

TPM Main

Part 3 IBM Commands

Specification Version 1.2
Revision 10

25 April 2005

Contact: Ken Goldman kgold@watson.ibm.com
Stefan Berger stefan@us.ibm.com

IBM

Table of Contents

1.	Scope and Audience.....	1
1.1	Key words.....	1
1.2	Statement Type.....	1
2.	Description.....	2
3.	Virtualization Commands.....	3
3.1	TPM_CreateInstance	4
3.2	TPM_DeleteInstance.....	6
3.3	TPM_SetupInstance	8
3.3.1	Action Mask.....	10
3.4	TPM_LockInstance	11
3.5	TPM_GetInstanceKey	13
3.6	TPM_SetInstanceKey.....	16
3.7	TPM_GetInstanceData	18
3.8	TPM_SetInstanceData	21
3.9	TPM_TransportInstance	24

End of Introduction do not delete

1. Scope and Audience

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).

2. Description

Start of informative comment:

This section contains a usage rationale for the TPM virtualization commands. These commands allow a virtual TPM instance to be created, deleted, swapped out and back, or migrated from one physical TPM to another.

TPM instance 0 is always present, and the owner of TPM 0 has sole authority for creating further virtual TPM's.

Typical usage:

1. Use TPM_CreateInstance to create a virtual TPM
2. Save the instance:
 - a. Use TPM_GetInstanceKey to create and retrieve the symmetric key used to encrypt sensitive data.
 - b. Use TPM_TransportInstance (TPM_GetCapability) to enumerate owner evict keys
 - c. Use TPM_TransportInstance (TPM_GetCapability) to enumerate NV defined space
 - d. Use TPM_TransportInstance (TPM_SaveState) to save volatile state in non-volatile memory
 - e. Use TPM_GetInstanceData to retrieve encrypted versions of non-volatile memory (NV defined space, owner evict keys, permanent data and flags, and saved volatile state.)
3. Use TPM_DeleteInstance to delete the instance
4. Use TPM_CreateInstance to create a virtual TPM on the same or another physical TPM.
5. Restore the instance:
 - a. Use TPM_SetInstanceKey to load the symmetric key used to decrypt the sensitive data.
 - b. Use TPM_SetInstanceData to restore non-volatile memory (NV defined space, owner evict keys, permanent data and flags, and saved volatile state.)
 - c. Use TPM_TransportInstance (TPM_Startup(STATE)) to restore volatile data

End of informative comment.

3. Virtualization Commands

Start of informative comment:

This section contains the commands that virtualize a TPM.

End of informative comment.

3.1 TPM_CreateInstance

Start of informative comment:

TPM_CreateInstance allows the TPM 0 owner to create an additional virtual TPM instance. After this command, the created instance state is that of a new TPM that has received TPM_Init.

End of informative comment.

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_CreateInstance
5	4			TPM_AUTHHANDLE	authHandle	The authorization session handle used for owner authentication.
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
6	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
7	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization session handle
8	20			TPM_AUTHDATA	ownerAuth	The authorization session digest for inputs and owner authentication. HMAC key: ownerAuth.

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_CreateInstance
4	4	3S	4	TPM_INSTANCE_HANDLE	instanceHandle	Internal TPM handle where the instance was loaded
4	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
5	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
6	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: ownerAuth.

Action

1. Validate that the command came from instance 0. If unsuccessful, return TPM_AUTHFAIL.
2. Validate the owner authorization. If unsuccessful, return TPM_AUTHFAIL.
3. If insufficient space exists, return TPM_RESOURCES.
4. Create a new TPM instance, initialized as per TPM_Init.
5. Return TPM_SUCCESS.

3.2 TPM_DeleteInstance

Start of informative comment:

TPM_DeleteInstance allows the TPM 0 owner to delete a virtual TPM created by TPM_CreateInstance.

End of informative comment.

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_DeleteInstance
4	4	2S	4	TPM_INSTANCE_HANDLE	instanceHandle	Internal TPM handle where the instance was loaded
5	4			TPM_AUTHHANDLE	authHandle	The authorization session handle used for owner authentication.
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
6	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
7	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization session handle
8	20			TPM_AUTHDATA	ownerAuth	The authorization session digest for inputs and owner authentication. HMAC key: ownerAuth.

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_DeleteInstance
4	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
5	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
6	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: ownerAuth.

