Appendix (based on TSS Project for the TCPA TPM Specification version 1.1b)

RC 3.0
September 18, 2003

Change log:

Date	Version	Author	Reason
Dec 4, 2002	0.20	Monty	Added macro to create header files
Dec 5, 2002	0.21	Monty	Added 'C' source files and Makefile to exersize the headers
April 8, 2003	0.22, 0.23	Monty	Released to WG under TCG
April 8, 2003	0.23A	Monty	Moved extraction macro to document.
May 20, 2003	0.23B	markwi	Added some missing IDL definitions; also made a few minor changes to get consistency with the spec.
May 27, 2003	0.35	Monty	Merged MSFT changes into .30 Change UINT to UINT
June 3, 2003	0.36A	Monty	Merge of IFX and IBM submissions. Remove header files that only include header files. Remove documentation only header files.
July 1, 2003	0.36B	Monty	Backed out unapproved changes by IBM in tss_defines.h
July 1, 2003	0.37	Monty	Misc fixes
July 1, 2003	0.38	Monty	Merged in changes from 0.36A to 0.37
July 1, 2003	0.40	Monty	Removed TSS defines to TPM defines to provide abstraction

			from TPM layer. Added legal notice as required for all TCG docs.
July 2, 2003	0.41	Monty	Editor's checkpoint vesion. Not released. Added macros back into file.
July 5, 2003	0.42	Monty	Editor's checkpoint vesion. Not released. Changed TCPA to TCG. Edit title page.
July 5, 2003	0.43	Monty	Editor's checkpoint vesion. Not released. Added IFX changes submitted June 30.
July 5, 2003	0.44	Monty	Add IBM changes submitted July 2, 2003 minus those changes challenged by IFX in e-mail dated 7/3/2003
July 29, 2003	0.45	markwi	Made Microsoft's proposed changes to TCS layer
???	0.46	IBM	Corrections to some defines and missing parameters
Aug 19, 2003	0.47	Monty	Integration of all changes above. Responded to and removal of all "ballon" comments. This is a "check" version and was not released.
Aug 19, 2003	0.48	Monty	Renamed TCSP_ function to Tcsip_. Renamed TCS_ function to Tcsi_. Make usage of T*_AUTH conssistem. Make usage of

			_VERSION consistent with context. Started process of synchronizing TPM defines and structs names with TCPA main spec.
Aug 24, 2003	0.49	Monty	Change TPM labels to TCPA labels. Not released
Aug 25, 2003	0.50	Monty	Added more test cases. Corrected found errors. All .h files compile. Not released
Aug 25, 2003	0.51	Monty	Added TSP 'C' prototypes

```
September 16, 20003   RC 1.0   David Challener  Fixed all errors reported in last meeting, added #define
TSS_KEY_TYPE_DEFAULT    (0x00000000)tss_defines.h
/*
   Work left to do:
      None unless something comes up
*/
```

+++platform.h

```
/*++

There are platform dependent and general defines.

--*/

/*++

On Windows platforms the types are:

--*/

#ifdef _WINDOWS_ // --- This should be used on Windows platforms
   typedef  unsigned char  BYTE;
   typedef  signed char    TSS_BOOL;  // Make specific to TSS to avoid potential conflicts
   typedef  unsigned short UINT16;
   typedef  unsigned long  UINT32;
   typedef  unsigned short UNICODE;
   typedef  void*          PVOID;
#endif

// On Linux platforms the types are:

#ifdef __GNUC__
   typedef u_char      BYTE;
   typedef signed char TSS_BOOL;  // Make specific to TSS to avoid potential conflicts
   typedef u_short     UINT16;
   typedef u_int       UINT32;
   typedef u_short     UNICODE;
   typedef void*       PVOID;
#endif
```

+++tcpa_typedef.h

```
/*++
TCPA typedefs basically extracted from TCPA Main Specification V1.1b
and TCPA Software Stack (TSS) Specification.

Before including this file it is necessary that the platform-specific typedef's are defined. Be advised
that these definitions can depend on both the platform and the compiler used. The defines used in this and
some other TCPA include file are derived from 4 Basic-Types:
--*/

#ifndef __TCPA_TYPEDEF_H__
#define __TCPA_TYPEDEF_H__

#define TRUE  0x01
#define FALSE 0x00

//-----------------------------------------------------------------------
// 4.2.3   Helper redefinitions
typedef UINT32   TCPA_PCRINDEX;     // Index to a PCR register
typedef UINT32   TCPA_DIRINDEX;     // Index to a DIR register
typedef UINT32   TCPA_AUTHHANDLE;   // Handle to an authorization session
typedef UINT32   TSS_HASHHANDLE;    // Handle to a hash session
typedef UINT32   TSS_HMACHHANDLE;   // Handle to a HMAC session
typedef UINT32   TCPA_ENCHANDLE;    // Handle to a encryption/decryption session
typedef UINT32   TCPA_KEY_HANDLE;   // The area where a key is held assigned by the TPM.
typedef UINT32   TCPA_RESULT;       // The return code from a function


// 4.2.4   Enumerated Helper redefinitions

typedef UINT32   TCPA_COMMAND_CODE;       // The command ordinal. See 4.33
typedef UINT16   TCPA_PROTOCOL_ID;        // The protocol in use. See 4.17
typedef UINT32   TCPA_EVENTTYPE;          // Type of PCR event. See 4.25.2
typedef BYTE     TCPA_AUTH_DATA_USAGE;    // Indicates the conditions where it is required that
authorization
                                          //    be presented. See 4.11
```

```
typedef UINT16   TCPA_ENTITY_TYPE;        // Indicates the types of entity that are supported by the TPM.
See 4.15
typedef UINT32   TCPA_ALGORITHM_ID;       // Indicates the type of algorithm. See 4.18
typedef UINT16   TCPA_KEY_USAGE;          // Indicates the permitted usage of the key. See 4.10
typedef UINT16   TCPA_STARTUP_TYPE;       // Indicates the start state. See 4.16
typedef UINT32   TCPA_CAPABILITY_AREA;    // Identifies a TPM capability area.  See 4.31
typedef UINT16   TCPA_ENC_SCHEME;         // The definition of the encryption scheme. See 8.4
typedef UINT16   TCPA_SIG_SCHEME;         // The definition of the signature scheme. See 8.5
typedef UINT16   TCPA_MIGRATE_SCHEME;     // The definition of the migration scheme 4.22
typedef UINT16   TCPA_PHYSICAL_PRESENCE;  // Sets the state of the physical presence mechanism. See section
4.19
typedef UINT32   TCPA_KEY_FLAGS;          // Indicates information regarding a key. See 4.12

typedef UINT16   TCPA_TAG;              // command tag identifier Errata: missing in Main spec
typedef BYTE     TCPA_PAYLOAD_TYPE;     // Errata: should be relocated to here in Main spec


#endif // __TCPA_TYPEDEF_H__
```

+++tcpa_defines.h

```
/*++

TMP defines basically extracted from TCPA Main Specification V1.1

--*/

#ifndef __TCPA_DEFINES_H__
#define __TCPA_DEFINES_H__


//////////////////////////////////////////////////////////////////////
// Parameter List Tag Identifiers
// A command with no authentication
#define TPM_TAG_RQU_COMMAND        (UINT16)(0x00C1)

// An authenticated command with one authentication handle
#define TPM_TAG_RQU_AUTH1_COMMAND  (UINT16)(0x00C2)

//An authenticated command with two authentication handles
#define TPM_TAG_RQU_AUTH2_COMMAND  (UINT16)(0x00C3)

// A response from a command with no authentication
#define TPM_TAG_RSP_COMMAND        (UINT16)(0x00C4)

// An authenticated response with one authentication handle
#define TPM_TAG_RSP_AUTH1_COMMAND  (UINT16)(0x00C5)

// An authenticated response with two authentication handles
#define TPM_TAG_RSP_AUTH2_COMMAND  (UINT16)(0x00C6)

//////////////////////////////////////////////////////////////////////
// vendor specific
//
#define TCPA_Vendor_Specific32  0x00000400
```

```
#define TCPA_Vendor_Specific8   0x80

//////////////////////////////////////////////////////////////////
// section 4.10 - key usage values - TPM_KEY_USAGE
#define TPM_KEY_SIGNING    (UINT16)(0x0010)
#define TPM_KEY_STORAGE    (UINT16)(0x0011)
#define TPM_KEY_IDENTITY   (UINT16)(0x0012)
#define TPM_KEY_AUTHCHANGE (UINT16)(0x0013)
#define TPM_KEY_BIND       (UINT16)(0x0014)
#define TPM_KEY_LEGACY     (UINT16)(0x0015)

//////////////////////////////////////////////////////////////////
// section 4.11 - auth data usage values - TPM_AUTH_DATA_USAGE
#define TPM_AUTH_NEVER   (UINT8) (0x00)
#define TPM_AUTH_ALWAYS  (UINT8) (0x01)

//////////////////////////////////////////////////////////////////
// section 4.14 - payload type values - TPM_PAYLOAD_TYPE
#define TCPA_PT_ASYM      0x01
#define TCPA_PT_BIND      0x02
#define TCPA_PT_MIGRATE   0x03
#define TCPA_PT_MAINT     0x04
#define TCPA_PT_SEAL      0x05

//////////////////////////////////////////////////////////////////
// section 4.15 - TPM_ENTITY_TYPE values
#define TCPA_ET_KEYHANDLE (UINT16)(0x0001)
#define TCPA_ET_OWNER     (UINT16)(0x0002)
#define TCPA_ET_DATA      (UINT16)(0x0003)
#define TCPA_ET_SRK       (UINT16)(0x0004)
#define TCPA_ET_KEY       (UINT16)(0x0005)

// The entity type TPM_ET_OWNER and TPM_ET_SRK are associated with
// specific key handles
// Errata: Not in spec
#define TPM_KEYHND_OWNER TCPA_KEY_HANDLE(0x40000001)
#define TPM_KEYHND_SRK   TPM_KEY_HANDLE(0x40000000)
```

```
//////////////////////////////////////////////////////////////////
// section 4.17 - TPM_PROTOCOL_ID values
#define TCPA_PID_OIAP  (UINT16)(0x0001)
#define TCPA_PID_OSAP  (UINT16)(0x0002)
#define TCPA_PID_ADIP  (UINT16)(0x0003)
#define TCPA_PID_ADCP  (UINT16)(0x0004)
#define TCPA_PID_OWNER (UINT16)(0x0005)

//////////////////////////////////////////////////////////////////
// section 4.18 - algorithm identifiers
#define TCPA_ALG_RSA   (UINT32)(0x00000001)
#define TCPA_ALG_DES   (UINT32)(0x00000002)
#define TCPA_ALG_3DES  (UINT32)(0x00000003)
#define TCPA_ALG_SHA   (UINT32)(0x00000004)
#define TCPA_ALG_HMAC  (UINT32)(0x00000005)
#define TCPA_ALG_AES   (UINT32)(0x00000006)

//////////////////////////////////////////////////////////////////
// section 4.19 - TPM_PHYSICAL_PRESENCE values
#define TCPA_PHYSICAL_PRESENCE_LIFETIME_LOCK  0x0080
#define TCPA_PHYSICAL_PRESENCE_HW_ENABLE      0x0040
#define TCPA_PHYSICAL_PRESENCE_CMD_ENABLE     0x0020
#define TCPA_PHYSICAL_PRESENCE_NOTPRESENT     0x0010
#define TCPA_PHYSICAL_PRESENCE_PRESENT        0x0008
#define TCPA_PHYSICAL_PRESENCE_LOCK           0x0004

//////////////////////////////////////////////////////////////////
// section 4.31 - capability identifiers
#define TCPA_CAP_ORD           (UINT32)(0x00000001)
#define TCPA_CAP_ALG           (UINT32)(0x00000002)
#define TCPA_CAP_PID           (UINT32)(0x00000003)
#define TCPA_CAP_FLAG          (UINT32)(0x00000004)
#define TCPA_CAP_PROPERTY      (UINT32)(0x00000005)
#define TCPA_CAP_VERSION       (UINT32)(0x00000006)
#define TCPA_CAP_KEY_HANDLE    (UINT32)(0x00000007)
#define TCPA_CAP_CHECK_LOADED  (UINT32)(0x00000008)

//////////////////////////////////////////////////////////////////
```

```
// section 8.11.1 - IDL Definitions of subCap
#define TPM_CAP_PROP_PCR           (UINT32)(0x00000101)
#define TPM_CAP_PROP_DIR           (UINT32)(0x00000102)
#define TPM_CAP_PROP_MANUFACTURER  (UINT32)(0x00000103)
#define TPM_CAP_PROP_SLOTS         (UINT32)(0x00000104)

//////////////////////////////////////////////////////////////////
// section 4.33 - command ordinals
#define TCPA_PROTECTED_COMMAND    (UINT32)(0x00000000)
#define TCPA_UNPROTECTED_COMMAND  (UINT32)(0x80000000)
#define TCPA_CONNECTION_COMMAND   (UINT32)(0x40000000)
#define TCPA_VENDOR_COMMAND       (UINT32)(0x20000000)

#define TCPA_MAIN       (UINT16)(0x0000) // Command is from the main specification
#define TCPA_PC         (UINT16)(0x0001) // Command is specific to the PC
#define TCPA_PDA        (UINT16)(0x0002) // Command is specific to a PDA
#define TCPA_CELL_PHONE (UINT16)(0x0003) // Command is specific to a cell phone

#define TCPA_PROTECTED_ORDINAL    TCPA_PROTECTED_COMMAND | TCPA_MAIN
#define TCPA_UNPROTECTED_ORDINAL  TCPA_UNPROTECTED_COMMAND | TCPA_MAIN
#define TCPA_CONNECTION_ORDINAL   TCPA_CONNECTION_COMMAND | TCPA_MAIN

//////////////////////////////////////////////////////////////////
// section 8.5 - TPM_SIG_SCHEME values
// For TCPA_ALG_RSA
#define TCPA_SS_NONE                  (UINT16)(0x0001)
#define TCPA_SS_RSASSAPKCS1v15_SHA1   (UINT16)(0x0002)
#define TCPA_SS_RSASSAPKCS1v15_DER    (UINT16)(0x0003)

//////////////////////////////////////////////////////////////////
// section 8.4 - TPM_ENC_SCHEME values
#define TCPA_ES_NONE                  (UINT16)(0x0001)
#define TCPA_ES_RSAESPKCSv15          (UINT16)(0x0002)
#define TCPA_ES_RSAESOAEP_SHA1_MGF1   (UINT16)(0x0003)

//////////////////////////////////////////////////////////////////
// section 4.22 - TPM_MIGRATE_SCHEME values
#define TCPA_MS_MIGRATE  (UINT16)(0x0001)
```

```
#define TCPA_MS_REWRAP   (UINT16)(0x0002)
#define TCPA_MS_MAINT    (UINT16)(0x0003)

// without any TPM_ like in Main Spec, strange
#define redirection    (UINT32)(0x00000001)
#define migratable     (UINT32)(0x00000002)
#define volatileKey    (UINT32)(0x00000004)

// empty defines Errata: What are these for?
#define AUTH  // paramDigest of HMAC1/HMAC2
#define HMAC1
#define HMAC2

// byte size definition for 160Bit SHA1 hash value
// Errata: Add these to Main Spec
#define TCPA_SHA1_160_HASH_LEN    0x14
#define TCPA_SHA1BASED_NONCE_LEN  TCPA_SHA1_160_HASH_LEN

#endif // __TCPA_DEFINES_H__
```

+++tcpa_struct.h

```
/*++

TPM structures basically extracted from  TCPA Main Specification V1.1b

*/

#ifndef __TCPA_STRUCT_H__
#define __TCPA_STRUCT_H__

//-----------------------------------------------------------------------

//*********************************************************************
// structures

//-----------------------------------------------------------------------
```

```
// section 4.5
typedef struct tdTCPA_VERSION
{
   BYTE   major;
   BYTE   minor;
   BYTE   revMajor;
   BYTE   revMinor;
} TCPA_VERSION;


//-----------------------------------------------------------------------
// section 4.6
// digest size is 20 or greater

typedef struct tdTCPA_DIGEST
{
   BYTE  digest[TCPA_SHA1_160_HASH_LEN];
} TCPA_DIGEST;

//-----------------------------------------------------------------------
typedef TCPA_DIGEST  TCPA_PCRVALUE;
typedef TCPA_DIGEST  TCPA_COMPOSITE_HASH;
typedef TCPA_DIGEST  TCPA_DIRVALUE;
typedef TCPA_DIGEST  TCPA_HMAC;
typedef TCPA_DIGEST  TCPA_CHOSENID_HASH;

//-----------------------------------------------------------------------
// section 4.7
typedef struct tdTCPA_NONCE
{
   BYTE  nonce[TCPA_SHA1BASED_NONCE_LEN];
} TCPA_NONCE;

typedef struct tdTCPA_AUTHDATA
{
   BYTE  authdata[TCPA_SHA1_160_HASH_LEN];
} TCPA_AUTHDATA;
```

```
typedef TCPA_AUTHDATA TCPA_SECRET;
typedef TCPA_AUTHDATA TCPA_ENCAUTH;

//--------------------------------------------------------------------
// section 4.9
typedef struct tdTCPA_KEY_HANDLE_LIST
{
   UINT16  loaded;
#ifdef __midl
   [size_is(loaded)]
#endif
   TCPA_KEY_HANDLE* handle;
} TCPA_KEY_HANDLE_LIST;

//--------------------------------------------------------------------
// section 4.12

// TPM_KEY_FLAGS has been moved to tpm_typedef.h

//--------------------------------------------------------------------
// section 4.20
typedef struct tdTCPA_KEY_PARMS
{
   TCPA_ALGORITHM_ID  algorithmID;
   TCPA_ENC_SCHEME    encScheme;
   TCPA_SIG_SCHEME    sigScheme;
   UINT32             parmSize;
#ifdef __midl
   [size_is(parmSize)]
#endif
   BYTE*              parms;
} TCPA_KEY_PARMS;

typedef struct tdTCPA_RSA_KEY_PARMS
{
   UINT32   keyLength;
   UINT32   numPrimes;
   UINT32   exponentSize;
```

```
#ifdef __midl
   [size_is(exponentSize)]
#endif
   BYTE*    exponent;
} TCPA_RSA_KEY_PARMS;

//-----------------------------------------------------------------
// section 4.25

typedef struct tdTCPA_PCR_SELECTION
{
   UINT16   sizeOfSelect;
#ifdef __midl
   [size_is(sizeOfSelect)]
#endif
   BYTE*    pcrSelect;
} TCPA_PCR_SELECTION;

typedef struct tdTCPA_PCR_COMPOSITE
{
   TCPA_PCR_SELECTION select;
   UINT32             valueSize;
#ifdef __midl
  [size_is(valueSize)]
#endif
   TCPA_PCRVALUE*      pcrValue;
} TCPA_PCR_COMPOSITE;

typedef struct tdTCPA_PCR_INFO
{
   TCPA_PCR_SELECTION  pcrSelection;
   TCPA_COMPOSITE_HASH digestAtRelease;
   TCPA_COMPOSITE_HASH digestAtCreation;
}  TCPA_PCR_INFO;

//-----------------------------------------------------------------
// section 4.26
typedef struct tdTCPA_STORED_DATA
```

```
{
   TCPA_VERSION  ver;
   UINT32   sealInfoSize;
#ifdef __midl
   [size_is(sealInfoSize)]
#endif
   BYTE*    sealInfo;
   UINT32   encDataSize;
#ifdef __midl
   [size_is(encDataSize)]
#endif
   BYTE*    encData;
} TCPA_STORED_DATA;

typedef struct tdTCPA_SEALED_DATA
{
   TCPA_PAYLOAD_TYPE  payload;
   TCPA_SECRET        authData;
   TCPA_NONCE         tpmProof;
   TCPA_DIGEST        storedDigest;
   UINT32             dataSize;
#ifdef __midl
   [size_is(dataSize)]
#endif
   BYTE*              data;
} TCPA_SEALED_DATA;

typedef struct tdTCPA_SYMMETRIC_KEY
{
   TCPA_ALGORITHM_ID  algId;
   TCPA_ENC_SCHEME    encScheme;
   UINT16             size;
#ifdef __midl
   [size_is(size)]
#endif
   BYTE*              data;
} TCPA_SYMMETRIC_KEY;
```

```
//----------------------------------------------------------------------
// section 4.27
typedef struct tdTCPA_STORE_PUBKEY
{
   UINT32   keyLength;
#ifdef __midl
   [size_is(keyLength)]
#endif
   BYTE*    key;
} TCPA_STORE_PUBKEY;

typedef struct tdTCPA_PUBKEY
{
   TCPA_KEY_PARMS     algorithmParms;
   TCPA_STORE_PUBKEY  pubKey;
} TCPA_PUBKEY;

typedef struct tdTCPA_STORE_PRIVKEY
{
   UINT32   keyLength;
#ifdef __midl
   [size_is(keyLength)]
#endif
   BYTE*    key;
} TCPA_STORE_PRIVKEY;

typedef struct tdTCPA_STORE_ASYMKEY
{
   TCPA_PAYLOAD_TYPE  payload;
   TCPA_SECRET        usageAuth;
   TCPA_SECRET        migrationAuth;
   TCPA_DIGEST        pubDataDigest;
   TCPA_STORE_PRIVKEY privKey;
} TCPA_STORE_ASYMKEY;

typedef struct tdTCPA_KEY
{
   TCPA_VERSION         ver;
```

```
   TCPA_KEY_USAGE       keyUsage;
   TCPA_KEY_FLAGS       keyFlags;
   TCPA_AUTH_DATA_USAGE authDataUsage;
   TCPA_KEY_PARMS       algorithmParms;
   UINT32               PCRInfoSize;
#ifdef __midl
   [size_is(PCRInfoSize)]
#endif
   BYTE*                PCRInfo;
   TCPA_STORE_PUBKEY    pubKey;
   UINT32               encSize;
#ifdef __midl
   [size_is(encSize)]
#endif
   BYTE*                encData;
} TCPA_KEY;

//---------------------------------------------------------------------
// section 4.28
typedef struct tdTCPA_CERTIFY_INFO
{
   TCPA_VERSION         version;
   TCPA_KEY_USAGE       keyUsage;
   TCPA_KEY_FLAGS       keyFlags;
   TCPA_AUTH_DATA_USAGE authDataUsage;
   TCPA_KEY_PARMS       algorithmParms;
   TCPA_DIGEST          pubkeyDigest;
   TCPA_NONCE           data;
   TSS_BOOL             parentPCRStatus;
   UINT32               PCRInfoSize;
#ifdef __midl
   [size_is(PCRInfoSize)]
#endif
   BYTE*                PCRInfo;
} TCPA_CERTIFY_INFO;

//---------------------------------------------------------------------
// section 4.23
```

```
typedef struct tdTCPA_MIGRATIONKEYAUTH
{
   TCPA_PUBKEY          migrationKey;
   TCPA_MIGRATE_SCHEME  migrationScheme;
   TCPA_DIGEST          digest;
} TCPA_MIGRATIONKEYAUTH;

//--------------------------------------------------------------------
// section 4.30.2
typedef struct tdTCPA_IDENTITY_REQ
{
   UINT32          asymSize;
   UINT32          symSize;
   TCPA_KEY_PARMS  asymAlgorithm;
   TCPA_KEY_PARMS  symAlgorithm;
#ifdef __midl
   [size_is(asymSize)]
#endif
   BYTE*           asymBlob;
#ifdef __midl
   [size_is(symSize)]
#endif
   BYTE*           symBlob;
} TCPA_IDENTITY_REQ;

// Errata: Where is TCPA_IDENTITY_CONTENTS?

//--------------------------------------------------------------------
// section 4.29
typedef struct tdTCPA_QUOTE_INFO
{
   TCPA_VERSION         version;
   BYTE                 fixed[4]; // Shall always be the ASCII string 'QUOT'
   TCPA_COMPOSITE_HASH  compositeHash;
   TCPA_NONCE           externalData;
} TCPA_QUOTE_INFO;

#endif // __TCPA_STRUCT_H__
```

