PARMS	algorithmParms	This SHALL be the information regarding the algorithm for this key
UINT32	PCRInfoSize	This SHALL be the length of the pcrInfo parameter. If the key is not bound to a PCR this value SHOULD be 0.
BYTE*	PCRInfo	This SHALL be a structure of type TPM_PCR_INFO, or an empty array if the key is not bound to PCRs.
TPM_STORE_PUBKEY	pubKey	This SHALL be the public portion of the key
UINT32	encDataSize	This SHALL be the size of the encData parameter.
BYTE*	encData	This SHALL be an encrypted TPM_STORE_ASYMKEY structure or TPM_MIGRATE_ASYMKEY structure

1183 **Version handling**

- 1184 1. A TPM MUST be able to read and create TPM_KEY structures
- 1185 2. A TPM MUST not allow a TPM_KEY structure to contain a TPM_PCR_INFO_LONG
- 1186 structure

1187 **10.3 TPM_KEY12**1188 **Start of informative comment**

1189 This provides the same functionality as TPM_KEY but uses the new PCR_INFO_LONG
1190 structures and the new structure tagging. In all other aspects this is the same structure.

1191 **End of informative comment:**1192 **Definition**

```
1193 typedef struct tdTPM_KEY12{
1194     TPM_STRUCTURE_TAG tag;
1195     UINT16 fill;
1196     TPM_KEY_USAGE keyUsage;
1197     TPM_KEY_FLAGS keyFlags;
1198     TPM_AUTH_DATA_USAGE authDataUsage;
1199     TPM_KEY_PARMS algorithmParms;
1200     UINT32 PCRInfoSize;
1201     BYTE* PCRInfo;
1202     TPM_STORE_PUBKEY pubKey;
1203     UINT32 encDataSize;
1204     [size_is(encDataSize)] BYTE* encData;
1205 } TPM_KEY12;
```

1206 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_KEY12
UINT16	fill	MUST be 0x0000
TPM_KEY_USAGE	keyUsage	This SHALL be the TPM key usage that determines the operations permitted with this key
TPM_KEY_FLAGS	keyFlags	This SHALL be the indication of migration, redirection etc.
TPM_AUTH_DATA_USAGE	authDataUsage	This SHALL Indicate the conditions where it is required that authorization be presented.
TPM_KEY_PARMS	algorithmParms	This SHALL be the information regarding the algorithm for this key
UINT32	PCRInfoSize	This SHALL be the length of the pcrInfo parameter. If the key is not bound to a PCR this value SHOULD be 0.
BYTE*	PCRInfo	This SHALL be a structure of type TPM_PCR_INFO_LONG,
TPM_STORE_PUBKEY	pubKey	This SHALL be the public portion of the key
UINT32	encDataSize	This SHALL be the size of the encData parameter.
BYTE*	encData	This SHALL be an encrypted TPM_STORE_ASYMKEY structure TPM_MIGRATE_ASYMKEY structure

1207 **Version handling**

- 1208 1. The TPM MUST be able to read and create TPM_KEY12 structures
- 1209 2. The TPM MUST not allow a TPM_KEY12 structure to contain a TPM_PCR_INFO structure

1210 **10.4 TPM_STORE_PUBKEY**

1211 **Start of informative comment**

1212 This structure can be used in conjunction with a corresponding TPM_KEY_PARMS to
1213 construct a public key which can be unambiguously used.

1214 **End of informative comment**

```
1215 typedef struct tdTPM_STORE_PUBKEY {
1216     UINT32 keyLength;
1217     BYTE[] key;
1218 } TPM_STORE_PUBKEY;
```

1219 **Parameters**

Type	Name	Description
UINT32	keyLength	This SHALL be the length of the key field.
BYTE[]	key	This SHALL be a structure interpreted according to the algorithm Id in the corresponding TPM_KEY_PARMS structure.

1220 **Descriptions**

1221 The contents of the ‘key’ field will vary depending upon the corresponding key algorithm:

Algorithm Id	‘Key’ Contents
TPM_ALG_RSA	The RSA public modulus

1222 **10.5 TPM_PUBKEY**1223 **Start of informative comment**

1224 The TPM_PUBKEY structure contains the public portion of an asymmetric key pair. It
 1225 contains all the information necessary for it's unambiguous usage. It is possible to
 1226 construct this structure from a TPM_KEY, using the algorithmParms and pubKey fields.

1227 **End of informative comment**1228 **Definition**

```
1229 typedef struct tdTPM_PUBKEY{
1230     TPM_KEY_PARMS algorithmParms;
1231     TPM_STORE_PUBKEY pubKey;
1232 } TPM_PUBKEY;
```

1233 **Parameters**

Type	Name	Description
TPM_KEY_PARMS	algorithmParms	This SHALL be the information regarding this key
TPM_STORE_PUBKEY	pubKey	This SHALL be the public key information

1234 **Descriptions**

1235 The pubKey member of this structure shall contain the public key for a specific algorithm.

1236 10.6 TPM_STORE_ASYMKEY

1237 Start of informative comment

1238 The TPM_STORE_ASYMKEY structure provides the area to identify the confidential
1239 information related to a key. This will include the private key factors for an asymmetric
1240 key.

1241 The structure is designed so that encryption of a TPM_STORE_ASYMKEY structure
1242 containing a 2048 bit RSA key can be done in one operation if the encrypting key is 2048
1243 bits.

1244 Using typical RSA notation the structure would include P, and when loading the key include
1245 the unencrypted P*Q which would be used to recover the Q value.

1246 To accommodate the future use of multiple prime RSA keys the specification of additional
1247 prime factors is an optional capability.

1248 This structure provides the basis of defining the protection of the private key.

1249 Changes in this structure MUST be reflected in the TPM_MIGRATE_ASYMKEY structure
1250 (section 10.8).

1251 End of informative comment

1252 Definition

```
1253 typedef struct tdTPM_STORE_ASYMKEY { // pos len total
1254     TPM_PAYLOAD_TYPE payload; // 0 1 1
1255     TPM_SECRET usageAuth; // 1 20 21
1256     TPM_SECRET migrationAuth; // 21 20 41
1257     TPM_DIGEST pubDataDigest; // 41 20 61
1258     TPM_STORE_PRIVKEY privKey; // 61 132-151 193-214
1259 } TPM_STORE_ASYMKEY;
```

1260 Parameters

Type	Name	Description
TPM_PAYLOAD_TYPE	payload	This SHALL set to TPM_PT_ASYM to indicate an asymmetric key. If used in TPM_CMK_ConvertMigration the value SHALL be TPM_PT_MIGRATE_EXTERNAL If used in TPM_CMK_CreateKey the value SHALL be TPM_PT_MIGRATE_RESTRICTED
TPM_SECRET	usageAuth	This SHALL be the AuthData data necessary to authorize the use of this value
TPM_SECRET	migrationAuth	This SHALL be the migration AuthData data for a migratable key, or the TPM secret value tpmProof for a non-migratable key created by the TPM. If the TPM sets this parameter to the value tpmProof, then the TPM_KEY.keyFlags.migratable of the corresponding TPM_KEY structure MUST be set to 0. If this parameter is set to the migration AuthData data for the key in parameter PrivKey, then the TPM_KEY.keyFlags.migratable of the corresponding TPM_KEY structure SHOULD be set to 1.
TPM_DIGEST	pubDataDigest	This SHALL be the digest of the corresponding TPM_KEY structure, excluding the fields TPM_KEY.encSize and TPM_KEY.encData. When TPM_KEY->pcrInfoSize is 0 then the digest calculation has no input from the pcrInfo field. The pcrInfoSize field MUST always be part of the digest calculation.
TPM_STORE_PRIVKEY	privKey	This SHALL be the private key data. The privKey can be a variable length which allows for differences in the key format. The maximum size of the area would be 151 bytes.

1261 **10.7 TPM_STORE_PRIVKEY**1262 **Start of informative comment**

1263 This structure can be used in conjunction with a corresponding TPM_PUBKEY to construct
1264 a private key which can be unambiguously used.

1265 **End of informative comment**

```
1266 typedef struct tdTPM_STORE_PRIVKEY {
1267     UINT32 keyLength;
1268     [size_is(keyLength)] BYTE* key;
1269 } TPM_STORE_PRIVKEY;
```

1270 **Parameters**

Type	Name	Description
UINT32	keyLength	This SHALL be the length of the key field.
BYTE*	key	This SHALL be a structure interpreted according to the algorithm Id in the corresponding TPM_KEY structure.

1271 **Descriptions**

1272 All migratable keys MUST be RSA keys with two (2) prime factors.

1273 For non-migratable keys, the size, format and contents of privKey.key MAY be vendor
1274 specific and MAY not be the same as that used for migratable keys. The level of
1275 cryptographic protection MUST be at least as strong as a migratable key.

Algorithm Id	key Contents
TPM_ALG_RSA	When the numPrimes defined in the corresponding TPM_RSA_KEY_PARMS field is 2, this shall be one of the prime factors of the key. Upon loading of the key the TPM calculates the other prime factor by dividing the modulus, TPM_RSA_PUBKEY, by this value. The TPM MAY support RSA keys with more than two prime factors. Definition of the storage structure for these keys is left to the TPM Manufacturer.

1276 **10.8 TPM_MIGRATE_ASYMKEY**

1277 **Start of informative comment**

1278 The TPM_MIGRATE_ASYMKEY structure provides the area to identify the private key factors
1279 of a asymmetric key while the key is migrating between TPM's.

1280 This structure provides the basis of defining the protection of the private key.

1281 **End of informative comment**

1282 **Definition**

```

1283 typedef struct tdTPM_MIGRATE_ASYMKEY {           // pos   len   total
1284     TPM_PAYLOAD_TYPE payload;                   //    0    1     1
1285     TPM_SECRET usageAuth;                       //    1   20    21
1286     TPM_DIGEST pubDataDigest;                  //   21   20    41
1287     UINT32 partPrivKeyLen;                      //   41    4    45
1288     [size_is(partPrivKeyLen)] BYTE* partPrivKey; //   45  112-127 157-172
1289 } TPM_MIGRATE_ASYMKEY;

```

1290 **Parameters**

Type	Name	Description
TPM_PAYLOAD_TYPE	payload	This SHALL set to TPM_PT_MIGRATE or TPM_PT_CMK_MIGRATE to indicate an migrating asymmetric key or TPM_PT_MAINT to indicate a maintenance key.
TPM_SECRET	usageAuth	This SHALL be a copy of the usageAuth from the TPM_STORE_ASYMKEY structure.
TPM_DIGEST	pubDataDigest	This SHALL be a copy of the pubDataDigest from the TPM_STORE_ASYMKEY structure.
UINT32	partPrivKeyLen	This SHALL be the size of the partPrivKey field
BYTE*	partPrivKey	This SHALL be the k2 area as described in TPM_CreateMigrationBlob

1291 **10.9 TPM_KEY_CONTROL**1292 **Start of informative comment**

1293 Attributes that can control various aspects of key usage and manipulation

1294 **End of informative comment**

Bit	Name	Description
31:1	Reserved	Must be 0
0	TPM_KEY_CONTROL_OWNER_EVICT	Owner controls when the key is evicted from the TPM. When set the TPM MUST preserve key the key across all TPM_Init invocations.

1295 11. Signed Structures

1296 11.1 TPM_CERTIFY_INFO Structure

1297 Start of informative comment

1298 When the TPM certifies a key, it must provide a signature with a TPM identity key on
1299 information that describes that key. This structure provides the mechanism to do so.

1300 Key usage and keyFlags must have their upper byte set to null to avoid collisions with the
1301 other signature headers.

1302 End of informative comment

1303 Definition

```
1304 typedef struct tdTPM_CERTIFY_INFO{
1305     TPM_STRUCTURE_VERSION version;
1306     TPM_KEY_USAGE keyUsage;
1307     TPM_KEY_FLAGS keyFlags;
1308     TPM_AUTH_DATA_USAGE authDataUsage;
1309     TPM_KEY_PARMS algorithmParms;
1310     TPM_DIGEST pubkeyDigest;
1311     TPM_NONCE data;
1312     BOOL parentPCRStatus;
1313     UINT32 PCRInfoSize;
1314     [size_is(PCRInfoSize)] BYTE* PCRInfo;
1315 } TPM_CERTIFY_INFO;
```

1316 Parameters

Type	Name	Description
TPM_STRUCTURE_VERSION	version	This MUST be 1.1.0.0
TPM_KEY_USAGE	keyUsage	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified. The upper byte MUST be NULL
TPM_KEY_FLAGS	keyFlags	This SHALL be set to the same value as the corresponding parameter in the TPM_KEY structure that describes the public key that is being certified. The upper byte MUST be NULL.
TPM_AUTH_DATA_USAGE	authDataUsage	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified
TPM_KEY_PARMS	algorithmParms	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified
TPM_DIGEST	pubKeyDigest	This SHALL be a digest of the value TPM_KEY -> pubKey -> key in a TPM_KEY representation of the key to be certified
TPM_NONCE	data	This SHALL be externally provided data.
BOOL	parentPCRStatus	This SHALL indicate if any parent key was wrapped to a PCR
UINT32	PCRInfoSize	This SHALL be the size of the PCRInfo parameter. A value of zero indicates that the key is not wrapped to a PCR
BYTE*	PCRInfo	This SHALL be the TPM_PCR_INFO structure.

1317 **11.2 TPM_CERTIFY_INFO2 Structure**1318 **Start of informative comment**

1319 When the TPM certifies a key, it must provide a signature with a TPM identity key on
1320 information that describes that key. This structure provides the mechanism to do so.

1321 Key usage and keyFlags must have their upper byte set to null to avoid collisions with the
1322 other signature headers.

1323 **End of informative comment**1324 **Definition**

```
1325 typedef struct tdTPM_CERTIFY_INFO2{
1326     TPM_STRUCTURE_TAG tag;
1327     BYTE fill;
1328     TPM_PAYLOAD_TYPE payloadType;
1329     TPM_KEY_USAGE keyUsage;
1330     TPM_KEY_FLAGS keyFlags;
1331     TPM_AUTH_DATA_USAGE authDataUsage;
1332     TPM_KEY_PARMS algorithmParms;
1333     TPM_DIGEST pubkeyDigest;
1334     TPM_NONCE data;
1335     BOOL parentPCRStatus;
1336     UINT32 PCRInfoSize;
1337     [size_is(PCRInfoSize)] BYTE* PCRInfo;
1338     UINT32 migrationAuthoritySize ;
1339     [size_is(migrationAuthoritySize)] BYTE migrationAuthority;
1340 } TPM_CERTIFY_INFO2;
```

1341 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_CERTIFY_INFO2
BYTE	fill	MUST be 0x00
TPM_PAYLOAD_TYPE	payloadType	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified
TPM_KEY_USAGE	keyUsage	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified. The upper byte MUST be NULL
TPM_KEY_FLAGS	keyFlags	This SHALL be set to the same value as the corresponding parameter in the TPM_KEY structure that describes the public key that is being certified. The upper byte MUST be NULL.
TPM_AUTH_DATA_USAGE	authDataUsage	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified
TPM_KEY_PARMS	algorithmParms	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified
TPM_DIGEST	pubKeyDigest	This SHALL be a digest of the value TPM_KEY -> pubKey -> key in a TPM_KEY representation of the key to be certified
TPM_NONCE	data	This SHALL be externally provided data.
BOOL	parentPCRStatus	This SHALL indicate if any parent key was wrapped to a PCR
UINT32	PCRInfoSize	This SHALL be the size of the PCRInfo parameter.

Type	Name	Description
BYTE*	PCRInfo	This SHALL be the TPM_PCR_INFO_SHORT structure.
UINT32	migrationAuthoritySize	This SHALL be the size of migrationAuthority
BYTE[]	migrationAuthority	If the key to be certified has [payload == TPM_PT_MIGRATE_RESTRICTED or payload == TPM_PT_MIGRATE_EXTERNAL], migrationAuthority is the digest of the TPM_MSA_COMPOSITE and has TYPE == TPM_DIGEST. Otherwise it is NULL.