Action

1. Validate that the command came from instance 0. If unsuccessful, return TPM_AUTHFAIL.
2. Validate the owner authorization. If unsuccessful, return TPM_AUTHFAIL.
3. If instanceHandle = 0, return TPM_BAD_PARAMETER
4. If instanceHandle does not point to a created instance, return TPM_BAD_PARAMETER
5. Delete the TPM instance for instanceHandle.
 - a. Free all NVRAM non-volatile memory.

- i. NV defined space
- ii. Owner Evict Keys
- iii. Saved state
- iv. TPM_PERMANENT_DATA
- v. TPM_PERMANENT_FLAGS
- b. Free all volatile memory.
 - i. TPM_PERMANENT_DATA
 - ii. TPM_PERMANENT_FLAGS
 - iii. TPM_STCLEAR_DATA
 - iv. TPM_STCLEAR_FLAGS
 - v. TPM_STANY_DATA
 - vi. TPM_STLEAR_DATA
 - vii. Key handle entries
6. Return TPM_SUCCESS.

3.3 TPM_SetupInstance

Start of informative comment:

TPM_SetupInstance allows the TPM 0 owner to initialize a newly created or rebooted instance. If the command fails, there is no requirement that the TPM instance MUST roll back its state.

End of informative comment.

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SetupInstance
4	4	2S	4	TPM_INSTANCE_HANDLE	instanceHandle	Internal TPM handle where the instance was loaded
5	4	3S	4	UINT32	pcrListSize	The size of the pcrList
6	<>	4S	<>	BYTE[]	pcrList	A list of TPM_PCRINDEX, TPM_DIGEST pairs
7	4	5S	4	UINT32	actionMask	A bit map of optional actions
8	4			TPM_AUTHHANDLE	authHandle	The authorization session handle used for owner authentication.
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
9	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
10	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization session handle
11	20			TPM_AUTHDATA	ownerAuth	The authorization session digest for inputs and owner authentication. HMAC key: ownerAuth.

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SetupInstance
4	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
5	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
6	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: ownerAuth.

Action

1. Validate that the command came from instance 0. If unsuccessful, return TPM_AUTHFAIL.
2. Validate the owner authorization. If unsuccessful, return TPM_AUTHFAIL.
3. If instanceHandle = 0, return TPM_BAD_PARAMETER

4. If instanceHandle does not point to a created instance, return TPM_BAD_PARAMETER
5. Set T1 to the TPM instance indicated by instanceHandle
6. Process the actionMask. See 3.3.1.
7. Validate that pcrListSize is consistent with a list of TPM_PCRINDEX, TPM_DIGEST pairs. On error, return TPM_BAD_PARAMETER.
8. For each pair in pcrList
 - a. Set P1 to the next TPM_PCRINDEX value
 - b. Set H1 to the next TPM_DIGEST value
 - c. For TPM T1, extend PCR P1 with the digest H1 using the Actions in TPM_Extend
9. Return TPM_SUCCESS.

3.3.1 Action Mask

Name	Value	Description
TPM_INSTANCE_ACTIVATE	0x00000001	Set both the TPM_PERMANENT_FLAGS and TPM_STCLEAR_FLAGS deactivated to FALSE
TPM_INSTANCE_ENABLE	0x00000002	Set TPM_PERMANENT_FLAGS disabled to FALSE
TPM_INSTANCE_STARTUP	0x00000004	Run the equivalent of TPM_Startup(ST_CLEAR). If multiple flags are set, this flag is processed first.

3.4 TPM_LockInstance

Start of informative comment:

TPM_LockInstance allows the TPM 0 owner to lock and unlock a virtual TPM created by TPM_CreateInstance. A locked TPM instance cannot process commands directed to it. This protects an atomic save of the instance state from an intervening command.

End of informative comment.

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_LockInstance
4	4	2S	4	TPM_INSTANCE_HANDLE	instanceHandle	Internal TPM handle where the instance was loaded
5	1	3S	1	BOOL	lock	TRUE: lock, FALSE: unlock
5	4			TPM_AUTHHANDLE	authHandle	The authorization session handle used for owner authentication.
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
6	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
7	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization session handle
8	20			TPM_AUTHDATA	ownerAuth	The authorization session digest for inputs and owner authentication. HMAC key: ownerAuth.

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_LockInstance
4	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
5	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
6	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: ownerAuth.