+++tcpa_error.h

```
/*++

TPM error return codes basically extracted from TCPA Main Specification V1.1b

--*/

#ifndef __TCPA_ERROR_H__
#define __TCPA_ERROR_H__


////////////////////////////////////////////////////////////////////
// error codes

#ifndef TCPA_E_BASE
#define TCPA_E_BASE    0x00000000L
#endif

#ifndef TCPA_E_NON_FATAL
#define TCPA_E_NON_FATAL   0x00000800L
#endif


// Successful completion of the TCPA operation.
#define TCPA_SUCCESS    TCPA_E_BASE

//
// MessageId: TCPA_E_AUTHFAIL
//
// MessageText:
//
// Authentication failed.
//
#define TCPA_E_AUTHFAIL    (UINT32)(TCPA_E_BASE + 0x1)
```

```
//
// MessageId: TCPA_E_BADINDEX
//
// MessageText:
//
// The index to a PCR, DIR or other register is incorrect.
//
#define TCPA_E_BADINDEX    (UINT32)(TCPA_E_BASE + 0x2)

//
// MessageId: TCPA_E_BAD_PARAMETER
//
// MessageText:
//
// One or more TCPA command parameter is bad.
//
#define TCPA_E_BAD_PARAMETER   (UINT32)(TCPA_E_BASE + 0x3)

//
// MessageId: TCPA_E_AUDITFAILURE
//
// MessageText:
//
// An operation completed successfully but the auditing of that operation failed.
//
#define TCPA_E_AUDITFAILURE   (UINT32)(TCPA_E_BASE + 0x4)

//
// MessageId: TCPA_E_CLEAR_DISABLED
//
// MessageText:
//
// The clear disable flag is set and all clear operations now require physical access.
//
#define TCPA_E_CLEAR_DISABLED   (UINT32)(TCPA_E_BASE + 0x5)

//
// MessageId: TCPA_E_DEACTIVATED
```

```
//
// MessageText:
//
// The TCPA is deactivated.
//
#define TCPA_E_DEACTIVATED   (UINT32)(TCPA_E_BASE + 0x6)

//
// MessageId: TCPA_E_DISABLED
//
// MessageText:
//
// The TCPA is disabled.
//
#define TCPA_E_DISABLED    (UINT32)(TCPA_E_BASE + 0x7)

//
// MessageId: TCPA_E_DISABLED_CMD
//
// MessageText:
//
// The target TCPA command has been disabled.
//
#define TCPA_E_DISABLED_CMD   (UINT32)(TCPA_E_BASE + 0x8)

//
// MessageId: TCPA_E_FAIL
//
// MessageText:
//
// The TCPA operation failed.
//
#define TCPA_E_FAIL    (UINT32)(TCPA_E_BASE + 0x9)

//
// MessageId: TCPA_E_INACTIVE
//
// MessageText:
```

```
//
// The TCPA is inactive.
//
#define TCPA_E_INACTIVE    (UINT32)(TCPA_E_BASE + 0xA)

//
// MessageId: TCPA_E_INSTALL_DISABLED
//
// MessageText:
//
// The ability to install an owner is disabled.
//
#define TCPA_E_INSTALL_DISABLED  (UINT32)(TCPA_E_BASE + 0xB)

//
// MessageId: TCPA_E_INVALID_HANDLE
//
// MessageText:
//
// The TCPA key handle presented was invalid.
//
#define TCPA_E_INVALID_KEYHANDLE  (UINT32)(TCPA_E_BASE + 0xC)

//
// MessageId: TCPA_E_KEYNOTFOUND
//
// MessageText:
//
// The target key was not found in the TCPA.
//
#define TCPA_E_KEYNOTFOUND   (UINT32)(TCPA_E_BASE + 0xD)

//
// MessageId: TCPA_E_NEED_SELFTEST
//
// MessageText:
//
// The capability requires an untested function,
```

```
// additional self-test is required before the capability may execute.
//
#define TCPA_E_NEED_SELFTEST   (UINT32)(TCPA_E_BASE + 0xE)

//
// MessageId: TCPA_E_MIGRATEFAIL
//
// MessageText:
//
// Migration authorization failed.
//
#define TCPA_E_MIGRATEFAIL   (UINT32)(TCPA_E_BASE + 0xF)

//
// MessageId: TCPA_E_NO_PCR_INFO
//
// MessageText:
//
// A list of PCR values was not supplied.
//
#define TCPA_E_NO_PCR_INFO   (UINT32)(TCPA_E_BASE + 0x10)

//
// MessageId: TCPA_E_NOSPACE
//
// MessageText:
//
// No room in the TCPA to load a key.
//
#define TCPA_E_NOSPACE    (UINT32)(TCPA_E_BASE + 0x11)

//
// MessageId: TCPA_E_NOSRK
//
// MessageText:
//
// There is no SRK set.
//
```

```
#define TCPA_E_NOSRK    (UINT32)(TCPA_E_BASE + 0x12)

//
// MessageId: TCPA_E_NOTSEALED_BLOB
//
// MessageText:
//
// An encrypted blob is invalid or was not created by this TCPA.
//
#define TCPA_E_NOTSEALED_BLOB   (UINT32)(TCPA_E_BASE + 0x13)

//
// MessageId: TCPA_E_OWNER_SET
//
// MessageText:
//
// An Owner is already set in the TCPA.
//
#define TCPA_E_OWNER_SET   (UINT32)(TCPA_E_BASE + 0x14)

//
// MessageId: TCPA_E_RESOURCES
//
// MessageText:
//
// The TPM has insufficient internal resources to perform the requested action.
//
#define TCPA_E_RESOURCES   (UINT32)(TCPA_E_BASE + 0x15)

//
// MessageId: TCPA_E_SHORTRANDOM
//
// MessageText:
//
// A random string supplied to the TPM was too short.
//
#define TCPA_E_SHORTRANDOM   (UINT32)(TCPA_E_BASE + 0x16)
```

```
//
// MessageId: TCPA_E_SIZE
//
// MessageText:
//
// The TPM does not have the space to perform the operation.
//
#define TCPA_E_SIZE    (UINT32)(TCPA_E_BASE + 0x17)

//
// MessageId: TCPA_E_WRONGPCRVAL
//
// MessageText:
//
// The named PCR value does not match the current PCR value.
//
#define TCPA_E_WRONGPCRVAL   (UINT32)(TCPA_E_BASE + 0x18)

//
// MessageId: TCPA_E_BUSY
//
// MessageText:
//
// The TPM is too busy to respond to the command.
//
//#define TCPA_E_BUSY    (UINT32)(TCPA_E_BASE + 0x19)

//
// MessageId: TCPA_E_BAD_PARAM_SIZE
//
// MessageText:
//
// The paramSize argument to the command has the incorrect value
//
#define TCPA_E_BAD_PARAM_SIZE   (UINT32)(TCPA_E_BASE + 0x19)

//
// MessageId: TCPA_E_SHA_THREAD
```

```
//
// MessageText:
//
// There is no existing SHA-1 thread in the TPM.
//
#define TCPA_E_SHA_THREAD   (UINT32)(TCPA_E_BASE + 0x1A)

//
// MessageId: TCPA_E_SHA_ERROR
//
// MessageText:
//
// The calculation is unable to proceed because the existing SHA-1
// thread has already encountered an error.
//
#define TCPA_E_SHA_ERROR   (UINT32)(TCPA_E_BASE + 0x1B)

//
// MessageId: TCPA_E_FAILEDSELFTEST
//
// MessageText:
//
// Self-test has failed and the TPM has shutdown.
//
#define TCPA_E_FAILEDSELFTEST   (UINT32)(TCPA_E_BASE + 0x1C)

//
// MessageId: TCPA_E_AUTH2FAIL
//
// MessageText:
//
// The authorization for the second key in a 2 key function failed authorization.
//
#define TCPA_E_AUTH2FAIL   (UINT32)(TCPA_E_BASE + 0x1D)

//
// MessageId: TCPA_E_BADTAG
//
```

```
// MessageText:
//
// The tag value sent to the TPM for a command is invalid.
//
#define TCPA_E_BADTAG    (UINT32)(TCPA_E_BASE + 0x1E)

//
// MessageId: TCPA_E_IOERROR
//
// MessageText:
//
//  An IO error occurred transmitting information to the TPM.
//
#define TCPA_E_IOERROR    (UINT32)(TCPA_E_BASE + 0x1F)

//
// MessageId: TCPA_E_ENCRYPT_ERROR
//
// MessageText:
//
// The TPM encryption process had a problem.
//
#define TCPA_E_ENCRYPT_ERROR   (UINT32)(TCPA_E_BASE + 0x20)

//
// MessageId: TCPA_E_DECRYPT_ERROR
//
// MessageText:
//
// The TPM decryption process did not complete.
//
#define TCPA_E_DECRYPT_ERROR   (UINT32)(TCPA_E_BASE + 0x21)

//
// MessageId: TCPA_E_INVALID_AUTHHANDLE
//
// MessageText:
//
```

```
// The TPM auth handle was invalid.
//
#define TCPA_E_INVALID_AUTHHANDLE  (UINT32)(TCPA_E_BASE + 0x22)

//
// MessageId: TCPA_E_NO_ENDORSEMENT
//
// MessageText:
//
// The TPM does not have an Endorsement Key installed.
//
#define TCPA_E_NO_ENDORSEMENT   (UINT32)(TCPA_E_BASE + 0x23)

//
// MessageId: TCPA_E_INVALID_KEYUSAGE
//
// MessageText:
//
// The usage of a key is not allowed.
//
#define TCPA_E_INVALID_KEYUSAGE  (UINT32)(TCPA_E_BASE + 0x24)

//
// MessageId: TCPA_E_WRONG_ENTITYTYPE
//
// MessageText:
//
//  The submitted entity type is not allowed.
//
#define TCPA_E_WRONG_ENTITYTYPE  (UINT32)(TCPA_E_BASE + 0x25)

//
// MessageId: TCPA_INVALID_POSTINIT
//
// MessageText:
//
// The command was received in the wrong sequence relative to TPM_Init and a subsequent TPM_Startup.
//
```

```
#define TCPA_E_INVALID_POSTINIT  (UINT32)(TCPA_E_BASE + 0x26)

//
// MessageId: TCPA_E_INAPPROPRIATE_SIG
//
// MessageText:
//
// Signed data cannot include additional DER information.
//
#define TCPA_E_INAPPROPRIATE_SIG  (UINT32)(TCPA_E_BASE + 0x27)

//
// MessageId: TCPA_E_BAD_KEY_PROPERTY
//
// MessageText:
//
//  The key properties in TCPA_KEY_PARMs are not supported by this TPM.
//
#define TCPA_E_BAD_KEY_PROPERTY  (UINT32)(TCPA_E_BASE + 0x28)

//
// MessageId: TCPA_E_BAD_MIGRATION
//
// MessageText:
//
//  The migration properties of this key are incorrect.
//
#define TCPA_E_BAD_MIGRATION   (UINT32)(TCPA_E_BASE + 0x29)

//
// MessageId: TCPA_E_BAD_SCHEME
//
// MessageText:
//
// The signature or encryption scheme for this key is incorrect or not permitted in this situation.
//
#define TCPA_E_BAD_SCHEME   (UINT32)(TCPA_E_BASE + 0x2A)
```

```
//
// MessageId: TCPA_E_BAD_DATASIZE
//
// MessageText:
//
//  The size of the data (or blob) parameter is bad or inconsistent with the referenced key.
//
#define TCPA_E_BAD_DATASIZE   (UINT32)(TCPA_E_BASE + 0x2B)

//
// MessageId: TCPA_E_BAD_MODE
//
// MessageText:
//
// A mode parameter is bad, such as capArea or subCapArea for TPM_GetCapability,
// phsicalPresence parameter for TPM_PhysicalPresence,
// or migrationType for TPM_CreateMigrationBlob.
//
#define TCPA_E_BAD_MODE    (UINT32)(TCPA_E_BASE + 0x2C)

//
// MessageId: TCPA_E_BAD_PRESENCE
//
// MessageText:
//
// Either the physicalPresence or physicalPresenceLock bits have the wrong value.
//
#define TCPA_E_BAD_PRESENCE   (UINT32)(TCPA_E_BASE + 0x2D)

//
// MessageId: TCPA_E_BAD_VERSION
//
// MessageText:
//
// The TPM cannot perform this version of the capability.
//
#define TCPA_E_BAD_VERSION   (UINT32)(TCPA_E_BASE + 0x2E)
```

```
//////////////////////////////////////////////////////////////////////
// non fatal errors

//
// MessageId: TCPA_E_RETRY
//
// MessageText:
//
// The TPM is too busy to respond to the command immediately,
// but the command could be resubmitted at a later time.
//
#define TCPA_E_RETRY  (UINT32)(TCPA_E_BASE + TCPA_E_NON_FATAL)

#endif // __TCPA_ERROR_H__
```

+++tss_error.h

```
/*++

TSS error return codes

--*/

#ifndef __TSS_ERROR_H__
#define __TSS_ERROR_H__

//
// error coding scheme for a Microsoft Windows platform -
// refer to the TSS Specification Parts
//
//  Values are 32 bit values layed out as follows:
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//  +---+-+-+-----------------------+-------+-----------------------+
//  |Lev|C|R|     Facility          | Layer |         Code          |
//  +---+-+-+-----------------------+-------+-----------------------+
```

```
// | Platform specific coding      | TSS error coding system       |
//  +---+-+-+-----------------------+-------+-----------------------+
//
//      Lev - is the Level code
//
//          00 - Success
//          01 - Informational
//          10 - Warning
//          11 - Error
//
//      C - is the Customer code flag  (must actually be set)
//
//      R - is a reserved bit    (unused)
//
//      Facility - is the facility code: TCPA: proposal 0x028
//
//      Code - is the facility's status code
//

//
// definitions for the code level information
//
#define TSS_LEVEL_SUCCESS  0x00000000L    // code level success
#define TSS_LEVEL_INFO     0x40000000L    // code level information
#define TSS_LEVEL_WARNING  0x80000000L    // code level warning
#define TSS_LEVEL_ERROR    0xC0000000L    // code level error

//
// some defines for the platform specific information
//
#define FACILITY_TSS            0x28L     // facility number for TCPA return codes
#define FACILITY_TSS_CODEPOS   (FACILITY_TSS << 16)  // shift the facility info to the code
                                                                    // position

#define TSS_CUSTOM_CODEFLAG     0x20000000L    // bit position for the custom flag in
                                                                    // return code

//
```

```
//
// TSS error return codes
//
//
#ifndef TSS_E_BASE
#define TSS_E_BASE    0x00000000L
#endif // TSS_E_BASE
#ifndef TSS_W_BASE
#define TSS_W_BASE    0x00000000L
#endif // TSS_W_BASE
#ifndef TSS_I_BASE
#define TSS_I_BASE    0x00000000L
#endif // TSS_I_BASE

//
// basic error return codes common to all TSS Service Provider Interface methods
// and returned by all TSS SW stack components
//

//
// MessageId: TSS_SUCCESS
//
// MessageText:
//
//  Successful completion of the operation.
//
#define TSS_SUCCESS     0x00000000L

//
// MessageId: TSS_E_FAIL
//
// MessageText:
//
//  An internal error has been detected, but the source is unknown.
//
#define TSS_E_FAIL     (UINT32)(TSS_E_BASE + 0x002L)

//
```

```
// MessageId: TSS_E_BAD_PARAMETER
//
// MessageText:
//
// One or more parameter is bad.
//
#define TSS_E_BAD_PARAMETER    (UINT32)(TSS_E_BASE + 0x003L)

//
// MessageId: TSS_E_INTERNAL_ERROR
//
// MessageText:
//
//  An internal SW error has been detected.
//
#define TSS_E_INTERNAL_ERROR    (UINT32)(TSS_E_BASE + 0x004L)

//
// MessageId: TSS_E_OUTOFMEMORY
//
// MessageText:
//
// Ran out of memory.
//
#define TSS_E_OUTOFMEMORY    (UINT32)(TSS_E_BASE + 0x005L)

//
// MessageId: TSS_E_NOTIMPL
//
// MessageText:
//
// Not implemented.
//
#define TSS_E_NOTIMPL     (UINT32)(TSS_E_BASE + 0x006L)

//
// MessageId: TSS_E_KEY_ALREADY_REGISTERED
//
```

```
// MessageText:
//
//  Key is already registered
//
#define TSS_E_KEY_ALREADY_REGISTERED  (UINT32)(TSS_E_BASE + 0x008L)


//
// MessageId: TSS_E_TPM_UNEXPECTED
//
// MessageText:
//
//  An unexpected TPM error has occurred.
//
#define TSS_E_TPM_UNEXPECTED    (UINT32)(TSS_E_BASE + 0x010L)

//
// MessageId: TSS_E_COMM_FAILURE
//
// MessageText:
//
//  A communications error with the TPM has been detected.
//
#define TSS_E_COMM_FAILURE    (UINT32)(TSS_E_BASE + 0x011L)

//
// MessageId: TSS_E_TIMEOUT
//
// MessageText:
//
//  The operation has timed out.
//
#define TSS_E_TIMEOUT     (UINT32)(TSS_E_BASE + 0x012L)

//
// MessageId: TSS_E_TPM_UNSUPPORTED_FEATURE
//
// MessageText:
```

```
//
//  The TPM does not support the requested feature.
//
#define TSS_E_TPM_UNSUPPORTED_FEATURE  (UINT32)(TSS_E_BASE + 0x014L)

//
// MessageId: TSS_E_CANCELED
//
// MessageText:
//
//  The action was canceled by request.
//
#define TSS_E_CANCELED     (UINT32)(TSS_E_BASE + 0x016L)

//
// MessageId: TSS_E_PS_KEY_NOTFOUND
//
// MessageText:
//
// The key cannot be found in the persistent storage database.
//
#define TSS_E_PS_KEY_NOTFOUND    (UINT32)(TSS_E_BASE + 0x020L)
//
// MessageId: TSS_E_PS_KEY_EXISTS
//
// MessageText:
//
// The key already exists in the persistent storage database.
//
#define TSS_E_PS_KEY_EXISTS            (UINT32)(TSS_E_BASE + 0x021L)

//
// MessageId: TSS_E_PS_BAD_KEY_STATE
//
// MessageText:
//
// The key data set not valid in the persistent storage database.
//
```

```
#define TSS_E_PS_BAD_KEY_STATE          (UINT32)(TSS_E_BASE + 0x022L)


//
// error codes returned by specific TSS Service Provider Interface methods
// offset TSS_TSPI_OFFSET
//

//
// MessageId: TSS_E_INVALID_OBJECT_TYPE
//
// MessageText:
//
// Object type not valid for this operation.
//
#define TSS_E_INVALID_OBJECT_TYPE   (UINT32)(TSS_E_BASE + 0x101L)

//
// MessageId: TSS_E_NO_CONNECTION
//
// MessageText:
//
// Core Service connection doesn't exist.
//
#define TSS_E_NO_CONNECTION    (UINT32)(TSS_E_BASE + 0x102L)

//
// MessageId: TSS_E_CONNECTION_FAILED
//
// MessageText:
//
// Core Service connection failed.
//
#define TSS_E_CONNECTION_FAILED   (UINT32)(TSS_E_BASE + 0x103L)

//
// MessageId: TSS_E_CONNECTION_BROKEN
//
```

```
// MessageText:
//
// Communication with Core Service failed.
//
#define TSS_E_CONNECTION_BROKEN   (UINT32)(TSS_E_BASE + 0x104L)

//
// MessageId: TSS_E_HASH_INVALID_ALG
//
// MessageText:
//
// Invalid hash algorithm.
//
#define TSS_E_HASH_INVALID_ALG   (UINT32)(TSS_E_BASE + 0x105L)

//
// MessageId: TSS_E_HASH_INVALID_LENGTH
//
// MessageText:
//
// Hash length is inconsistent with hash algorithm.
//
#define TSS_E_HASH_INVALID_LENGTH   (UINT32)(TSS_E_BASE + 0x106L)

//
// MessageId: TSS_E_HASH_NO_DATA
//
// MessageText:
//
// Hash object has no internal hash value.
//
#define TSS_E_HASH_NO_DATA    (UINT32)(TSS_E_BASE + 0x107L)


//
// MessageId: TSS_E_INVALID_ATTRIB_FLAG
//
// MessageText:
```

```
//
// Flag value for attrib-functions inconsistent.
//
#define TSS_E_INVALID_ATTRIB_FLAG   (UINT32)(TSS_E_BASE + 0x109L)

//
// MessageId: TSS_E_INVALID_ATTRIB_SUBFLAG
//
// MessageText:
//
// Subflag value for attrib-functions inconsistent.
//
#define TSS_E_INVALID_ATTRIB_SUBFLAG  (UINT32)(TSS_E_BASE + 0x10AL)

//
// MessageId: TSS_E_INVALID_ATTRIB_DATA
//
// MessageText:
//
// Data for attrib-functions invalid.
//
#define TSS_E_INVALID_ATTRIB_DATA   (UINT32)(TSS_E_BASE + 0x10BL)

//
// MessageId: TSS_E_INVALID_OBJECT_INITFLAG
//
// MessageText:
//
// Wrong flag information for object creation.
//
#define TSS_E_INVALID_OBJECT_INITFLAG  (UINT32)(TSS_E_BASE + 0x10CL)

//
// MessageId: TSS_E_NO_PCRS_SET
//
// MessageText:
//
// No PCR register are selected or set.
```

```
//
#define TSS_E_NO_PCRS_SET    (UINT32)(TSS_E_BASE + 0x10DL)

//
// MessageId: TSS_E_KEY_NOT_LOADED
//
// MessageText:
//
// The addressed key is currently not loaded.
//
#define TSS_E_KEY_NOT_LOADED    (UINT32)(TSS_E_BASE + 0x10EL)

//
// MessageId: TSS_E_KEY_NOT_SET
//
// MessageText:
//
// No key information is currently available.
//
#define TSS_E_KEY_NOT_SET    (UINT32)(TSS_E_BASE + 0x10FL)

//
// MessageId: TSS_E_VALIDATION_FAILED
//
// MessageText:
//
// Internal validation of data failed.
//
#define TSS_E_VALIDATION_FAILED   (UINT32)(TSS_E_BASE + 0x110L)

//
// MessageId: TSS_E_TSP_AUTHREQUIRED
//
// MessageText:
//
// Authorization is required.
//
#define TSS_E_TSP_AUTHREQUIRED   (UINT32)(TSS_E_BASE + 0x111L)
```

```
//
// MessageId: TSS_E_TSP_AUTH2REQUIRED
//
// MessageText:
//
// Multiple authorization is required.
//
#define TSS_E_TSP_AUTH2REQUIRED   (UINT32)(TSS_E_BASE + 0x112L)

//
// MessageId: TSS_E_TSP_AUTHFAIL
//
// MessageText:
//
// Authorization failed.
//
#define TSS_E_TSP_AUTHFAIL    (UINT32)(TSS_E_BASE + 0x113L)

//
// MessageId: TSS_E_TSP_AUTH2FAIL
//
// MessageText:
//
// Multiple authorization failed.
//
#define TSS_E_TSP_AUTH2FAIL    (UINT32)(TSS_E_BASE + 0x114L)

//
// MessageId: TSS_E_KEY_NO_MIGRATION_POLICY
//
// MessageText:
//
// There's no migration policy object set for the addressed key.
//
#define TSS_E_KEY_NO_MIGRATION_POLICY  (UINT32)(TSS_E_BASE + 0x115L)

//
```

```
// MessageId: TSS_E_POLICY_NO_SECRET
//
// MessageText:
//
// No secret information is currently available for the addressed policy object.
//
#define TSS_E_POLICY_NO_SECRET   (UINT32)(TSS_E_BASE + 0x116L)

//
// MessageId: TSS_E_INVALID_OBJ_ACCESS
//
// MessageText:
//
// The operation failed due to an invalid object status.
//
#define TSS_E_INVALID_OBJ_ACCESS   (UINT32)(TSS_E_BASE + 0x117L)

//
// MessageId: TSS_E_INVALID_ENCSCHEME
//
// MessageText:
//
//
//
#define TSS_E_INVALID_ENCSCHEME   (UINT32)(TSS_E_BASE + 0x118L)


//
// MessageId: TSS_E_INVALID_SIGSCHEME
//
// MessageText:
//
//
//
#define TSS_E_INVALID_SIGSCHEME   (UINT32)(TSS_E_BASE + 0x119L)

//
// MessageId: TSS_E_ENC_INVALID_LENGTH
```

```
//
// MessageText:
//
//
//
#define TSS_E_ENC_INVALID_LENGTH   (UINT32)(TSS_E_BASE + 0x120L)


//
// MessageId: TSS_E_ENC_NO_DATA
//
// MessageText:
//
//
//
#define TSS_E_ENC_NO_DATA    (UINT32)(TSS_E_BASE + 0x121L)

//
// MessageId: TSS_E_ENC_INVALID_TYPE
//
// MessageText:
//
//
//
#define TSS_E_ENC_INVALID_TYPE   (UINT32)(TSS_E_BASE + 0x122L)


//
// MessageId: TSS_E_INVALID_KEYUSAGE
//
// MessageText:
//
//
//
#define TSS_E_INVALID_KEYUSAGE   (UINT32)(TSS_E_BASE + 0x123L)

//
// MessageId: TSS_E_VERIFICATION_FAILED
```

```
//
// MessageText:
//
//
//
#define TSS_E_VERIFICATION_FAILED   (UINT32)(TSS_E_BASE + 0x124L)

//
// MessageId: TSS_E_HASH_NO_IDENTIFIER
//
// MessageText:
//
// Hash algorithm identifier not set.
//
#define TSS_E_HASH_NO_IDENTIFIER   (UINT32)(TSS_E_BASE + 0x125L)

//
// MessageId: TSS_E_INVALID_HANDLE
//
// MessageText:
//
//  An invalid handle
//
#define TSS_E_INVALID_HANDLE    (UINT32)(TSS_E_BASE + 0x126L)

//
// MessageId: TSS_E_SILENT_CONTEXT
//
// MessageText:
//
//  A silent context requires user input
//
#define TSS_E_SILENT_CONTEXT           (UINT32)(TSS_E_BASE + 0x127L)

//
// MessageId: TSS_E_EK_CHECKSUM
//
// MessageText:
```

```
//
// TSP is instructed to verify the EK checksum and it does not verify.
//
#define TSS_E_EK_CHECKSUM             (UINT32)(TSS_E_BASE + 0x128L)



#endif // __TSS_ERROR_H__
```