1342 **11.3 TPM_QUOTE_INFO Structure**1343 **Start of informative comment**

1344 This structure provides the mechanism for the TPM to quote the current values of a list of
1345 PCRs.

1346 **End of informative comment**1347 **Definition**

```
1348 typedef struct tdTPM_QUOTE_INFO{
1349     TPM_STRUCTURE_VER version;
1350     BYTE fixed[4];
1351     TPM_COMPOSITE_HASH digestValue;
1352     TPM_NONCE externalData;
1353 } TPM_QUOTE_INFO;
```

1354 **Parameters**

Type	Name	Description
TPM_STRUCTURE_VER	version	This MUST be 1.1.0.0
BYTE	fixed	This SHALL always be the string 'QUOT'
TPM_COMPOSITE_HASH	digestValue	This SHALL be the result of the composite hash algorithm using the current values of the requested PCR indices.
TPM_NONCE	externalData	160 bits of externally supplied data

1355 **11.4 TPM_QUOTE_INFO2 Structure**

1356 **Start of informative comment**

1357 This structure provides the mechanism for the TPM to quote the current values of a list of
1358 PCRs.

1359 **End of informative comment**

1360 **Definition**

```
1361 typedef struct tdTPM_QUOTE_INFO2{  
1362     TPM_STRUCTURE_TAG tag;  
1363     BYTE fixed[4];  
1364     TPM_NONCE externalData;  
1365     TPM_PCR_INFO_SHORT infoShort;  
1366 } TPM_QUOTE_INFO2;
```

1367 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL be TPM_TAG_QUOTE_INFO2
BYTE	fixed	This SHALL always be the string 'OUT2'
TPM_NONCE	externalData	160 bits of externally supplied data
TPM_PCR_INFO_SHORT	infoShort	the quoted PCR registers

1368 **12. Identity Structures**1369 **12.1 TPM_EK_BLOB**1370 **Start of informative comment**

1371 This structure provides a wrapper to each type of structure that will be in use when the
1372 endorsement key is in use.

1373 **End of informative comment**1374 **Definition**

```
1375 typedef struct tdTPM_EK_BLOB{
1376     TPM_STRUCTURE_TAG  tag;
1377     TPM_EK_TYPE  ekType;
1378     UINT32  blobSize;
1379     [size_is(blobSize)] byte*  blob;
1380 } TPM_EK_BLOB;
```

1381 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_EK_BLOB
TPM_EK_TYPE	ekType	This SHALL be set to reflect the type of blob in use
UINT32	blobSize	The size of the blob field
BYTE*	blob	The blob of information depending on the type

1382 **12.2 TPM_EK_BLOB_ACTIVATE**

1383 **Start of informative comment**

1384 This structure contains the symmetric key to encrypt the identity credential.

1385 This structure always is contained in a TPM_EK_BLOB.

1386 **End of informative comment**

1387 **Definition**

```
1388 typedef struct tdTPM_EK_BLOB_ACTIVATE{
1389     TPM_STRUCTURE_TAG  tag;
1390     TPM_SYMMETRIC_KEY sessionKey;
1391     TPM_DIGEST idDigest;
1392     TPM_PCR_INFO_SHORT pcrInfo;
1393 } TPM_EK_BLOB_ACTIVATE;
```

1394 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_EK_BLOB_ACTIVATE
TPM_SYMMETRIC_KEY	sessionKey	This SHALL be the session key used by the CA to encrypt the TPM_IDENTITY_CREDENTIAL
TPM_DIGEST	idDigest	This SHALL be the digest of the TPM_PUBKEY that is being certified by the CA
TPM_PCR_INFO_SHORT	pcrInfo	This SHALL indicate the PCR's and localities

1395 **12.3 TPM_EK_BLOB_AUTH**1396 **Start of informative comment**

1397 This structure contains the symmetric key to encrypt the identity credential.

1398 This structure always is contained in a TPM_EK_BLOB.

1399 **End of informative comment**1400 **Definition**

```

1401 typedef struct tdTPM_EK_BLOB_AUTH{
1402     TPM_STRUCTURE_TAG  tag;
1403     TPM_SECRET  authValue;
1404 } TPM_EK_BLOB_AUTH;

```

1405 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_EK_BLOB_AUTH
TPM_SECRET	authValue	This SHALL be the AuthData value

1406

1407 **12.4 TPM_CHOSENID_HASH**

1408 This definition specifies the operation necessary to create a TPM_CHOSENID_HASH
1409 structure.

1410 **Parameters**

Type	Name	Description
BYTE []	identityLabel	The label chosen for a new TPM identity
TPM_PUBKEY	privacyCA	The public key of a TTP chosen to attest to a new TPM identity

1411 **Action**

1412 1. $TPM_CHOSENID_HASH = SHA(identityLabel || privacyCA)$

1413 **12.5 TPM_IDENTITY_CONTENTS**1414 **Start of informative comment**

1415 TPM_MakeIdentity uses this structure and the signature of this structure goes to a privacy
 1416 CA during the certification process. There is no reason to update the version as this
 1417 structure did not change for version 1.2.

1418 **End of informative comment**1419 **Definition**

```
1420 typedef struct tdTPM_IDENTITY_CONTENTS {
1421     TPM_STRUCT_VER         ver;
1422     UINT32                 ordinal;
1423     TPM_CHOSENID_HASH     labelPrivCADigest;
1424     TPM_PUBKEY            identityPubKey;
1425 } TPM_IDENTITY_CONTENTS;
```

1426 **Parameters**

Type	Name	Description
TPM_STRUCT_VER	ver	This MUST be 1.1.0.0. This is the version information for this structure and not the underlying key.
UINT32	ordinal	This SHALL be the ordinal of the TPM_MakeIdentity command.
TPM_CHOSENID_HASH	labelPrivCADigest	This SHALL be the result of hashing the chosen identityLabel and privacyCA for the new TPM identity
TPM_PUBKEY	identityPubKey	This SHALL be the public key structure of the identity key

1427 **12.6 TPM_IDENTITY_REQ**

1428 **Start of informative comment**

1429 This structure is sent by the TSS to the Privacy CA to create the identity credential.

1430 This structure is informative only.

1431 **End of informative comment**

1432 **Parameters**

Type	Name	Description
UINT32	asymSize	This SHALL be the size of the asymmetric encrypted area created by TSS_CollatIdentityRequest
UINT32	symSize	This SHALL be the size of the symmetric encrypted area created by TSS_CollatIdentityRequest
TPM_KEY_PARMS	asymAlgorithm	This SHALL be the parameters for the asymmetric algorithm used to create the asymBlob
TPM_KEY_PARMS	symAlgorithm	This SHALL be the parameters for the symmetric algorithm used to create the symBlob
BYTE*	asymBlob	This SHALL be the asymmetric encrypted area from TSS_CollatIdentityRequest
BYTE*	symBlob	This SHALL be the symmetric encrypted area from TSS_CollatIdentityRequest

1433 **12.7 TPM_IDENTITY_PROOF**1434 **Start of informative comment**

1435 Structure in use during the AIK credential process.

1436 **End of informative comment**

Type	Name	Description
TPM_STRUCT_VER	ver	This MUST be 1.1.0.0
UINT32	labelSize	This SHALL be the size of the label area
UINT32	identityBindingSize	This SHALL be the size of the identitybinding area
UINT32	endorsementSize	This SHALL be the size of the endorsement credential
UINT32	platformSize	This SHALL be the size of the platform credential
UINT32	conformanceSize	This SHALL be the size of the conformance credential
TPM_PUBKEY	identityKey	This SHALL be the public key of the new identity
BYTE*	labelArea	This SHALL be the text label for the new identity
BYTE*	identityBinding	This SHALL be the signature value of TPM_IDENTITY_CONTENTS structure from the TPM_MakeIdentity command
BYTE*	endorsementCredential	This SHALL be the TPM endorsement credential
BYTE*	platformCredential	This SHALL be the TPM platform credential
BYTE*	conformanceCredential	This SHALL be the TPM conformance credential

1437 12.8 TPM_ASYM_CA_CONTENTS

1438 **Start of informative comment**

1439 This structure contains the symmetric key to encrypt the identity credential.

1440 **End of informative comment**

1441 **Definition**

```
1442 typedef struct tdTPM_ASYM_CA_CONTENTS{  
1443     TPM_SYMMETRIC_KEY sessionKey;  
1444     TPM_DIGEST idDigest;  
1445 } TPM_ASYM_CA_CONTENTS;
```

1446 **Parameters**

Type	Name	Description
TPM_SYMMETRIC_KEY	sessionKey	This SHALL be the session key used by the CA to encrypt the TPM_IDENTITY_CREDENTIAL
TPM_DIGEST	idDigest	This SHALL be the digest of the TPM_PUBKEY of the key that is being certified by the CA

1447 **12.9 TPM_SYM_CA_ATTESTATION**1448 **Start of informative comment**

1449 This structure returned by the Privacy CA with the encrypted identity credential.

1450 **End of informative comment**

Type	Name	Description
UINT32	credSize	This SHALL be the size of the credential parameter
TPM_KEY_PARMS	algorithm	This SHALL be the indicator and parameters for the symmetric algorithm
BYTE*	credential	This is the result of encrypting TPM_IDENTITY_CREDENTIAL using the session_key and the algorithm indicated "algorithm"

1451 13. Transport structures

1452 13.1 TPM_TRANSPORT_PUBLIC

1453 Start of informative comment

1454 The public information relative to a transport session

1455 End of informative comment

1456 Definition

```
1457 typedef struct tdTPM_TRANSPORT_PUBLIC{
1458     TPM_STRUCTURE_TAG tag;
1459     TPM_TRANSPORT_ATTRIBUTES transAttributes;
1460     TPM_ALGORITHM_ID algID;
1461     TPM_ENC_SCHEME encScheme;
1462 } TPM_TRANSPORT_PUBLIC;
```

1463 Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_PUBLIC
TPM_TRANSPORT_ATTRIBUTES	transAttributes	The attributes of this session
TPM_ALGORITHM_ID	algId	This SHALL be the algorithm identifier of the symmetric key.
TPM_ENC_SCHEME	encScheme	This SHALL fully identify the manner in which the key will be used for encryption operations.

1464 13.1.1 TPM_TRANSPORT_ATTRIBUTES Definitions

Name	Value	Description
TPM_TRANSPORT_ENCRYPT	0x00000001	The session will provide encryption using the internal encryption algorithm
TPM_TRANSPORT_LOG	0x00000002	The session will provide a log of all operations that occur in the session
TPM_TRANSPORT_EXCLUSIVE	0x00000004	The transport session is exclusive and any command executed outside the transport session causes the invalidation of the session

1465 **13.2 TPM_TRANSPORT_INTERNAL**1466 **Start of informative comment**

1467 The internal information regarding transport session

1468 **End of informative comment**1469 **Definition**

```

1470 typedef struct tdTPM_TRANSPORT_INTERNAL{
1471     TPM_STRUCTURE_TAG  tag;
1472     TPM_AUTHDATA authData;
1473     TPM_TRANSPORT_PUBLIC tranPublic;
1474     TPM_TRANSHANDLE transHandle;
1475     TPM_NONCE transEven;
1476     TPM_DIGEST transDigest;
1477 } TPM_TRANSPORT_INTERNAL;

```

1478 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_INTERNAL
TPM_AUTHDATA	authData	The shared secret for this session
TPM_TRANSPORT_PUBLIC	tranPublic	The public information of this session
TPM_TRANSHANDLE	transHandle	The handle for this session
TPM_NONCE	transEven	The even nonce for the rolling protocol
TPM_DIGEST	transDigest	The log of transport events

1479 **13.3 TPM_TRANSPORT_LOG_IN structure**

1480 **Start of informative comment**

1481 The logging of transport commands occurs in two steps, before execution with the input
1482 parameters and after execution with the output parameters.

1483 This structure is in use for input log calculations.

1484 **End of informative comment**

1485 **Definition**

```
1486 typedef struct tdTPM_TRANSPORT_LOG_IN{
1487     TPM_STRUCTURE_TAG tag;
1488     TPM_DIGEST parameters;
1489     TPM_DIGEST pubKeyHash;
1490 } TPM_TRANSPORT_LOG_IN;
```

1491 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_LOG_IN
TPM_DIGEST	parameters	The actual parameters contained in the digest are subject to the rules of the command using this structure. To find the exact calculation refer to the actions in the command using this structure.
TPM_DIGEST	pubKeyHash	The hash of any keys in the transport command

1492 **13.4 TPM_TRANSPORT_LOG_OUT structure**1493 **Start of informative comment**

1494 The logging of transport commands occurs in two steps, before execution with the input
1495 parameters and after execution with the output parameters.

1496 This structure is in use for output log calculations.

1497 This structure is in use for the INPUT logging during releaseTransport.

1498 **End of informative comment**1499 **Definition**

```
1500 typedef struct tdTPM_TRANSPORT_LOG_OUT{
1501     TPM_STRUCTURE_TAG tag;
1502     TPM_CURRENT_TICKS currentTicks;
1503     TPM_DIGEST parameters;
1504     TPM_MODIFIER_INDICATOR locality;
1505 } TPM_TRANSPORT_LOG_OUT
```

1506 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_LOG_OUT
TPM_CURRENT_TICKS	currentTicks	The current tick count. This SHALL be the value of the current TPM tick counter. The value is set to 0 on input to ExecuteTransport to avoid timing attacks.
TPM_DIGEST	parameters	The actual parameters contained in the digest are subject to the rules of the command using this structure. To find the exact calculation refer to the actions in the command using this structure.
TPM_MODIFIER_INDICATOR	locality	The locality that called TPM_ExecuteTransport

1507 13.5 TPM_TRANSPORT_AUTH structure

1508 Start of informative comment

1509 This structure provides the validation for the encrypted AuthData value.

1510 End of informative comment

1511 Definition

```
1512 typedef struct tdTPM_TRANSPORT_AUTH {  
1513     TPM_STRUCTURE_TAG tag;  
1514     TPM_AUTHDATA authData;  
1515 } TPM_TRANSPORT_AUTH;
```

1516 Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_AUTH
TPM_AUTHDATA	authData	The AuthData value

1517 **14. Audit Structures**1518 **14.1 TPM_AUDIT_EVENT_IN structure**1519 **Start of informative comment**

1520 This structure provides the auditing of the command upon receipt of the command. It
1521 provides the information regarding the input parameters.

1522 **End of informative comment**1523 **Definition**

```
1524 typedef struct tdTPM_AUDIT_EVENT_IN {
1525     TPM_STRUCTURE_TAG tag;
1526     TPM_DIGEST inputParms;
1527     TPM_COUNTER_VALUE auditCount;
1528 } TPM_AUDIT_EVENT_IN;
```

1529 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_AUDIT_EVENT_IN
TPM_DIGEST	inputParms	Digest value according to the HMAC digest rules of the "above the line" parameters (i.e. the first HMAC digest calculation). When there are no HMAC rules, the input digest includes all parameters including and after the ordinal.
TPM_COUNTER_VALUE	auditCount	The current value of the audit monotonic counter

1530 **14.2 TPM_AUDIT_EVENT_OUT structure**

1531 **Start of informative comment**

1532 This structure reports the results of the command execution. It includes the return code
1533 and the output parameters.

1534 **End of informative comment**

1535 **Definition**

```
1536 typedef struct tdTPM_AUDIT_EVENT_OUT {
1537     TPM_STRUCTURE_TAG tag;
1538     TPM_DIGEST outputParms;
1539     TPM_COUNTER_VALUE auditCount;
1540 } TPM_AUDIT_EVENT_OUT;
```

1541 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_AUDIT_EVENT_OUT
TPM_DIGEST	outputParms	Digest value according to the HMAC digest rules of the "above the line" parameters (i.e. the first HMAC digest calculation). When there are no HMAC rules, the output digest includes the return code, the ordinal, and all parameters after the return code.
TPM_COUNTER_VALUE	auditCount	The current value of the audit monotonic counter

1542

1543 **15. Tick Structures**1544 **15.1 TPM_CURRENT_TICKS**1545 **Start of informative comment**

1546 This structure holds the current number of time ticks in the TPM. The value is the number
1547 of time ticks from the start of the current session. Session start is a variable function that is
1548 platform dependent. Some platforms may have batteries or other power sources and keep
1549 the TPM clock session across TPM initialization sessions.

1550 The <tickRate> element of the TPM_CURRENT_TICKS structure provides the relationship
1551 between ticks and seconds.

1552 No external entity may ever set the current number of time ticks held in
1553 TPM_CURRENT_TICKS. This value is always reset to 0 when a new clock session starts and
1554 increments under control of the TPM.

1555 Maintaining the relationship between the number of ticks counted by the TPM and some
1556 real world clock is a task for external software.

1557 **End of informative comment**1558 **Definition**

```
1559 typedef struct tdTPM_CURRENT_TICKS {
1560     TPM_STRUCTURE_TAG tag;
1561     UINT64 currentTicks;
1562     UINT16 tickRate;
1563     TPM_NONCE tickNonce;
1564 } TPM_CURRENT_TICKS;
```

1565 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_CURRENT_TICKS
UINT64	currentTicks	The number of ticks since the start of this tick session
UINT16	tickRate	One tick represents x microseconds. The maximum resolution of the TPM tick counter would then be 1 microsecond. The minimum resolution SHOULD be 1 millisecond.
TPM_NONCE	tickNonce	The nonce created by the TPM when resetting the currentTicks to 0. This indicates the beginning of a time session. This value MUST be valid before the first use of TPM_CURRENT_TICKS. The value can be set at TPM_Startup or just prior to first use.