Action

1. Validate that the command came from instance 0. If unsuccessful, return TPM_AUTHFAIL.
2. Validate the owner authorization. If unsuccessful, return TPM_AUTHFAIL.
3. If instanceHandle = 0, return TPM_BAD_PARAMETER
4. If instanceHandle does not point to a created instance, return TPM_BAD_PARAMETER

5. Set T1 to the TPM instance indicated by instanceHandle
6. If lock is TRUE, lock T1
7. If lock is FALSE, unlock T1
8. Return TPM_SUCCESS.

3.5 TPM_GetInstanceKey

Start of informative comment:

TPM_GetInstanceKey generates, stores, and returns a symmetric key that the TPM_GetInstanceData command uses to encrypt instance data being stored externally.

The symmetric key is encrypted using the specified TPM 0 asymmetric public key.

End of informative comment.

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH2_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_ORD_GetInstanceKey.
4	4			TPM_KEY_HANDLE	keyHandle	Handle of a loaded key that can encrypt the data.
5	4	2S	4	TPM_INSTANCE_HANDLE	instanceHandle	Internal TPM handle where the instance was loaded
6	4			TPM_AUTHHANDLE	keyAuthHandle	The authorization session handle used for keyHandle.
		2H1	20	TPM_NONCE	keyLastNonceEven	Even nonce previously generated by TPM to cover inputs
7	20	3H1	20	TPM_NONCE	keyNonceOdd	Nonce generated by system associated with keyAuthHandle
8	1	4H1	1	BOOL	continueKeySession	The continue use flag for the authorization session handle
9	20			TPM_AUTHDATA	keyAuth	The authorization session digest that authorizes the use of the public key in keyHandle. HMAC key: key.usageAuth.
10	4			TPM_AUTHHANDLE	ownerAuthHandle	The authorization session handle used to authorize inData.
		2H2	20	TPM_NONCE	ownerLastNonceEven	Even nonce previously generated by TPM
11	20	3H2	20	TPM_NONCE	ownerNonceOdd	Nonce generated by system associated with entityAuthHandle
12	1	4H2	1	BOOL	continueOwnerSession	Continue usage flag for dataAuthHandle.
13	20			TPM_AUTHDATA	ownerAuth	The authorization session digest for inputs and owner authorization. HMAC key: ownerAuth.

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH2_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_ORD_GetInstanceKey.
4	4	3S	4	UINT32	secretSize	The size of the secret parameter in bytes
5	<>	4S	<>	BYTE[]	secret	Encrypted instance key
6	20	2H1	20	TPM_NONCE	keyNonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	keyNonceOdd	Nonce generated by system associated with keyAuthHandle
7	1	4H1	1	BOOL	continueKeySession	Continue use flag, TRUE if handle is still active
8	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: key.usageAuth.
9	20	2H2	20	TPM_NONCE	ownerNonceEven	Even nonce newly generated by TPM.
		3H2	20	TPM_NONCE	ownerNonceOdd	Nonce generated by system associated with ownerAuthHandle
10	1	4H2	1	BOOL	continueOwnerSession	Continue use flag, TRUE if handle is still active
11	20			TPM_AUTHDATA	ownerAuth	The authorization session digest used for the ownerAuth session. HMAC key: ownerAuth.

Actions

1. Validate that the command came from instance 0. If unsuccessful, return TPM_AUTHFAIL
2. Validate the owner authorization. If unsuccessful, return TPM_AUTHFAIL
3. If instanceHandle = 0, return TPM_BAD_PARAMETER
4. If instanceHandle does not point to a created instance, return TPM_BAD_PARAMETER
5. Set T1 to the TPM instance indicated by instanceHandle
6. Set K1 to the key indicated by keyHandle
 - a. Validate the key authorization. If unsuccessful, return TPM_AUTHFAIL
 - b. If K1 -> keyUsage != TPM_KEY_STORAGE, return TPM_INVALID_KEYUSAGE
 - c. If K1 -> keyFlags -> migratable is FALSE, return TPM_INVALID_KEYUSAGE
7. Create K2 a TPM_SYMMETRIC_KEY structure
 - a. Set K2 -> algId to the symmetric key algorithm identifier
 - b. Set K2 -> encScheme to the encryption scheme
 - c. Generate a symmetric key according to the algorithm identifier
 - i. Set K2 -> size to the size of the symmetric key
 - ii. Set K2 -> data to the symmetric key
8. Store K2 in T1 -> TPM_STCLEAR_DATA -> instanceKey
9. Create S1 the encryption of K2 using K1
10. Return S1 as secret

11. Return TPM_SUCCESS

3.6 TPM_SetInstanceKey

Start of informative comment:

TPM_SetInstanceKey loads the symmetric key returned by the TPM_GetInstanceKey command.