+++tss_error_basics.h

```
/*++

Basic defines for TSS error return codes

--*/

#ifndef __TSS_ERROR_BASICS_H__
#define __TSS_ERROR_BASICS_H__


//
// definitions for the various TSS-SW layers
//
#ifndef TSS_LAYER_TPM
#define TSS_LAYER_TPM   0x0000L     // definition for TPM layer
#endif // TSS_LAYER_TPM

#define TSS_LAYER_TDDL   0x1000L     // definition for TDDL layer
#define TSS_LAYER_TCS   0x2000L     // definition for TCS layer

#ifndef TSS_LAYER_TSP
#define TSS_LAYER_TSP   0x3000L     // definition for TSP layer
#endif // TSS_LAYER_TSP
```

```
//
// definitions for the start points of layer specific error codes
//
#ifndef TSS_COMMON_OFFSET
#define TSS_COMMON_OFFSET   0x000L
#endif // TSS_COMMON_OFFSET

#define TSS_TDDL_OFFSET    0x080L
#define TSS_TCSI_OFFSET    0x0C0L

#ifndef TSS_TSPI_OFFSET
#define TSS_TSPI_OFFSET    0x100L
#endif // TSS_TSPI_OFFSET

#ifndef TSS_VENDOR_OFFSET
#define TSS_VENDOR_OFFSET   0x800L
#endif // TSS_VENDOR_OFFSET

// do not exceed TSS_MAX_ERROR for vendor specific code values:
#ifndef TSS_MAX_ERROR
#define TSS_MAX_ERROR    0xFFFL
#endif // TSS_MAX_ERROR

#endif // __TSS_ERROR_BASICS_H__
```

+++tss_defines.h

```
/*++

Global defines for TSS.

--*/

#ifndef __TSS_DEFINES_H__
#define __TSS_DEFINES_H__

typedef UINT32  TSS_HASHHANDLE;        // handle to a hash session
```

```
typedef UINT32  TSS_HMACHANDLE;        // handle to a HMAC session

//
// definition of the object types that can be created via CreateObject
//
#define   TSS_OBJECT_TYPE_POLICY      (0x01)      // Policy object
#define   TSS_OBJECT_TYPE_RSAKEY      (0x02)      // RSA-Key object
#define   TSS_OBJECT_TYPE_ENCDATA      (0x03)      // Encrypted data object
#define   TSS_OBJECT_TYPE_PCRS      (0x04)      // PCR composite object
#define   TSS_OBJECT_TYPE_HASH      (0x05)      // Hash object
//
////////////////////////////////////////////////////////////////////////
// CreateObject: Flags
////////////////////////////////////////////////////////////////////////
//
// for RSAKEY object:
//
// Authorization:
//
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//   -----------------------------------------------------------------
// Authorization:
//   Never                              |0 0|
//   Always                            |0 1|
//
#define   TSS_KEY_NO_AUTHORIZATION   (0x00000000)   // no authorization for this key
#define   TSS_KEY_AUTHORIZATION      (0x00000001)   // key needs authorization
//
// Volatility
//
//   Non Volatile                                          |0|
//   Volatile                                              |1|
//
#define    TSS_KEY_NON_VOLATILE           (0x00000000)    // Key is non-volatile
#define    TSS_KEY_VOLATILE               (0x00000004)    // Key is volatile
//
```

```
// Migration:
//
//   Non Migratable                                          |0|
//   Migratable                                              |1|
//
#define   TSS_KEY_NOT_MIGRATABLE      (0x00000000)   // key is not migratable
#define   TSS_KEY_MIGRATABLE      (0x00000008)   // key is migratable
//
// Usage:
//
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//   ---------------------------------------------------------------
// Usage:
//    Default (Legacy)                                 |0 0 0 0|
//   Signing                                          |0 0 0 1|
//   Storage                                          |0 0 1 0|
//   Identity                                         |0 0 1 0|
//   AuthChange                                       |0 1 0 0|
//   Bind                                             |0 1 0 1|
//   Legacy                                           |0 1 1 0|
//
//
#define   TSS_KEY_TYPE_DEFAULT      (0x00000000)   // indicate a default key (Legacy-Key)
#define   TSS_KEY_TYPE_SIGNING      (0x00000010)   // indicate a signing key
#define   TSS_KEY_TYPE_STORAGE      (0x00000020)   // used as storage key
#define   TSS_KEY_TYPE_IDENTITY      (0x00000030)   // indicate an idendity key
#define   TSS_KEY_TYPE_AUTHCHANGE      (0x00000040)   // indicate an ephemeral key
#define   TSS_KEY_TYPE_BIND         (0x00000050)   // indicate a key for TPM_Bind
#define   TSS_KEY_TYPE_LEGACY      (0x00000060)   // indicate a key that can perfom signing
//                                                                    and binding
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//   ---------------------------------------------------------------
// Size:
//   512                                    |0 0 0 1|
```

```
//  1024                                    |0 0 1 0|
//  2048                                    |0 0 1 1|
//  4096                                    |0 1 0 0|
//  8192                                    |0 1 0 1|
// 16286                                    |0 1 1 0|
//
#define TSS_KEY_SIZE_512    UINT32( 0x00000100 ) // indicate a key with 512 bit
#define TSS_KEY_SIZE_1024   UINT32( 0x00000200 ) // indicate a key with 1024 bit
#define TSS_KEY_SIZE_2048   UINT32( 0x00000300 ) // indicate a key with 2048 bit
#define TSS_KEY_SIZE_4096   UINT32( 0x00000400 ) // indicate a key with 4096 bit
#define TSS_KEY_SIZE_8192   UINT32( 0x00000500 ) // indicate a key with 8192 bit
#define TSS_KEY_SIZE_16384  UINT32( 0x00000600 ) // indicate a key with 16286 bit
//
// fixed KeyTypes (templates)
//
//
//                      3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//                      1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//   ------------------------------------------------------------------------------
//    Reserved:                     |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0|
//   Empty Key        |0 0 0 0 0 0|
//   Storage root key |0 0 0 0 0 1|
//
#define   TSS_KEY_EMPTY_KEY         (0x00000000)   // no TCPA key template (empty TSP key
//                                                                          object)
#define   TSS_KEY_TSP_SRK         (0x04000000)   // use a TCPA SRK template (TSP key object
//                                                                          for SRK)
//
// Flags for ENCDATA:
//
// Type:
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//   ---------------------------------------------------------------
// Type:
//   Seal                            |0 0 1|
//   Bind                            |0 1 0|
```

```
//   Legacy                          |0 1 1|
//
// ENCDATA Reserved:
//  |x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x|
//
#define   TSS_ENCDATA_SEAL         (0x00000001)   // data for seal operation
#define   TSS_ENCDATA_BIND         (0x00000002)   // data for bind operation
#define   TSS_ENCDATA_LEGACY      (0x00000003)   // data for legacy bind operation
//
//
// Flags for POLICY:
//
// Type:
//
//
// Flags for POLICY:
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//   ---------------------------------------------------------------
// Type:
//   Usage                                                    |0 1|
//   Migration                                                |1 0|
//
// POLICY Reserved:
//  |x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x|

#define   TSS_POLICY_USAGE         (0x00000001)   // usage policy object
#define   TSS_POLICY_MIGRATION      (0x00000002)   // migration policy object
//
//
// Flags for HASH:
//
//
// Flags for HASH:
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
```

```
//   ---------------------------------------------------------------
// Algorithm:
//   DEFAULT
//  |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0|
//   SHA1
//  |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1|
//   OTHER
//  |1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1|
//
#define   TSS_HASH_DEFAULT         (0x00000000)   // Default hash algorithm
#define   TSS_HASH_SHA1          (0x00000001)   // Sha1 with 20 bytes
#define   TSS_HASH_OTHER          (0xFFFFFFFF)   // Not specified hash algorithm
//
//////////////////////////////////////////////////////////////////////////
// SetAttribField and GetAttribField: Flags
//////////////////////////////////////////////////////////////////////////
//
// Object Context:
//
//        3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
//        1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//        ---------------------------------------------------------------
//   TSS_TSPATTRIB_CONTEXT_SILENT_MODE                                |0 0 1|
//   TSS_TSPATTRIB_CONTEXT_MACHINE_NAME                               |0 1 0|
//
#define TSS_TSPATTRIB_CONTEXT_SILENT_MODE      (0x00000001)   // TSP dialog display control
#define TSS_TSPATTRIB_CONTEXT_MACHINE_NAME   (0x00000002)
//
// Object Policy:
//
//                  3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
//                  1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//                  ---------------------------------------------------------------
//   TSS_TSPATTRIB_POLICY_CALLBACK_HMAC                          |0 0 1|
//   TSS_TSPATTRIB_POLICY_CALLBACK_XOR_ENC                       |0 1 0|
//   TSS_TSPATTRIB_POLICY_CALLBACK_TAKEOWNERSHIP                 |0 1 1|
//   TSS_TSPATTRIB_POLICY_CALLBACK_CHANGEAUTHASYM                |1 0 0|
//   TSS_TSPATTRIB_POLICY_SECRET_LIFETIME                        |1 0 1|
```

```
//   TSS_TSPATTRIB_POLICY_POPUPSTRING                        |1 1 0|
//
#define TSS_TSPATTRIB_POLICY_CALLBACK_HMAC         (0x00000080)   // enable/disable callback
//                                                                                  function
#define TSS_TSPATTRIB_POLICY_CALLBACK_XOR_ENC         (0x00000100)   // enable/disable callback
//                                                                                  function
#define TSS_TSPATTRIB_POLICY_CALLBACK_TAKEOWNERSHIP      (0x00000180)   // enable/disable callback
//                                                                                  function
#define TSS_TSPATTRIB_POLICY_CALLBACK_CHANGEAUTHASYM      (0x00000200)   // enable/disable callback
//                                                                                  function
#define TSS_TSPATTRIB_POLICY_SECRET_LIFETIME         (0x00000280)   // set lifetime mode for
//                                                                                  policy secret
#define TSS_TSPATTRIB_POLICY_POPUPSTRING            (0x00000300)   // set a NULL terminated
//                                                                                  UNICODE string which is
displayed
//                                                                                  in the TSP policy popup
dialog
//
//   Definition of policy mode flags that can be used with the method Tspi_Policy_SetSecret( )
//
//   TSS_SECRET_MODE_NONE                              |0 0 0 1|
//   TSS_SECRET_MODE_SHA1                              |0 0 1 0|
//   TSS_SECRET_MODE_PLAIN                             |0 0 1 1|
//   TSS_SECRET_MODE_POPUP                             |0 1 0 0|
//   TSS_SECRET_MODE_CALLBACK                          |0 1 0 1|
//
#define TSS_SECRET_MODE_NONE      (0x00000800)      // No authorization will be processed
#define TSS_SECRET_MODE_SHA1      (0x00001000)      // Secret string will not be touched by TSP
#define TSS_SECRET_MODE_PLAIN      (0x00001800)      // Secret string will be hashed using SHA1
#define TSS_SECRET_MODE_POPUP      (0x00002000)      // TSS SP will ask for a secret
#define TSS_SECRET_MODE_CALLBACK   (0x00002800)      // Application has to provide a call back
//                                                                                  function
//
////////////////////////////////////////////////////////////////////////
// SetAttribField and GetAttribField: SubFlags
////////////////////////////////////////////////////////////////////////
//
// SubFlags for Flag TSS_TSPATTRIB_POLICY_SECRET_LIFETIME
```

```
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//   ---------------------------------------------------------------
// SubFlags for Flag TSS_TSPATTRIB_POLICY_SECRET_LIFETIME
//
//   TSS_TSPATTRIB_POLICYSECRET_LIFETIME_ALWAYS            |0 0 0 1|
//   TSS_TSPATTRIB_POLICYSECRET_LIFETIME_COUNTER           |0 0 1 0|
//   TSS_TSPATTRIB_POLICYSECRET_LIFETIME_TIMER             |0 0 1 1|
//
#define   TSS_SECRET_LIFETIME_ALWAYS   (0x00000001)   // secret will not be invalidated
#define   TSS_SECRET_LIFETIME_COUNTER   (0x00000002)   // secret lifetime controled be counter
#define   TSS_SECRET_LIFETIME_TIMER   (0x00000003)   // secret lifetime controled be time
//
//////////////////////////////////////////////////////////////////////////
// SetAttribField and GetAttribField: Attrib
//////////////////////////////////////////////////////////////////////////
//
// for Flag TSS_TSPATTRIB_CONTEXT_SILENT_MODE
//
#define   TSS_TSPATTRIB_CONTEXT_NOT_SILENT   (0x00000000)   // TSP dialogs enabled
#define   TSS_TSPATTRIB_CONTEXT_SILENT      (0x00000001)   // TSP dialogs disabled
//
// Object EncData:
//
#define TSS_TSPATTRIB_ENCDATA_BLOB         (0x00000008)   // data blob for seal or bind
#define TSS_TSPATTRIB_ENCDATA_PCR             (0x00000010)
//
// Object Key:
//
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//   ---------------------------------------------------------------
//
// Object Key:
//   TSS_TSPATTRIB_KEY_BLOB                     |0 0 0 1|
//   TSS_TSPATTRIB_KEY_PARAM                    |0 0 1 0|
```

```
//   TSS_TSPATTRIB_KEY_GUID                    |0 0 1 1|
//   TSS_TSPATTRIB_KEY_PCR                     |0 1 0 0|
//   TSS_TSPATTRIB_RSAKEY_INFO                 |0 1 0 1|
//   TSS_TSPATTRIB_KEY_REGISTER                |0 1 1 0|
//
#define TSS_TSPATTRIB_KEY_BLOB      (0x00000040)   // key info as blob data
#define TSS_TSPATTRIB_KEY_INFO      (0x00000080)   // key param info as blob data
#define TSS_TSPATTRIB_KEY_UUID      (0x000000C0)   // key GUID info as blob data
#define TSS_TSPATTRIB_KEY_PCR         (0x00000100)   // composite digest value for the key
#define TSS_TSPATTRIB_RSAKEY_INFO      (0x00000140)   // public exponent of the key
#define TSS_TSPATTRIB_KEY_REGISTER      (0x00000180)   // register location for the key data
//
// Object Hash:
//
#define TSS_TSPATTRIB_HASH_IDENTIFIER   (0x00001000)   // Hash algorithm identifier
//
// SubFlags for Flag TSS_TSPATTRIB_ENCDATA_BLOB
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//   ---------------------------------------------------------------
// SubFlags for Flag TSS_TSPATTRIB_ENCDATA_BLOB
//
//   TSS_TSPATTRIB_ENCDATABLOB_BLOB                          |0 0 1|
//   TSS_TSPATTRIB_ENCDATAPCR_DIGEST_ATCREATION              |0 1 0|
//
#define TSS_TSPATTRIB_ENCDATABLOB_BLOB         (0x00000001)   // encrypted data blob
#define TSS_TSPATTRIB_ENCDATAPCR_DIGEST_ATCREATION    (0x00000002)   // PCR digest at creation
#define TSS_TSPATTRIB_ENCDATAPCR_DIGEST_RELEASE      (0x00000003)
#define TSS_TSPATTRIB_ENCDATAPCR_SELECTION      (0x00000004)
//
// SubFlags for Flag TSS_TSPATTRIB_KEY_BLOB
//
//
//     TSS_TSPATTRIB_KEYBLOB_BLOB                    |0 0 0 1|
//    TSS_TSPATTRIB_KEYBLOB_PUBLIC_KEY              |0 0 1 0|
//    TSS_TSPATTRIB_KEYBLOB_PLAIN                   |0 0 1 1|
//    TSS_TSPATTRIB_KEYBLOB_GUID                    |0 1 0 0|
```

```
//    TSS_TSPATTRIB_KEYBLOB_PRIVATE_KEY             |0 1 0 1|
//
#define TSS_TSPATTRIB_KEYBLOB_BLOB         (0x00000008)   // key info using the key blob
#define TSS_TSPATTRIB_KEYBLOB_PUBLIC_KEY      (0x00000010)   // public key info using the blob
#define TSS_TSPATTRIB_KEYBLOB_PRIVATE_KEY      (0x00000028)   // encrypted private key blob
//
// SubFlags for Flag TSS_TSPATTRIB_KEY_INFO
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//   ---------------------------------------------------------------
//
//    TSS_TSPATTRIB_KEYINFO_SIZ           |0 0 0 0 1|
//    TSS_TSPATTRIB_KEYINFO_USAGE         |0 0 0 1 0|
//    TSS_TSPATTRIB_KEYINFO_KEYFLAGS      |0 0 0 1 1|
//    TSS_TSPATTRIB_KEYINFO_AUTHUSAGE     |0 0 1 0 0|
//    TSS_TSPATTRIB_KEYINFO_ALGORITHM     |0 0 1 0 1|
//    TSS_TSPATTRIB_KEYINFO_SIGSCHEME     |0 0 1 1 0|
//    TSS_TSPATTRIB_KEYINFO_ENCSCHEME     |0 0 1 1 1|
//    TSS_TSPATTRIB_KEYINFO_MIGRATABLE    |0 1 0 0 0|
//    TSS_TSPATTRIB_KEYINFO_REDIRECTED    |0 1 0 0 1|
//    TSS_TSPATTRIB_KEYINFO_VOLATILE      |0 1 0 1 0|
//    TSS_TSPATTRIB_KEYINFO_AUTHDATAUSAGE |0 1 0 1 1|
//    TSS_TSPATTRIB_KEYINFO_VERSION       |0 1 0 100|
//
#define TSS_TSPATTRIB_KEYINFO_SIZ          (0x00000080)   // key size in bits
#define TSS_TSPATTRIB_KEYINFO_USAGE         (0x00000100)   // key usage info
#define TSS_TSPATTRIB_KEYINFO_KEYFLAGS      (0x00000180)   // key flags
#define TSS_TSPATTRIB_KEYINFO_AUTHUSAGE      (0x00000200)   // key auth usage info
#define TSS_TSPATTRIB_KEYINFO_ALGORITHM          (0x00000280)   // key algorithm ID
#define TSS_TSPATTRIB_KEYINFO_SIGSCHEME          (0x00000300)   // key sig scheme
#define TSS_TSPATTRIB_KEYINFO_ENCSCHEME          (0x00000380)   // key enc scheme
#define TSS_TSPATTRIB_KEYINFO_MIGRATABLE      (0x00000400)   // if true then key is migratable
#define TSS_TSPATTRIB_KEYINFO_REDIRECTED      (0x00000480)   // key is redirected
#define TSS_TSPATTRIB_KEYINFO_VOLATILE      (0x00000500)   // if true key is volatile
#define TSS_TSPATTRIB_KEYINFO_AUTHDATAUSAGE   (0x00000580)   // if true authorization is required
#define TSS_TSPATTRIB_KEYINFO_VERSION      (0x00000600)   // version info as TSS version struct
//
```

```
//////////////////////////////////////////////////////////////////////
//      3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//      1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//      --------------------------------------------------------------
//
// SubFlags for Flag TSS_TSPATTRIB_RSAKEY_INFO
//
//  TSS_TSPATTRIB_KEYINFO_RSA_EXPONENT   |0 0 1|
//  TSS_TSPATTRIB_KEYINFO_RSA_MODULUS    |0 1 0|
//  TSS_TSPATTRIB_KEYINFO_RSA_KEYSIZE    |0 1 1|
//  TSS_TSPATTRIB_KEYINFO_RSA_PRIMES     |1 0 0|
//
#define TSS_TSPATTRIB_KEYINFO_RSA_EXPONENT   (0x00001000)
#define TSS_TSPATTRIB_KEYINFO_RSA_MODULUS      (0x00002000)
#define TSS_TSPATTRIB_KEYINFO_RSA_KEYSIZE      (0x00003000)
#define TSS_TSPATTRIB_KEYINFO_RSA_PRIMES      (0x00004000)
//
// SubFlags for Flag TSS_TSPATTRIB_KEY_PCR
//
////////////////////////////////////////////////////////////////////////////
//              3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//              1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//              --------------------------------------------------------------
//
// SubFlags for Flag TSS_TSPATTRIB_KEY_PCR
//  TSS_TSPATTRIB_KEYPCR_DIGEST_ATCREATION  |0 0 1|
//  TSS_TSPATTRIB_KEYPCR_DIGEST_ATRELEASE   |0 1 0|
//  TSS_TSPATTRIB_KEYPCR_SELECTION          |0 1 1|
//
#define TSS_TSPATTRIB_KEYPCR_DIGEST_ATCREATION   (0x00008000)
#define TSS_TSPATTRIB_KEYPCR_DIGEST_ATRELEASE   (0x00010000)
#define TSS_TSPATTRIB_KEYPCR_SELECTION      (0x00018000)
//
// SubFlags for TSS_TSPATTRIB_KEY_REGISTER
//
// SubFlags for TSS_TSPATTRIB_KEY_REGISTER
////////////////////////////////////////////////////////////////////////////////////////
//                            3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
```

```
//                           1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//                           ------------------------------------------------------------------
//
// TSS_TSPATTRIB_KEYREGISTER_USER    |0 0 1|
// TSS_TSPATTRIB_KEYREGISTER_SYSTEM  |0 1 0|
// TSS_TSPATTRIB_KEYREGISTER_NO      |0 1 1|
//
#define TSS_TSPATTRIB_KEYREGISTER_USER      (0x02000000)
#define TSS_TSPATTRIB_KEYREGISTER_SYSTEM      (0x04000000)
#define TSS_TSPATTRIB_KEYREGISTER_NO      (0x06000000)
//
//   Attribute definition for the tsp key object
//
//  Algorithm ID Definitions
//

//
//   This table defines the algo id's
//      Values intentional moved away from corresponding TPM values to avoid possible misuse
//
#define   TSS_ALG_RSA            (0x20)
#define   TSS_ALG_DES            (0x21)
#define   TSS_ALG_3DES         (0x22)
#define   TSS_ALG_SHA            (0x23)
#define   TSS_ALG_HMAC         (0x24)
#define   TSS_ALG_AES            (0x25)
//
//
// persisten storage registration definitions
//
#define TSS_PS_TYPE_USER             (1)         // Key is registered persistantly in the user
//                                                                              storage database.

#define TSS_PS_TYPE_SYSTEM           (2)         // Key is registered persistantly in the
//                                                                              system storage database.
// migration scheme definitions
//      Values intentional moved away from corresponding TPM values to avoid possible misuse
//
```

```
#define TSS_MS_MIGRATE            (0x20)
#define TSS_MS_REWRAP            (0x21)
#define TSS_MS_MAINT            (0x22)
//
//
//   TCPA key authorization
//      Values intentional moved away from corresponding TPM values to avoid possible misuse
//
#define TSS_KEYAUTH_AUTH_NEVER      (0x10)
#define TSS_KEYAUTH_AUTH_ALWAYS      (0x11)
//
// key usage definitions
//      Values intentional moved away from corresponding TPM values to avoid possible misuse
//
#define TSS_KEYUSAGE_BIND         (0x00)
#define TSS_KEYUSAGE_IDENTITY         (0x01)
#define TSS_KEYUSAGE_LEGACY         (0x02)
#define TSS_KEYUSAGE_SIGN         (0x03)
#define TSS_KEYUSAGE_STORAGE         (0x04)
#define TSS_KEYUSAGE_AUTHCHANGE      (0x07)
//
// key encrypten and signature scheme definitions
//
#define TSS_ES_NONE            (0x10)
#define TSS_ES_RSAESPKCSV15         (0x11)
#define TSS_ES_RSAESOAEP_SHA1_MGF1      (0x12)
//
#define TSS_SS_NONE            (0x10)
#define TSS_SS_RSASSAPKCS1V15_SHA1      (0x11)
#define TSS_SS_RSASSAPKCS1V15_DER      (0x12)
//
// Flags for TPM status information (Get- and SetStatus)
//
#define TSS_TPMSTATUS_DISABLEOWNERCLEAR      (0x00000001)   // persistent flag
#define TSS_TPMSTATUS_DISABLEFORCECLEAR      (0x00000002)   // volatile flag
#define TSS_TPMSTATUS_DISABLED         (0x00000003)   // persistent flag
#define TSS_TPMSTATUS_DEACTIVATED         (0x00000004)   // volatile flag
```

```
#define TSS_TPMSTATUS_OWNERSETDISABLE      (0x00000005)   // persistent flag for SetStatus
//                                                                          (disable flag)
#define TSS_TPMSTATUS_SETOWNERINSTALL      (0x00000006)   // persistent flag (ownership flag)
#define TSS_TPMSTATUS_DISABLEPUBEKREAD      (0x00000007)   // persistent flag
#define TSS_TPMSTATUS_ALLOWMAINTENANCE      (0x00000008)   // persistent flag
#define TSS_TPMSTATUS_PHYSPRES_LIFETIMELOCK   (0x00000009)   // persistent flag
#define TSS_TPMSTATUS_PHYSPRES_HWENABLE      (0x0000000A)   // persistent flag
#define TSS_TPMSTATUS_PHYSPRES_CMDENABLE      (0x0000000B)   // persistent flag
#define TSS_TPMSTATUS_PHYSPRES_LOCK         (0x0000000C)   // volatile flag
#define TSS_TPMSTATUS_PHYSPRESENCE         (0x0000000D)   // volatile flag
#define TSS_TPMSTATUS_PHYSICALDISABLE      (0x0000000E)   // persistent flag (SetStatus-Fkt
//                                                                          disable flag)
#define TSS_TPMSTATUS_CEKP_USED         (0x0000000F)   // persistent flag
#define TSS_TPMSTATUS_PHYSICALSETDEACTIVATED   (0x00000010)   // persistent flag (deactivated flag)
#define TSS_TPMSTATUS_SETTEMPDEACTIVATED      (0x00000011)   // volatile flag (deactivated flag)
#define TSS_TPMSTATUS_POSTINITIALISE      (0x00000012)   // volatile flag
#define TSS_TPMSTATUS_TPMPOST            (0x00000013)   // persistent flag
#define TSS_TPMSTATUS_TPMPOSTLOCK         (0x00000014)   // persistent flag
//
// Capability flag definitions
//
// TPM capabilities
#define TSS_TPMCAP_ORD               (0x10)
#define TSS_TPMCAP_ALG               (0x11)
#define TSS_TPMCAP_FLAG               (0x12)
#define TSS_TPMCAP_PROPERTY            (0x13)
#define TSS_TPMCAP_VERSION            (0x14)
//
// Sub-Capability Flags TPM-Capabilities
#define TSS_TPMCAP_PROP_PCR            (0x10)
#define TSS_TPMCAP_PROP_DIR            (0x11)
#define TSS_TPMCAP_PROP_MANUFACTURER      (0x12)
#define TSS_TPMCAP_PROP_SLOTS            (0x13)
//
// TSS Core Service Capabilities
#define TSS_TCSCAP_ALG               (0x00000001)
#define TSS_TCSCAP_VERSION            (0x00000002)
#define TSS_TCSCAP_CACHING            (0x00000003)
```

```
#define TSS_TCSCAP_PERSSTORAGE         (0x00000004)
#define TSS_TCSCAP_MANUFACTURER         (0x00000005)
//
// Sub-Capability Flags TSS-CoreService-Capabilities
#define TSS_TCSCAP_PROP_KEYCACHE         (0x00000100)
#define TSS_TCSCAP_PROP_AUTHCACHE         (0x00000101)
#define TSS_TCSCAP_PROP_MANUFACTURER_STR      (0x00000102)
#define TSS_TCSCAP_PROP_MANUFACTURER_ID      (0x00000103)
//
// TSS Service Provider Capabilities
#define TSS_TSPCAP_ALG               (0x00000010)
#define TSS_TSPCAP_VERSION            (0x00000011)
#define TSS_TSPCAP_PERSSTORAGE         (0x00000012)
//
// Event type definitions
#define TSS_EV_CODE_CERT            (0x00000001)
#define TSS_EV_CODE_NOCERT            (0x00000002)
#define TSS_EV_XML_CONFIG            (0x00000003)
#define TSS_EV_NO_ACTION            (0x00000004)
#define TSS_EV_SEPARATOR            (0x00000005)
#define TSS_EV_ACTION               (0x00000006)
#define TSS_EV_PLATFORM_SPECIFIC         (0x00000007)
//
//
// TSS random number limits
//
#define TSS_TSPCAP_RANDOMLIMIT         (0x00001000)   / Errata: Missing from spec
//
// UUIDs
//
// Errata: This are not in the spec
#define TSS_UUID_SRK   L"00000000-0000-0000-0000-000000000001"      // Storage root key
#define TSS_UUID_SK   L"00000000-0000-0000-0000-000000000002"      // System key
#define TSS_UUID_RK   L"00000000-0000-0000-0000-000000000003"      // roaming key
#define TSS_UUID_USK1   L"00000000-0000-0000-0000-000000000004"      // user storage key 1
#define TSS_UUID_USK2   L"00000000-0000-0000-0000-000000000005"      // user storage key 2
#define TSS_UUID_USK3   L"00000000-0000-0000-0000-000000000006"      // user storage key 3
#define TSS_UUID_USK4   L"00000000-0000-0000-0000-000000000007"      // user storage key 4
```

```
//
// TCPA well known secret
//
#define TSS_WELL_KNOWN_SECRET {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \\
                0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}


#endif // __TSS_DEFINES_H__
```