1566 16. Return codes

1567 **Start of informative comment**

1568 The TPM has five types of return code. One indicates successful operation and four indicate
1569 failure. TPM_SUCCESS (00000000) indicates successful execution. The failure reports are:
1570 TPM defined fatal errors (00000001 to 000003FF), vendor defined fatal errors (00000400 to
1571 000007FF), TPM defined non-fatal errors (00000800 to 00000BFF), vendor defined non-fatal
1572 errors (00000C00 to 00000FFF).

1573 The range of vendor defined non-fatal errors was determined by the TSS-WG, which defined
1574 XXXX YCCC with XXXX as OS specific and Y defining the TSS SW stack layer (0: TPM layer)

1575 All failure cases return a non-authenticated fixed set of information, only. This is due to the
1576 fact that the failure may have been due to authentication or other factors and there is no
1577 possibility of producing an authenticated response.

1578 Fatal errors also terminate any authorization sessions. This is a result of returning only the
1579 error code as there is no way to return and continue the nonce's necessary to maintain an
1580 authorization session. Non-fatal errors do not terminate authorization sessions.

1581 **End of informative comment**

1582 **Description**

1583 1. When a command fails for ANY reason, the TPM MUST return only the following three
1584 items:

- 1585 a. TPM_TAG_RQU_COMMAND (2 bytes)
- 1586 b. ParamLength(4 bytes, fixed at 10)
- 1587 c. Return Code (4 bytes, never TPM_SUCCESS)

1588 2. When a capability has failed to complete successfully, the TPM MUST return a legal
1589 error code. Otherwise the TPM SHOULD return TPM_SUCCESS. If a TPM returns an
1590 error code after executing a capability, it SHOULD be the error code specified by the
1591 capability or another legal error code that is appropriate to the error condition

1592 3. A fatal failure SHALL cause termination of the associated authorization or transport
1593 session. A non-fatal failure SHALL NOT cause termination of the associated
1594 authorization or transport session.

1595 4. A fatal failure of a wrapped command SHALL not cause any disruption of a transport
1596 session that wrapped the failing command. The exception to this is when the failure
1597 causes the TPM itself to go into failure mode (selftest failure etc.)

1598 The return code MUST use the following base. The return code MAY be TCG defined or
1599 vendor defined.

1600

Mask Parameters

Name	Value	Description
TPM_BASE	0x0	The start of TPM return codes
TPM_SUCCESS	TPM_BASE	Successful completion of the operation
TPM_VENDOR_ERROR	TPM_Vendor_Specific32	Mask to indicate that the error code is vendor specific for vendor specific commands.
TPM_NON_FATAL	0x00000800	Mask to indicate that the error code is a non-fatal failure.

1601

TPM-defined fatal error codes

Name	Value	Description
TPM_AUTHFAIL	TPM_BASE + 1	Authentication failed
TPM_BADINDEX	TPM_BASE + 2	The index to a PCR, DIR or other register is incorrect
TPM_BAD_PARAMETER	TPM_BASE + 3	One or more parameter is bad
TPM_AUDITFAILURE	TPM_BASE + 4	An operation completed successfully but the auditing of that operation failed.
TPM_CLEAR_DISABLED	TPM_BASE + 5	The clear disable flag is set and all clear operations now require physical access
TPM_DEACTIVATED	TPM_BASE + 6	The TPM is deactivated
TPM_DISABLED	TPM_BASE + 7	The TPM is disabled
TPM_DISABLED_CMD	TPM_BASE + 8	The target command has been disabled
TPM_FAIL	TPM_BASE + 9	The operation failed
TPM_BAD_ORDINAL	TPM_BASE + 10	The ordinal was unknown or inconsistent
TPM_INSTALL_DISABLED	TPM_BASE + 11	The ability to install an owner is disabled
TPM_INVALID_KEYHANDLE	TPM_BASE + 12	The key handle can not be interpreted
TPM_KEYNOTFOUND	TPM_BASE + 13	The key handle points to an invalid key
TPM_INAPPROPRIATE_ENC	TPM_BASE + 14	Unacceptable encryption scheme
TPM_MIGRATEFAIL	TPM_BASE + 15	Migration authorization failed
TPM_INVALID_PCR_INFO	TPM_BASE + 16	PCR information could not be interpreted
TPM_NOSPACE	TPM_BASE + 17	No room to load key.
TPM_NOSRK	TPM_BASE + 18	There is no SRK set
TPM_NOTSEALED_BLOB	TPM_BASE + 19	An encrypted blob is invalid or was not created by this TPM
TPM_OWNER_SET	TPM_BASE + 20	There is already an Owner
TPM_RESOURCES	TPM_BASE + 21	The TPM has insufficient internal resources to perform the requested action.
TPM_SHORTRANDOM	TPM_BASE + 22	A random string was too short
TPM_SIZE	TPM_BASE + 23	The TPM does not have the space to perform the operation.
TPM_WRONGPCRVAL	TPM_BASE + 24	The named PCR value does not match the current PCR value.
TPM_BAD_PARAM_SIZE	TPM_BASE + 25	The paramSize argument to the command has the incorrect value
TPM_SHA_THREAD	TPM_BASE + 26	There is no existing SHA-1 thread.
TPM_SHA_ERROR	TPM_BASE + 27	The calculation is unable to proceed because the existing SHA-1 thread has already encountered an error.
TPM_FAILEDSELFTEST	TPM_BASE + 28	Self-test has failed and the TPM has shutdown.
TPM_AUTH2FAIL	TPM_BASE + 29	The authorization for the second key in a 2 key function failed authorization
TPM_BADTAG	TPM_BASE + 30	The tag value sent to for a command is invalid
TPM_IOERROR	TPM_BASE + 31	An IO error occurred transmitting information to the TPM
TPM_ENCRYPT_ERROR	TPM_BASE + 32	The encryption process had a problem.
TPM_DECRYPT_ERROR	TPM_BASE + 33	The decryption process did not complete.
TPM_INVALID_AUTHHANDLE	TPM_BASE + 34	An invalid handle was used.
TPM_NO_ENDORSEMENT	TPM_BASE + 35	The TPM does not a EK installed
TPM_INVALID_KEYUSAGE	TPM_BASE + 36	The usage of a key is not allowed

Name	Value	Description
TPM_WRONG_ENTITYTYPE	TPM_BASE + 37	The submitted entity type is not allowed
TPM_INVALID_POSTINIT	TPM_BASE + 38	The command was received in the wrong sequence relative to TPM_Init and a subsequent TPM_Startup
TPM_INAPPROPRIATE_SIG	TPM_BASE + 39	Signed data cannot include additional DER information
TPM_BAD_KEY_PROPERTY	TPM_BASE + 40	The key properties in TPM_KEY_PARMS are not supported by this TPM
TPM_BAD_MIGRATION	TPM_BASE + 41	The migration properties of this key are incorrect.
TPM_BAD_SCHEME	TPM_BASE + 42	The signature or encryption scheme for this key is incorrect or not permitted in this situation.
TPM_BAD_DATASIZE	TPM_BASE + 43	The size of the data (or blob) parameter is bad or inconsistent with the referenced key
TPM_BAD_MODE	TPM_BASE + 44	A mode parameter is bad, such as capArea or subCapArea for TPM_GetCapability, physicalPresence parameter for TPM_PhysicalPresence, or migrationType for TPM_CreateMigrationBlob.
TPM_BAD_PRESENCE	TPM_BASE + 45	Either the physicalPresence or physicalPresenceLock bits have the wrong value
TPM_BAD_VERSION	TPM_BASE + 46	The TPM cannot perform this version of the capability
TPM_NO_WRAP_TRANSPORT	TPM_BASE + 47	The TPM does not allow for wrapped transport sessions
TPM_AUDITFAIL_UNSUCCESSFUL	TPM_BASE + 48	TPM audit construction failed and the underlying command was returning a failure code also
TPM_AUDITFAIL_SUCCESSFUL	TPM_BASE + 49	TPM audit construction failed and the underlying command was returning success
TPM_NOTRESETABLE	TPM_BASE + 50	Attempt to reset a PCR register that does not have the resettable attribute
TPM_NOTLOCAL	TPM_BASE + 51	Attempt to reset a PCR register that requires locality and locality modifier not part of command transport
TPM_BAD_TYPE	TPM_BASE + 52	Make identity blob not properly typed
TPM_INVALID_RESOURCE	TPM_BASE + 53	When saving context identified resource type does not match actual resource
TPM_NOTFIPS	TPM_BASE + 54	The TPM is attempting to execute a command only available when in FIPS mode
TPM_INVALID_FAMILY	TPM_BASE + 55	The command is attempting to use an invalid family ID
TPM_NO_NV_PERMISSION	TPM_BASE + 56	The permission to manipulate the NV storage is not available
TPM_REQUIRES_SIGN	TPM_BASE + 57	The operation requires a signed command
TPM_KEY_NOTSUPPORTED	TPM_BASE + 58	Wrong operation to load an NV key
TPM_AUTH_CONFLICT	TPM_BASE + 59	NV_LoadKey blob requires both owner and blob authorization
TPM_AREA_LOCKED	TPM_BASE + 60	The NV area is locked and not writable
TPM_BAD_LOCALITY	TPM_BASE + 61	The locality is incorrect for the attempted operation
TPM_READ_ONLY	TPM_BASE + 62	The NV area is read only and can't be written to
TPM_PER_NOWRITE	TPM_BASE + 63	There is no protection on the write to the NV area
TPM_FAMILYCOUNT	TPM_BASE + 64	The family count value does not match
TPM_WRITE_LOCKED	TPM_BASE + 65	The NV area has already been written to
TPM_BAD_ATTRIBUTES	TPM_BASE + 66	The NV area attributes conflict
TPM_INVALID_STRUCTURE	TPM_BASE + 67	The structure tag and version are invalid or inconsistent
TPM_KEY_OWNER_CONTROL	TPM_BASE + 68	The key is under control of the TPM Owner and can only be evicted by the TPM Owner.
TPM_BAD_COUNTER	TPM_BASE + 69	The counter handle is incorrect
TPM_NOT_FULLWRITE	TPM_BASE + 70	The write is not a complete write of the area
TPM_CONTEXT_GAP	TPM_BASE + 71	The gap between saved context counts is too large
TPM_MAXNVWRITES	TPM_BASE + 72	The maximum number of NV writes without an owner has been exceeded
TPM_NOOPERATOR	TPM_BASE + 73	No operator AuthData value is set

Name	Value	Description
TPM_RESOURCEMISSING	TPM_BASE + 74	The resource pointed to by context is not loaded
TPM_DELEGATE_LOCK	TPM_BASE + 75	The delegate administration is locked
TPM_DELEGATE_FAMILY	TPM_BASE + 76	Attempt to manage a family other than the delegated family
TPM_DELEGATE_ADMIN	TPM_BASE + 77	Delegation table management not enabled
TPM_TRANSPORT_NOTEXCLUSIVE	TPM_BASE + 78	There was a command executed outside of an exclusive transport session
TPM_OWNER_CONTROL	TPM_BASE + 79	Attempt to context save a owner evict controlled key
TPM_DAA_RESOURCES	TPM_BASE + 80	The DAA command has no resources available to execute the command
TPM_DAA_INPUT_DATA0	TPM_BASE + 81	The consistency check on DAA parameter inputData0 has failed.
TPM_DAA_INPUT_DATA1	TPM_BASE + 82	The consistency check on DAA parameter inputData1 has failed.
TPM_DAA_ISSUER_SETTINGS	TPM_BASE + 83	The consistency check on DAA_issuerSettings has failed.
TPM_DAA_TPM_SETTINGS	TPM_BASE + 84	The consistency check on DAA_tpmSpecific has failed.
TPM_DAA_STAGE	TPM_BASE + 85	The atomic process indicated by the submitted DAA command is not the expected process.
TPM_DAA_ISSUER_VALIDITY	TPM_BASE + 86	The issuer's validity check has detected an inconsistency
TPM_DAA_WRONG_W	TPM_BASE + 87	The consistency check on w has failed.
TPM_BAD_HANDLE	TPM_BASE + 88	The handle is incorrect
TPM_BAD_DELEGATE	TPM_BASE + 89	Delegation is not correct
TPM_BADCONTEXT	TPM_BASE + 90	The context blob is invalid
TPM_TOOMANYCONTEXTS	TPM_BASE + 91	Too many contexts held by the TPM
TPM_MA_TICKET_SIGNATURE	TPM_BASE + 92	Migration authority signature validation failure
TPM_MA_DESTINATION	TPM_BASE + 93	Migration destination not authenticated
TPM_MA_SOURCE	TPM_BASE + 94	Migration source incorrect
TPM_MA_AUTHORITY	TPM_BASE + 95	Incorrect migration authority
TPM_PERMANENTEK	TPM_BASE + 97	Attempt to revoke the EK and the EK is not revocable
TPM_BAD_SIGNATURE	TPM_BASE + 98	Bad signature of CMK ticket

1602 TPM-defined non-fatal errors

Name	Value	Description
TPM_RETRY	TPM_BASE + TPM_NON_FATAL	The TPM is too busy to respond to the command immediately, but the command could be resubmitted at a later time The TPM MAY return TPM_Retry for any command at any time.
TPM_NEEDS_SELFTEST	TPM_BASE + TPM_NON_FATAL + 1	SelfTestFull has not been run
TPM_DOING_SELFTEST	TPM_BASE + TPM_NON_FATAL + 2	The TPM is currently executing a full selftest
TPM_DEFEND_LOCK_RUNNING	TPM_BASE + TPM_NON_FATAL + 3	The TPM is defending against dictionary attacks and is in some time-out period.

1627 The following Purviews have been defined:

Value	Event Name	Comments
0x00	TPM_MAIN	Command is from the main specification
0x01	TPM_PC	Command is specific to the PC
0x02	TPM_PDA	Command is specific to a PDA
0x03	TPM_CELL_PHONE	Command is specific to a cell phone
0x04	TPM_SERVER	Command is specific to servers

1628

1629 Combinations for the main specification would be

Value	Event Name
TPM_PROTECTED_COMMAND TPM_MAIN	TPM_PROTECTED_ORDINAL
TPM_UNPROTECTED_COMMAND TPM_MAIN	TPM_UNPROTECTED_ORDINAL
TPM_CONNECTION_COMMAND TPM_MAIN	TPM_CONNECTION_ORDINAL

1630

1631 If a command is tagged from the audit column the default state is that use of that command
1632 SHALL be audited. Otherwise, the default state is that use of that command SHALL NOT be
1633 audited.