The symmetric key is decrypted using the specified TPM 0 asymmetric private key. If the virtual TPM is being migrated to a different physical TPM, the asymmetric key must be migrated from the source to the destination TPM 0 before this command can be executed.

End of informative comment.

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH2_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_ORD_SetInstanceKey.
4	4			TPM_KEY_HANDLE	keyHandle	Handle of a loaded key that can decrypt the data.
5	4	2S	4	TPM_INSTANCE_HANDLE	instanceHandle	Internal TPM handle where the instance was loaded
6	4	3S	4	UINT32	secretSize	The size of the secret parameter in bytes
7	<>	4S	<>	BYTE[]	secret	Encrypted instance key
8	4			TPM_AUTHHANDLE	keyAuthHandle	The authorization session handle used for keyHandle.
		2H1	20	TPM_NONCE	keyLastNonceEven	Even nonce previously generated by TPM to cover inputs
9	20	3H1	20	TPM_NONCE	keyNonceOdd	Nonce generated by system associated with keyAuthHandle
10	1	4H1	1	BOOL	continueKeySession	The continue use flag for the authorization session handle
11	20			TPM_AUTHDATA	keyAuth	The authorization session digest that authorizes the use of the public key in keyHandle. HMAC key: key.usageAuth.
12	4			TPM_AUTHHANDLE	ownerAuthHandle	The authorization session handle used to authorize inData.
		2H2	20	TPM_NONCE	ownerLastNonceEven	Even nonce previously generated by TPM
13	20	3H2	20	TPM_NONCE	ownerNonceOdd	Nonce generated by system associated with entityAuthHandle
14	1	4H2	1	BOOL	continueOwnerSession	Continue usage flag for dataAuthHandle.
15	20			TPM_AUTHDATA	ownerAuth	The authorization session digest for inputs and owner authorization. HMAC key: ownerAuth.

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH2_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_ORD_SetInstanceKey.
4	20	2H1	20	TPM_NONCE	keyNonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	keyNonceOdd	Nonce generated by system associated with keyAuthHandle
5	1	4H1	1	BOOL	continueKeySession	Continue use flag, TRUE if handle is still active
6	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: key.usageAuth.
7	20	2H2	20	TPM_NONCE	ownerNonceEven	Even nonce newly generated by TPM.
		3H2	20	TPM_NONCE	ownerNonceOdd	Nonce generated by system associated with ownerAuthHandle
8	1	4H2	1	BOOL	continueOwnerSession	Continue use flag, TRUE if handle is still active
9	20			TPM_AUTHDATA	ownerAuth	The authorization session digest used for the ownerAuth session. HMAC key: ownerAuth.

Actions

1. Validate that the command came from instance 0. If unsuccessful, return TPM_AUTHFAIL
2. Validate the owner authorization. If unsuccessful, return TPM_AUTHFAIL
3. If instanceHandle = 0, return TPM_BAD_PARAMETER
4. If instanceHandle does not point to a created instance, return TPM_BAD_PARAMETER
5. Set T1 to the TPM instance indicated by instanceHandle
6. Set K1 to the key indicated by keyHandle
 - a. Validate the key authorization. If unsuccessful, return TPM_AUTHFAIL
 - b. If K1 -> keyUsage != TPM_KEY_STORAGE, return TPM_INVALID_KEYUSAGE
 - c. If K1 -> keyFlags -> migratable is FALSE, return TPM_INVALID_KEYUSAGE
7. Create K2 a TPM_SYMMETRIC_KEY by decrypting secret using K1
8. Validate K2
 - a. K2 MUST be a valid TPM_SYMMETRIC_KEY structure. If invalid return TPM_INVALID_STRUCTURE
 - b. K2 -> algId MUST be a supported symmetric key algorithm identifier. If invalid return TPM_BAD_KEY_PROPERTY
 - c. K2 -> encScheme MUST be a supported encryption scheme. If invalid return TPM_INAPPROPRIATE_ENC
9. Store K2 in T1 -> TPM_STCLEAR_DATA -> instanceKey
10. Return TPM_SUCCESS

3.7 TPM_GetInstanceData

Start of informative comment:

TPM_GetInstanceData gets TPM state data associated with an instance. All data is encrypted with the symmetric instance key.