+++tss_typedef.h

```
/*++

Global typedefs for TSS

*/

#ifndef __TSS_TYPEDEF_H__
#define __TSS_TYPEDEF_H__


//--------------------------------------------------------------------
// definitions for TSS Service Provider (TSP)
//
typedef  UINT32  TSS_HANDLE;

typedef  UINT32  TSS_FLAG;  // object attributes
typedef  UINT32  TSS_RESULT;  // the return code from a TSS function

typedef  UINT32      TSS_HOBJECT; // basic object handle
typedef  TSS_HOBJECT     TSS_HCONTEXT; // context object handle
typedef  TSS_HOBJECT     TSS_HPOLICY; // policy object handle
typedef  TSS_HOBJECT     TSS_HTPM;  // TPM object handle
typedef  TSS_HOBJECT     TSS_HKEY;  // key object handle
typedef  TSS_HOBJECT     TSS_HENCDATA; // encrypted data object handle
typedef  TSS_HOBJECT     TSS_HPCRS;  // PCR composite object handle
typedef  TSS_HOBJECT     TSS_HHASH;  // hash object handle
typedef  TSS_HOBJECT     TSS_HPS;  // persistent storage object handle
```

```
typedef UINT32  TSS_EVENTTYPE;
typedef UINT16  TSS_MIGRATE_SCHEME;
typedef UINT32  TSS_ALGORITHM_ID;
typedef UINT32  TSS_KEY_USAGE_ID;
typedef UINT16  TSS_KEY_ENC_SCHEME;
typedef UINT16  TSS_KEY_SIG_SCHEME;

#endif // __TSS_TYPEDEF_H__
```

+++tss_structs.h

```
/*++

TSS structures for TSS

*/

#ifndef __TSS_STRUCTS_H__
#define __TSS_STRUCTS_H__

typedef struct tdTSS_VERSION
{
 BYTE   bMajor;
 BYTE   bMinor;
 BYTE   bRevMajor;
 BYTE   bRevMinor;
} TSS_VERSION;

typedef struct tdTSS_PCR_EVENT
{
 TSS_VERSION  versionInfo;
 UINT32  ulPcrIndex;
 TSS_EVENTTYPE eventType;
 UINT32  ulPcrValueLength;
 #ifdef __midl
```

```
    [size_is(ulPcrValueLength)]
 #endif
 BYTE*      rgbPcrValue;
 UINT32  ulEventLength;
 #ifdef __midl
    [size_is(ulEventLength)]
 #endif
 BYTE*      rgbEvent;
} TSS_PCR_EVENT;


typedef struct tdTSS_EVENT_CERT
{
 TSS_VERSION       versionInfo;
 UINT32  ulCertificateHashLength;
 BYTE*   rgbCertificateHash;
 UINT32  ulEntityDigestLength;
 #ifdef __midl
   [size_is(ulEntityDigestLength)]
 #endif
 BYTE*   rgbentityDigest;
 TSS_BOOL  fDigestChecked;
 TSS_BOOL  fDigestVerified;
 UINT32  ulIssuerLength;
 #ifdef __midl
   [size_is(ulIssuerLength)]
 #endif
 BYTE*   rgbIssuer;
} TSS_EVENT_CERT;
```

```
typedef struct tdTSS_UUID
{
      UINT32  ulTimeLow;
 UINT16  usTimeMid;
 UINT16  usTimeHigh;
 BYTE   bClockSeqHigh;
 BYTE   bClockSeqLow;
 BYTE   rgbNode[6];
} TSS_UUID;

typedef struct tdTSS_KM_KEYINFO
{
TSS_VERSION  versionInfo;
TSS_UUID  keyUUID;
TSS_UUID  parentKeyUUID;
BYTE   bAuthDataUsage;
TSS_BOOL  fIsLoaded;    // TRUE: actually loaded in TPM
UINT32  ulVendorDataLength;  // may be 0
#ifdef __midl
  [size_is(ulVendorDataLength)]
#endif
BYTE*   rgbVendorData;   // may be NULL
} TSS_KM_KEYINFO;


typedef struct tdTSS_VALIDATION
{
 TCPA_NONCE    ExternalData;
 UINT32    DataLength;
#ifdef __midl
 [size_is(DataLength)]
#endif
 BYTE*     Data;
 UINT32    ValidationDataLength;
#ifdef __midl
 [size_is(ValidationDataLength)]
#endif
 BYTE*     ValidationData;
```

```
} TSS_VALIDATION;


#endif // __TSS_STRUCTS_H__
```

+++tspi.h

```
#if !defined( _TSPI_H_ )
#define _TSPI_H_


#if !defined( EXTERN_C )
#if defined ( __cplusplus )
#define EXTERN_C extern "C"
#else
#define EXTERN_C
#endif //__cplusplus
#endif //EXTERN_C

#if !defined( TSPICALL )
#define TSPICALL EXTERN_C TSS_RESULT
#endif //TSPICALL

TSPICALL Tspi_SetAttribUint32
(
   TSS_HOBJECT      hObject,      // in
   TSS_FLAG         attribFlag,   // in
   TSS_FLAG         subFlag,      // in
   UINT32           ulAttrib      // in
);

TSPICALL Tspi_GetAttribUint32
(
   TSS_HOBJECT      hObject,     // in
   TSS_FLAG         attribFlag,  // in
   TSS_FLAG         subFlag,     // in
   UINT32*          pulAttrib    // out
```

```
);

TSPICALL Tspi_SetAttribData
(
   TSS_HOBJECT      hObject,            // in
   TSS_FLAG         attribFlag,         // in
   TSS_FLAG         subFlag,            // in
   UINT32           ulAttribDataSize,   // in
   BYTE*            rgbAttribData       // in
);

TSPICALL Tspi_GetAttribData
(
   TSS_HOBJECT      hObject,               // in
   TSS_FLAG         attribFlag,            // in
   TSS_FLAG         subFlag,               // in
   UINT32*          pulAttribDataSize,     // out
   BYTE**           prgbAttribData         // out
);


TSPICALL Tspi_ChangeAuth
(
   TSS_HOBJECT      hObjectToChange,   // in
   TSS_HOBJECT      hParentObject,     // in
   TSS_HPOLICY      hNewPolicy         // in
);


TSPICALL Tspi_ChangeAuthAsym
(
   TSS_HOBJECT      hObjectToChange,   // in
   TSS_HOBJECT      hParentObject,     // in
   TSS_HKEY         hIdentKey,         // in
   TSS_HPOLICY      hNewPolicy         // in
);

TSPICALL Tspi_GetPolicyObject
```

```
(
   TSS_HOBJECT      hObject,      // in
   TSS_FLAG         policyType,   // in
   TSS_HPOLICY*     phPolicy      // out
);


TSPICALL Tspi_Context_Create
(
   TSS_HCONTEXT*    phContext   // out
);

TSPICALL Tspi_Context_Close
(
   TSS_HCONTEXT     hContext    // in
);


TSPICALL Tspi_Context_Connect
(
   TSS_HCONTEXT     hContext,        // in
   UNICODE*         wszDestination   // in
);

TSPICALL Tspi_Context_FreeMemory
(
   TSS_HCONTEXT     hContext,    // in
   BYTE*            rgbMemory    // in
);


TSPICALL Tspi_Context_GetDefaultPolicy
(
   TSS_HCONTEXT     hContext,    // in
   TSS_HPOLICY*     phPolicy     // out
);

TSPICALL Tspi_Context_CreateObject
```

```
(
   TSS_HCONTEXT     hContext,     // in
   TSS_FLAG         objectType,   // in
   TSS_FLAG         initFlags,    // in
   TSS_HOBJECT*     phObject      // out
);

TSPICALL Tspi_Context_CloseObject
(
   TSS_HCONTEXT     hContext,    // in
   TSS_HOBJECT      hObject      // in
);

TSPICALL Tspi_Context_GetCapability
(
   TSS_HCONTEXT     hContext,            // in
   TSS_FLAG         capArea,             // in
   UINT32           ulSubCapLength,      // in
   BYTE*            rgbSubCap,           // in
   UINT32*          pulRespDataLength,   // out
   BYTE**           prgbRespData         // out
);


TSPICALL Tspi_Context_GetTpmObject
(
   TSS_HCONTEXT     hContext,     // in
   TSS_HTPM*        phTPM         // out
);

TSPICALL Tspi_Context_LoadKeyByBlob
(
   TSS_HCONTEXT     hContext,         // in
   TSS_HKEY         hUnwrappingKey,   // in
   UINT32           ulBlobLength,     // in
   BYTE*            rgbBlobData,      // in
   TSS_HKEY*        phKey             // out
);
```

```
TSPICALL Tspi_Context_LoadKeyByUUID
(
   TSS_HCONTEXT     hContext,                // in
   TSS_FLAG         persistentStorageType,   // in
   TSS_UUID         uuidData,                // in
   TSS_HKEY*        phKey                    // out
);

TSPICALL Tspi_Context_RegisterKey
(
   TSS_HCONTEXT     hContext,                     // in
   TSS_HKEY         hKey,                         // in
   TSS_FLAG         persistentStorageType,        // in
   TSS_UUID         uuidKey,                      // in
   TSS_FLAG         persistentStorageTypeParent,  // in
   TSS_UUID         uuidParentKey                 // in
);

TSPICALL Tspi_Context_UnregisterKey
(
   TSS_HCONTEXT     hContext,                // in
   TSS_FLAG         persistentStorageType,   // in
   TSS_UUID         uuidKey,                 // in
   TSS_HKEY*        phkey                    // out
);

TSPICALL Tspi_Context_GetKeyByUUID
(
   TSS_HCONTEXT     hContext,                // in
   TSS_FLAG         persistentStorageType,   // in
   TSS_UUID         uuidData,                // in
   TSS_HKEY*        phKey                    // out
);

TSPICALL Tspi_Context_GetKeyByPublicInfo
(
```

```
   TSS_HCONTEXT     hContext,                // in
   TSS_FLAG         persistentStorageType,   // in
   TSS_ALGORITHM_ID algID,                   // in
   UINT32           ulPublicInfoLength,      // in
   BYTE*            rgbPublicInfo,           // in
   TSS_HKEY*        phKey                    // out
);

TSPICALL Tspi_Context_GetRegisteredKeysByUUID
(
   TSS_HCONTEXT     hContext,                // in
   TSS_FLAG         persistentStorageType,   // in
   TSS_UUID*        pGuidData,               // in
   UINT32*          pulKeyHierarchySize,     // out
   TSS_KM_KEYINFO** ppKeyHierarchy           // out
);

TSPICALL Tspi_Policy_SetSecret
(
   TSS_HPOLICY      hPolicy,           // in
   TSS_FLAG         secretMode,        // in
   UINT32           ulSecretLength,    // in
   BYTE*            rgbSecret          // in
);

TSPICALL Tspi_Policy_FlushSecret
(
   TSS_HPOLICY      hPolicy    // in
);

TSPICALL Tspi_Policy_AssignToObject
(
   TSS_HPOLICY      hPolicy,   // in
   TSS_HOBJECT      hObject    // in
);

TSPICALL Tspi_TPM_CreateEndorsementKey
(
```

```
   TSS_HTPM         hTPM,               // in
   TSS_HKEY         hKey,               // in
   TSS_VALIDATION*  pValidationData     // in, out
);

TSPICALL Tspi_TPM_GetPubEndorsementKey
(
   TSS_HTPM         hTPM,                   // in
   TSS_BOOL         fOwnerAuthorized,       // in
   TSS_VALIDATION*  pValidationData,        // in, out
   TSS_HKEY*        phEndorsementPubKey     // out
);

TSPICALL Tspi_TPM_TakeOwnership
(
   TSS_HTPM         hTPM,                   // in
   TSS_HKEY         hKeySRK,                // in
   TSS_HKEY         hEndorsementPubKey      // in
);


TSPICALL Tspi_TPM_CollateIdentityRequest
(
   TSS_HTPM         hTPM,                       // in
   TSS_HKEY         hKeySRK,                    // in
   TSS_HKEY         hCAPubKey,                  // in
   UINT32           ulIdentityLabelLength,      // in
   BYTE*            rgbIdentityLabelData,       // in
   TSS_HKEY         hIdentityKey,               // in
   TSS_ALGORITHM_ID algid,                      // in
   UINT32*          pulTcpaIdentityReqLength,   // out
   BYTE**           prgbTcpaIdentityReq         // out
);


TSPICALL Tspi_TPM_ActivateIdentity
(
   TSS_HTPM         hTPM,                         // in
```

```
   TSS_HKEY         hIdentKey,                     // in
   UINT32           ulAsymCAContentsBlobLength,    // in
   BYTE*            rgbAsymCAContentsBlob,         // in
   UINT32           ulSymCAAttestationBlobLength, // in
   BYTE*            rgbSymCAAttestationBlob,       // in
   UINT32*          pulCredentialLength,           // out
   BYTE**           prgbCredential                 // out
);


TSPICALL Tspi_TPM_ClearOwner
(
   TSS_HTPM         hTPM,            // in
   TSS_BOOL         fForcedClear     // in
);


TSPICALL Tspi_TPM_SetStatus
(
   TSS_HTPM         hTPM,         // in
   TSS_FLAG         statusFlag,   // in
   TSS_BOOL         fTpmState     // in
);

TSPICALL Tspi_TPM_GetStatus
(
   TSS_HTPM         hTPM,         // in
   TSS_FLAG         statusFlag,   // in
   TSS_BOOL*        pfTpmState    // out
);

TSPICALL Tspi_TPM_SelfTestFull
(
   TSS_HTPM         hTPM   // in
);

TSPICALL Tspi_TPM_CertifySelfTest
(
```

```
   TSS_HTPM         hTPM,               // in
   TSS_HKEY         hKey,               // in
   TSS_VALIDATION*  pValidationData     // in, out
);

TSPICALL Tspi_TPM_GetTestResult
(
   TSS_HTPM         hTPM,                  // in
   UINT32*          pulTestResultLength,   // out
   BYTE**           prgbTestResult         // out
);

TSPICALL Tspi_TPM_GetCapability
(
   TSS_HTPM         hTPM,                // in
   TSS_FLAG         capArea,             // in
   UINT32           ulSubCapLength,      // in
   BYTE*            rgbSubCap,           // in
   UINT32*          pulRespDataLength,   // out
   BYTE**           prgbRespData         // out
);

TSPICALL Tspi_TPM_GetCapabilitySigned
(
   TSS_HTPM         hTPM,                // in
   TSS_HTPM         hKey,                // in
   TSS_FLAG         capArea,             // in
   UINT32           ulSubCapLength,      // in
   BYTE*            rgbSubCap,           // in
   TSS_VALIDATION*  pValidationData,     // in, out
   UINT32*          pulRespDataLength,   // out
   BYTE**           prgbRespData         // out
);

TSPICALL Tspi_TPM_CreateMaintenanceArchive
(
   TSS_HTPM         hTPM,                   // in
   TSS_BOOL         fGenerateRndNumber,     // in
```

```
   UINT32*          pulRndNumberLength,     // out
   BYTE**           prgbRndNumber,          // out
   UINT32*          pulArchiveDataLength,   // out
   BYTE**           prgbArchiveData         // out
);

TSPICALL Tspi_TPM_KillMaintenanceFeature
(
   TSS_HTPM         hTPM    // in
);

TSPICALL Tspi_TPM_LoadMaintenancePubKey
(
   TSS_HTPM         hTPM,               // in
   TSS_HKEY         hMaintenanceKey,    // in
   TSS_VALIDATION*  pValidationData     // in, out
);

TSPICALL Tspi_TPM_CheckMaintenancePubKey
(
   TSS_HTPM         hTPM,              // in
   TSS_HKEY         hMaintenanceKey,   // in
   TSS_VALIDATION*  pValidationData    // in, out
);

TSPICALL Tspi_TPM_GetRandom
(
   TSS_HTPM         hTPM,                  // in
   UINT32           ulRandomDataLength,    // in
   BYTE**           prgbRandomData         // out
);

TSPICALL Tspi_TPM_StirRandom
(
   TSS_HTPM         hTPM,                  // in
   UINT32           ulEntropyDataLength,   // in
   BYTE*            rgbEntropyData         // in
);
```

```
TSPICALL Tspi_TPM_AuthorizeMigrationTicket
(
   TSS_HTPM            hTPM,                // in
   TSS_HKEY            hMigrationKey,       // in
   TSS_MIGRATE_SCHEME  migrationScheme,     // in
   UINT32*             pulMigTicketLength,  // out
   BYTE**              prgbMigTicket        // out
);


TSPICALL Tspi_TPM_GetEvent
(
   TSS_HTPM         hTPM,             // in
   UINT32           ulPcrIndex,       // in
   UINT32           ulEventNumber,    // in
   TSS_PCR_EVENT*   pPcrEvent         // out
);

TSPICALL Tspi_TPM_GetEvents
(
   TSS_HTPM         hTPM,              // in
   UINT32           ulPcrIndex,        // in
   UINT32           ulStartNumber,     // in
   UINT32*          pulEventNumber,    // in, out
   TSS_PCR_EVENT**  prgPcrEvents       // out
);


TSPICALL Tspi_TPM_GetEventLog
(
   TSS_HTPM         hTPM,                // in
   UINT32*          pulEventNumber,      // out
   TSS_PCR_EVENT**  prgPcrEvents         // out
);

TSPICALL Tspi_TPM_Quote
(
```

```
   TSS_HTPM         hTPM,               // in
   TSS_HKEY         hIdentKey,          // in
   TSS_HPCRS        hPcrComposite,      // in
   TSS_VALIDATION*  pValidationData     // in, out
);

TSPICALL Tspi_TPM_PcrExtend
(
      TSS_HTPM       hTPM,              // in
      UINT32         ulPcrIndex,        // in
      UINT32         ulPcrDataLength,   // in
      BYTE*          pbPcrData,         // in
      TSS_PCR_EVENT* pPcrEvent,         // in
      UINT32*        pulPcrValueLength, // out
      BYTE**         prgbPcrValue       // out
);

TSPICALL Tspi_TPM_PcrRead
(
   TSS_HTPM         hTPM,               // in
   UINT32           ulPcrIndex,         // in
   UINT32*          pulPcrValueLength,  // out
   BYTE**           prgbPcrValue        // out
);


TSPICALL Tspi_TPM_DirWrite
(
   TSS_HTPM         hTPM,              // in
   UINT32           ulDirIndex,        // in
   UINT32           ulDirDataLength,   // in
   BYTE*            rgbDirData         // in
);


TSPICALL Tspi_TPM_DirRead
(
   TSS_HTPM         hTPM,               // in
```

```
   UINT32           ulDirIndex,          // in
   UINT32*          pulDirDataLength,    // out
   BYTE**           prgbDirData          // out
);


TSPICALL Tspi_Key_LoadKey
(
   TSS_HKEY         hKey,             // in
   TSS_HKEY         hUnwrappingKey    // in
);

TSPICALL Tspi_Key_UnloadKey
(
   TSS_HKEY         hKey              // in
);


TSPICALL Tspi_Key_GetPubKey
(
   TSS_HKEY         hKey,              // in
   UINT32*          pulPubKeyLength,   // out
   BYTE**           prgbPubKey         // out
);


TSPICALL Tspi_Key_CertifyKey
(
   TSS_HKEY         hKey,              // in
   TSS_HKEY         hCertifyingKey,    // in
   TSS_VALIDATION*  pValidationData    // in, out
);

TSPICALL Tspi_Key_CreateKey
(
   TSS_HKEY         hKey,            // in
   TSS_HKEY         hWrappingKey,    // in
   TSS_HPCRS        hPcrComposite    // in, may be NULL
```

```
);

TSPICALL Tspi_Key_WrapKey
(
   TSS_HKEY         hKey,           // in
   TSS_HKEY         hWrappingKey,   // in
   TSS_HPCRS        hPcrComposite   // in, may be NULL
);

TSPICALL Tspi_Key_CreateMigrationBlob
(
   TSS_HKEY         hKeyToMigrate,            // in
   TSS_HKEY         hParentKey,               // in
   UINT32           ulMigTicketLength,        // in
   BYTE*            rgbMigTicket   ,          // in
   UINT32*          pulRandomLength,          // out
   BYTE**           prgbRandom,               // out
   UINT32*          pulMigrationBlobLength,   // out
   BYTE**           prgbMigrationBlob         // out
);

TSPICALL Tspi_Key_ConvertMigrationBlob
(
   TSS_HKEY         hKeyToMigrate,           // in
   TSS_HKEY         hParentKey,              // in
   UINT32           ulRandomLength,          // in
   BYTE*            rgbRandom,               // in
   UINT32           ulMigrationBlobLength,   // in
   BYTE*            rgbMigrationBlob         // in
);

TSPICALL Tspi_Hash_Sign
(
   TSS_HHASH        hHash,                // in
   TSS_HKEY         hKey,                 // in
   UINT32*          pulSignatureLength,   // out
   BYTE**           prgbSignature         // out
);
```

```
TSPICALL Tspi_Hash_VerifySignature
(
   TSS_HHASH        hHash,               // in
   TSS_HKEY         hKey,                // in
   UINT32           ulSignatureLength,   // in
   BYTE*            rgbSignature         // in
);

TSPICALL Tspi_Hash_SetHashValue
(
   TSS_HHASH        hHash,               // in
   UINT32           ulHashValueLength,   // in
   BYTE*            rgbHashValue         // in
);

TSPICALL Tspi_Hash_GetHashValue
(
   TSS_HHASH        hHash,                // in
   UINT32*          pulHashValueLength,   // out
   BYTE**           prgbHashValue         // out
);

TSPICALL Tspi_Hash_UpdateHashValue
(
   TSS_HHASH        hHash,          // in
   UINT32           ulDataLength,   // in
   BYTE*            rgbData         // in
);


TSPICALL Tspi_Data_Bind
(
   TSS_HENCDATA     hEncData,       // in
   TSS_HKEY         hEncKey,        // in
   UINT32           ulDataLength,   // in
   BYTE*            rgbDataToBind   // in
);
```

```
TSPICALL Tspi_Data_Unbind
(
   TSS_HENCDATA     hEncData,               // in
   TSS_HKEY         hKey,                   // in
   UINT32*          pulUnboundDataLength,   // out
   BYTE**           prgbUnboundData         // out
);

TSPICALL Tspi_Data_Seal
(
   TSS_HENCDATA     hEncData,        // in
   TSS_HKEY         hEncKey,         // in
   UINT32           ulDataLength,    // in
   BYTE*            rgbDataToSeal,   // in
   TSS_HPCRS        hPcrComposite    // in
);

TSPICALL Tspi_Data_Unseal
(
   TSS_HENCDATA     hEncData,                // in
   TSS_HKEY         hKey,                    // in
   UINT32*          pulUnsealedDataLength,   // out
   BYTE**           prgbUnsealedData         // out
);

TSPICALL Tspi_PcrComposite_SelectPcrIndex
(
   TSS_HPCRS        hPcrComposite,   // in
   UINT32           ulPcrIndex       // in
);

TSPICALL Tspi_PcrComposite_SetPcrValue
(
   TSS_HPCRS        hPcrComposite,      // in
   UINT32           ulPcrIndex,         // in
   UINT32           ulPcrValueLength,   // in
   BYTE*            rgbPcrValue         // in
```

```
);

TSPICALL Tspi_PcrComposite_GetPcrValue
(
   TSS_HPCRS        hPcrComposite,       // in
   UINT32           ulPcrIndex,          // in
   UINT32*          pulPcrValueLength,   // out
   BYTE**           prgbPcrValue         // out
);

/******/
TSPICALL Tspicb_CallbackHMACAuth
(
   PVOID            lpAppData,           // in
   TSS_HOBJECT      hAuthorizedObject,   // in
   TSS_BOOL         ReturnOrVerify,      // in
   UINT32           ulPendingFunction,   // in
   TSS_BOOL         ContinueUse,         // in
   UINT32           ulSizeNonces,        // in
   BYTE*            rgbNonceEven,        // in
   BYTE*            rgbNonceOdd,         // in
   BYTE*            rgbNonceEvenOSAP,    // in
   BYTE*            rgbNonceOddOSAP,     // in
   UINT32           ulSizeDigestHmac,    // in
   BYTE*            rgbParamDigest,      // in
   BYTE*            rgbHmacData          // in, out
);


TSPICALL Tspicb_CallbackXorEnc
(
   PVOID            lpAppData,            // in
   TSS_HOBJECT      hOSAPObject,          // in
   TSS_HOBJECT      hObject,              // in
   TSS_FLAG         PurposeSecret,        // in
   UINT32           ulSizeNonces,         // in
   BYTE*            rgbNonceEven,         // in
   BYTE*            rgbNonceOdd,          // in
```

```
   BYTE*            rgbNonceEvenOSAP,     // in
   BYTE*            rgbNonceOddOSAP,      // in
   UINT32           ulSizeEncAuth,        // in
   BYTE*            rgbEncAuthUsage,      // out
   BYTE*            rgbEncAuthMigration   // out
);


TSPICALL Tspicb_CallbackTakeOwnership
(
   PVOID            lpAppData,         // in
   TSS_HOBJECT      hObject,           // in
   TSS_HKEY         hObjectPubKey,     // in
   UINT32           ulSizeEncAuth,     // in
   BYTE*            rgbEncAuth         // out
);


TSPICALL Tspicb_CallbackChangeAuthAsym
(
   PVOID            lpAppData,        // in
   TSS_HOBJECT      hObject,          // in
   TSS_HKEY         hObjectPubKey,    // in
   UINT32           ulSizeEncAuth,    // in
   UINT32           ulSizeAuthLink,   // in
   BYTE*            rgbEncAuth,       // out
   BYTE*            rgbAuthLink       // out
);

TSPICALL Tspicb_CollateIdentity
(
```

```
      PVOID             lpAppData,                       // in
      UINT32            ulTCPAPlainIdentityProofLength, // in
      BYTE*             rgbTCPAPlainIdentityProof,      // in
      TSS_ALGORITHM_ID  algID,                           // in
      UINT32            ulSessionKeyLength,              // out
      BYTE*             rgbSessionKey,                   // out
      UINT32*           pulTCPAIdentityProofLength,      // out
      BYTE*             rgbTCPAIdentityProof             // out
);


TSPICALL Tspicb_ActivateIdentity
(
   PVOID            lpAppData,                    // in
   UINT32           ulSessionKeyLength,           // in
   BYTE*            rgbSessionKey,                // in
   UINT32           ulSymCAAttestationBlobLength, // in
   BYTE*            rgbSymCAAttestationBlob,      // in
   UINT32*          pulCredentialLength,          // out
   BYTE*            rgbCredential                 // out
);

#endif //_TSPI_H_
```