Column	Column Values	Comments and valid column entries
AUTH2	x	Does the command support two authorization entities, normally two keys
AUTH1	x	Does the commands support an single authorization session
RQU	x	Does the command execute without any authorization
Optional	x	Is the command optional
No Owner	x	Is the command executable when no owner is present
PCR Use Enforced	x	Does the command enforce PCR restrictions when executed
Audit	x, N	Is the default for auditing enabled N = Never the ordinal is never audited
Duration	S, M, L	What is the expected duration of the command, S = Short implies no asymmetric cryptography M = Medium implies an asymmetric operation L = Long implies asymmetric key generation
1.2 Changes	N, D, X, C	N = New for 1.2 X = Deleted in 1.2 D = Deprecated in 1.2 C = Changed in 1.2
FIPS changes	x	Ordinal has change to satisfy FIPS 140 requirements
Avail Deactivated	x, A	Ordinal will execute when deactivated A = Authorization means that command will only work if the underlying NV store does not require authorization
Avail Disabled	x, A	Ordinal will execute when disabled A = Authorization means that command will only work if the underlying NV store does not require authorization The TPM MUST return TPM_DISABLED for all commands other than those marked as available

1634

	TPM_PROTECTED_ORDINAL +	Complete ordinal	AUTH2	AUTH1	RQU	Optional	No Owner	PCR Use enforced	Audit	Duration	1.2 New Functionality	FIPS Changes	Avail Deactivated	Avail Disabled
TPM_ORD_ActivateIdentity	122	0x0000007A	X	X				X	X	M				
TPM_ORD_AuthorizeMigrationKey	43	0x0000002B		X					X	S				
TPM_ORD_CertifyKey	50	0x00000032	X	X	X			X		M				
TPM_ORD_CertifyKey2	51	0x00000033	X	X	X			X		M	N			
TPM_ORD_CertifySelfTest	82	0x00000052		X	X			X		M	X			
TPM_ORD_ChangeAuth	12	0x0000000C	X					X		M				
TPM_ORD_ChangeAuthAsymFinish	15	0x0000000F		X	X			X		M	D			
TPM_ORD_ChangeAuthAsymStart	14	0x0000000E		X	X			X		L	D			
TPM_ORD_ChangeAuthOwner	16	0x00000010		X				X	X	S				
TPM_ORD_CMK_ApproveMA	29	0x0000001D		X						S	N			
TPM_ORD_CMK_ConvertMigration	36	0x00000024		X				X		M	N			
TPM_ORD_CMK_CreateBlob	27	0x0000001B		X						M	N			
TPM_ORD_CMK_CreateKey	19	0x00000013		X				X		L	N	X		
TPM_ORD_CMK_CreateTicket	18	0x00000012		X						M	N			
TPM_ORD_CMK_SetRestrictions	28	0x0000001C		X						S	N			
TPM_ORD_ContinueSelfTest	83	0x00000053			X		X			L		X	X	X
TPM_ORD_ConvertMigrationBlob	42	0x0000002A		X	X			X	X	M				
TPM_ORD_CreateCounter	220	0x000000DC		X						S	N			
TPM_ORD_CreateEndorsementKeyPair	120	0x00000078			X		X			L				
TPM_ORD_CreateMaintenanceArchive	44	0x0000002C		X		X			X	S				
TPM_ORD_CreateMigrationBlob	40	0x00000028	X	X				X	X	M				
TPM_ORD_CreateRevocableEK	127	0x0000007F			X	X	X			L	N			
TPM_ORD_CreateWrapKey	31	0x0000001F		X				X	X	L		X		
TPM_ORD_DAA_JOIN	41	0x00000029		X		X				L	N			
TPM_ORD_DAA_SIGN	49	0x00000031		X		X				L	N			
TPM_ORD_Delegate_CreateKeyDelegation	212	0x000000D4		X						M	N			
TPM_ORD_Delegate_CreateOwnerDelegation	213	0x000000D5		X						M	N			
TPM_ORD_Delegate_LoadOwnerDelegation	216	0x000000D8		X	X		X			M	N			
TPM_ORD_Delegate_Manage	210	0x000000D2		X	X		X			M	N			
TPM_ORD_Delegate_ReadTable	219	0x000000DB			X		X			S	N			
TPM_ORD_Delegate_UpdateVerification	209	0x000000D1		X						S	N			
TPM_ORD_Delegate_VerifyDelegation	214	0x000000D6			X					M	N			

	TPM_PROTECTED_ORDINAL	Complete ordinal	AUTH2	AUTH1	RQU	Optional	No Owner	PCR Use enforced	Audit	Duration	1.2 New Functionality	FIPS Changes	Avail Deactivated	Avail Disabled
TPM_ORD_DirRead	26	0x0000001A			X					S	D			
TPM_ORD_DirWriteAuth	25	0x00000019		X						S	D			
TPM_ORD_DisableForceClear	94	0x0000005E		X			X	X	S					
TPM_ORD_DisableOwnerClear	92	0x0000005C		X				X	S					
TPM_ORD_DisablePubekRead	126	0x0000007E		X				X	S					
TPM_ORD_DSAP	17	0x00000011			X				S	N		X	X	
TPM_ORD_EstablishTransport	230	0x000000E6		X				X	S	N				
TPM_ORD_EvictKey	34	0x00000022			X				S	D				
TPM_ORD_ExecuteTransport	231	0x000000E7		X					? L	N				
TPM_ORD_Extend	20	0x00000014			X		X		S					X
TPM_ORD_FieldUpgrade	170	0x000000AA	X	X	X	X	X		?					
TPM_ORD_FlushSpecific	186	0x000000BA			X		X		S	N		X	X	
TPM_ORD_ForceClear	93	0x0000005D			X		X	X	S					
TPM_ORD_GetAuditDigest	133	0x00000085			X	X	X	N	S	N				
TPM_ORD_GetAuditDigestSigned	134	0x00000086		X	X	X		N	M	N				
TPM_ORD_GetAuditEvent	130	0x00000082			X	X		N	S	X				
TPM_ORD_GetAuditEventSigned	131	0x00000083		X	X	X		N	M	X				
TPM_ORD_GetCapability	101	0x00000065			X		X		S	C		X	X	
TPM_ORD_GetCapabilityOwner	102	0x00000066		X					S	X				
TPM_ORD_GetCapabilitySigned	100	0x00000064		X	X			X	M	X				
TPM_ORD_GetOrdinalAuditStatus	140	0x0000008C			X	X		N	S	X				
TPM_ORD_GetPubKey	33	0x00000021		X	X			X	S					
TPM_ORD_GetRandom	70	0x00000046			X		X		S					
TPM_ORD_GetTestResult	84	0x00000054			X		X		S			X	X	
TPM_ORD_GetTicks	241	0x000000F1			X		X		S	N				
TPM_ORD_IncrementCounter	221	0x000000DD		X					S	N				
TPM_ORD_Init	151	0x00000097			X				M			X	X	
TPM_ORD_KeyControlOwner	35	0x00000023		X					S	N				
TPM_ORD_KillMaintenanceFeature	46	0x0000002E		X		X		X	S					
TPM_ORD_LoadAuthContext	183	0x000000B7			X	X	X		M	D				
TPM_ORD_LoadContext	185	0x000000B9			X		X		M	N				
TPM_ORD_LoadKey	32	0x00000020		X	X			X	M	D	X			
TPM_ORD_LoadKey2	65	0x00000041		X	X			X	M	C	X			
TPM_ORD_LoadKeyContext	181	0x000000B5			X	X	X		S	D				

	TPM_PROTECTED_ORDINAL	Complete ordinal	AUTH2	AUTH1	RQU	Optional	No Owner	PCR Use enforced	Audit	Duration	1.2 New Functionality	FIPS Changes	Avail Deactivated	Avail Disabled
TPM_ORD_LoadMaintenanceArchive	45	0x0000002D		X		X			X	S				
TPM_ORD_LoadManuMaintPub	47	0x0000002F			X	X			X	S				
TPM_ORD_MakeIdentity	121	0x00000079	X	X				X	X	L		X		
TPM_ORD_MigrateKey	37	0x00000025		X	X			X		M	C			
TPM_ORD_NV_DefineSpace	204	0x000000CC		X	X		X			S	N		A	A
TPM_ORD_NV_ReadValue	207	0x000000CF		X	X		X	X		S	N		A	A
TPM_ORD_NV_ReadValueAuth	208	0x000000D0		X				X		S	N			
TPM_ORD_NV_WriteValue	205	0x000000CD		X	X		X	X		S	N		A	A
TPM_ORD_NV_WriteValueAuth	206	0x000000CE		X				X		S	N			
TPM_ORD_OIAP	10	0x0000000A			X		X			S			X	X
TPM_ORD_OSAP	11	0x0000000B			X					S			X	X
TPM_ORD_OwnerClear	91	0x0000005B		X					X	S				
TPM_ORD_OwnerReadInternalPub	129	0x00000081		X						S	C			
TPM_ORD_OwnerReadPubek	125	0x0000007D		X					X	S	D			
TPM_ORD_OwnerSetDisable	110	0x0000006E		X					X	S			X	X
TPM_ORD_PCR_Reset	200	0x000000C8			X		X			S	X		X	X
TPM_ORD_PcrRead	21	0x00000015			X		X			S				
TPM_ORD_PhysicalDisable	112	0x00000070			X		X		X	S			X	
TPM_ORD_PhysicalEnable	111	0x0000006F			X		X		X	S			X	X
TPM_ORD_PhysicalSetDeactivated	114	0x00000072			X		X		X	S			X	
TPM_ORD_Quote	22	0x00000016		X	X			X		M				
TPM_ORD_Quote2	62	0x0000003E		X		X		X		M	N			
TPM_ORD_ReadCounter	222	0x000000DE			X		X			S	N			
TPM_ORD_ReadManuMaintPub	48	0x00000030			X	X			X	S				
TPM_ORD_ReadPubek	124	0x0000007C			X		X		X	S				
TPM_ORD_ReleaseCounter	223	0x000000DF		X			X			S	N			
TPM_ORD_ReleaseCounterOwner	224	0x000000E0		X						S	N			
TPM_ORD_ReleaseTransportSigned	232	0x000000E8	X	X				X		M	N			
TPM_ORD_Reset	90	0x0000005A			X		X			S	C		X	X
TPM_ORD_ResetLockValue	64	0x00000040		X						S	N			
TPM_ORD_RevokeTrust	128	0x00000080			X	X	X			S	N			
TPM_ORD_SaveAuthContext	182	0x000000B6			X	X	X			M	D			
TPM_ORD_SaveContext	184	0x000000B8			X		X			M	N			
TPM_ORD_SaveKeyContext	180	0x000000B4			X	X	X			M	D			
TPM_ORD_SaveState	152	0x00000098			X		X			M			X	X

	TPM_PROTECTED_ORDINAL	Complete ordinal	AUTH2	AUTH1	RQU	Optional	No Owner	PCR Use enforced	Audit	Duration	1.2 New Functionality	FIPS Changes	Avail Deactivated	Avail Disabled
TPM_ORD_Seal	23	0x00000017		X						M				
TPM_ORD_Sealx	61	0x0000003D		X	X			X		M	N			
TPM_ORD_SelfTestFull	80	0x00000050			X		X			L			X	X
TPM_ORD_SetCapability	63	0x0000003F		X	X			X		S	N		X	X
TPM_ORD_SetOperatorAuth	116	0x00000074			X		X			S	N			
TPM_ORD_SetOrdinalAuditStatus	141	0x0000008D		X		X			X	S				
TPM_ORD_SetOwnerInstall	113	0x00000071			X		X	X	S					
TPM_ORD_SetOwnerPointer	117	0x00000075			X					S	N			
TPM_ORD_SetRedirection	154	0x0000009A			X	X			X	S				
TPM_ORD_SetTempDeactivated	115	0x00000073		X	X		X	X	S					
TPM_ORD_SHA1Complete	162	0x000000A2			X		X			S			X	X
TPM_ORD_SHA1CompleteExtend	163	0x000000A3			X		X			S				X
TPM_ORD_SHA1Start	160	0x000000A0			X		X			S			X	X
TPM_ORD_SHA1Update	161	0x000000A1			X		X			S			X	X
TPM_ORD_Sign	60	0x0000003C		X	X			X		M				
TPM_ORD_Startup	153	0x00000099			X		X			S			X	X
TPM_ORD_StirRandom	71	0x00000047			X		X			S				
TPM_ORD_TakeOwnership	13	0x0000000D		X			X	X	L				X	
TPM_ORD_Terminate_Handle	150	0x00000096			X		X			S	D		X	X
TPM_ORD_TickStampBlob	242	0x000000F2		X	X			X		M	N			
TPM_ORD_UnBind	30	0x0000001E		X	X			X		M				
TPM_ORD_Unseal	24	0x00000018	X	X				X		M	C			
UNUSED	38	0x00000026												
UNUSED	39	0x00000027												
UNUSED	66	0x00000042												
UNUSED	67	0x00000043												
UNUSED	68	0x00000044												
UNUSED	69	0x00000045												
UNUSED	72	0x00000048												
UNUSED	73	0x00000049												
UNUSED	74	0x0000004A												
UNUSED	75	0x0000004B												
UNUSED	76	0x0000004C												
UNUSED	77	0x0000004D												
UNUSED	78	0x0000004E												

	TPM_PROTECTED_ORDINAL_+	Complete ordinal	AUTH2	AUTH1	RQU	Optional		No Owner	PCR Use enforced	Audit	Duration	1.2 New Functionality	FIPS Changes	Avail Deactivated	Avail Disabled
UNUSED	79	0x0000004F													
UNUSED	81	0x00000051													
UNUSED	85	0x00000055													
UNUSED	86	0x00000056													
UNUSED	87	0x00000057													
UNUSED	88	0x00000058													
UNUSED	89	0x00000059													
UNUSED	95	0x0000005F													
UNUSED	96	0x00000060													
UNUSED	97	0x00000061													
UNUSED	98	0x00000062													
UNUSED	99	0x00000063													
UNUSED	103	0x00000067													
UNUSED	104	0x00000068													
UNUSED	105	0x00000069													
UNUSED	106	0x0000006A													
UNUSED	107	0x0000006B													
UNUSED	108	0x0000006C													
UNUSED	109	0x0000006D													
UNUSED	118	0x00000076													
UNUSED	119	0x00000077													
UNUSED	132	0x00000084													
UNUSED	135	0x00000087													
UNUSED	136	0x00000088													
UNUSED	137	0x00000089													
UNUSED	138	0x0000008A													
UNUSED	139	0x0000008B													
UNUSED	142	0x0000008E													
UNUSED	143	0x0000008F													
UNUSED	144	0x00000090													
UNUSED	145	0x00000091													
UNUSED	146	0x00000092													
UNUSED	147	0x00000093													
UNUSED	148	0x00000094													
UNUSED	149	0x00000095													

	TPM_PROTECTED_ORDINAL_+	Complete ordinal	AUTH2	AUTH1	RQU	Optional		No Owner	PCR Use enforced	Audit	Duration	1.2 New Functionality	FIPS Changes	Avail Deactivated	Avail Disabled
UNUSED	155	0x0000009B													
UNUSED	156	0x0000009C													
UNUSED	157	0x0000009D													
UNUSED	158	0x0000009E													
UNUSED	159	0x0000009F													
UNUSED	164	0x000000A4													
UNUSED	165	0x000000A5													
UNUSED	166	0x000000A6													
UNUSED	167	0x000000A7													
UNUSED	168	0x000000A8													
UNUSED	169	0x000000A9													
UNUSED	171	0x000000AB													
UNUSED	172	0x000000AC													
UNUSED	173	0x000000AD													
UNUSED	174	0x000000AE													
UNUSED	175	0x000000AF													
UNUSED	176	0x000000B0													
UNUSED	177	0x000000B1													
UNUSED	178	0x000000B2													
UNUSED	179	0x000000B3													
UNUSED	187	0x000000BB													
UNUSED	188	0x000000BC													
UNUSED	189	0x000000BD													
UNUSED	190	0x000000BE													
UNUSED	191	0x000000BF													
UNUSED	192	0x000000C0													
UNUSED	193	0x000000C1													
UNUSED	194	0x000000C2													
UNUSED	195	0x000000C3													
UNUSED	196	0x000000C4													
UNUSED	197	0x000000C5													
UNUSED	198	0x000000C6													
UNUSED	199	0x000000C7													
UNUSED	202	0x000000CA													
UNUSED	203	0x000000CB													

	TPM_PROTECTED_ORDINAL_+	Complete ordinal	AUTH2	AUTH1	RQU	Optional		No Owner	PCR Use enforced	Audit	Duration	1.2 New Functionality	FIPS Changes	Avail Deactivated	Avail Disabled
UNUSED	211	0x000000D3													
UNUSED	215	0x000000D7													
Unused	217	0x000000D9			x					S					
UNUSED	218	0x000000DA													
UNUSED	225	0x000000E1													
UNUSED	233	0x000000E9													
UNUSED	234	0x000000EA													
UNUSED	235	0x000000EB													
UNUSED	236	0x000000EC													
UNUSED	237	0x000000ED													
UNUSED	238	0x000000EE													
UNUSED	239	0x000000EF													
UNUSED	240	0x000000F0													
UNUSED	201	0x000000C9													

1635 **17.1 TSC Ordinals**

1636 The connection commands manage the TPM's connection to the TBB.

	TPM_PROTECTED_ORDINAL +	Complete ordinal	AUTH2	AUTH1	RQU	Optional		No Owner	PCR Use enforced	Audit	Duration	1.2 New Functionality	FIPS Changes	Avail Deactivated	Avail Disabled
TSC_ORD_PhysicalPresence	10	0x4000000A			X	X		X			S				
TSC_ORD_ResetEstablishmentBit	11	0x4000000B			X	X		X			S				

1637 **18. Context structures**1638 **18.1 TPM_CONTEXT_BLOB**1639 **Start of informative comment**

1640 This is the header for the wrapped context. The blob contains all information necessary to
1641 reload the context back into the TPM.

1642 The additional data is in use by the TPM manufacturer to save information that will assist
1643 in the reloading of the context. This area must not contain any shielded data. For instance,
1644 the field could contain some size information that allows the TPM more efficient loads of the
1645 context. The additional area could not contain one of the primes for a RSA key.

1646 To ensure integrity of the blob when using symmetric encryption the TPM vendor could use
1647 some valid cipher chaining mechanism. To ensure the integrity without depending on
1648 correct implementation the TPM_CONTEXT_BLOB structure uses a HMAC of the entire
1649 structure using tpmProof as the secret value.