State data includes:

Non-volatile data such as:

TPM_PERMANENT_FLAGS

TPM_PERMANENT_DATA

NVRAM defined space

Owner evict keys

Volatile saved state, including:

PCR's

Audit digest

TPM_STCLEAR_FLAGS

TPM_STCLEAR_DATA

End of informative comment.

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_GetInstanceData
4	4	2S	4	TPM_INSTANCE_HANDLE	instanceHandle	Internal TPM handle where the instance was loaded
5	2	3S	2	TPM_STRUCTURE_TAG	structureTag	Denotes the instance structure to be returned
6	4	4S	4	UINT32	value	Meaning dependent on structureTag
7	4			TPM_AUTHHANDLE	authHandle	The authorization session handle used for owner authentication.
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
8	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
9	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization session handle
10	20			TPM_AUTHDATA	ownerAuth	The authorization session digest for inputs and owner authentication. HMAC key: ownerAuth.

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_GetInstanceData
4	4	3S	4	UINT32	encDataSize	The size of the data parameter
5	<>	4S	<>	BYTE[]	encData	Encrypted instance data
6	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
7	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
8	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: ownerAuth.

Action

1. Validate that the command came from instance 0. If unsuccessful, return TPM_AUTHFAIL.
2. Validate the owner authorization. If unsuccessful, return TPM_AUTHFAIL.
3. If instanceHandle = 0, return TPM_BAD_PARAMETER
4. If instanceHandle does not point to a created instance, return TPM_BAD_PARAMETER.
5. Set T1 to the TPM instance indicated by instanceHandle
6. If T1 -> TPM_STCLEAR_DATA -> instanceKey is NULL, return TPM_KEYNOTFOUND
7. If structureTag = TPM_TAG_PERMANENT_DATA
 - a. Set E1 to T1 -> TPM_PERMANENT_DATA
8. Else if structureTag = TPM_TAG_PERMANENT_FLAGS
 - a. Set E1 to T1 -> TPM_PERMANENT_FLAGS
9. Else if structureTag = TPM_TAG_NV_DATA_SENSITIVE
 - a. Set E1 to the T1 -> TPM_NV_DATA_SENSITIVE that corresponds to value as an nvIndex, on error return TPM_BAD_INDEX
10. if structureTag = TPM_TAG_KEY
 - a. Set E1 to the T1 -> TPM_KEY_HANDLE_ENTRY that corresponds to value as an owner evict key handle, on error return TPM_INVALID_KEYHANDLE
11. Else if structureTag = TPM_TAG_STCLEAR_DATA
 - a. Set E1 to the state information previously stored in T1 non-volatile memory by TPM_SaveState, on error return TPM_RESOURCEMISSING
12. Else return TPM_BAD_PARAMETER
13. Create S1 the encryption of E1 using T1 -> TPM_STCLEAR_DATA -> instanceKey
14. Return S1 as encData

15. Return TPM_SUCCESS.

3.8 TPM_SetInstanceData

Start of informative comment:

TPM_SetInstanceData sets TPM state data associated with an instance. The data is generated using TPM_GetInstanceData.

State data includes:

Non-volatile data such as:

TPM_PERMANENT_FLAGS

TPM_PERMANENT_DATA

NVRAM defined space

Owner evict keys

Volatile saved state, including:

PCR's

Audit digest

TPM_STCLEAR_FLAGS

TPM_STCLEAR_DATA

End of informative comment.

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SetInstanceData
4	4	2S	4	TPM_INSTANCE_HANDLE	instanceHandle	Internal TPM handle where the instance was loaded
5	2	3S	2	TPM_STRUCTURE_TAG	structureTag	Denotes the instance structure to be loaded
6	4	4S	4	UINT32	encDataSize	The size of the data parameter
7	<>	5S	<>	BYTE[]	encData	Encrypted instance data
8	4			TPM_AUTHHANDLE	authHandle	The authorization session handle used for owner authentication.
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
9	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
10	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization session handle
11	20			TPM_AUTHDATA	ownerAuth	The authorization session digest for inputs and owner authentication. HMAC key: ownerAuth.

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SetInstanceData
4	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
5	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
6	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: ownerAuth.