+++tspi.idl

```
/*++

Interface declarations for the TSS Service Provider - COM interface for Windows based platforms

--*/

import "oaidl.idl";
import "ocidl.idl";

import "TPM_struct.h";
import "tss_typedef.h";
```

```
import "tss_structs.h";

// forward declaration
interface ITCPAPolicy;
interface ITCPAKey;


   //ITCPAAttrib Interface
   [
      object,
      uuid(FBCD9C2E-72CB-47BB-99DD-2317551491DE),

      helpstring("ITCPAAttrib Interface"),
      pointer_default(unique)
   ]
   interface ITCPAAttrib : IUnknown
   {
      [helpstring("method SetAttribUint32")]
      HRESULT SetAttribUint32([in] TSS_FLAG attribFlag,
                     [in] TSS_FLAG subFlags,
                  [in] UINT32 ulAttrib);

      [helpstring("method GetAttribUint32")]
      HRESULT GetAttribUint32([in] TSS_FLAG attribFlag,
                  [in] TSS_FLAG subFlags,
                  [out] UINT32* pulAttrib);

      [helpstring("method SetAttribData")]
      HRESULT SetAttribData([in] TSS_FLAG attribFlag,
                   [in] TSS_FLAG subFlags,
                   [in] UINT32 ulAttribDataSize,
                   [in, ptr, size_is(ulAttribDataSize)] BYTE* rgbAttribData);

      [helpstring("method GetAttribData")]
      HRESULT GetAttribData([in] TSS_FLAG attribFlag,
                   [in] TSS_FLAG subFlags,
                   [out] UINT32* pulAttribDataSize,
                   [out, size_is(, *pulAttribDataSize)] BYTE** prgbAttribData);
```

```
}

//ITCPAAuth Interface
[
   object,
   uuid(FBCD9C2F-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAAuth Interface"),
   pointer_default(unique)
]
interface ITCPAAuth : IUnknown
{
   [helpstring("method GetPolicyObject")]
   HRESULT GetPolicyObject([in] TSS_FLAG PolicyType,
               [out] ITCPAPolicy** ppPolicyObject);

   [helpstring("method ChangeAuth")]
   HRESULT ChangeAuth([in] IUnknown* PpParentObject,
             [in] ITCPAPolicy* PpNewPolicy);

   [helpstring("method ChangeAuthAsym")]
   HRESULT ChangeAuth([in] IUnknown* PpParentObject,
             [in] ITCPAKey* hIdentkey,
             [in] ITCPAPolicy* PpNewPolicy);
};

//ITCPAPcrs Interface
[
   object,
   uuid(FBCD9C2D-72CB-47BB-99DD-2317551491DE),
   helpstring("ITCPAPcrs Interface"),
   pointer_default(unique)
]
interface ITCPAPcrs : IUnknown
{
   [helpstring("method SetPcrValue")]
   HRESULT SetPcrValue([in] UINT32 ulPCRIndex,
               [in] UINT32 ulPcrValueLength,
```

```
            [in, size_is(ulPcrValueLength)] BYTE* rgbPcrValue);

   [helpstring("method GetPcrValue")]
   HRESULT GetPcrValue([in] UINT32 ulPCRIndex,
            [out] UINT32* pulPcrValueLength,
            [out, size_is(, *pulPcrValueLength)] BYTE** prgbPcrValue);

   [helpstring("method SelectPcrIndex")]
   HRESULT SelectPcrIndex([in] UINT32 ulPCRIndex);

};

//ITCPAKey Interface
[
   object,
   uuid(FBCD9C27-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAKey Interface"),
   pointer_default(unique)
]
interface ITCPAKey : IUnknown
{
   [helpstring("method LoadKey")]
   HRESULT LoadKey([in] ITCPAKey* pUnwrappingKey);

   [helpstring("method CreateKey")]
   HRESULT CreateKey([in] ITCPAKey* pUnwrappingKey,
            [in] ITCPAPcrs* pPcrComosite);

   [helpstring("method WrapKey")]
   HRESULT WrapKey([in] ITCPAKey* pWrappinKey,
            [in] ITCPAPcrs* pPcrComposite);

   [helpstring("method CertifyKey")]
   HRESULT CertifyKey([in] ITCPAKey* pCertifyingKey,
            [in, out, ptr] TSS_VALIDATION* pValidationData);

   [helpstring("method GetPubKey")]
```

```
   HRESULT GetPubKey([out] UINT32* pulPubKeyLength,
            [out, size_is(, *pulPubKeyLength)] BYTE** prgbPubKey);

   [helpstring("method UnLoadKey")]
   HRESULT UnLoadKey();

};

// ITCPAMigration
[
   object,
   uuid(FBCD9C30-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAMigration Interface"),
   pointer_default(unique)
]
interface ITCPAMigration : IUnknown
{
   [helpstring("method CreateMigrationBlob")]
   HRESULT CreateMigrationBlob([in] ITCPAKey *pParentKey,
                  [in] UINT32 ulMigTicketLength,
                  [in, size_is(ulMigTicketLength)] BYTE* rgbMigTicket,
                  [out] UINT32 *pulRandomLength,
                  [out, size_is(, *pulRandomLength)] BYTE **prgbRandom,
                  [out] UINT32 *pulMigrationBlobLength,
                  [out, size_is(, *pulMigrationBlobLength)] BYTE **prgbMigrationBlob);

   [helpstring("method ConvertMigrationBlob")]
   HRESULT ConvertMigrationBlob([in] ITCPAKey *pParentKey,
                   [in] UINT32 ulRandomLength,
                   [in, size_is(ulRandomLength)] BYTE *rgbRandom,
                   [in] UINT32 ulMigrationBlobLength,
                   [in, size_is(ulMigrationBlobLength)] BYTE *rgbMigrationBlob);
};


//ITCPAEncData Interface
[
```

```
   object,
   uuid(FBCD9C29-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAEncData Interface"),
   pointer_default(unique)
]
interface ITCPAEncData : IUnknown
{
   [helpstring("method Seal")]
   HRESULT Seal([in] ITCPAKey* pEncKey,
          [in] UINT32 ulDataLength,
          [in, size_is(ulDataLength)] BYTE* rgbDataToSeal,
          [in] ITCPAPcrs* pPcrComposite);

   [helpstring("method Unseal")]
   HRESULT Unseal([in] ITCPAKey* pKey,
            [out] UINT32* pulUnsealedDataLength,
            [out, size_is(, *pulUnsealDataLength)] BYTE** prgbUnsealedData);

   [helpstring("method Bind")]
   HRESULT Bind([in] ITCPAKey* pEncKey,
          [in] UINT32 ulDataLength,
          [in, size_is(ulDataLength)] BYTE* rgbDataToBind);

   [helpstring("method Unbind")]
   HRESULT Unbind([in] ITCPAKey* pKey,
            [out] UINT32* pulUnboundDataLength,
            [out, size_is(, *pulUnboundDataLength)] BYTE** prgbUnboundData);

};

//ITCPAHash Interface
[
   object,
   uuid(FBCD9C2B-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAHash Interface"),
   pointer_default(unique)
```

```
]
interface ITCPAHash : IUnknown
{
   [helpstring("method SetHashValue")]
   HRESULT SetHashValue([in] UINT32 ulHashValueLength,
               [in, size_is(ulHashValueLength)] BYTE* rgbHashValue);

   [helpstring("method GetHashValue")]
   HRESULT GetHashValue([out] UINT32* pulHashValueLength,
               [out, size_is(, *pulHashValueLength)] BYTE** prgbHashValue);

   [helpstring("method UpdateHashValue")]
   HRESULT UpdateHashValue([in] UINT32 ulDataLength,
               [in, size_is(ulDataLength)] BYTE* rgbData);

   [helpstring("method Sign")]
   HRESULT Sign([in] ITCPAKey* pKey,
          [out] UINT32* pulSignatureLength,
          [out, size_is(, *pulSignatureLength)] BYTE** prgbSignature);

   [helpstring("method VerifySignature")]
   HRESULT VerifySignature([in] ITCPAKey* pKey,
               [in] UINT32 ulSignatureLength,
               [in, size_is(ulSignatureLength)] BYTE* rgbSignature);

};


//ITCPAPolicy Interface
[
   object,
   uuid(FBCD9C1E-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAPolicy Interface"),
   pointer_default(unique)
]
interface ITCPAPolicy : IUnknown
{
```

```
   [helpstring("method SetSecret")]
   HRESULT SetSecret([in] TSS_FLAG SecretMode,
            [in] UINT32 ulSecretLength,
            [in, ptr, size_is(ulSecretLength)] BYTE* rgbSecret);

   [helpstring("method FlushSecret")]
   HRESULT FlushSecret();

   [helpstring("method AssignToObject")]
   HRESULT AssignToObject([in] IUnknown* pUnkObject);

};

//ITCPAAdministration Interface
[
   object,
   uuid(FBCD9C24-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAAdministration Interface"),
   pointer_default(unique)
]
interface ITCPAAdministration : IUnknown
{
   [helpstring("method SelfTestFull")]
   HRESULT SelfTestFull();

   [helpstring("method GetTestResult")]
   HRESULT GetTestResult([out] UINT32* pulTestResultLength,
               [out, size_is(, *pulTestResultLength)] BYTE** prgbTestResult);

   [helpstring("method CertifySelfTest")]
   HRESULT CertifySelfTest([in] ITCPAKey* phKey,
              [in, out, ptr] TSS_VALIDATION* pValidationData);

   [helpstring("method CreateEndorsementKey")]
   HRESULT CreateEndorsementKey([in] ITCPAKey* pEndorsementKey,
                   [in, out, ptr] TSS_VALIDATION* pValidationData);
```

```
   [helpstring("method GetPubEndorsementKey")]
   HRESULT GetPubEndorsementKey([in] BOOL fOwnerAuthorized,
                   [in, out, ptr] TSS_VALIDATION* pValidationData,
                   [out] ITCPAKey** ppEndorsementKey);

   [helpstring("method TakeOwnerShip")]
   HRESULT TakeOwnerShip([in] ITCPAKey* pKeySRK,
                [in] ITCPAKey* pEndorsementKeyPubKey);

   [helpstring("method ClearOwner")]
   HRESULT ClearOwner([in] BOOL fForcedClear);

   [helpstring("method SetStatus")]
   HRESULT SetStatus([in] TSS_FLAG statusFlag,
            [in] BOOL fTpmState);

   [helpstring("method GetStatus")]
   HRESULT GetStatus([in] TSS_FLAG statusFlag,
            [out] BOOL* pfTpmState);

   [helpstring("method AuthorizeMigrationTicket")]
   HRESULT AuthorizeMigrationTicket([in] ITCPAKey* pMigrationKey,
                       [in] UINT32 MigrationScheme,
                    [out] UINT32* pulMigTicketLength,
                    [out, size_is(, *pulMigTicketLength)] BYTE** prgbMigTicket);
}

//ITCPAIdentityCreation Interface
[
   object,
   uuid(FBCD9C23-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAIdentityCreation Interface"),
   pointer_default(unique)
]
interface ITCPAIdentityCreation: IUnknown
{
   [helpstring("method CollateIdentityRequest")]
```

```
   HRESULT CollateIdentityRequest([in] ITCPAKey* pKeySRK,
            [in] ITCPAKey* pCAPubKey,
            [in] UINT32 ulIdentityLabelLength,
            [in, size_is(ulIdentityLabelLength)] BYTE* rgbIdentityLabelData,
            [in] ITCPAKey* pIdentityKey,
            [in] TSS_ALGORITHM_ID algID,
            [out] UINT32* pulTCPAIdentityReqLength,
            [out, size_is(, *pulTCPAIdentityReqLength)] BYTE** prgbTCPAIdentityReq);

   [helpstring("method ActivateIdentity")]
   HRESULT ActivateIdentity([in] ITCPAKey* pIdentityKey,
            [in] UINT32 ulAsymCAContentsBlobLength,
            [in, size_is(ulAsymCAContentsBlobLength)] BYTE* rgbAsymCAContentsBlob,
            [in] UINT32 ulSymCAAttestationBlobLength,
            [in, size_is(ulSymCAAttestationBlobLength)] BYTE* rgbSymCAAttestationBlob,
            [out] UINT32* pulCredentialLength,
            [out, size_is(, *pulCredentialLength)] BYTE** prgbCredential);
};


//ITCPAMaintenance Interface
[
   object,
   uuid(FBCD9C25-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAMaintenance Interface"),
   pointer_default(unique)
]
interface ITCPAMaintenance: IUnknown
{
   [helpstring("method CreateMaintenanceArchive")]
```

```
   HRESULT CreateMaintenanceArchive([in] TSS_BOOL fGenerateRndNumber,
            [out] UINT32* pulRndNumberLength,
            [out, size_is(, *pulRndNumberLength)] BYTE** prgbRndNumber,
            [out] UINT32* pulArchiveDataLength,
            [out, size_is(, *pulArchiveDataLength)] BYTE** prgbArchiveData);

   [helpstring("method KillMaintenanceFeature")]
   HRESULT KillMaintenanceFeature();

   [helpstring("method LoadMaintenancePubKey")]
   HRESULT LoadMaintenancePubKey([in] ITCPAKey* pMaintenanceKey,
            [in, out, ptr] TSS_VALIDATION* pValidationData);

   [helpstring("method CheckMaintenancePubKey")]
   HRESULT CheckMaintenancePubKey([in] ITCPAKey* pMaintenanceKey,
            [in, out, ptr] TSS_VALIDATION* pValidationData);
};

//ITCPAIntegrity Interface
[
   object,
   uuid(FBCD9C22-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAIntegrity Interface"),
   pointer_default(unique)
]
interface ITCPAIntegrity : IUnknown
{
   [helpstring("method PcrExtend")]
   HRESULT PcrExtend([in] UINT32 ulPcrIndex,
                          [in] UINT32 ulPcrDataLength,
            [in, size_is(ulPcrDataLength)] BYTE* pbPcrData,
            [in, ptr] TSS_PCR_EVENT* pEvent,
            [out] UINT32* pulPcrValueLength,
            [out, size_is(, *pulPcrValueLength)] BYTE** ppbPcrValue);

   [helpstring("method PcrRead")]
   HRESULT PcrRead([in] UINT32 ulPcrIndex,
```

```
            [out] UINT32* pulPcrValueLength,
            [out, size_is(, *pulPcrValueLength)] BYTE** ppbPcrValue);

   [helpstring("method DirWrite")]
   HRESULT DirWrite([in] UINT32 ulDirIndex,
             [in] UINT32 ulDirDataLength,
             [in, size_is(ulDirDataLength)] BYTE* rgbDirData);

   [helpstring("method DirRead")]
   HRESULT DirRead([in] UINT32 ulDirIndex,
            [out] UINT32* pulDirDataLength,
            [out, size_is(, *pulDirDataLength)] BYTE** prgbDirData);

   [helpstring("method Quote")]
   HRESULT Quote([in] ITCPAKey* pIdentKey,
           [in] ITCPAPcrs* pPcrComposite,
           [in, out, ptr] TSS_VALIDATION* pValidationData);
};

//ITCPATpm Interface
[
   object,
   uuid(FBCD9C21-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPATpm Interface"),
   pointer_default(unique)
]
interface ITCPATpm : IUnknown
{
   [helpstring("method GetRandom")]
   HRESULT GetRandom([in] UINT32 ulRandomDataLength,
            [out, size_is(, ulRandomDataLength] BYTE** prgbRandomData);

   [helpstring("method StirRandom")]
   HRESULT StirRandom([in] UINT32 ulEntropyDataLength,
               [in, size_is(ulEntropyDataLength)] BYTE* rgbEntropyData);

   [helpstring("method GetCapability")]
```

```
   HRESULT GetCapability([in] TSS_FLAG CapArea,
                [in] UINT32 ulSubCapLength,
                [in, ptr, size_is(ulSubCapLength)] BYTE* rgbSubCap,
                [out] UINT32* pulRespDataLength,
                [out, size_is(, *pulRespDataLength)] BYTE** prgbRespData);

   [helpstring("method GetCapabilitySigned")]
   HRESULT GetCapabilitySigned([in] ITCPAKey* pKey,
                   [in] TSS_FLAG CapArea,
                   [in] UINT32 ulSubCapLength,
                   [in, ptr, size_is(ulSubCapLength)] BYTE* rgbSubCap,
                   [in, out, ptr] TSS_VALIDATION *pValidationData,
                   [out] UINT32* pulRespDataLength,
                   [out, size_is(, *pulRespDataLength)] BYTE** prgbRespData);
};

//ITCPAPcrEvent Interface
[
   object,
   uuid(FBCD9C31-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAPcrEvent Interface"),
   pointer_default(unique)
]
interface ITCPAPcrEvent : IUnknown
{
   [helpstring("method GetEvent")]
   HRESULT GetEvent([in] UINT32 ulPcrIndex,
            [in] UINT32 ulEventNumber,
            [out] TSS_PCR_EVENT* pPcrEvent);

   [helpstring("method GetEvents")]
   HRESULT GetEvents([in] UINT32 ulPcrIndex,
            [in] UINT32 ulStartNumber,
            [in, out] UINT32* pulEventNumber,
            [out, size_is(, *pulEventNumber)] TSS_PCR_EVENT** prgPcrEvents);

   [helpstring("method GetEventLog")]
```

```
   HRESULT GetEventLog([out] UINT32* pulEventNumber,
                       [out, size_is(, *pulEventNumber)] TSS_PCR_EVENT** prgPcrEvents);
};

//ITCPAPersistentStorage Interface
[
   object,
   uuid(FBCD9C1C-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAPersistentStorage Interface"),
   pointer_default(unique)
]
interface ITCPAPersistentStorage: IUnknown
{
   [helpstring("method LoadKeyByUUID")]
   HRESULT LoadKeyByUUID([in] TSS_FLAG persistentStorageType,
                [in] TSS_UUID uuidData,
                             [out] ITCPAKey** ppKey);

   [helpstring("method RegisterKey")]
   HRESULT RegisterKey([in] ITCPAKey* pKey,
               [in] TSS_FLAG persistentStorageType,
               [in] TSS_UUID uuidKey,
               [in] TSS_FLAG persistentStorageTypeParent,
               [in] TSS_UUID uuidParentKey);

   [helpstring("method UnregisterKey")]
   HRESULT UnregisterKey([in] TSS_FLAG persistentStorageType,
                [in] TSS_UUID uuidKey,
                [out] ITCPAKey** ppKey);

   [helpstring("method DeleteKeyByUUID")]
   HRESULT DeleteKeyByUUID([in] TSS_FLAG persistentStorageType,
               [in] TSS_UUID uuidData);

   [helpstring("method GetKeyByUUID")]
   HRESULT GetKeyByUUID([in] TSS_FLAG persistentStorageType,
               [in] TSS_UUID uuidData,
```

```
              [out] ITCPAKey** ppKey);

   [helpstring("method GetKeyByPublicInfo")]
   HRESULT GetKeyByPublicInfo([in] TSS_FLAG persistentStorageType,
                  [in] UINT32  ulAlgId,
                     [in] UINT32 ulPublicInfoLength,
                     [in, size_is(ulPublicInfoLength)] BYTE* rgbPublicInfo,
                     [out] ITCPAKey** ppKey);

         [helpstring("method GetRegisteredKeysByUUID")]
   HRESULT GetRegisteredKeysByUUID([in] TSS_FLAG ulPersistentStorageType,
                   [in] LPOLESTR wszKeyGuid,
                   [out] UINT32* pulKeyHierarchySize,
                   [out, size_is(, *pulKeyHierarchySize)] TSS_KM_KEYINFO** ppKeyHierarchy);
}

//ITCPAContext Interface
[
   object,
   uuid(FBCD9C1B-72CB-47BB-99DD-2317551491DE),

   helpstring("ITCPAContext Interface"),
   pointer_default(unique)
]
interface ITCPAContext : IUnknown
{
   [helpstring("method Connect")]
   HRESULT Connect([in, ptr] LPOLESTR wszDestination);

   [helpstring("method CreateObject")]
   HRESULT CreateObject([in] UINT32 ulObjectType,
               [in] UINT32 ulInitFlags,
               [out] IUnknown** ppUnkObject);

   [helpstring("method LoadKeyByBlob")]
   HRESULT LoadKeyByBlob([in] ITCPAKey* pUnwrappingKey,
                [in] UINT32 ulBlobLength,
                [in, size_is(ulBlobLength)] BYTE* rgbBlobData,
```

```
                    [out] ITCPAKey** ppKey);

      [helpstring("method GetTPMObject")]
      HRESULT GetTPMObject([out] ITCPATpm** ppTPMObject);

      [helpstring("method GetDefaultPolicy")]
      HRESULT GetDefaultPolicy([out] ITCPAPolicy** ppPolicyObject);

      [helpstring("method GetCapability")]
      HRESULT GetCapability([in] TSS_FLAG ulCapArea,
                    [in] UINT32 ulSubCapLength,
                    [in, ptr, size_is(ulSubCapLength)] BYTE* rgbSubCap,
                    [out] UINT32* pulRespDataLength,
                    [out, size_is(, *pulRespDataLength)] BYTE** prgbRespData);
   };


[
   uuid(FBCD9C19-72CB-47BB-99DD-2317551491DE),
   version(1.0),
   helpstring("TSS Service Provider 1.0 Type Library")
]
library TSPLib
{
   importlib("stdole32.tlb");
   importlib("stdole2.tlb");

      interface ITCPAContext;

   //TCPAContext Class
   [
      uuid(FBCD9C1A-72CB-47BB-99DD-2317551491DE),
      helpstring("TCPAContext Class")
   ]
   coclass TCPAContext
   {
      [default] interface ITCPAContext;
      interface ITCPAAttrib;
```

```
   interface ITCPAPersistentStorage;
};

   // ITCPACallback Interface
[
   uuid(FBCD9C1F-72CB-47BB-99DD-2317551491DE),

   helpstring("_ITCPACallback Interface"),
   pointer_default(unique)
]
interface _ITCPACallback
{
   [helpstring("method Tspicb_CallbackHMACAuth"), callback]
   HRESULT Tspicb_CallbackHMACAuth( [in] UINT32 PulAppData,
                     [in] IUnknown *PpAuthorizedObject,
                     [in] BOOL PfReturnOrVerify,
                     [in] UINT32 PulPendingFunction,
                     [in] BOOL PfContinueUse,
                     [in] UINT32 PulSizeNonces,
                     [in, size_is(PulSizeNonces)] BYTE* PrgbNonceEven,
                     [in, size_is(PulSizeNonces)] BYTE* PrgbNonceOdd,
                     [in, size_is(PulSizeNonces)] BYTE* PrgbNonceEvenOSAP,
                     [in, size_is(PulSizeNonces)] BYTE* PrgbNonceOddOSAP,
                     [in] UINT32 PulSizeDigestHmac,
                     [in, size_is(PulSizeDigestHmac)] BYTE* PrgbParamDigest,
                     [in, out, size_is(PulSizeDigestHmac)] BYTE* PrgbHmacData);

   [helpstring("method Tspicb_CallbackXorEnc"), callback]
   HRESULT Tspicb_CallbackXorEnc( [in] UINT32 PulAppData,
                    [in] IUnknown *PpOSAPObject,
                    [in] IUnknown *PpObject,
                    [in] BOOL PfPurposeSecret,
                    [in] UINT32   PulSizeNonces,
                    [in, size_is(PulSizeNonces)] BYTE* PrgbNonceEven,
                    [in, size_is(PulSizeNonces)] BYTE* PrgbNonceOdd,
                    [in, size_is(PulSizeNonces)] BYTE* PrgbNonceEvenOSAP,
                    [in, size_is(PulSizeNonces)] BYTE* PrgbNonceOddOSAP,
                    [in] UINT32 PulSizeEncAuth,
```

```
                    [out, size_is(PulSizeEncAuth)] BYTE* PrgbEncAuthUsage,
                    [out, size_is(PulSizeEncAuth)] BYTE* PrgbEncAuthMigration);

   [helpstring("method Tspicb_CallbackTakeOwnership"), callback]
   HRESULT Tspicb_CallbackTakeOwnership( [in] UINT32 PulAppData,
                        [in] IUnknown *PpObject,
                        [in] IUnknown *PpObjectPubKey,
                        [in] UINT32 ulSizeEncAuth,
                        [out, size_is(PulSizeEncAuth)] BYTE* PrgbEncAuth );

   [helpstring("method Tspicb_CallbackChangeAuthAsym"), callback]
   HRESULT Tspicb_CallbackChangeAuthAsym( [in] UINT32 PulAppData,
                         [in] IUnknown *PpObject,
                         [in] IUnknown *PpObjectPubKey,
                         [in] UINT32 PulSizeEncAuth,
                         [in] UINT32 PulSizeAuthLink,
                         [out, size_is(PulSizeEncAuth)] BYTE* PrgbEncAuth,
                         [out, size_is(PulSizeAuthLink)] BYTE* PrgbAuthLink);
};   // end of _ITCPACallback

//TCPAPolicy Class
[
   uuid(FBCD9C1D-72CB-47BB-99DD-2317551491DE),
   helpstring("TCPAPolicy Class"),
   noncreatable
]
coclass TCPAPolicy
{
   [default] interface ITCPAPolicy;
         [default, source] interface _ITCPACallback;
};

   // ITCPACallbackTpm Interface
[
   uuid(FBCD9C31-72CB-47BB-99DD-2317551491DE),

   helpstring("_ITCPACallbackTpm Interface"),
   pointer_default(unique)
```

```
]
interface _ITCPACallbackTpm
{
   [helpstring("method Tspicb_CallbackCollateIdentity"), callback]
   HRESULT Tspicb_CallbackCollateIdentity( [in] UINT32 PulAppData,
                  [in] UINT32 PulSessionKeyLength,
                  [in, size_is(PulSessionKeyLength)] BYTE* PrgbSessionKey,
                  [in] UINT32 PulTCPAPlainIdentityReqLength,
                  [in, size_is(PulTCPAPlainIdentityReqLength)] BYTE* PrgbTCPAPlainIdentityReq,
                  [out] UINT32* PulTCPAIdentityReqLength,
                  [out, size_is(, *PulTCPAIdentityReqLength)] BYTE** PrgbTCPAIdentityReq);

   [helpstring("method Tspicb_CallbackActivateIdentity"), callback]
   HRESULT Tspicb_CallbackActivateIdentity( [in] UINT32 PulAppData,
                   [in] UINT32 PulSessionKeyLength,
                  [in, size_is(PulSessionKeyLength)] BYTE* PrgbSessionKey,
                  [in] UINT32 PulSymCAAttestationBlobLength,
                  [in, size_is(PulSymCAAttestationBlobLength)] BYTE* PrgbSymCAAttestationBlob,
                  [out] UINT32* PulCredentialLength,
                  [out, size_is(, *PulCredentialLength)] BYTE** PrgbCredential);

};   // end of _ITCPACallbackTpm

//TCPATpm Class
[
   uuid(FBCD9C20-72CB-47BB-99DD-2317551491DE),
   helpstring("TCPATpm Class"),
   noncreatable
]
coclass TCPATpm
{
   [default] interface ITCPATpm;
         [default, source] interface _ITCPACallbackTpm;
   interface ITCPAAttrib;
   interface ITCPAAuth;
   interface ITCPAIntegrity;
   interface ITCPAAdministration;
         interface ITCPAIdentityCreation;
```

```
        interface ITCPAMaintenance;
        interface ITCPAPcrEvent;
};

//TCPAKey Class
[
   uuid(FBCD9C26-72CB-47BB-99DD-2317551491DE),
   helpstring("TCPAKey Class"),
   noncreatable
]
coclass TCPAKey
{
   [default] interface ITCPAKey;
   interface ITCPAAttrib;
   interface ITCPAAuth;
   interface ITCPAMigration;
};

//TCPAEncData Class
[
   uuid(FBCD9C28-72CB-47BB-99DD-2317551491DE),
   helpstring("TCPAEncData Class"),
   noncreatable
]
coclass TCPAEncData
{
   [default] interface ITCPAEncData;
   interface ITCPAAttrib;
   interface ITCPAAuth;
};

//TCPAHash Class
[
   uuid(FBCD9C2A-72CB-47BB-99DD-2317551491DE),
   helpstring("TCPAHash Class"),
   noncreatable
]
coclass TCPAHash
```

```
   {
      [default] interface ITCPAHash;
      interface ITCPAAttrib;
   };

   //TCPAPcrs Class
   [
      uuid(FBCD9C2C-72CB-47BB-99DD-2317551491DE),
      helpstring("TCPAPcrs Class"),
      noncreatable
   ]
   coclass TCPAPcrs
   {
      [default] interface ITCPAPcrs;
   };

};   // end of library TSPLib
```