1650 **End of informative comment**1651 **Definition**

```
1652 typedef struct tdTPM_CONTEXT_BLOB {
1653     TPM_STRUCTURE_TAG tag;
1654     TPM_RESOURCE_TYPE resourceType;
1655     TPM_HANDLE handle;
1656     BYTE[16] label;
1657     UINT32 contextCount;
1658     TPM_DIGEST integrityDigest;
1659     UINT32 additionalSize;
1660     [size_is(additionalSize)] BYTE* additionalData;
1661     UINT32 sensitiveSize;
1662     [size_is(sensitiveSize)] BYTE* sensitiveData;
1663 }TPM_CONTEXT_BLOB;
```

1664 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_CONTEXTBLOB
TPM_RESOURCE_TYPE	resourceType	The resource type
TPM_HANDLE	handle	Previous handle of the resource
BYTE[16]	label	Label for identification of the blob. Free format area.
UINT32	contextCount	MUST be TPM_STANY_DATA -> contextCount when creating the structure. This value is ignored for context blobs that reference a key.
TPM_DIGEST	integrityDigest	The integrity of the entire blob including the sensitive area. This is a HMAC calculation with the entire structure (including sensitiveData) being the hash and tpmProof is the secret
UINT32	additionalSize	The size of additionalData
BYTE	additionalData	Additional information set by the TPM that helps define and reload the context. The information held in this area MUST NOT expose any

Type	Name	Description
		information held in shielded locations. This should include any IV for symmetric encryption
UINT32	sensitiveSize	The size of sensitiveData
BYTE	sensitiveData	The normal information for the resource that can be exported

1665 **18.2 TPM_CONTEXT_SENSITIVE**1666 **Start of informative comment**

1667 The internal areas that the TPM needs to encrypt and store off the TPM.

1668 This is an informative structure and the TPM can implement in any manner they wish.

1669 **End of informative comment**1670 **Definition**

```

1671 typedef struct tdTPM_CONTEXT_SENSITIVE {
1672     TPM_STRUCTURE_TAG tag;
1673     TPM_NONCE contextNonce;
1674     UINT32 internalSize;
1675     [size_is(internalSize)] BYTE* internalData;
1676 }TPM_CONTEXT_SENSITIVE;

```

1677 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_CONTEXT_SENSITIVE
TPM_NONCE	contextNonce	On context blobs other than keys this MUST be TPM_STANY_DATA -> contextNonceSession For keys the value is TPM_STCLEAR_DATA -> contextNonceKey
UINT32	internalSize	The size of the internalData area
BYTE	internalData	The internal data area

1678 **19. NV storage structures**

1679 **19.1 TPM_NV_INDEX**

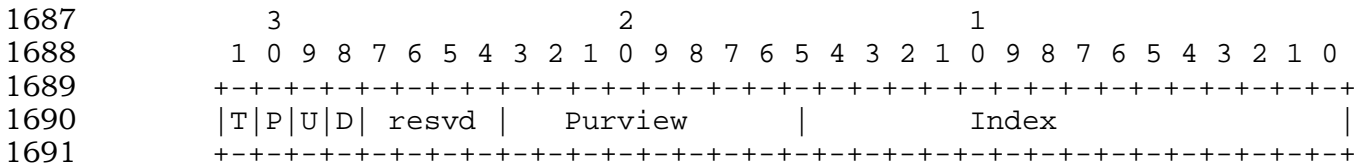
1680 **Start of informative comment**

1681 The index provides the handle to identify the area of storage. The reserved bits allow for a
1682 segregation of the index name space to avoid name collisions.

1683 The TCG defines the space where the high order bits (T, P, U) are 0. The other spaces are
1684 controlled by the indicated entity.

1685 **End of informative comment**

1686 The TPM_NV_INDEX is a 32-bit value.



1692 **Where:**

- 1693 1. All reserved area bits are set to 0
 - 1694 a. T is the TPM manufacturer reserved bit. 0 indicates TCG defined value 1 indicates a
1695 TPM manufacturer specific value
 - 1696 b. P is the platform manufacturer reserved bit. 1 indicates that the index controlled by
1697 the platform manufacturer.
 - 1698 c. U is for the platform user. 1 indicates that the index controlled by the platform user.
 - 1699 d. D indicates defined. 1 indicates that the index is permanently defined and that any
1700 defineSpace operation will fail.
 - 1701 e. TCG reserved areas have T/P/U set to 0
 - 1702 f. TCG reserved areas MAY have D set to 0 or 1
- 1703 2. Purview is the same value used to indicate the platform specific area. This value is the
1704 same purview as in use for command ordinals.
 - 1705 a. The TPM MUST reject index values that do not match the purview of the TPM. This
1706 means that a index value for a PDA is rejected by a TPM designed to work on the PC.

1707 **19.1.1 Required TPM_NV_INDEX values**1708 **Start of informative comment**

1709 The required index values must be found on each TPM regardless of platform. These areas
1710 are always present and do not require a TPM_DefineSpace command to allocate.

1711 A platform specific specification may add additional required index values for the platform.

1712 **End of informative comment**

1713 1. The TPM MUST reserve the space as indicated for the required index values

1714 **Required Index values**

Value	Index Name	Default Size	Attributes
0xFFFFFFFF	TPM_NV_INDEX_LOCK	Size for this MUST be 0. This value turns on the NV authorization protections. Once executed all NV areas use the protections as defined. This value never resets Attempting to execute TPM_NV_DefineSpace on this value after it is set, or at any time with non-zero size MUST result in a TPM_BADINDEX response	None
0x00000000	TPM_NV_INDEX0	Size for this MUST be 0. This value allows for the setting of the persistent lock bit which is only reset on TPM_Startup(ST_Clear) Attempting to execute TPM_NV_DefineSpace with a size other than zero MUST result in the TPM_BADINDEX error code Attempts to execute TPM_NV_DefineSpace when the lock is already set MUST return TPM_SUCCESS	None
0x10000001	TPM_NV_INDEX_DIR	Size MUST be 20. This index points to the deprecated DIR command area from 1.1. The TPM MUST map this reserved space to be the area operated on by the 1.1 DIR commands. As the DIR commands are deprecated any additional DIR functionality MUST use the NV commands and not the DIR command. Attempts to execute TPM_NV_DefineSpace with this index MUST result in TPM_BADINDEX	TPM_NV_PER_OWNERWRITE TPM_NV_PER_WRITEALL

1715 **19.1.2 Reserved Index values**

1716 **Start of informative comment**

1717 The reserved values are defined to avoid index collisions. These values are not in each and
1718 every TPM.

1719 **End of informative comment**

- 1720 1. The reserved index values are to avoid index value collisions.
1721 2. These index values require a TPM_DefineSpace to have the area for the index allocated
1722 3. A platform specific specification MAY indicate that reserved values are required.
1723 4. The reserved index values MAY have their D bit set by the TPM vendor to permanently
1724 reserve the index in the TPM

Value	Event Name	Default Size
0x0000F000	TPM_NV_INDEX_EKCert	The Endorsement credential
0x0000F001	TPM_NV_INDEX_TPM_CC	The TPM Conformance credential
0x0000F002	TPM_NV_INDEX_PlatformCert	The platform credential
0x0000F003	TPM_NV_INDEX_Platform_CC	The Platform conformance credential
0x000111xx	TPM_NV_INDEX_TSS	Reserved for TSS use
0x000112xx	TPM_NV_INDEX_PC	Reserved for PC Client use
0x000113xx	TPM_NV_INDEX_SERVER	reserved for Server use
0x000114xx	TPM_NV_INDEX_MOBILE	Reserved for mobile use
0x000115xx	TPM_NV_INDEX_PERIPHERAL	Reserved for peripheral use
0x000116xx	TPM_NV_INDEX_GPIO_xx	Reserved for GPIO pins
0x0001xxxx	TPM_NV_INDEX_GROUP_RESV	Reserved for TCG WG's

1725 **19.2 TPM_NV_ATTRIBUTES**1726 **Start of informative comment**

1727 This structure allows the TPM to keep track of the data and permissions to manipulate the
1728 area.

1729 A write once per lifetime of the TPM attribute, while attractive, is simply too dangerous
1730 (attacker allocates all of the NV area and uses it). The locked attribute adds close to that
1731 functionality. This allows the area to be “locked” and only changed when unlocked. The lock
1732 bit would be set for all indexes sometime during the initialization of a platform. The use
1733 model would be that the platform BIOS would lock the TPM and only allow changes in the
1734 BIOS setup routine.

1735 There are no locality bits to allow for a locality to define space. The rationale behind this is
1736 that the define space includes the permissions so that would mean any locality could define
1737 space. The use model for localities would assume that the platform owner was opting into
1738 the use of localities and would define the space necessary to operate when the opt-in was
1739 authorized.

1740 **End of informative comment**1741 **Definition**

```
1742 typedef struct tdTPM_NV_ATTRIBUTES{
1743     TPM_STRUCTURE_TAG tag;
1744     UINT32 attributes;
1745 } TPM_NV_ATTRIBUTES;
```

1746 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	TPM_TAG_NV_ATTRIBUTES
UINT32	attributes	The attribute area

1747 **Attributes values**

Bit	Name	Description
31	TPM_NV_PER_READ_STCLEAR	The value can be read until locked by a read with a data size of 0. It can only be unlocked by TPM_Startup(ST_Clear) or a successful write.
30:19	Reserved	
18	TPM_NV_PER_AUTHREAD	The value requires authorization to read
17	TPM_NV_PER_OWNERREAD	The value requires TPM Owner authorization to read.
16	TPM_NV_PER_PPREAD	The value requires physical presence to read
15	TPM_NV_PER_GLOBALLOCK	The value is writeable until a write to index 0 is successful. The lock of this attribute is reset by TPM_Startup(ST_CLEAR). Lock held by SV -> bGlobalLock
14	TPM_NV_PER_WRITE_STCLEAR	The value is writeable until a write to the specified index with a datasize of 0 is successful. The lock of this attribute is reset by TPM_Startup(ST_CLEAR). Lock held for each area in bWriteSTClear
13	TPM_NV_PER_WRITEDEFINE	The value can only be written once after performing the TPM_NV_DefineSpace command. Lock held for each area as bWriteDefine. Lock set by writing to the index with a datasize of 0

Bit	Name	Description
12	TPM_NV_PER_WRITEALL	The value must be written in a single operation
11:3	Reserved for write additions	
2	TPM_NV_PER_AUTHWRITE	The value requires authorization to write
1	TPM_NV_PER_OWNERWRITE	The value requires TPM Owner authorization to write
0	TPM_NV_PER_PPWRITE	The value requires physical presence to write

1748 **19.3 TPM_NV_DATA_PUBLIC**1749 **Start of informative comment**

1750 This structure represents the public description and controls on the NV area.

1751 **End of informative comment**1752 **Definition**

```

1753 typedef struct tdTPM_NV_DATA_PUBLIC {
1754     TPM_STRUCTURE_TAG tag;
1755     TPM_NV_INDEX nvIndex;
1756     TPM_PCR_INFO_SHORT pcrInfoRead;
1757     TPM_PCR_INFO_SHORT pcrInfoWrite;
1758     TPM_NV_ATTRIBUTES permission;
1759     BOOL bReadSTClear;
1760     BOOL bWriteSTClear;
1761     BOOL bWriteDefine;
1762     UINT32 dataSize;
1763 } TPM_NV_DATA_PUBLIC;

```

1764 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_NV_DATA_PUBLIC
TPM_NV_INDEX	nvIndex	The index of the data area
TPM_PCR_INFO_SHORT	pcrInfoRead	The PCR selection that allows reading of the area
TPM_PCR_INFO_SHORT	pcrInfoWrite	The PCR selection that allows writing of the area
TPM_NV_ATTRIBUTES	permission	The permissions for manipulating the area
BOOL	bReadSTClear	Set to FALSE on each TPM_Startup(ST_Clear) and set to TRUE after a ReadValuexxx with datasize of 0
BOOL	bWriteSTClear	Set to FALSE on each TPM_Startup(ST_CLEAR) and set to TRUE after a WriteValuexxx with a datasize of 0. Set to FALSE on a WriteValuexxx with a datasize other than 0.
BOOL	bWriteDefine	Set to FALSE after TPM_NV_DefineSpace and set to TRUE after a successful WriteValue with a datasize of 0
UINT32	dataSize	The size of the data area in bytes

1765 **Actions**

- 1766 1. On read of this structure (through GetCapability) if pcrInfoRead -> pcrSelect is 0 then
1767 pcrInfoRead -> digestAtRelease MUST be 0x00...00
- 1768 2. On read of this structure (through GetCapability) if pcrInfoWrite -> pcrSelect is 0 then
1769 pcrInfoWrite -> digestAtRelease MUST be 0x00...00

1770 **19.4 TPM_NV_DATA_SENSITIVE**

1771 **Start of informative comment**

1772 This is an internal structure that the TPM uses to keep the actual NV data and the controls
1773 regarding the area.

1774 This entire section is informative

1775 **End of informative comment**

1776 **Definition**

```
1777 typedef struct tdTPM_NV_DATA_SENSITIVE {
1778     TPM_STRUCTURE_TAG tag;
1779     TPM_NV_DATA_PUBLIC pubInfo;
1780     TPM_AUTHDATA authValue;
1781     [size_is(dataSize)] BYTE* data;
1782 } TPM_NV_DATA_SENSITIVE;
```

1783 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_NV_DATA_SENSITIVE
TPM_NV_DATA_PUBLIC	pubInfo	The public information regarding this area
TPM_AUTHDATA	authValue	The AuthData value to manipulate the value
BYTE*	data	The data area. This MUST not contain any sensitive information as the TPM does not provide any confidentiality on the data.

1784 **19.5 Max NV Size**

1785 The value `TPM_MAX_NV_SIZE` is a value where the minimum value is set by the platform
1786 specific specification. The TPM vendor can design a TPM with a size that is larger than the
1787 minimum.

1788 **19.6 TPM_NV_DATA_AREA**

1789 **Start of informative comment**

1790 TPM_NV_DATA_AREA is an indication of the internal structure the TPM uses to track NV
1791 areas. The structure definition is TPM vendor specific and never leaves the TPM. The
1792 structure would contain both the TPM_NV_DATA_PUBLIC and TPM_NV_DATA_SENSITIVE
1793 areas.

1794 **End of informative comment**

1795 **20. Delegate Structures**1796 **20.1 Structures and encryption**1797 **Start of informative comment**

1798 The TPM is responsible for encrypting various delegation elements when stored off the TPM.
1799 When the structures are TPM internal structures and not in use by any other process (i.e.
1800 TPM_DELEGATE_SENSITIVE) the structure is merely an informative comment as to the
1801 information necessary to make delegation work. The TPM may put additional, or possibly,
1802 less information into the structure and still obtain the same result.

1803 Where the structures are in use across TPM's or in use by outside processes (i.e.
1804 TPM_DELEGATE_PUBLIC) the structure is normative and the must use the structure
1805 without modification.

1806 **End of informative comment**

- 1807 1. The TPM MUST provide encryption of sensitive areas held outside of the TPM. The
1808 encryption MUST be comparable to AES 128-bit key or a full three key triple DES.

1809 20.2 Delegate Definitions

1810 Informative comment

1811 The delegations are in a 64-bit field. Each bit describes a capability that the TPM Owner
1812 can delegate to a trusted process by setting that bit. Each delegation bit setting is
1813 independent of any other delegation bit setting in a row.

1814 If a TPM command is not listed in the following table, then the TPM Owner cannot delegate
1815 that capability to a trusted process. For the TPM commands that are listed in the following
1816 table, if the bit associated with a TPM command is set to zero in the row of the table that
1817 identifies a trusted process, then that process has not been delegated to use that TPM
1818 command.

1819 The minimum granularity for delegation is at the ordinal level. It is not possible to delegate
1820 an option of an ordinal. This implies that if the options present a difficulty and there is a
1821 need to separate the delegations then there needs to be a split into two separate ordinals.