Action

1. Validate that the command came from instance 0. If unsuccessful, return TPM_AUTHFAIL.
2. Validate the owner authorization. If unsuccessful, return TPM_AUTHFAIL.
3. If instanceHandle = 0, return TPM_BAD_PARAMETER
4. If instanceHandle does not point to a created instance, return TPM_BAD_PARAMETER
5. Set T1 to the TPM instance indicated by instanceHandle
6. If T1 -> TPM_STCLEAR_DATA -> instanceKey is NULL, return TPM_INVALID_KEYHANDLE.
7. Create E1 the decryption of encData using T1 -> TPM_STCLEAR_DATA -> instanceKey
8. If structureTag = TPM_TAG_PERMANENT_DATA
 - a. Validate that E1 is a TPM_PERMANENT_DATA structure, on error return TPM_INVALID_STRUCTURE
 - b. Store E1 as T1 -> TPM_PERMANENT_DATA
9. Else if structureTag = TPM_TAG_PERMANENT_FLAGS
 - a. Validate that E1 is a TPM_PERMANENT_FLAGS structure, on error return TPM_INVALID_STRUCTURE
 - b. Store E1 as T1 -> TPM_PERMANENT_FLAGS
10. Else if structureTag = TPM_TAG_NV_DATA_SENSITIVE
 - a. Validate that E1 is a TPM_NV_DATA_SENSITIVE structure, on error return TPM_INVALID_STRUCTURE
 - b. If E1 -> pubInfo -> nvIndex corresponds to an already defined T1 NV area, return TPM_BADINDEX.
 - c. If E1 -> pubInfo -> nvIndex corresponds to an illegal index, return TPM_BADINDEX
 - d. Store E1 as T1 -> TPM_NV_DATA_SENSITIVE
11. Else if structureTag = TPM_TAG_KEY
 - a. Validate that E1 is a TPM_KEY_HANDLE_ENTRY structure, on error return TPM_INVALID_STRUCTURE
 - b. If E1 -> handle corresponds to an already defined T1 owner evict key, return TPM_INVALID_KEYHANDLE.
 - c. Store E1 as T1 -> TPM_KEY_HANDLE_ENTRY
12. Else if structureTag = TPM_TAG_STCLEAR_DATA

- a. Store E1 as T1 -> volatile data to be used by TPM_Startup(TPM_ST_STATE)
13. Else return TPM_BAD_PARAMETER
 14. Return TPM_SUCCESS.

3.9 TPM_TransportInstance

Start of informative comment:

TPM_TransportInstance wraps a command to a TPM instance in a TPM 0 owner authorized command.

The commands that can be wrapped are:

- TPM_Extend
- TPM_GetCapability
- TPM_LoadContext
- TPM_PhysicalEnable
- TPM_PhysicalSetDeactivated
- TPM_SaveContext
- TPM_Startup
- TSC_PhysicalPresence

End of informative comment.

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_TransportInstance
4	4	2S	4	UINT32	instanceCommandSize	The size of the instanceCommand
5	<>	3S	<>	BYTE[]	instanceCommand	The wrapped command to be routed to the instance
6	4			TPM_AUTHHANDLE	authHandle	The authorization session handle used for owner authentication.
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
7	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
8	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization session handle
9	20			TPM_AUTHDATA	ownerAuth	The authorization session digest for inputs and owner authentication. HMAC key: ownerAuth.

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_TransportInstance
4	4	3S	4	UINT32	instanceResponseSize	The size of the instanceResponse
5	<>	4S	<>	BYTE[]	instanceResponse	The wrapped response from the instance
6	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
7	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
8	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: ownerAuth.

Action

1. Validate that the command came from instance 0. If unsuccessful, return TPM_AUTHFAIL.
2. Validate the owner authorization. If unsuccessful, return TPM_AUTHFAIL.
3. Set H1 to the instance handle specified by instanceCommand -> tag.
4. If H1 does not point to a created instance, return TPM_BAD_PARAMETER.
5. If H1 = 0, return TPM_BAD_PARAMETER.
6. Set T1 to the TPM instance indicated by H1.
7. Set O1 to the ordinal specified by instanceCommand -> ordinal.
8. If O1 is not a wrapable ordinal, return TPM_BAD_ORDINAL.
9. Set P1 to the parameter size specified by instanceCommand -> paramSize.
10. If P1 does not equal instanceCommandSize. Return TPM_BAD_PARAM_SIZE.
11. Execute ordinal O1 for instance T1.
12. Return the response as instanceResponse.

End of Document - Do Not Delete