+++tcs_typedef.h

```
/*++

Global typedefs for TSS Core Service

*/

#ifndef __TCS_TYPEDEF_H__
#define __TCS_TYPEDEF_H__


typedef UINT32  TCS_AUTHHANDLE;
typedef UINT32  TCS_CONTEXT_HANDLE;
typedef UINT32  TCS_KEY_HANDLE;

#endif // __TCS_TYPEDEF_H__
```

+++tcs_structs.h

```
/*++

TSS Core Service structures

*/

#ifndef __TCS_STRUCT_H__
#define __TCS_STRUCT_H__

// Errata: This should be named TSS_AUTH to avoid confussion with TPM structures
typedef struct tdTPM_AUTH
{
 TCS_AUTHHANDLE  AuthHandle;
 TCPA_NONCE   NonceOdd;   // system
 TCPA_NONCE   NonceEven;   // TPM
 TSS_BOOL   fContinueAuthSession;
 TCPA_AUTHDATA    HMAC;
} TPM_AUTH;


typedef struct tdTCS_LOADKEY_INFO
{
TSS_UUID   keyUUID;
TSS_UUID   parentKeyUUID;
TCPA_DIGEST   paramDigest; // SHA1 digest of the TPM_LoadKey
        // Command input parameters
        // As defined in TCPA Main
        // Specification
TPM_AUTH   authData;  // Data regarding a valid auth
        // Session including the
        // HMAC digest
} TCS_LOADKEY_INFO;

#endif // __TCS_STRUCT_H__
```

+++tcs_error.h

```
/*++

TSS Core Service error return codes

--*/

#ifndef __TCS_ERROR_H__
#define __TCS_ERROR_H__


#ifndef TSS_E_BASE
#define TSS_E_BASE      0x00000000L
#endif // TSS_E_BASE

//
// specific error codes returned by the TSS Core Service
// offset TSS_TCSI_OFFSET
//

// The context handle supplied is invalid.
#define TCS_E_INVALID_CONTEXTHANDLE  (UINT32)(TSS_E_BASE + 0x0C1L)

// The key handle supplied is invalid.
#define TCS_E_INVALID_KEYHANDLE  (UINT32)(TSS_E_BASE + 0x0C2L)

// The authorization session handle supplied is invalid.
#define TCS_E_INVALID_AUTHHANDLE  (UINT32)(TSS_E_BASE + 0x0C3L)

// the auth session has been closed by the TPM
#define TCS_E_INVALID_AUTHSESSION  (UINT32)(TSS_E_BASE + 0x0C4L)

// the key has been unloaded
#define TCS_E_INVALID_KEY   (UINT32)(TSS_E_BASE + 0x0C5L)

// Key addressed by the application key handle does not match the key addressed
// by the given UUID.
```

```
#define TCS_E_KEY_MISMATCH   (UINT32)(TSS_E_BASE + 0x0C8L)

// Key adressed by Key's UUID cannot be loaded because one of the required
// parent keys needs authorization.
#define TCS_E_KM_LOADFAILED   (UINT32)(TSS_E_BASE + 0x0CAL)

// The Key Cache Manager could not reload the key into the TPM.
#define TCS_E_KEY_CONTEXT_RELOAD  (UINT32)(TSS_E_BASE + 0x0CCL)

#endif // __TCS_ERROR_H__
```