1822 End of informative comment

```
1823 #define TPM_DEL_OWNER_BITS 0x00000001
1824 #define TPM_DEL_KEY_BITS 0x00000002
1825
1826 typedef struct tdTPM_DELEGATIONS{
1827     TPM_STRUCTURE_TAG tag;
1828     UINT32 delegateType;
1829     UINT32 per1;
1830     UINT32 per2;
1831 } TPM_DELEGATIONS;
```

1832 Parameters

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_DELEGATIONS
UINT32	delegateType	Owner or key
UNIT32	per1	The first block of permissions
UINT32	per2	The second block of permissions

1833 **20.2.1 Owner Permission Settings**1834 **Informative comment**

1835 This section is going to remove any ambiguity as to the order of bits in the permission array

1836 **End of informative comment**1837 **Per1 bits**

Bit Number	Ordinal	Bit Name
31	Reserved	Reserved MUST be 0
30	TPM_ORD_SetOrdinalAuditStatus	TPM_DELEGATE_SetOrdinalAuditStatus
29	TPM_ORD_DirWriteAuth	TPM_DELEGATE_DirWriteAuth
28		
27		
26	TPM_ORD_CMD_CreateTicket	TPM_DELEGATE_CMD_CreateTicket
25		
24	TPM_ORD_Delegate_LoadOwnerDelegation	TPM_DELEGATE_LoadOwnerDelegation
23	TPM_ORD_DAA_Join	TPM_DELEGATE_DAA_Join
22	TPM_ORD_AuthorizeMigrationKey	TPM_DELEGATE_AuthorizeMigrationKey
21	TPM_ORD_CreateMaintenanceArchive	TPM_DELEGATE_CreateMaintenanceArchive
20	TPM_ORD_LoadMaintenanceArchive	TPM_DELEGATE_LoadMaintenanceArchive
19	TPM_ORD_KillMaintenanceFeature	TPM_DELEGATE_KillMaintenanceFeature
18	TPM_ORD_OwnerReadInternalPub	TPM_DELEGATE_OwnerReadInternalPub
17	TPM_ORD_ResetLockValue	TPM_DELEGATE_ResetLockValue
16	TPM_ORD_OwnerClear	TPM_DELEGATE_OwnerClear
15	TPM_ORD_DisableOwnerClear	TPM_DELEGATE_DisableOwnerClear
14	TPM_ORD_DisableForceClear	TPM_DELEGATE_DisableForceClear
13	TPM_ORD_OwnerSetDisable	TPM_DELEGATE_OwnerSetDisable
12		
11	TPM_ORD_MakeIdentity	TPM_DELEGATE_MakeIdentity
10	TPM_ORD_ActivateIdentity	TPM_DELEGATE_ActivateIdentity
9	TPM_ORD_OwnerReadPubek	TPM_DELEGATE_OwnerReadPubek
8	TPM_ORD_DisablePubekRead	TPM_DELEGATE_DisablePubekRead
7	TPM_ORD_SetRedirection	TPM_DELEGATE_SetRedirection
6	TPM_ORD_FieldUpgrade	TPM_DELEGATE_FieldUpgrade
5	TPM_ORD_Delegate_UpdateVerification	TPM_DELEGATE_UpdateVerification
4	TPM_ORD_CreateCounter	TPM_DELEGATE_CreateCounter
3	TPM_ORD_ReleaseCounterOwner	TPM_DELEGATE_ReleaseCounterOwner
2	TPM_ORD_Delegate_Manage	TPM_DELEGATE_Delegate_Manage
1	TPM_ORD_Delegate_CreateOwnerDelegation	TPM_DELEGATE_Delegate_CreateOwnerDelegation
0	TPM_ORD_DAA_Sign	TPM_DELEGATE_DAA_Sign

1838 **Per2 bits**

Bit Number	Ordinal	Bit Name
31:0	Reserved	Reserved MUST be 0

1839 **20.2.2 Owner commands not delegated**

1840 **Start of informative comment**

1841 Not all TPM Owner authorized commands can be delegated. The following table lists those
1842 commands the reason why the command is not delegated.

1843 **End of informative comment**

Command	Rationale
TPM_ChangeAuthOwner	Delegating change owner allows the delegatee to control the TPM Owner. This implies that the delegate has more control than the owner. The owner can create the same situation by merely having the process that the owner wishes to control the TPM to perform ChangeOwner with the current owners permission.
TPM_TakeOwnership	If you don't have an owner how can the current owner delegate the command.
TPM_CMK_SetRestrictions	This command allows the owner to restrict what processes can be delegated the ability to create and manipulate CMK keys
TPM_CertifySelfTest	This command has a security hole and is deleted

1844 **20.2.3 Key Permission settings**1845 **Informative comment**

1846 This section is going to remove any ambiguity as to the order of bits in the permission array

1847 **End of informative comment**1848 **Per1 bits**

Bit Number	Ordinal	Bit Name
31:28	Reserved	Reserved MUST be 0
27	TPM_ORD_TickStampBlob	TPM_KEY_DELEGATE_TickStampBlob
26	TPM_ORD_ChangeAuthAsymStart	TPM_KEY_DELEGATE_ChangeAuthAsymStart
25	TPM_ORD_ChangeAuthAsymFinish	TPM_KEY_DELEGATE_ChangeAuthAsymFinish
24	TPM_ORD_CMK_CreateKey	TPM_KEY_DELEGATE_CMK_CreateKey
23	TPM_ORD_MigrateKey	TPM_KEY_DELEGATE_MigrateKey
22	TPM_ORD_LoadKey2	TPM_KEY_DELEGATE_LoadKey2
21	TPM_ORD_EstablishTransport	TPM_KEY_DELEGATE_EstablishTransport
20	TPM_ORD_ReleaseTransportSigned	TPM_KEY_DELEGATE_ReleaseTransportSigned
19	TPM_ORD_Quote2	TPM_KEY_DELEGATE_Quote2
18	TPM_ORD_Sealx	TPM_KEY_DELEGATE_Sealx
17	TPM_ORD_MakeIdentity	TPM_KEY_DELEGATE_MakeIdentity
16	TPM_ORD_ActivateIdentity	TPM_KEY_DELEGATE_ActivateIdentity
15	TPM_ORD_GetAuditDigestSigned	TPM_KEY_DELEGATE_GetAuditDigestSigned
14	TPM_ORD_Sign	TPM_KEY_DELEGATE_Sign
13	TPM_ORD_CertifyKey2	TPM_KEY_DELEGATE_CertifyKey2
12	TPM_ORD_CertifyKey	TPM_KEY_DELEGATE_CertifyKey
11	TPM_ORD_CreateWrapKey	TPM_KEY_DELEGATE_CreateWrapKey
10	TPM_ORD_CMK_CreateBlob	TPM_KEY_DELEGATE_CMK_CreateBlob
9	TPM_ORD_CreateMigrationBlob	TPM_KEY_DELEGATE_CreateMigrationBlob
8	TPM_ORD_ConvertMigrationBlob	TPM_KEY_DELEGATE_ConvertMigrationBlob
7	TPM_ORD_Delete_CreateKeyDelegation	TPM_KEY_DELEGATE_CreateKeyDelegation
6	TPM_ORD_ChangeAuth	TPM_KEY_DELEGATE_ChangeAuth
5	TPM_ORD_GetPubKey	TPM_KEY_DELEGATE_GetPubKey
4	TPM_ORD_Unbind	TPM_KEY_DELEGATE_Unbind
3	TPM_ORD_Quote	TPM_KEY_DELEGATE_Quote
2	TPM_ORD_Unseal	TPM_KEY_DELEGATE_Unseal
1	TPM_ORD_Seal	TPM_KEY_DELEGATE_Seal
0	TPM_ORD_LoadKey	TPM_KEY_DELEGATE_LoadKey

1849 **Per2 bits**

Bit Number	Ordinal	Bit Name
31:0	Reserved	Reserved MUST be 0

1850 **20.2.4 Key commands not delegated**

1851 **Start of informative comment**

1852 Not all TPM key commands can be delegated. The following table lists those commands the
1853 reason why the command is not delegated.

1854 **End of informative comment**

Command	Rationale
None	

1855 **20.3 TPM_FAMILY_FLAGS**1856 **Start of informative comment**

1857 These flags indicate the operational state of the delegation and family table. These flags are
1858 additions to TPM_PERMANENT_FLAGS and are not standalone values.

1859 **End of informative comment**1860 **TPM_FAMILY_FLAGS bit settings**

Bit Number	Bit Name	Comments
31:2	Reserved MUST be 0	
1	TPM_DELEGATE_ADMIN_LOCK	TRUE: Some TPM_Delegate_XXX commands are locked and return TPM_DELEGATE_LOCK FALSE: TPM_Delegate_XXX commands are available Default is FALSE
0	TPM_FAMFLAG_ENABLED	When TRUE the table is enabled. The default vaue is FALSE.

1861 **20.4 TPM_FAMILY_LABEL**

1862 **Start of informative comment**

1863 Used in the family table to hold a one-byte numeric value (sequence number) that software
1864 can map to a string of bytes that can be displayed or used by applications.

1865 This is not sensitive data.

1866 **End of informative comment**

```
1867 typedef struct tdTPM_FAMILY_LABEL{  
1868     BYTE label;  
1869 } TPM_FAMILY_LABEL;
```

1870 **Parameters**

Type	Name	Description
BYTE	label	A sequence number that software can map to a string of bytes that can be displayed or used by the applications. This MUST not contain sensitive information.

1871 **20.5 TPM_FAMILY_TABLE_ENTRY**1872 **Start of informative comment**

1873 The family table entry is an individual row in the family table. There are no sensitive values
1874 in a family table entry.

1875 Each family table entry contains values to facilitate table management: the familyID
1876 sequence number value that associates a family table row with one or more delegate table
1877 rows, a verification sequence number value that identifies when rows in the delegate table
1878 were last verified, and a BYTE family label value that software can map to an ASCII text
1879 description of the entity using the family table entry

1880 **End of informative comment**

```
1881 typedef struct tdTPM_FAMILY_TABLE_ENTRY{
1882     TPM_STRUCTURE_TAG tag;
1883     TPM_FAMILY_LABEL familyLabel;
1884     TPM_FAMILY_ID familyID;
1885     TPM_FAMILY_VERIFICATION verificationCount;
1886     TPM_FAMILY_FLAGS flags;
1887 } TPM_FAMILY_TABLE_ENTRY;
```

1888 **Description**

1889 The default value of all fields in a family row at TPM manufacture SHALL be null.

1890 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_FAMILY_TABLE_ENTRY
TPM_FAMILY_LABEL	familyLabel	The text description of the entity using this family row. This is not a sensitive value.
TPM_FAMILY_ID	familyID	The family ID in use to tie values together. This is not a sensitive value.
TPM_FAMILY_VERIFICATION	verificationCount	The value inserted into delegation rows to indicate that they are the current generation of rows. Used to identify when a row in the delegate table was last verified. This is not a sensitive value.
TPM_FAMILY_FLAGS	flags	See section on TPM_FAMILY_FLAGS.

1891 **20.6 TPM_FAMILY_TABLE**

1892 **Start of informative comment**

1893 The family table is stored in a TPM shielded location. There are no confidential values in the
1894 family table. The family table contains a minimum of 8 rows.

1895 **End of informative comment**

```
1896 #define TPM_NUM_FAMILY_TABLE_ENTRY_MIN 8  
1897  
1898 typedef struct tdTPM_FAMILY_TABLE{  
1899     TPM_FAMILY_TABLE_ENTRY famTableRow[TPM_NUM_FAMILY_TABLE_ENTRY_MIN];  
1900 } TPM_FAMILY_TABLE;
```

1901 **Parameters**

Type	Name	Description
TPM_FAMILY_TABLE_ENTRY	famTableRow	The array of family table entries

1902 **20.7 TPM_DELEGATE_LABEL**1903 **Start of informative comment**

1904 Used in both the delegate table and the family table to hold a sequence of bytes that can be
1905 displayed or used by applications. This is not sensitive data.

1906 **End of informative comment**

```
1907 typedef struct tdTPM_DELEGATE_LABEL{
1908     BYTE label;
1909 } TPM_DELEGATE_LABEL;
```

1910 **Parameters**

Type	Name	Description
BYTE	label	A byte that can be displayed or used by the applications. This MUST not contain sensitive information.

1911 20.8 TPM_DELEGATE_PUBLIC

1912 **Start of informative comment**

1913 The information of a delegate row that is public and does not have any sensitive
1914 information.

1915 PCR_INFO_SHORT is appropriate here as the command to create this is done using owner
1916 authorization, hence the owner authorized the command and the delegation. There is no
1917 need to validate what configuration was controlling the platform during the blob creation.

1918 **End of informative comment**

```
1919 typedef struct tdTPM_DELEGATE_PUBLIC{
1920     TPM_STRUCTURE_TAG tag;
1921     TPM_DELEGATE_LABEL rowLabel;
1922     TPM_PCR_INFO_SHORT pcrInfo;
1923     TPM_DELEGATIONS permissions;
1924     TPM_FAMILY_ID familyID;
1925     TPM_FAMILY_VERIFICATION verificationCount
1926 } TPM_DELEGATE_PUBLIC;
```

1927 **Description**

1928 The default value of all fields of a delegate row at TPM manufacture SHALL be null. The
1929 table MUST NOT contain any sensitive information.

1930 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_DELEGATE_PUBLIC
TPM_DELEGATE_LABEL	rowlabel	This SHALL be the label for the row. It MUST not contain any sensitive information.
TPM_PCR_INFO_SHORT	pcrInfo	This SHALL be the designation of the process that can use the permission. This is a not sensitive value. PCR_SELECTION may be NULL. If selected the pcrInfo MUST be checked on each use of the delegation. Use of the delegation is where the delegation is passed as an authorization handle.
TPM_DELEGATIONS	permissions	This SHALL be the permissions that are allowed to the indicated process. This is not a sensitive value.
TPM_FAMILY_ID	familyID	This SHALL be the family ID that identifies which family the row belongs to. This is not a sensitive value.
TPM_FAMILY_VERIFICATION	verificationCount	A copy of verificationCount from the associated family table. This is not a sensitive value.

1931 **20.9 TPM_DELEGATE_TABLE_ROW**1932 **Start of informative comment**

1933 A row of the delegate table.

1934 **End of informative comment**

```

1935 typedef struct tdTPM_DELEGATE_TABLE_ROW{
1936     TPM_STRUCTURE_TAG tag;
1937     TPM_DELEGATE_PUBLIC pub;
1938     TPM_SECRET authValue;
1939 } TPM_DELEGATE_TABLE_ROW;

```

1940 **Description**

1941 The default value of all fields of a delegate row at TPM manufacture SHALL be empty

1942 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This SHALL TPM_TAG_DELEGATE_TABLE_ROW
TPM_DELEGATE_PUBLIC	pub	This SHALL be the public information for a table row.
TPM_SECRET	authValue	This SHALL be the AuthData value that can use the permissions. This is a sensitive value.

1943 **20.10 TPM_DELEGATE_TABLE**

1944 **Start of informative comment**

1945 This is the delegate table. The table contains a minimum of 2 rows.

1946 This will be an entry in the TPM_PERMANENT_DATA structure.

1947 **End of informative comment**

```
1948 #define TPM_NUM_DELEGATE_TABLE_ENTRY_MIN 2
```

```
1949
```

```
1950 typedef struct tdTPM_DELEGATE_TABLE{
```

```
1951     TPM_DELEGATE_TABLE_ROW delRow[TPM_NUM_DELEGATE_TABLE_ENTRY_MIN];
```

```
1952 } TPM_DELEGATE_TABLE;
```

1953 **Parameters**

Type	Name	Description
TPM_DELEGATE_TABLE_ROW	delRow	The array of delegations

1954 **20.11 TPM_DELEGATE_SENSITIVE**1955 **Start of informative comment**

1956 The TPM_DELEGATE_SENSITIVE structure is the area of a delegate blob that contains
1957 sensitive information.

1958 This structure is informative as the TPM vendor can include additional information. This
1959 structure is under complete control of the TPM and is never seen by any entity other than
1960 internal TPM processes.

1961 **End of informative comment**

```
1962 typedef struct tdTPM_DELEGATE_SENSITIVE {
1963     TPM_STRUCTURE_TAG tag;
1964     TPM_SECRET authValue;
1965 } TPM_DELEGATE_SENSITIVE;
```

1966 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This MUST be TPM_TAG_DELEGATE_SENSITIVE
TPM_SECRET	authValue	AuthData value

1967 **20.12 TPM_DELEGATE_OWNER_BLOB**

1968 **Start of informative comment**

1969 This data structure contains all the information necessary to externally store a set of owner
1970 delegation rights that can subsequently be loaded or used by this TPM.

1971 The encryption mechanism for the sensitive area is a TPM choice. The TPM may use
1972 asymmetric encryption and the SRK for the key. The TPM may use symmetric encryption
1973 and a secret key known only to the TPM.

1974 **End of informative comment**

```

1975 typedef struct tdTPM_DELEGATE_OWNER_BLOB{
1976     TPM_STRUCTURE_TAG tag;
1977     TPM_DELEGATE_PUBLIC pub;
1978     TPM_DIGEST integrityDigest;
1979     UINT32 additionalSize;
1980     [size_is(additionalSize)] BYTE* additionalArea;
1981     UINT32 sensitiveSize;
1982     [size_is(sensitiveSize)] BYTE* sensitiveArea;
1983 } TPM_DELEGATE_OWNER_BLOB;

```

1984 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This MUST be TPM_TAG_DELEGATE_OWNER_BLOB
TPM_DELEGATE_PUBLIC	pub	The public information for this blob
TPM_DIGEST	integrityDigest	The HMAC to guarantee the integrity of the entire structure
UINT32	additionalSize	The size of additionalArea
BYTE	additionalArea	An area that the TPM can add to the blob which MUST NOT contain any sensitive information. This would include any IV material for symmetric encryption
UINT32	sensitiveSize	The size of the sensitive area
BYTE	sensitiveArea	The area that contains the encrypted TPM_DELEGATE_SENSITIVE

1985 **20.13 TPM_DELEGATE_KEY_BLOB**1986 **Start of informative comment**

1987 A structure identical to TPM_DELEGATE_OWNER_BLOB but which stores delegation
1988 information for user keys. As compared to TPM_DELEGATE_OWNER_BLOB, it adds a hash
1989 of the corresponding public key value to the public information.