+++tcsi.idl

```
/*++

Interface declarations for the TSS Core Service - COM interface for Windows based platforms

--*/

import "oaidl.idl";
import "ocidl.idl";
// header file for the basic TCPA data types

#include "tpm_defines.h"
#include "tpm_typedefs.h"
#include "tpm_struct.h"

#include "tss_typedef.h";
#include "tss_structs.h";

#include "tcs_typedef.h";
#include "tcs_structs.h";


[
 object,
 uuid(FBCD9C02-72CB-47BB-99DD-2317551491DE),
```

```
 helpstring("ITCSBase Interface"),
 pointer_default(unique)
]
interface ITCSBase : IUnknown
{

 [helpstring("method Tcsip_GetRandom")]
 HRESULT Tcsip_GetRandom(   [in, out] UINT32* bytesRequested,
       [out, size_is(,*bytesRequested)] BYTE** randomBytes);


 [helpstring("method Tcsip__StirRandom")]
 HRESULT TCSIP_StirRandom(   [in] UINT32 InDataSize,
       [in, size_is(InDataSize)] BYTE* InData);



 [helpstring("method Tcsi_GetCapability")]
 HRESULT Tcsi_GetCapability(  [in] TPM_CAPABILITY_AREA capArea,
       [in] UINT32 subCapSize,
       [in, ptr, size_is(subCapSize)] BYTE* subCap,
       [out] UINT32* respSize,
       [out, size_is(, *respSize)] BYTE** resp);
};


[
 object,
 uuid(FBCD9C0D-72CB-47BB-99DD-2317551491DE),
 helpstring("ITCSKey Interface"),
 pointer_default(unique)
]
interface ITCSKey : IUnknown
{
 [helpstring("method Tcsip_LoadKeyByBlob")]
 HRESULT Tcsip_LoadKeyByBlob(  [in] TCS_KEY_HANDLE hUnwrappingKey,
       [in] UINT32  cWrappedKeyBlobSize,
       [in, size_is(cWrappedKeyBlobSize)] BYTE* rgbWrappedKeyBlob,
```

```
      [in, out, ptr] TPM_AUTH* pAuth,
      [out] TCS_KEY_HANDLE* phKeyTCSI,
      [out] TCS_KEY_HANDLE* phKeyHMAC);

// The [in] and [out] attributes specify the direction in which the parameters are passed. A parameter can
// be defined as [in]-  // only, [out]-only, or [in, out].
// A related attribute, [in, out], indicates that the parameter is passed from the calling procedure to the
// called procedure and
// afterwards that the parameter acts as a pointer to data to be passed back from the called procedure to
// the calling procedure // this is a valid optimization for COM.
// For that reason the name for such type of parameters can differ from the parameter name in the
// specification.

 [helpstring("method Tcsip_CreateWrapKey")]
 HRESULT Tcsip_CreateWrapKey(  [in] TCS_KEY_HANDLE hWrappingKey,
      [in] TPM_ENCAUTH KeyUsageAuth,
      [in] TPM_ENCAUTH KeyMigrationAuth,
      [in, out] TCPA_UINT32* pcKeySize,
      [in, out, size_is(, *pcKeySize)] BYTE** prgbKey,
      [in, out, ptr] TCPA_AUTH* pAuth);

 [helpstring("method Tcsip_GetPubKey")]
 HRESULT Tcsip_GetPubKey(   [in] TCS_KEY_HANDLE hKey,
      [in, out, ptr] TPM_AUTH* pAuth,
      [out] UINT32* pcPubKeySize,
      [out, size_is(, *pcPubKeySize)] BYTE** prgbPubKey);

 [helpstring("method Tcsip_EvictKey")]
 HRESULT Tcsip_EvictKey([in] TCS_KEY_HANDLE hKey);

 [helpstring("method Tcsip_CertifyKey")]
 HRESULT Tcsip_CertifyKey(  [in] TCS_KEY_HANDLE certHandle,
```

```
        [in] TCS_KEY_HANDLE keyHandle,
        [in] TCPA_NONCE antiReplay,
        [in, out] TPM_AUTH* certAuth,
        [in, out] TPM_AUTH* keyAuth,
        [out] UINT32* CertifyInfoSize,
        [out, size_is(, *CertifyInfoSize)] BYTE** CertifyInfo,
        [out] UINT32* outDataSize,
        [out, size_is(, *outDataSize)] BYTE** outData);

};

[
 object,
 uuid(FBCD9C0E-72CB-47BB-99DD-2317551491DE),
 helpstring("ITCSKeyManage Interface"),
 pointer_default(unique)
]
interface ITCSKeyManage : IUnknown
{
 [helpstring("method Tcsip_LoadKeyByUUID")]
 HRESULT Tcsip_LoadKeyByUUID(  [in] TSS_UUID KeyUUID,
       [in, out, ptr] TCS_LOADKEY_INFO* pLoadKeyInfo,
       [out] TCS_KEY_HANDLE* phKeyTCSI);

 [helpstring("method Tcsi_RegisterKey")]
 HRESULT Tcsi_RegisterKey(  [in] TSS_UUID WrappingKeyUUID,
       [in] TSS_UUID KeyUUID,
       [in] UINT32 cKeySize,
       [in, size_is(cKeySize)] BYTE* rgbKey,
       [in, defaultvalue(0)] UINT32 cVendorData,
       [in, ptr, size_is(cVendorData), defaultvalue(0)] BYTE* rgbVendorData);

 [helpstring("method Tcsi_UnregisterKey")]
 HRESULT Tcsi_UnregisterKey(  [in] TSS_UUID KeyUUID);


 [helpstring("method Tcsi_EnumRegisteredKeys")]
 HRESULT Tcsi_EnumRegisteredKeys( [in, ptr] TSS_UUID* pKeyUUID,
```

```
      [out] UINT32* pcKeyHierarchySize,
      [out, size_is(, *pcKeyHierarchySize)] TSS_KM_KEYINFO** ppKeyHierarchy);

 [helpstring("method Tcsi_GetRegisteredKey")]
 HRESULT Tcsi_GetRegisteredKey(  [in] TSS_UUID KeyUUID,
      [out] TSS_KM_KEYINFO** ppKeyInfo);

 [helpstring("method Tcsi_GetRegisteredKeyBlob")]
 HRESULT Tcsi_GetRegisteredKeyBlob( [in] TSS_UUID KeyUUID,
      [out] UINT32* pcKeySize,
      [out, size_is(, *pcKeySize)] BYTE** prgbKey);

 [helpstring("method Tcsip_GetRegisteredKeyByPublicInfo ")]
 HRESULT Tcsip_GetRegisteredKeyByPublicInfo( [in] TPM_ALGORITHM_ID algID,
      [in] UINT32 ulPublicInfoLength,
      [in] BYTE* rgbPublicInfo,
      [out] UINT32* keySize,
      [out, size_is(, *keySize)] BYTE** keyBlob);

};


[
 object,
 uuid(FBCD9C05-72CB-47BB-99DD-2317551491DE),
 helpstring("ITCSCryptography Interface"),
 pointer_default(unique)
]
interface ITCSCryptography : IUnknown
{
 [helpstring("method Tcsip_Sign")]
 HRESULT Tcsip_Sign(   [in] TCS_KEY_HANDLE keyHandle,
      [in] UINT32 areaToSignSize,
      [in, size_is(areaToSignSize)] BYTE* areaToSign,
      [in, out, ptr] TPM_AUTH* privAuth,
      [out] UINT32* sigSize,
      [out, size_is(, *sigSize)] BYTE** sig);
```

```
 [helpstring("method Tcsip_Unbind")]
 HRESULT Tcsip_Unbind(   [in] TCS_KEY_HANDLE keyHandle,
       [in] UINT32 inDataSize,
       [in, size_is(inDataSize)] BYTE* inData,
       [in, out, ptr] TPM_AUTH* privAuth,
       [out] UINT32* outDataSize,
       [out, size_is(, *outDataSize)] BYTE** outData);

 [helpstring("method Tcsip_Seal")]
 HRESULT Tcsip_Seal(   [in] TCS_KEY_HANDLE keyHandle,
       [in] TPM_ENCAUTH encAuth,
       [in] UINT32 pcrInfoSize,
       [in, ptr, size_is(pcrInfoSize)] BYTE* PcrInfo,
       [in] UINT32 inDataSize,
       [in, size_is(inDataSize)] BYTE* inData,
       [in, out, ptr] TPM_AUTH* pubAuth,
       [out] UINT32* SealedDataSize,
       [out, size_is(, *SealedDataSize)] BYTE** SealedData);

 [helpstring("method Tcsip_Unseal")]
 HRESULT Tcsip_Unseal(   [in] TCS_KEY_HANDLE parentHandle,
       [in] UINT32 SealedDataSize,
       [in, size_is(SealedDataSize)] BYTE* SealedData,
       [in, out, ptr] TPM_AUTH* parentAuth,
       [in, out, ptr] TPM_AUTH* dataAuth,
       [out] UINT32* DataSize,
       [out, size_is(, *DataSize)] BYTE** Data);

};


[
 object,
 uuid(FBCD9C0F-72CB-47BB-99DD-2317551491DE),
 helpstring("ITCSAuthorization Interface"),
 pointer_default(unique)
]
interface ITCSAuthorization : IUnknown
```

```
{
 [helpstring("method Tcsip_ChangeAuth")]
 HRESULT Tcsip_ChangeAuth(  [in] TCS_KEY_HANDLE parentHandle,
       [in] TPM_PROTOCOL_ID protocolID,
       [in] TPM_ENCAUTH newAuth,
       [in] TPM_ENTITY_TYPE entityType,
       [in] UINT32 encDataSize,
       [in, size_is(encDataSize)] BYTE* encData,
       [in, out, ptr] TPM_AUTH* ownerAuth,
       [in, out, ptr] TPM_AUTH* entityAuth,
       [out] UINT32* outDataSize,
       [out, size_is(, *outDataSize)] BYTE** outData);

 [helpstring("method Tcsip_ChangeAuthOwner")]
 HRESULT Tcsip_ChangeAuthOwner(  [in] TPM_PROTOCOL_ID protocolID,
       [in] TPM_ENCAUTH newAuth,
       [in] TPM_ENTITY_TYPE entityType,
       [in, out, ptr] TPM_AUTH* ownerAuth);


 [helpstring("method Tcsip_OIAP")]
 HRESULT Tcsip_OIAP(   [out] TCS_AUTHHANDLE* authHandle,
       [out] TCPA_NONCE* nonce0);

 [helpstring("method Tcsip_OSAP")]
 HRESULT Tcsip_OSAP(   [in] TPM_ENTITY_TYPE entityType,
       [in] UINT32 entityValue,
       [in] TCPA_NONCE nonceOddOSAP,
       [out] TCS_AUTHHANDLE* authHandle,
       [out] TCPA_NONCE* nonceEven,
       [out] TCPA_NONCE* nonceEvenOSAP);

 [helpstring("method Tcsip_TerminateHandle")]
 HRESULT Tcsip_TerminateHandle(  [in] TCS_AUTHHANDLE handle);
```

```
        [helpstring("method Tcsip_ChangeAuthAsymStart")]
        HRESULT Tcsip_ChangeAuthAsymStart(   [in]TCS_KEY_HANDLE idHandle,
           [in]TCPA_NONCE antiReplay,
           [in]UINT32 TempKeyInfoSize,
           [in, size_is( TempKeyInfoSize )]BYTE* TempKeyInfoData,
           [in, out]TPM_AUTH* pAuth,
           [out]UINT32* TempKeySize,
              [out, size_is(, *TempKeySize )]BYTE** TempKeyData,
              [out]UINT32* CertifyInfoSize,
              [out, size_is(, *CertifyInfoSize )]BYTE** CertifyInfo,
              [out]UINT32* sigSize,
              [out, size_is(, *sigSize )]BYTE** sig,
              [out]TCS_KEY_HANDLE* ephHandle);


        [helpstring("method Tcsip_ChangeAuthAsymFinish")]
        HRESULT Tcsip_ChangeAuthAsymFinish( [in]TCS_KEY_HANDLE parentHandle,
           [in]TCS_KEY_HANDLE ephHandle,
              [in]TPM_ENTITY_TYPE entityType,
              [in]TPM_HMAC newAuthLink,
              [in]UINT32 newAuthSize,
              [in, size_is( newAuthSize )]BYTE* encNewAuth,
              [in]UINT32 encDataSizeIn,
              [in, size_is( encDataSizeIn )]BYTE* encDataIn,
              [in, out]TPM_AUTH* ownerAuth,
              [out]UINT32* encDataSizeOut,
              [out, size_is(, *encDataSizeOut )]BYTE** encDataOut,
              [out]TCPA_NONCE* saltNonce,
              [out]TCPA_DIGEST* changeProof);

};

[
 object,
 uuid(FBCD9C07-72CB-47BB-99DD-2317551491DE),
 helpstring("ITCSIntegrity Interface"),
 pointer_default(unique)
]
```

```
interface ITCSIntegrity : IUnknown
{
 [helpstring("method Tcsip_Extend")]
 HRESULT Tcsip_Extend(   [in] TPM_PCRINDEX pcrNum,
       [in] TCPA_DIGEST inDigest,
       [out] TPM_PCRVALUE* outDigest);

 [helpstring("method Tcsip_PcrRead")]
 HRESULT Tcsip_PcrRead(   [in] TPM_PCRINDEX pcrNum,
       [out] TPM_PCRVALUE* outDigest);

 [helpstring("method Tcsip_DirWriteAuth")]
 HRESULT Tcsip_DirWriteAuth(  [in] TPM_DIRINDEX dirIndex,
       [in] TPM_DIRVALUE newContents,
       [in, out, ptr] TPM_AUTH* ownerAuth);

 [helpstring("method Tcsip_DirRead")]
 HRESULT Tcsip_DirRead(   [in] TPM_DIRINDEX dirIndex,
       [out] TPM_DIRVALUE* dirValue);



// The [in] and [out] attributes specify the direction in which the parameters are passed. A parameter can
// be defined as [in]-  // only, [out]-only, or [in, out].
// A related attribute, [in, out], indicates that the parameter is passed from the calling procedure to the
// called procedure and
// afterwards that the parameter acts as a pointer to data to be passed back from the called procedure to
// the calling procedure // this is a valid optimization for COM.
// For that reason the name for such type of parameters can differ from the parameter name in the
// specification.



 [helpstring("method Tcsip_Quote")]
```

```
      HRESULT Tcsip_Quote(
         [in]TCS_KEY_HANDLE      keyHandle,                          // in
         [in]TCPA_NONCE          antiReplay,                         // in
         [in, out]UINT32*        pcrTargetSize,                      // in, out
         [in, out, size_is(, *pcrTargetSize)]BYTE**   pcrTarget,     // in, out
         [in, out, ptr]TPM_AUTH* privAuth,                           // in, out
         [out]UINT32*            sigSize,                            // out
         [out, size_is(, *sigSize)]BYTE** ppSignature);              // out
};


[
 object,
 uuid(FBCD9C10-72CB-47BB-99DD-2317551491DE),
 helpstring("ITCSTpm Interface"),
 pointer_default(unique)
]
interface ITCSTpm : IUnknown
{

// The [in] and [out] attributes specify the direction in which the parameters are passed. A parameter can
// be defined as [in]-  // only, [out]-only, or [in, out].
// A related attribute, [in, out], indicates that the parameter is passed from the calling procedure to the
// called procedure and
// afterwards that the parameter acts as a pointer to data to be passed back from the called procedure to
// the calling procedure // this is a valid optimization for COM.
// For that reason the name for such type of parameters can differ from the parameter name in the
// specification.

 [helpstring("method Tcsip_CreateEndorsementKeyPair")]
 HRESULT Tcsip_CreateEndorsementKeyPair([in] TCPA_NONCE antiReplay,                    // in
      [in, out] UINT32*             endorsementKeyInfoSize,                          // in, out
      [in, out, size_is(, *endorsementKeyInfoSize )]BYTE**  endorsementKeyInfo,      // in, out
      [out] TCPA_DIGEST*                  checksum);                                 // out

 [helpstring("method Tcsip_DisableForceClear")]
 HRESULT Tcsip_DisableForceClear();
```

```
[helpstring("method Tcsip_DisablePubekRead")]
HRESULT Tcsip_DisablePubekRead([in, out, ptr] TPM_AUTH* ownerAuth);

[helpstring("method Tcsip_DisableOwnerClear")]
HRESULT Tcsip_DisableOwnerClear([in, out, ptr] TPM_AUTH* ownerAuth);

[helpstring("method Tcsip_ForceClear")]
HRESULT Tcsip_ForceClear();

[helpstring("method Tcsip_GetCapability")]
HRESULT Tcsip_GetCapability([in] TPM_CAPABILITY_AREA capArea,
       [in] UINT32 subCapSize,
       [in, ptr, size_is(subCapSize)] BYTE* subCap,
       [out] UINT32* respSize,
       [out, size_is(, *respSize)] BYTE** resp);

[helpstring("method Tcsip_GetCapabilityOwner")]
HRESULT Tcsip_GetCapabilityOwner([in, out, ptr] TPM_AUTH* pOwnerAuth,
       [out] TCPA_VERSION* pVersion,
       [out] UINT32* pNonVolatileFlags,
       [out] UINT32* pVolatileFlags);

[helpstring("method Tcsip_GetTestResult")]
HRESULT Tcsip_GetTestResult([out] UINT32* outDataSize,
       [out, size_is(, *outDataSize)] BYTE** outData);

[helpstring("method Tcsip_OwnerClear")]
HRESULT Tcsip_OwnerClear([in, out, ptr] TPM_AUTH* ownerAuth);

[helpstring("method Tcsip_OwnerReadPubek")]
HRESULT Tcsip_OwnerReadPubek([in, out, ptr] TPM_AUTH* ownerAuth,
        [out] UINT32* pubEndorsementKeySize,
        [out, size_is(, *pubEndorsementKeySize)] BYTE** pubEndorsementKey);

[helpstring("method Tcsip_OwnerSetDisable")]
HRESULT Tcsip_OwnerSetDisable([in] TSS_BOOL disableState,
         [in, out, ptr] TPM_AUTH* ownerAuth);
```

```
 [helpstring("method Tcsip_PhysicalDisable")]
 HRESULT Tcsip_PhysicalDisable();

 [helpstring("method Tcsip_PhysicalEnable")]
 HRESULT Tcsip_PhysicalEnable();

 [helpstring("method Tcsip_PhysicalPresence")]
 HRESULT Tcsip_PhysicalPresence([in] TPM_PHYSICAL_PRESENCE fPhysicalPresence);

 [helpstring("method Tcsip_PhysicalSetDeactivated")]
 HRESULT Tcsip_PhysicalSetDeactivated([in] TSS_BOOL state);

 [helpstring("method Tcsip_ReadPubek")]
 HRESULT Tcsip_ReadPubek([in] TCPA_NONCE antiReplay,
         [out] UINT32* pubEndorsementKeySize,
         [out, size_is(, *pubEndorsementKeySize)] BYTE** pubEndorsementKey,
         [out] TCPA_DIGEST* checksum);

 [helpstring("method Tcsip_SetOwnerInstall")]
 HRESULT Tcsip_SetOwnerInstall([in] TSS_BOOL state);

 [helpstring("method Tcsip_SetTempDeactivated")]
 HRESULT Tcsip_SetTempDeactivated();


// The [in] and [out] attributes specify the direction in which the parameters are passed. A parameter can
// be defined as [in]-  // only, [out]-only, or [in, out].
// A related attribute, [in, out], indicates that the parameter is passed from the calling procedure to the
// called procedure and
// afterwards that the parameter acts as a pointer to data to be passed back from the called procedure to
// the calling procedure // this is a valid optimization for COM.
// For that reason the name for such type of parameters can differ from the parameter name in the
// specification.

 [helpstring("method Tcsip_TakeOwnership")]
 HRESULT Tcsip_TakeOwnership([in] UINT16 protocolID,                    // in
        [in] UINT32 encOwnerAuthSize,                                   // in
        [in, size_is(encOwnerAuthSize)] BYTE* encOwnerAuth,             // in
```

```
        [in] UINT32 encSrkAuthSize,                                     // in
        [in, ptr, size_is(encSrkAuthSize)] BYTE* encSrkAuth,            // in
        [in, out]UINT32*                       srkKeyInfoSize,          // in, out
        [in, out, size_is(, *srkKeyInfoSize )]BYTE** srkKeyInfo,        // in, out
        [in, out, ptr]TPM_AUTH*       ownerAuth);                       // in, out


[helpstring("method Tcsip_FieldUpgrade")]
HRESULT Tcsip_FieldUpgrade([in] UINT32 dataInSize,
      [in, ptr, size_is(dataInSize)] BYTE* dataIn,
      [in, out, ptr] TPM_AUTH* ownerAuth,
      [out] UINT32* dataOutSize,
      [out, size_is(, *dataOutSize)] BYTE** dataOut);

[helpstring("method Tcsip_SetRedirection")]
HRESULT Tcsip_SetRedirection([in] TCS_KEY_HANDLE keyHandle,
```

```
        [in] UINT32 c1,
        [in] UINT32 c2,
        [in, out] TPM_AUTH* privAuth);

 [helpstring("method Tcsip_GetCapabilitySigned")]
 HRESULT Tcsip_GetCapabilitySigned([in] TCS_KEY_HANDLE keyHandle,
        [in] TCPA_NONCE antiReplay,
        [in] TPM_CAPABILITY_AREA capArea,
        [in] UINT32 subCapSize,
        [in, size_is(subCapSize)] BYTE* subCap,
        [in, out] TPM_AUTH* privAuth,
        [out] TCPA_VERSION* Version,
        [out] UINT32* respSize,
        [out, size_is(, *respSize)] BYTE** resp,
        [out] UINT32* sigSize,
        [out, size_is(, *sigSize)] BYTE** sig);

 [helpstring("method Tcsip_CertifySelfTest")]
 HRESULT Tcsip_CertifySelfTest([in] TCS_KEY_HANDLE keyHandle,
        [in] TCPA_NONCE antiReplay,
        [in, out] TPM_AUTH* privAuth,
        [out] UINT32* sigSize,
        [out, size_is(, *sigSize)] BYTE** sig);

 [helpstring("method Tcsip_SelfTestFull")]
 HRESULT Tcsip_SelfTestFull();

 [helpstring("method Tcsip_ContinueSelfTest")]
 HRESULT Tcsip_ContinueSelfTest();
};

[
 object,
 uuid(FBCD9C09-72CB-47BB-99DD-2317551491DE),
 helpstring("ITCSIdentityKey Interface"),
 pointer_default(unique)
]
interface ITCSIdentityKey : IUnknown
```

```
{

// The [in] and [out] attributes specify the direction in which the parameters are passed. A parameter can
// be defined as [in]-  // only, [out]-only, or [in, out].
// A related attribute, [in, out], indicates that the parameter is passed from the calling procedure to the
// called procedure and
// afterwards that the parameter acts as a pointer to data to be passed back from the called procedure to
// the calling procedure // this is a valid optimization for COM.
// For that reason the name for such type of parameters can differ from the parameter name in the
// specification.

 [helpstring("method Tcsip_MakeIdentity")]
 HRESULT Tcsip_MakeIdentity([in] TPM_ENCAUTH IdentityAuth,
       [in] TPM_CHOSENID_HASH IDLabel_PrivCAHash,
       [in, out]UINT32*       idIdentityKeyInfoSize,                                  // in, out
       [in, out, size_is(, *idIdentityKeyInfoSize )]BYTE**  idIdentityKeyInfo,      // in, out
       [in, out, ptr] TPM_AUTH* pSrkAuth,
       [in, out, ptr] TPM_AUTH* pOwnerAuth,
       [out] UINT32* pcIdentityBindingSize,
       [out, size_is(, *pcIdentityBindingSize)] BYTE** prgbIdentityBinding,
       [out] UINT32* pcEndorsementCredentialSize,
       [out, size_is(, *pcEndorsementCredentialSize)] BYTE** prgbEndorsementCredential,
       [out] UINT32* pcPlatformCredentialSize,
       [out, size_is(, *pcPlatformCredentialSize)] BYTE** prgbPlatformCredential,
       [out] UINT32* pcConformanceCredentialSize,
       [out, size_is(, *pcConformanceCredentialSize)] BYTE** prgbConformanceCredential);


 [helpstring("method Tcsip_ActivateTPMIdentity")]
 HRESULT Tcsip_ActivateTPMIdentity([in] TCS_KEY_HANDLE idKey,
       [in] UINT32 blobSize,
       [in, size_is(blobSize)] BYTE* blob,
       [in, out] TPM_AUTH* idKeyAuth,
       [in, out] TPM_AUTH* ownerAuth,
       [out] UINT32* SymmetricKeySize,
       [out, size_is(, *SymmetricKeySize)] BYTE** SymmetricKey);
```

```
};


[
 object,
 uuid(FBCD9C0A-72CB-47BB-99DD-2317551491DE),
 helpstring("ITCSMigration Interface"),
 pointer_default(unique)
]
interface ITCSMigration : IUnknown
{
 [helpstring("method Tcsip_AuthorizeMigrationKey")]
 HRESULT Tcsip_AuthorizeMigrationKey([in] TPM_MIGRATE_SCHEME migrateScheme,
           [in] UINT32 MigrationKeySize,
           [in, size_is(MigrationKeySize)] BYTE* MigrationKey,
           [in, out, ptr] TPM_AUTH* ownerAuth,
           [out] UINT32* MigrationKeyAuthSize,
           [out, size_is(, *MigrationKeyAuthSize)] BYTE** MigrationKeyAuth);

 [helpstring("method Tcsip_ConvertMigrationBlob")]
 HRESULT Tcsip_ConvertMigrationBlob([in] TCS_KEY_HANDLE parentHandle,
          [in] UINT32 inDataSize,
          [in, size_is(inDataSize)] BYTE* inData,
          [in] UINT32 randomSize,
          [in, size_is(randomSize)] BYTE* random,
          [in, out, ptr] TPM_AUTH* parentAuth,
          [out] UINT32* outDataSize,
          [out, size_is(, *outDataSize)] BYTE** outData);

 [helpstring("method Tcsip_CreateMigrationBlob")]
 HRESULT Tcsip_CreateMigrationBlob([in] TCS_KEY_HANDLE parentHandle,
         [in] TPM_MIGRATE_SCHEME migrationType,
         [in] UINT32 MigrationKeyAuthSize,
         [in, size_is(MigrationKeyAuthSize)] BYTE* MigrationKeyAuth,
         [in] UINT32 encDataSize,
         [in, size_is(encDataSize)] BYTE* encData,
         [in, out, ptr] TPM_AUTH* parentAuth,
         [in, out, ptr] TPM_AUTH* entityAuth,
```

```
        [out] UINT32* randomSize,
        [out, size_is(, *randomSize)] BYTE** random,
        [out] UINT32* outDataSize,
        [out, size_is(, *outDataSize)] BYTE** outData);
};


[
 object,
 uuid(FBCD9C0B-72CB-47BB-99DD-2317551491DE),
 helpstring("ITCSEventManager Interface"),
 pointer_default(unique)
]
interface ITCSEventManager : IUnknown
{
```

```
      [helpstring("method Tcsi_LogPcrEvent")]
      HRESULT Tcsi_LogPcrEvent([in] TSS_PCR_EVENT Event,
          [out] UINT32* pNumber);


      [helpstring("method Tcsi_GetPcrEvent")]
      HRESULT Tcsi_GetPcrEvent([in] UINT32 PcrIndex,
           [in, out] UINT32* pNumber,
           [out] TSS_PCR_EVENT**  ppEvent);



      [helpstring("method Tcsi_GetPcrEventsByPcr")]
      HRESULT Tcsi_GetPcrEventByPcr([in] UINT32 PcrIndex,
           [in] UINT32 FirstEvent,
           [in, out] UINT32* pEventCount,
           [out, size_is(, *pEventCount)] TSS_PCR_EVENT** ppEvents);


      [helpstring("method Tcsi_GetPcrEventLog")]
      HRESULT Tcs_GetPcrEventLog([out] UINT32* pEventCount,
           [out, size_is(, *pEventCount)] TSS_PCR_EVENT** ppEvents);


};


[
 object,
 uuid(FBCD9C0C-72CB-47BB-99DD-2317551491DE),
 helpstring("ITCSMaintenance Interface"),
 pointer_default(unique)
]
interface ITCSMaintenance : IUnknown
{
      [helpstring("method Tcsip_CreateMaintenanceArchive")]
      HRESULT Tcsip_CreateMaintenanceArchive([in] TSS_BOOL generateRandom,
```

```
            [in, out] TPM_AUTH* ownerAuth,
            [out] UINT32* randomSize,
            [out, size_is(, *randomSize)] BYTE** random,
            [out] UINT32* archiveSize,
            [out, size_is(, *archiveSize)] BYTE** archive);

        [helpstring("method Tcsip_LoadMaintenanceArchive")]
        HRESULT Tcsip_LoadMaintenanceArchive([in] UINT32 dataInSize,
            [in, size_is(dataInSize)]] BYTE* dataIn,
            [in, out] TPM_AUTH* ownerAuth,
            [out] UINT32* dataOutSize,
            [out, size_is(, *dataOutSize)] BYTE** dataOut);

        [helpstring("method Tcsip_KillMaintenanceFeature")]
        HRESULT Tcsip_KillMaintenanceFeature([in, out] TPM_AUTH* ownerAuth);

        [helpstring("method Tcsi_LoadManuMaintPub")]
           HRESULT Tcsip_LoadManuMaintPub (
           [in] TCPA_NONCE                     antiReplay,
           [in] UINT32                        PubKeySize,
           [in, size_is(PubKeySize)] BYTE*   PubKey,
           [out] TCPA_DIGEST*                  checksum
        );

        [helpstring("method Tcsip_ReadManuMaintPub")]
        HRESULT Tcsip_ReadManuMaintPub ([in] TCPA_NONCE antiReplay,
            [out] TCPA_DIGEST* checksum);

};


// TSSCORELib
[
 uuid(FBCD9C00-72CB-47BB-99DD-2317551491DE),
 version(1.0),
 helpstring("TSSCore 1.0 Type Library")
]
library TSSCORELib
```

```
{
 importlib("stdole32.tlb");
 importlib("stdole2.tlb");
 [
  uuid(FBCD9C01-72CB-47BB-99DD-2317551491DE),
  helpstring("TSSCoreService Class")
 ]
 coclass TSSCoreService
 {
  [default] interface ITCSBase;
  interface ITCSKey;
  interface ITCSKeyManage;
  interface ITCSCryptography;
  interface ITCSAuthorization;
  interface ITCSIntegrity;
  interface ITCSTpm;
  interface ITCSIdentityKey;
  interface ITCSMigration;
  interface ITCSEventManager;
  interface ITCSMaintenance;
 };
};
```