1990 **End of informative comment**

```
1991 typedef struct tdTPM_DELEGATE_KEY_BLOB{
1992     TPM_STRUCTURE_TAG tag;
1993     TPM_DELEGATE_PUBLIC pub;
1994     TPM_DIGEST integrityDigest;
1995     TPM_DIGEST pubKeyDigest;
1996     UINT32 additionalSize;
1997     [size_is(additionalSize)] BYTE* additionalArea;
1998     UINT32 sensitiveSize;
1999     [size_is(sensitiveSize)] BYTE* sensitiveArea;
2000 } TPM_DELEGATE_KEY_BLOB;
```

2001 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	This MUST be TPM_TAG_DELG_KEY_BLOB
TPM_DELEGATE_PUBLIC	pub	The public information for this blob
TPM_DIGEST	integrityDigest	The HMAC to guarantee the integrity of the entire structure
TPM_DIGEST	pubKeyDigest	The digest, that uniquely identifies the key for which this usage delegation applies. This is a hash of the TPM_STORE_PUBKEY structure.
UINT32	additionalSize	The size of the integrity area
BYTE	additionalArea	An area that the TPM can add to the blob which MUST NOT contain any sensitive information. This would include any IV material for symmetric encryption
UINT32	sensitiveSize	The size of the sensitive area
BYTE	sensitiveArea	The area that contains the encrypted TPM_DELEGATE_SENSITIVE

2002 **20.14 TPM_FAMILY_OPERATION Values**

2003 **Start of informative comment**

2004 These are the opFlag values used by TPM_Delegate_Manage.

2005 **End of informative comment**

Value	Capability Name	Comments
0x00000001	TPM_FAMILY_CREATE	Create a new family
0x00000002	TPM_FAMILY_ENABLE	Set or reset the enable flag for this family.
0x00000003	TPM_FAMILY_ADMIN	Prevent administration of this family..
0x00000004	TPM_FAMILY_INVALIDATE	Invalidate a specific family row.

2006 **21. Capability areas**2007 **21.1 TPM_CAPABILITY_AREA for GetCapability**2008 **Start of informative comment**

2009 The TPM needs to provide to outside entities various pieces of information regarding the
2010 design and current state of the TPM. The process works by first supplying an area to look at
2011 and then optionally a refinement to further indicate the type of information requested. The
2012 documents use the terms capability and subcap to indicate the area and subarea in
2013 question.

2014 Some capabilities have a single purpose and the subcap is either ignored or supplies a
2015 handle or other generic piece of information.

2016 The following table contains both the values for the capabilities but also the sub
2017 capabilities. When providing the value for a subcap it appears in the capability name slot.
2018 Subcap is always a 32 bit value.

2019 **End of informative comment**2020 **TPM_CAPABILITY_AREA Values for TPM_GetCapability**

Value	Capability Name	Sub cap	Comments
0x00000001	TPM_CAP_ORD	A command ordinal	Boolean value. TRUE indicates that the TPM supports the ordinal. FALSE indicates that the TPM does not support the ordinal.
0x00000002	TPM_CAP_ALG	TPM_ALG_XX: A value from TPM_ALGORITHM_ID	Boolean value. TRUE means that the TPM supports the algorithm for TPM_Sign, TPM_Seal, TPM_UnSeal and TPM_Unbind and related commands. FALSE indicates that for these types of commands the algorithm is not supported.
0x00000003	TPM_CAP_PID	TPM_PID_XX: A value of TPM_PROTOCOL_ID:	Boolean value. TRUE indicates that the TPM supports the protocol, FALSE indicates that the TPM does not support the protocol.
0x00000004	TPM_CAP_FLAG		Either of the next two subcaps
	0x0000108	TPM_CAP_FLAG_PERMANENT	Return the TPM_PERMANENT_FLAGS structure. Each flag in the structure returns as a byte.
	0x0000109	TPM_CAP_FLAG_VOLATILE	Return the TPM_STCLEAR_FLAGS structure. Each flag in the structure returns as a byte.
0x00000005	TPM_CAP_PROPERTY		See following table for the subcaps
0x00000006	TPM_CAP_VERSION	Ignored	TPM_Version structure. The Major and Minor must indicate 1.1. The manufacturer information MUST indicate the firmware version of the TPM. Any software using this structure MUST be aware that when included in a structure the value MUST be 1.1.0.0, when reported by this command the manufacturer information MAY include firmware versions. The use of this value is deprecated, new software SHOULD use TPM_CAP_VERSION_VAL to obtain version information regarding the TPM.
0x00000007	TPM_CAP_KEY_HANDLE	Ignored	A TPM_KEY_HANDLE_LIST structure that enumerates all key handles loaded on the TPM. The list only contains the number of handles that an external manager can operate with and does not include the EK or SRK.

Value	Capability Name	Sub cap	Comments
			This is command is available for backwards compability. It is the same as TPM_CAP_HANDLE with a resource type of keys.
0x00000008	TPM_CAP_CHECK_LOADED	ALGORITHM: A value of TPM_KEY_PARMS	A Boolean value. TRUE indicates that the TPM has enough memory available to load a key of the type specified by ALGORITHM. FALSE indicates that the TPM does not have enough memory.
0x00000009	TPM_CAP_SYM_MODE	TPM_SYM_MODE	A Boolean value. TRUE indicates that the TPM supports the TPM_SYM_MODE, FALSE indicates the TPM does not support the mode.
0x0000000A	Unused		
0x0000000B	Unused		
0x0000000C	TPM_CAP_KEY_STATUS	handle	Boolean value of ownerEvict. The handle MUST point to a valid key handle.
0x0000000D	TPM_CAP_NV_LIST	ignored	A list of UINT32 that are the NV storage indexes.
0x0000000E	Unused		
0x0000000F	Unused		
0x00000010	TPM_CAP_MFR		Manufacturer specific. The manufacturer may provide any additional information regarding the TPM and the TPM state but MUST not expose any sensitive information.
0x00000011	TPM_CAP_NV_INDEX	TPM_NV_INDEX	A TPM_NV_DATA_PUBLIC structure that indicates the values for the TPM_NV_INDEX
0x00000012	TPM_CAP_TRANS_ALG	TPM_ALG_XXX	Boolean value. TRUE means that the TPM supports the algorithm for TPM_EstablishTransport, TPM_ExecuteTransport and TPM_ReleaseTransportSigned. FALSE indicates that for these three commands the algorithm is not supported."
0x00000013			
0x00000014	TPM_CAP_HANDLE	TPM_RESOURCE_TYPE	A TPM_KEY_HANDLE_LIST structure that enumerates all handles currently loaded in the TPM for the given resource type. When describing keys the handle list only contains the number of handles that an external manager can operate with and does not include the EK or SRK. Legal resources are TPM_RT_KEY, TPM_RT_AUTH, TPM_RT_TRANS,, TPM_RT_COUNTER TPM_RT_CONTEXT is valid and returns not a list of handles but a list of the context count values.
0x00000015	TPM_CAP_TRANS_ES	TPM_ES_XXX	Boolean value. TRUE means the TPM supports the encryption scheme in a transport session.
0x00000016			
0x00000017	TPM_CAP_AUTH_ENCRYPT	TPM_ALGORITHM_ID	Boolean value. TRUE indicates that the TPM supports the encryption algorithm in OSAP encryption of AuthData values
0x00000018	TPM_CAP_SELECT_SIZE	TPM_SELECT_SIZE	Boolean value. TRUE indicates that the TPM supports the size for the given version. For instance a request could ask for version 1.1 size 2 and the TPM would indicate TRUE. For 1.1 size 3 the TPM would indicate FALSE. For 1.2 size 3 the TPM would indicate TRUE.
0x00000019			
0x0000001A	TPM_CAP_VERSION_VAL	Ignored	TPM_CAP_VERSION_INFO structure. The TPM fills in the structure and returns the information indicating what the TPM currently supports.

2021 **21.2 CAP_PROPERTY Subcap values for GetCapability**2022 **Start of informative comment**

2023 The TPM_CAP_PROPERTY capability has numerous subcap values. The definition for all
2024 subcap values occurs in this table.

2025 **End of informative comment**2026 **TPM_CAP_PROPERTY Subcap Values for TPM_GetCapability**

Value	Capability Name	Comments
0x00000101	TPM_CAP_PROP_PCR	UINT32 value. Returns the number of PCR registers supported by the TPM
0x00000102	TPM_CAP_PROP_DIR	UNIT32. Deprecated. Returns the number of DIR, which is now fixed at 1
0x00000103	TPM_CAP_PROP_MANUFACTURER	UINT32 value. Returns the Identifier of the TPM manufacturer.
0x00000104	TPM_CAP_PROP_KEYS	UINT32 value. Returns the number of 2048-bit RSA keys that can be loaded. This MAY vary with time and circumstances.
0x00000107	TPM_CAP_PROP_MIN_COUNTER	UINT32. The minimum amount of time in 10ths of a second that must pass between invocations of incrementing the monotonic counter.
0x0000010A	TPM_CAP_PROP_AUTHSESS	UINT32. The number of available authorization sessions. This may vary with time and circumstances
0x0000010B	TPM_CAP_PROP_TRANSESS	UINT32. The number of transport sessions the TPM could currently support
0x0000010C	TPM_CAP_PROP_COUNTERS	UINT32. The number of available monotonic counters. This MAY vary with time and circumstances.
0x0000010D	TPM_CAP_PROP_MAX_AUTHSESS	UINT32. The maximum number of loaded authorization sessions the TPM supports
0x0000010E	TPM_CAP_PROP_MAX_TRANSESS	UINT32. The maximum number of loaded transport sessions the TPM supports. This MAY vary with time and circumstances
0x0000010F	TPM_CAP_PROP_MAX_COUNTERS	UINT32. The maximum number of monotonic counters under control of TPM_CreateCounter
0x00000110	TPM_CAP_PROP_MAX_KEYS	UINT32. The maximum number of 2048 RSA keys that the TPM can support. The number does not include the EK or SRK.
0x00000111	TPM_CAP_PROP_OWNER	BOOL. A value of TRUE indicates that the TPM has successfully installed an owner.
0x00000112	TPM_CAP_PROP_CONTEXT	UINT32. The number of available saved session slots. This MAY vary with time and circumstances.
0x00000113	TPM_CAP_PROP_MAX_CONTEXT	UINT32. The maximum number of saved session slots.
0x00000114	TPM_CAP_PROP_FAMILYROWS	UINT32. The number of rows in the family table
0x00000115	TPM_CAP_PROP_TIS_TIMEOUT	A 4 element array of UINT32 values each denoting the timeout value in microseconds for the following in this order: TIMEOUT_A, TIMEOUT_B, TIMEOUT_C, TIMEOUT_D Where these timeouts are to be used is determined by the platform specific TPM Interface Specification.
0x00000116	TPM_CAP_PROP_STARTUP_EFFECT	The TPM_STARTUP_EFFECTS structure
0x00000117	TPM_CAP_PROP_DELEGATE_ROW	UINT32. The size of the delegate table in rows.
0x00000118	open	
0x00000119	TPM_CAP_PROP_DAA_MAX	UINT32. The maximum number of DAA sessions (join or sign) that the TPM supports
0x0000011A	CAP_PROP_SESSION_DAA	UINT32. The number of available DAA sessions. This may vary with time and circumstances
0x0000011B	TPM_CAP_PROP_CONTEXT_DIST	UINT32. The maximum distance between context count values. This MUST be at least 2 ¹⁶ -1
0x0000011C	TPM_CAP_PROP_DAA_INTERRUPT	BOOL. A value of TRUE indicates that the TPM will accept ANY command while executing a DAA Join or Sign. A value of FALSE indicates that the TPM will invalidate the DAA Join or Sign upon the receipt of any command other than the next join/sign in the session or a TPM_SaveContext

Value	Capability Name	Comments
0x0000011D	TPM_CAP_PROP_SESSIONS	UINT32. The number of available sessions from the pool. This MAY vary with time and circumstances. Pool sessions include authorization and transport sessions.
0x0000011E	TPM_CAP_PROP_MAX_SESSIONS	UINT32. The maximum number of sessions the TPM supports.
0x0000011F	TPM_CAP_PROP_CMK_RESTRICTION	UINT32 TPM_Permanent_Data -> restrictDelegate
0x00000120	TPM_CAP_PROP_DURATION	A 3 element array of UINT32 values each denoting the duration value in microseconds of the duration of the three classes of commands: Small, Medium and Long in the following in this order: SMALL_DURATION, MEDIUM_DURATION, LONG_DURATION
0x00000121	open	
0x00000122	TPM_CAP_PROP_ACTIVE_COUNTER	TPM_COUNT_ID. The id of the current counter. 0xff..ff if no counter is active
0x00000123	TPM_CAP_PROP_MAX_NV_AVAILABLE	UINT32. The maximum number of NV space that can be allocated, MAY vary with time and circumstances.
0x00000124	TPM_CAP_PROP_INPUT_BUFFER	UINT32. The size of the TPM input buffer in bytes.
0x00000125	XX Next number	

2027 **21.3 Bit ordering for structures**2028 **Start of informative comment**

2029 When returning a structure the TPM will use the following bit ordering scheme

2030 **Sample structure**

```

2031 typedef struct tdSAMPLE {
2032     TPM_STRUCTURE_TAG      tag;
2033     UINT32                 N1;
2034     UINT32                 N2;
2035 } SAMPLE;

```

2036 **End of informative comment**

- 2037 1. Using the sample structure in the informative comment as a template the TPM performs
 2038 the following marshaling
- 2039 a. Bit 0 of the output is first bit following the open bracket. The first bit of tag is then
 2040 bit 0 of the output.
 - 2041 b. Bit-N of the output is the nth bit from the opening bracket
 - 2042 i. The bits of N1 appear before the bits of N2 in the output
- 2043 2. All structures use the endness defined in section 2.1 of this document

2044 **21.3.1 Deprecated GetCapability Responses**

Num	CapArea	subCap	Response
1	TPM_CAP_PROPERTY	TPM_CAP_PROP_DIR_AUTH	UINT32 value. Returns the number of DIR registers under control of the TPM owner supported by the TPM. As there is now only 1 DIR, this is deprecated to always return a value of 1 in version 1.2.

2045 **21.4 TPM_CAPABILITY_AREA Values for SetCapability**

2046 **TPM_CAPABILITY_AREA Values for TPM_SetCapability**

Value	Capability Name	Sub cap	Comments
0x00000001	TPM_SET_PERM_FLAGS	See TPM_PERMANENT_FLAGS structure	The ability to set a value is field specific and a review of the structure will disclose the ability and requirements to set a value
0x00000002	TPM_SET_PERM_DATA	See TPM_PERMANENT_DATA structure	The ability to set a value is field specific and a review of the structure will disclose the ability and requirements to set a value
0x00000003	TPM_SET_STCLEAR_FLAGS	See TPM_STCLEAR_FLAGS structure	The ability to set a value is field specific and a review of the structure will disclose the ability and requirements to set a value
0x00000004	TPM_SET_STCLEAR_DATA	See TPM_STCLEAR_DATA structure	The ability to set a value is field specific and a review of the structure will disclose the ability and requirements to set a value
0x00000005	TPM_SET_STANY_FLAGS	See TPM_STANY_FLAGS structure	The ability to set a value is field specific and a review of the structure will disclose the ability and requirements to set a value
0x00000006	TPM_SET_STANY_DATA	See TPM_STANY_DATA structure	The ability to set a value is field specific and a review of the structure will disclose the ability and requirements to set a value
0x00000007	TPM_SET_VENDOR	Vendor specific	This area allows the vendor to set specific areas in the TPM according to the normal shielded location requirements

2047

2048 The setValue type for SetCapability is determined by the definition of the SubCap value
 2049 listed in the structure definition of each flag section. The setValueSize is set according to
 2050 this type.

2051 21.5 SubCap Values for SetCapability

- 2052 1. SubCap values for SetCapability are found in each flag definition section under the table
2053 “Flag Restrictions for SetCapability”. Each table has the following column definitions:
- 2054 a. Flag SubCap Number 0x00000000+: Incremental flag value used in the SubCap field
 - 2055 b. Set: A “Y” in this column indicates that the flag can be set by SetCapability. A “N” in
2056 this column indicates that the flag can not be set by SetCapability.
 - 2057 c. Set restrictions: Restrictions on how and when SetCapability can set a flag. Each flag
2058 that can be set with SetCapability may have one or more restrictions on how and
2059 when the SetCapability can be used to change a value of a flag. A definition of
2060 common restrictions is listed below.
 - 2061 d. Actions From: This column contains information on other TPM command areas that
2062 can effect a flag
- 2063 2. Common Restriction Definitions
- 2064 a. Owner authorization: SetCapability must use owner authorization to change the
2065 value of a flag
 - 2066 b. Physical presence assertion: Physical presence must be asserted in order for
2067 SetCapability to change the value of a flag
 - 2068 c. No Authorization: SetCapability must be sent as TPM_TAG_RQU_COMMAND (no
2069 authorization)

2070 **21.6 TPM_CAP_VERSION_INFO**

2071 **Start of informative comment**

2072 This structure is in use both as input to a getcap and as an output to a getcap request.
 2073 When in use for input, the caller is asking if the TPM supports a certain version, or portion
 2074 of a version.
 2075 For instance the input for version could contain, 1.2.0.0, which means the question is does
 2076 the TPM support the main spec, version 1.2, regardless of manufacturer version. The TPM
 2077 would return TRUE if the TPM supported all 1.2 TPM main spec commands.
 2078 If the input was 1.2.3.1, the question is does the TPM support main spec version 1.2 AND
 2079 manufacturer firmware version 3.1. The TPM would return TRUE if the TPM supported the
 2080 main spec AND the proper firmware.
 2081 The spec letter allows the TPM to respond to 1.2a or 1.2b questions.
 2082 The vendor specific area allows the TPM vendor to provide support for vendor options. The
 2083 TPM vendor may define the area to the TPM vendor’s needs.
 2084 When in use for output, the TPM is returning the current version of the TPM

2085 **End of informative comment**

2086 **Definition**

```

2087 typedef struct tdTPM_CAP_VERSION_INFO {
2088     TPM_STRUCTURE_TAG tag;
2089     TPM_VERSION version;
2090     UINT16 specLevel;
2091     BYTE errataRev;
2092     BYTE tpmVendorID[4];
2093     UINT16 vendorSpecificSize;
2094     [size_is(vendorSpecificSize)] BYTE* vendorSpecific;
2095 } TPM_CAP_VERSION_INFO;
2096
    
```

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_CAP_VERSION_INFO
TPM_VERSION	version	The major spec version
UINT16	specLevel	The level of ordinals supported
TPM_VERSION_BYTE	errataRev	The letter version of the spec
BYTE	tpmVendorID	The vendor ID – Obtained from TCG
UINT16	vendorSpecificSize	The size of the vendor specific area
BYTE*	vendorSpecific	Vendor specific information

2097 **22. DAA Structures**

2098 All byte and bit areas are byte arrays treated as large integers

2099 **22.1 Size definitions**

```
2100 #define DAA_SIZE_r0      43 (Bytes)
2101 #define DAA_SIZE_r1      43 (Bytes)
2102 #define DAA_SIZE_r2     128 (Bytes)
2103 #define DAA_SIZE_r3     168 (Bytes)
2104 #define DAA_SIZE_r4     219 (Bytes)
2105 #define DAA_SIZE_NT      20 (Bytes)
2106 #define DAA_SIZE_v0     128 (Bytes)
2107 #define DAA_SIZE_v1     192 (Bytes)
2108 #define DAA_SIZE_NE     256 (Bytes)
2109 #define DAA_SIZE_w      256 (Bytes)
2110 #define DAA_SIZE_issuerModulus 256 (Bytes)
```

2111 **22.2 Constant definitions**

```
2112 #define DAA_power0      104
2113 #define DAA_power1     1024
```

2114 22.3 TPM_DAA_ISSUER

2115 Start of informative comment

2116 This structure is the abstract representation of non-secret settings controlling a DAA
2117 context. The structure is required when loading public DAA data into a TPM.

2118 TPM_DAA_ISSUER parameters are normally held outside the TPM as plain text data, and
2119 loaded into a TPM when a DAA session is required. A TPM_DAA_ISSUER structure contains
2120 no integrity check: the TPM_DAA_ISSUER structure at time of JOIN is indirectly verified by
2121 the issuer during the JOIN process, and a digest of the verified TPM_DAA_ISSUER structure
2122 is held inside the TPM_DAA_TPM structure created by the JOIN process.

2123 Parameters DAA_digest_X are digests of public DAA_generic_X parameters, and used to
2124 verify that the correct value of DAA_generic_X has been loaded. DAA_generic_q is stored in
2125 its native form to reduce command complexity.

2126 End of informative comment

2127 Definition

```
2128 typedef struct tdTPM_DAA_ISSUER {
2129     TPM_STRUCTURE_TAG  tag;
2130     TPM_DIGEST         DAA_digest_R0;
2131     TPM_DIGEST         DAA_digest_R1;
2132     TPM_DIGEST         DAA_digest_S0;
2133     TPM_DIGEST         DAA_digest_S1;
2134     TPM_DIGEST         DAA_digest_n;
2135     TPM_DIGEST         DAA_digest_gamma;
2136     BYTE[26]          DAA_generic_q;
2137 } TPM_DAA_ISSUER;
2138
```

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_DAA_ISSUER
TPM_DIGEST	DAA_digest_R0	A digest of the parameter "R0", which is not secret and may be common to many TPMs.
TPM_DIGEST	DAA_digest_R1	A digest of the parameter "R1", which is not secret and may be common to many TPMs.
TPM_DIGEST	DAA_digest_S0	A digest of the parameter "S0", which is not secret and may be common to many TPMs.
TPM_DIGEST	DAA_digest_S1	A digest of the parameter "S1", which is not secret and may be common to many TPMs.
TPM_DIGEST	DAA_digest_n	A digest of the parameter "n", which is not secret and may be common to many TPMs.
TPM_DIGEST	DAA_digest_gamma	A digest of the parameter "gamma", which is not secret and may be common to many TPMs.
BYTE[]	DAA_generic_q	The parameter q, which is not secret and may be common to many TPMs. Note that q is slightly larger than a digest, but is stored in its native form to simplify the TPM_DAA_join command. Otherwise, JOIN requires 3 input parameters.

2139 **22.4 TPM_DAA_TPM**2140 **Start of informative comment**

2141 This structure is the abstract representation of TPM specific parameters used during a DAA
2142 context. TPM-specific DAA parameters may be stored outside the TPM, and hence this
2143 structure is needed to save private DAA data from a TPM, or load private DAA data into a
2144 TPM.

2145 If a TPM_DAA_TPM structure is stored outside the TPM, it is stored in a confidential format
2146 that can be interpreted only by the TPM created it. This is to ensure that secret parameters
2147 are rendered confidential, and that both secret and non-secret data in TPM_DAA_TPM form
2148 a self-consistent set.

2149 TPM_DAA_TPM includes a digest of the public DAA parameters that were used during
2150 creation of the TPM_DAA_TPM structure. This is needed to verify that a TPM_DAA_TPM is
2151 being used with the public DAA parameters used to create the TPM_DAA_TPM structure.

2152 Parameters DAA_digest_v0 and DAA_digest_v1 are digests of public DAA_private_v0 and
2153 DAA_private_v1 parameters, and used to verify that the correct private parameters have
2154 been loaded.

2155 **Parameter DAA_count is stored in its native form, because it is smaller than a digest,
2156 and is required to enforce consistency.**

2157 **End of informative comment**2158 **Definition**

```
2159 typedef struct tdTPM_DAA_TPM {
2160     TPM_STRUCTURE_TAG tag;
2161     TPM_DIGEST      DAA_digestIssuer;
2162     TPM_DIGEST      DAA_digest_v0;
2163     TPM_DIGEST      DAA_digest_v1;
2164     TPM_DIGEST      DAA_rekey;
2165     UINT32          DAA_count;
2166 } TPM_DAA_TPM;
```

2167 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_DAA_TPM
TPM_DIGEST	DAA_digestIssuer	A digest of a TPM_DAA_ISSUER structure that contains the parameters used to generate this TPM_DAA_TPM structure.
TPM_DIGEST	DAA_digest_v0	A digest of the parameter "v0", which is secret and specific to this TPM. "v0" is generated during a JOIN phase.
TPM_DIGEST	DAA_digest_v1	A digest of the parameter "v1", which is secret and specific to this TPM. "v1" is generated during a JOIN phase.
TPM_DIGEST	DAA_rekey	A digest related to the rekeying process, which is not secret but is specific to this TPM, and must be consistent across JOIN/SIGN sessions. "rekey" is generated during a JOIN phase.
UINT32	DAA_count	The parameter "count", which is not secret but must be consistent across JOIN/SIGN sessions. "count" is an input to the TPM from the host system.

2168 **22.5 TPM_DAA_CONTEXT**

2169 **Start of informative comment**

2170 TPM_DAA_CONTEXT structure is created and used inside a TPM, and never leaves the TPM.
2171 This entire section is informative as the TPM does not expose this structure.

2172 TPM_DAA_CONTEXT includes a digest of the public and private DAA parameters that were
2173 used during creation of the TPM_DAA_CONTEXT structure. This is needed to verify that a
2174 TPM_DAA_CONTEXT is being used with the public and private DAA parameters used to
2175 create the TPM_DAA_CONTEXT structure.

2176 **End of informative comment**

2177 **Definition**

```
2178 typedef struct tdTPM_DAA_CONTEXT {
2179     TPM_STRUCTURE_TAG    tag;
2180     TPM_DIGEST           DAA_digestContext
2181     TPM_DIGEST           DAA_digest;
2182     TPM_DAA_CONTEXT_SEED DAA_contextSeed;
2183     BYTE[ 256 ]         DAA_scratch;
2184     BYTE                 DAA_stage;
2185 } TPM_DAA_CONTEXT;
```

2186 **Parameters**

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_DAA_CONTEXT
TPM_DIGEST	DAA_digestContext	A digest of parameters used to generate this structure. The parameters vary, depending on whether the session is a JOIN session or a SIGN session.
TPM_DIGEST	DAA_digest	A running digest of certain parameters generated during DAA computation; operationally the same as a PCR (which holds a running digest of integrity metrics).
TPM_DAA_CONTEXT_SEED	DAA_contextSeed	The seed used to generate other DAA session parameters
BYTE[]	DAA_scratch	Memory used to hold different parameters at different times of DAA computation, but only one parameter at a time. The maximum size of this field is 256 bytes
BYTE	DAA_stage	A counter, indicating the stage of DAA computation that was most recently completed. The value of the counter is zero if the TPM currently contains no DAA context. When set to zero (0) the TPM MUST clear all other fields in this structure. The TPM MUST set DAA_stage to 0 on TPM_Startup(ANY)

2187 **22.6 TPM_DAA_JOINDATA**2188 **Start of informative comment**

2189 This structure is the abstract representation of data that exists only during a specific JOIN
2190 session.

2191 **End of informative comment**2192 **Definition**

```
2193 typedef struct tdTPM_DAA_JOINDATA {
2194     BYTE[128]    DAA_join_u0;
2195     BYTE[138]    DAA_join_u1;
2196     TPM_DIGEST   DAA_digest_n0;
2197 } TPM_DAA_JOINDATA;
```

2198 **Parameters**

Type	Name	Description
BYTE[]	DAA_join_u0	A TPM-specific secret "u0", used during the JOIN phase, and discarded afterwards.
BYTE[]	DAA_join_u1	A TPM-specific secret "u1", used during the JOIN phase, and discarded afterwards.
TPM_DIGEST	DAA_digest_n0	A digest of the parameter "n0", which is an RSA public key with exponent $2^{16} + 1$

2199 **22.7 TPM_STANY_DATA Additions**

2200 **Informative comment**

2201 This shows that the volatile data areas are added to the TPM_STANY_DATA structure

2202 **End of informative comment**

2203 **Definition**

```
2204 typedef struct tdTPM_STANY_DATA{
2205     TPM_DAA_ISSUER      DAA_issuerSettings;
2206     TPM_DAA_TPM        DAA_tpmSpecific;
2207     TPM_DAA_CONTEXT    DAA_session;
2208     TPM_DAA_JOINDATA   DAA_joinSession
2209 }TPM_STANY_DATA;
```

2210 **Types of Volatile Data**

Type	Name	Description
TPM_DAA_ISSUER	DAA_issuerSettings	A set of DAA issuer parameters controlling a DAA session.
TPM_DAA_TPM	DAA_tpmSpecific	A set of DAA parameters associated with a specific TPM.
TPM_DAA_CONTEXT	DAA_session	A set of DAA parameters associated with a DAA session.
TPM_DAA_JOIN	DAA_joinSession	A set of DAA parameters used only during the JOIN phase of a DAA session, and generated by the TPM.

2211

2212 **22.8 TPM_DAA_BLOB**2213 **Informative comment**

2214 The structure passed during the join process

2215 **End of informative comment**2216 **Definition**

```

2217 typedef struct tdTPM_DAA_BLOB {
2218     TPM_STRUCTURE_TAG tag;
2219     TPM_RESOURCE_TYPE resourceType;
2220     BYTE[16] label;
2221     TPM_DIGEST blobIntegrity;
2222     UINT32 additionalSize;
2223     [size_is(additionalSize)] BYTE* additionalData;
2224     UINT32 sensitiveSize;
2225     [size_is(sensitiveSize)] BYTE* sensitiveData;
2226 }TPM_DAA_BLOB;
2227

```

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_DAA_BLOB
TPM_RESOURCE_TYPE	resourceType	The resource type: enc(DAA_tpmSpecific) or enc(v0) or enc(v1)
BYTE[16]	label	Label for identification of the blob. Free format area.
TPM_DIGEST	blobIntegrity	The integrity of the entire blob including the sensitive area. This is a HMAC calculation with the entire structure (including sensitiveData) being the hash and tpmProof is the secret
UINT32	additionalSize	The size of additionalData
BYTE	additionalData	Additional information set by the TPM that helps define and reload the context. The information held in this area MUST NOT expose any information held in shielded locations. This should include any IV for symmetric encryption
UINT32	sensitiveSize	The size of sensitiveData
BYTE	sensitiveData	A TPM_DAA_SENSITIVE structure

2228 **22.9 TPM_DAA_SENSITIVE**

2229 **Informative comment**

2230 The encrypted area for the DAA parameters

2231 **End of informative comment**

2232 **Definition**

```
2233 typedef struct tdTPM_DAA_SENSITIVE {  
2234     TPM_STRUCTURE_TAG tag;  
2235     UINT32 internalSize;  
2236     [size_is(internalSize)] BYTE* internalData;  
2237 }TPM_DAA_SENSITIVE;  
2238
```

Type	Name	Description
TPM_STRUCTURE_TAG	tag	MUST be TPM_TAG_DAA_SENSITIVE
UINT32	internalSize	The size of the internalData area
BYTE	internalData	DAA_tpmSpecific or DAA_private_v0 or DAA_private_v1

2239 **23. Redirection**2240 **23.1 TPM_REDIR_COMMAND**2241 **Informative comment**

2242 The types of redirections

2243 **End of informative comment**2244 **Command modes**

Name	Value	Description
	0x00000001	

2245 **24. Deprecated Structures**

2246 **24.1 Persistent Flags**

2247 **Start of Informative comment**

2248 Rewritten to be part of the PERMANENT, STANY and STCLEAR structures

2249 **End of informative comment**

```
2250 typedef struct tdTPM_PERSISTENT_FLAGS{  
2251 // deleted see version 1.1  
2252 } TPM_PERSISTENT_FLAGS;
```

2253 **24.2 Volatile Flags**

2254 **Start of Informative comment**

2255 Rewritten to be part of the PERMANENT, STANY and STCLEAR structures

2256 **End of informative comment**

```
2257 typedef struct tdTPM_VOLATILE_FLAGS{  
2258 // see version 1.1  
2259 } TPM_VOLATILE_FLAGS;
```

2260 **24.3 TPM persistent data**

2261 **Start of Informative comment**

2262 Rewritten to be part of the PERMANENT, STANY and STCLEAR structures

2263 **End of informative comment**

2264 **Definition**

```
2265 typedef struct tdTPM_PERSISTENT_DATA{  
2266 // see version 1.1  
2267 }TPM_PERSISTENT_DATA;
```

2268 **24.4 TPM volatile data**

2269 **Start of Informative comment**

2270 Rewritten to be part of the PERMANENT, STANY and STCLEAR structures

2271 **End of informative comment**

2272 **Definition**

```
2273 typedef struct tdTPM_VOLATILE_DATA{  
2274 // see version 1.1  
2275 }TPM_VOLATILE_DATA;
```

2276 **24.5 TPM SV data**2277 **Start of informative comment**

2278 Rewritten to be part of the PERMANENT, STANY and STCLEAR structures

2279 **End of informative comment**2280 **Definition**

```

2281 typedef struct tdTPM_SV_DATA{
2282 // see version 1.1
2283 }TPM_SV_DATA;
2284

```

2285 **24.6 TPM_SYM_MODE**2286 **Start of informative comment**2287 This indicates the mode of a symmetric encryption. Mode is Electronic CookBook (ECB) or
2288 some other such mechanism.2289 **End of informative comment**2290 **TPM_SYM_MODE values**

Value	Name	Description
0x00000001	TPM_SYM_MODE_ECB	The electronic cookbook mode (this requires no IV)
0x00000002	TPM_SYM_MODE_CBC	Cipher block chaining mode
0x00000003	TPM_SYM_MODE_CFB	Cipher feedback mode

2291 **Description**

2292 The TPM MAY support any of the symmetric encryption modes