+++tddlapi_error.h

```
/*++

TDDL error return codes for the TPM Device Driver Library Interface (TDDLI)

--*/

#ifndef __TDDLAPI_ERROR_H__
#define __TDDLAPI_ERROR_H__


//
// error coding scheme for a Microsoft Windows platform -
```

```
// refer to the TSS Specification Parts
//
//  Values are 32 bit values layed out as follows:
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//  +---+-+-+-----------------------+-------+-----------------------+
//  |Lev|C|R|     Facility          | Layer |         Code          |
//  +---+-+-+-----------------------+-------+-----------------------+
//  | Platform specific coding      | TSS error coding system       |
//  +---+-+-+-----------------------+-------+-----------------------+
//
//      Lev - is the Level code
//
//          00 - Success
//          01 - Informational
//          10 - Warning
//          11 - Error
//
//      C - is the Customer code flag  (must actually be set)
//
//      R - is a reserved bit    (unused)
//
//      Facility - is the facility code: TCPA: proposal 0x028
//
//      Code - is the facility's status code
//


// no macros are used below intentionally
// for a better error code recognition by the reader

// note that the values of TPM_E_BASE and TSS_E_BASE, TSS_W_BASE and TSS_I_BASE
// have to be adjusted for a platform other than Windows

//
// TPM specific error codes (layer nibble set to TPM layer TSS_LAYER_TPM)
//
```

```
#endif // __TDDLAPI_ERROR_H__
```

+++tddl_error.h

```
/*++

TPM Device Driver Library error return codes

--*/

#ifndef __TDDL_ERROR_H__
#define __TDDL_ERROR_H__


#ifndef TSS_E_BASE
#define TSS_E_BASE      0x00000000L
#endif // TSS_E_BASE


//
// specific error codes returned by the TPM device driver library
// offset TSS_TDDL_OFFSET
//
#define TDDL_E_FAIL     TSS_E_FAIL
#define TDDL_E_TIMEOUT  TSS_E_TIMEOUT

// The connection was already established.
#define TDDL_E_ALREADY_OPENED   (UINT32)(TSS_E_BASE + 0x081L)

// The device was not connected.
#define TDDL_E_ALREADY_CLOSED   (UINT32)(TSS_E_BASE + 0x082L)

// The receive buffer is too small.
#define TDDL_E_INSUFFICIENT_BUFFER  (UINT32)(TSS_E_BASE + 0x083L)
```

```
// The command has already completed.
#define TDDL_E_COMMAND_COMPLETED  (UINT32)(TSS_E_BASE + 0x084L)

// TPM aborted processing of command.
#define TDLL_E_COMMAND_ABORTED  (UINT32)(TSS_E_BASE + 0x085L)

//  The request could not be performed because of an I/O device error.
#define TDDL_E_IOERROR    (UINT32)(TSS_E_BASE + 0x087L)

// Unsupported TAG is requested
#define TDDL_E_BADTAG    (UINT32)(TSS_E_BASE + 0x088L)

// the requested TPM component was not found
#define TDDL_E_COMPONENT_NOT_FOUND  (UINT32)(TSS_E_BASE + 0x089L)


#endif // __TDDL_ERROR_H__
```

+++tddli.h

```
/*++

TPM Device Driver Library interface

--*/

#ifndef __TDDLI_H__
#define __TDDLI_H__

#ifdef _WINDOWS_

// --- This should be used on Windows platforms
#ifdef TDDLI_EXPORTS
#define TDDLI __declspec(dllexport)
#else
#define TDDLI __declspec(dllimport)
#endif
```

```
#endif


// Errata: Change spec from TCPA_CAP_PROP_MANUFACTURER to TPM_CAP_PROP_MANUFACTURER
#define TDDL_CAP_VERSION   0x0100
#define TDDL_CAP_VER_DRV   0x0101
#define TDDL_CAP_VER_FW    0x0102
#define TDDL_CAP_VER_FW_DATE   0x0103

#define TDDL_CAP_PROPERTY   0x0200
#define TDDL_CAP_PROP_MANUFACTURER  0x0201
#define TDDL_CAP_PROP_MODULE_TYPE  0x0202
#define TDDL_CAP_PROP_GLOBAL_STATE  0x0203

//------------------------------------------------------------------
// TDDL specific helper redefinitions

#ifdef __cplusplus
extern "C" {

   //establish a connection to the TPM device driver
   TDDLI TSS_RESULT Tddli_Open();

   //close a open connection to the TPM device driver
   TDDLI TSS_RESULT Tddli_Close();

   //cancels the last outstanding TPM command
   TDDLI TSS_RESULT Tddli_Cancel();

   // read the attributes returned by the TPM HW/FW
   TDDLI TSS_RESULT Tddli_GetCapability(
                           UINT32        CapArea,
                           UINT32        SubCap,
                           BYTE*         pCapBuf,
                           UINT32*       puntCapBufLen
  );
```

```
   // set parameters to the TPM HW/FW
   TDDLI TSS_RESULT Tddli_SetCapability(
                           UINT32        CapArea,
                           UINT32        SubCap,
                           BYTE*         pCapBuf,
                           UINT32*       puntCapBufLen
   );

   // get status of the TPM driver and device
   TDDLI TSS_RESULT Tddli_GetStatus(
                           UINT32        ReqStatusType,
                           UINT32*       puntStatus
   );

   // send any data to the TPM module
   TDDLI TSS_RESULT Tddli_TransmitData(
                           BYTE*         pTransmitBuf,
                           UINT32        TransmitBufLen,
                           BYTE*         pReceiveBuf,
                           UINT32*       puntReceiveBufLen
  );
}
#else

   //establish a connection to the TPM device driver
   extern TDDLI TSS_RESULT Tddli_Open();

   //close a open connection to the TPM device driver
   extern TDDLI TSS_RESULT Tddli_Close();

   //cancels the last outstanding TPM command
   extern TDDLI TSS_RESULT Tddli_Cancel();

   // read the attributes returned by the TPM HW/FW
   extern TDDLI TSS_RESULT Tddli_GetCapability(
                           UINT32       CapArea,
                           UINT32       SubCap,
                           BYTE*        pCapBuf,
```

```
                            UINT32*      puntCapBufLen
   );

   // set parameters to the TPM HW/FW
   extern TDDLI TSS_RESULT Tddli_SetCapability(
                            UINT32       CapArea,
                            UINT32       SubCap,
                            BYTE*        pCapBuf,
                            UINT32*      puntCapBufLen
   );

   // get status of the TPM driver and device
   extern TDDLI TSS_RESULT Tddli_GetStatus(
                            UINT32       ReqStatusType,
                            UINT32*      puntStatus
   );

   // send any data to the TPM module
   extern TDDLI TSS_RESULT Tddli_TransmitData(
                            BYTE*        pTransmitBuf,
                            UINT32       TransmitBufLen,
                            BYTE*        pReceiveBuf,
                            UINT32*      puntReceiveBufLen
   );
#endif

#endif // __TDDLI_H__
```

+++main.c

```
/* Test compile TSS files
   Monty Wiseman
   Aug 25, 2003
*/

#include <stdio.h>
#include <sys/types.h>
```

```
#include "platform.h"
#include "tcpa_typedef.h"
#include "tcpa_defines.h"
#include "tcpa_error.h"
#include "tcpa_struct.h"

#ifdef __GNUC__
#define UNUSED_PARAM __attribute__ ((unused))
#endif

int main(void)
{
   printf("This is TPM_TCPA test file\n\n");

   TCPA_VERSION UNUSED_PARAM a;
   TCPA_DIGEST UNUSED_PARAM b;
   TCPA_PCRVALUE UNUSED_PARAM c;
   TCPA_COMPOSITE_HASH UNUSED_PARAM d;
   TCPA_DIRVALUE UNUSED_PARAM e;
   TCPA_HMAC UNUSED_PARAM f;
   TCPA_CHOSENID_HASH UNUSED_PARAM g;
   TCPA_NONCE UNUSED_PARAM h;
   TCPA_AUTHDATA UNUSED_PARAM i;
   TCPA_SECRET UNUSED_PARAM j;
   TCPA_ENCAUTH UNUSED_PARAM k;
   TCPA_KEY_HANDLE_LIST UNUSED_PARAM l;
   TCPA_KEY_PARMS UNUSED_PARAM m;
   TCPA_RSA_KEY_PARMS UNUSED_PARAM n;
   TCPA_PCR_SELECTION UNUSED_PARAM o;
   TCPA_PCR_COMPOSITE UNUSED_PARAM p;
   TCPA_PCR_INFO UNUSED_PARAM q;
   TCPA_STORED_DATA UNUSED_PARAM r;
   TCPA_SEALED_DATA UNUSED_PARAM s;
   TCPA_SYMMETRIC_KEY UNUSED_PARAM t;
   TCPA_STORE_PUBKEY UNUSED_PARAM u;
   TCPA_PUBKEY UNUSED_PARAM v;
   TCPA_STORE_PRIVKEY UNUSED_PARAM w;
```

```
TCPA_STORE_ASYMKEY UNUSED_PARAM x;
TCPA_KEY UNUSED_PARAM y;
TCPA_CERTIFY_INFO UNUSED_PARAM z;
TCPA_MIGRATIONKEYAUTH UNUSED_PARAM aa;
TCPA_IDENTITY_REQ UNUSED_PARAM ab;
TCPA_QUOTE_INFO UNUSED_PARAM ac;

printf("\tthis is error code %u\n",TCPA_E_AUTHFAIL);
printf("\tthis is error code %u\n",TCPA_E_BADINDEX);
printf("\tthis is error code %u\n",TCPA_E_BAD_PARAMETER);
printf("\tthis is error code %u\n",TCPA_E_AUDITFAILURE);
printf("\tthis is error code %u\n",TCPA_E_CLEAR_DISABLED);
printf("\tthis is error code %u\n",TCPA_E_DEACTIVATED);
printf("\tthis is error code %u\n",TCPA_E_DISABLED);
printf("\tthis is error code %u\n",TCPA_E_DISABLED_CMD);
printf("\tthis is error code %u\n",TCPA_E_FAIL);
printf("\tthis is error code %u\n",TCPA_E_INACTIVE);
printf("\tthis is error code %u\n",TCPA_E_INSTALL_DISABLED);
printf("\tthis is error code %u\n",TCPA_E_INVALID_KEYHANDLE);
printf("\tthis is error code %u\n",TCPA_E_KEYNOTFOUND);
printf("\tthis is error code %u\n",TCPA_E_NEED_SELFTEST);
printf("\tthis is error code %u\n",TCPA_E_MIGRATEFAIL);
printf("\tthis is error code %u\n",TCPA_E_NO_PCR_INFO);
printf("\tthis is error code %u\n",TCPA_E_NOSPACE);
printf("\tthis is error code %u\n",TCPA_E_NOSRK);
printf("\tthis is error code %u\n",TCPA_E_NOTSEALED_BLOB);
printf("\tthis is error code %u\n",TCPA_E_OWNER_SET);
printf("\tthis is error code %u\n",TCPA_E_RESOURCES);
printf("\tthis is error code %u\n",TCPA_E_SHORTRANDOM);
printf("\tthis is error code %u\n",TCPA_E_SIZE);
printf("\tthis is error code %u\n",TCPA_E_WRONGPCRVAL);
//   printf("\tthis is error code %u\n",TCPA_E_BUSY);   // Errata: Why is this not in the spec?
printf("\tthis is error code %u\n",TCPA_E_BAD_PARAM_SIZE);
printf("\tthis is error code %u\n",TCPA_E_SHA_THREAD);
printf("\tthis is error code %u\n",TCPA_E_SHA_ERROR);
printf("\tthis is error code %u\n",TCPA_E_FAILEDSELFTEST);
printf("\tthis is error code %u\n",TCPA_E_AUTH2FAIL);
printf("\tthis is error code %u\n",TCPA_E_BADTAG);
```

```
   printf("\tthis is error code %u\n",TCPA_E_IOERROR);
   printf("\tthis is error code %u\n",TCPA_E_ENCRYPT_ERROR);
   printf("\tthis is error code %u\n",TCPA_E_DECRYPT_ERROR);
   printf("\tthis is error code %u\n",TCPA_E_INVALID_AUTHHANDLE);
   printf("\tthis is error code %u\n",TCPA_E_NO_ENDORSEMENT);
   printf("\tthis is error code %u\n",TCPA_E_INVALID_KEYUSAGE);
   printf("\tthis is error code %u\n",TCPA_E_WRONG_ENTITYTYPE);
   printf("\tthis is error code %u\n",TCPA_E_INVALID_POSTINIT);
   printf("\tthis is error code %u\n",TCPA_E_INAPPROPRIATE_SIG);
   printf("\tthis is error code %u\n",TCPA_E_BAD_KEY_PROPERTY);
   printf("\tthis is error code %u\n",TCPA_E_BAD_MIGRATION);
   printf("\tthis is error code %u\n",TCPA_E_BAD_SCHEME);
   printf("\tthis is error code %u\n",TCPA_E_BAD_DATASIZE);
   printf("\tthis is error code %u\n",TCPA_E_BAD_MODE);
   printf("\tthis is error code %u\n",TCPA_E_BAD_PRESENCE);
   printf("\tthis is error code %u\n",TCPA_E_BAD_VERSION);
   printf("\tthis is error code %u\n",TCPA_E_RETRY);

   return(0);
}
```

+++*tsp.c*

```
/* Test compile TSS files
   Monty Wiseman
   Aug 25, 2003
*/

#include <stdio.h>
#include <sys/types.h>

#include "platform.h"
#include "tcpa_typedef.h"
#include "tcpa_defines.h"
#include "tcpa_error.h"
#include "tcpa_struct.h"
```

```
#include "tss_defines.h"
#include "tss_error_basics.h"
#include "tss_error.h"
#include "tss_typedef.h"
#include "tss_structs.h"

#include "tspi.h"

#ifdef __GNUC__
#define UNUSED_PARAM __attribute__ ((unused))
#endif

int main(void)
{
   printf("This is TPM_TSP test file\n\n");

   printf("\tthis is error code %u\n", TSS_SUCCESS);
   printf("\tthis is error code %u\n", TSS_E_FAIL );
   printf("\tthis is error code %u\n", TSS_E_BAD_PARAMETER );
   printf("\tthis is error code %u\n", TSS_E_INTERNAL_ERROR );
   printf("\tthis is error code %u\n", TSS_E_OUTOFMEMORY);
   printf("\tthis is error code %u\n", TSS_E_NOTIMPL);
   printf("\tthis is error code %u\n", TSS_E_KEY_ALREADY_REGISTERED  );
   printf("\tthis is error code %u\n", TSS_E_TPM_UNEXPECTED );
   printf("\tthis is error code %u\n", TSS_E_COMM_FAILURE );
   printf("\tthis is error code %u\n", TSS_E_TIMEOUT);
   printf("\tthis is error code %u\n", TSS_E_TPM_UNSUPPORTED_FEATURE  );
   printf("\tthis is error code %u\n", TSS_E_CANCELED);
   printf("\tthis is error code %u\n", TSS_E_PS_KEY_NOTFOUND);
   printf("\tthis is error code %u\n", TSS_E_PS_KEY_EXISTS);
   printf("\tthis is error code %u\n", TSS_E_PS_BAD_KEY_STATE);
   printf("\tthis is error code %u\n", TSS_E_INVALID_OBJECT_TYPE);
   printf("\tthis is error code %u\n", TSS_E_NO_CONNECTION);
   printf("\tthis is error code %u\n", TSS_E_CONNECTION_FAILED);
   printf("\tthis is error code %u\n", TSS_E_CONNECTION_BROKEN);
   printf("\tthis is error code %u\n", TSS_E_HASH_INVALID_ALG);
   printf("\tthis is error code %u\n", TSS_E_HASH_INVALID_LENGTH);
   printf("\tthis is error code %u\n", TSS_E_HASH_NO_DATA);
```

```
printf("\tthis is error code %u\n", TSS_E_INVALID_ATTRIB_FLAG);
printf("\tthis is error code %u\n", TSS_E_INVALID_ATTRIB_SUBFLAG  );
printf("\tthis is error code %u\n", TSS_E_INVALID_ATTRIB_DATA);
printf("\tthis is error code %u\n", TSS_E_INVALID_OBJECT_INITFLAG  );
printf("\tthis is error code %u\n", TSS_E_NO_PCRS_SET);
printf("\tthis is error code %u\n", TSS_E_KEY_NOT_LOADED );
printf("\tthis is error code %u\n", TSS_E_KEY_NOT_SET);
printf("\tthis is error code %u\n", TSS_E_VALIDATION_FAILED);
printf("\tthis is error code %u\n", TSS_E_TSP_AUTHREQUIRED);
printf("\tthis is error code %u\n", TSS_E_TSP_AUTH2REQUIRED);
printf("\tthis is error code %u\n", TSS_E_TSP_AUTHFAIL);
printf("\tthis is error code %u\n", TSS_E_TSP_AUTH2FAIL);
printf("\tthis is error code %u\n", TSS_E_KEY_NO_MIGRATION_POLICY);
printf("\tthis is error code %u\n", TSS_E_POLICY_NO_SECRET);
printf("\tthis is error code %u\n", TSS_E_INVALID_OBJ_ACCESS);
printf("\tthis is error code %u\n", TSS_E_INVALID_ENCSCHEME);
printf("\tthis is error code %u\n", TSS_E_INVALID_SIGSCHEME);
printf("\tthis is error code %u\n", TSS_E_ENC_INVALID_LENGTH);
printf("\tthis is error code %u\n", TSS_E_ENC_NO_DATA);
printf("\tthis is error code %u\n", TSS_E_ENC_INVALID_TYPE);
printf("\tthis is error code %u\n", TSS_E_INVALID_KEYUSAGE);
printf("\tthis is error code %u\n", TSS_E_VERIFICATION_FAILED);
printf("\tthis is error code %u\n", TSS_E_HASH_NO_IDENTIFIER);
printf("\tthis is error code %u\n", TSS_E_INVALID_HANDLE );
printf("\tthis is error code %u\n", TSS_E_SILENT_CONTEXT);
printf("\tthis is error code %u\n", TSS_E_EK_CHECKSUM);

TSS_HANDLE UNUSED_PARAM a;
TSS_FLAG UNUSED_PARAM b;
TSS_RESULT UNUSED_PARAM c;
TSS_HOBJECT UNUSED_PARAM d;
TSS_HCONTEXT UNUSED_PARAM e;
TSS_HPOLICY UNUSED_PARAM f;
TSS_HTPM UNUSED_PARAM g;
TSS_HKEY UNUSED_PARAM h;
TSS_HENCDATA UNUSED_PARAM i;
TSS_HPCRS UNUSED_PARAM j;
TSS_HHASH UNUSED_PARAM k;
```

```
   TSS_HPS UNUSED_PARAM l;
   TSS_EVENTTYPE UNUSED_PARAM m;
   TSS_MIGRATE_SCHEME UNUSED_PARAM n;
   TSS_ALGORITHM_ID UNUSED_PARAM o;
   TSS_KEY_USAGE_ID UNUSED_PARAM p;
   TSS_KEY_ENC_SCHEME UNUSED_PARAM q;
   TSS_KEY_SIG_SCHEME UNUSED_PARAM r;
   TSS_VERSION UNUSED_PARAM s;
   TSS_PCR_EVENT UNUSED_PARAM t;
   TSS_EVENT_CERT UNUSED_PARAM u;
   TSS_UUID UNUSED_PARAM v;
   TSS_KM_KEYINFO UNUSED_PARAM w;
   TSS_VALIDATION UNUSED_PARAM x;

   return(0);
}
```

+++*tcs.c*

```
/* Test compile TSS files
   Monty Wiseman
   Aug 25, 2003
*/

#include <stdio.h>
#include <sys/types.h>

#include "platform.h"
#include "tcpa_typedef.h"
#include "tcpa_defines.h"
#include "tcpa_error.h"
#include "tcpa_struct.h"

#include "tss_typedef.h"
#include "tss_defines.h"
#include "tss_structs.h"
```

```
#include "tcs_typedef.h"
#include "tcs_structs.h"
#include "tcs_error.h"

#ifdef __GNUC__
#define UNUSED_PARAM __attribute__ ((unused))
#endif

int main(void)
{
   printf("This is TPM_TCS test file\n\n");

   printf("\tthis is error code %u\n", TCS_E_INVALID_CONTEXTHANDLE);
   printf("\tthis is error code %u\n", TCS_E_INVALID_KEYHANDLE);
   printf("\tthis is error code %u\n", TCS_E_INVALID_AUTHHANDLE);
   printf("\tthis is error code %u\n", TCS_E_INVALID_AUTHSESSION);
   printf("\tthis is error code %u\n", TCS_E_INVALID_KEY);
   printf("\tthis is error code %u\n", TCS_E_KEY_MISMATCH);
   printf("\tthis is error code %u\n", TCS_E_KM_LOADFAILED);
   printf("\tthis is error code %u\n", TCS_E_KEY_CONTEXT_RELOAD);

   TPM_AUTH UNUSED_PARAM a;
   TCS_LOADKEY_INFO UNUSED_PARAM b;

   return(0);
}
```

+++tddl.c

```
/* Test compile TSS files
   Monty Wiseman
   Aug 25, 2003
*/

#include <stdio.h>
#include <sys/types.h>

#include "platform.h"
```

```
#include "tcpa_typedef.h"
#include "tcpa_defines.h"
#include "tcpa_error.h"
#include "tcpa_struct.h"

#include "tss_typedef.h"
#include "tddl_error.h"
#include "tddlapi_error.h"
#include "tddli.h"

#ifdef __GNUC__
#define UNUSED_PARAM __attribute__ ((unused))
#endif

int main(void)
{
   printf("\tthis is error code %u\n", TDDL_E_FAIL);
   printf("\tthis is error code %u\n", TDDL_E_BAD_PARAMETER);
   printf("\tthis is error code %u\n", TDDL_E_OUTOFMEMORY);
   printf("\tthis is error code %u\n", TDDL_E_COMPONENT_NOT_FOUND);
   printf("\tthis is error code %u\n", TDDL_E_ALREADY_OPENED);
   printf("\tthis is error code %u\n", TDDL_E_BADTAG);
   printf("\tthis is error code %u\n", TDDL_E_TIMEOUT);
   printf("\tthis is error code %u\n", TDDL_E_INSUFFICIENT_BUFFER);
   printf("\tthis is error code %u\n", TDDL_E_COMMAND_COMPLETED);
   printf("\tthis is error code %u\n", TDDL_E_ALREADY_CLOSED);
   printf("\tthis is error code %u\n", TDDL_E_IOERROR);
   printf("\tthis is error code %u\n", TDLL_E_COMMAND_ABORTED);

   return(0);
}
```

+++Makefile

```
CFLAGS = -c -x c++ -pedantic -Wall -Wmissing-prototypes -Wmissing-declarations 2> errors.err

all: main tsp tcs tddl
```

```
main: main.o

tsp: tsp.o

tcs: tcs.o

tddl: tddl.o

cleanall:
	rm -f *.o *.exe *.err
	rm -f *.c *.idl *.h
	rm -f Makefile.*

clean:
	rm -f *.o *.exe *.err

+++eof
NOTE: Be sure the commands in the makefile contain a "tab" character
The +++eof marks the end of the macro processing.
```