

Trusted Application Programming Reference on Portable TEE

The National Institute of Advanced Industrial Science and Technology

2021-11-30

1 Overview of ta-ref	1
1.1 Features	1
1.1.1 What we did on RISC-V	1
1.1.2 Separate GP TEE Internal API	1
1.2 Diagram	2
1.2.1 Dependency of category	2
2 How to Program on ta-ref	2
2.1 Time Functions	2
2.2 Random Functions	2
2.3 Hash Functions	3
2.4 Symmetric Crypto Functions	3
2.5 Asymmetric Crypto Functions	4
2.6 Asymmetric Crypto Gcm Functions	5
2.7 Open, Read, Write, Close On Secure Storage	6
3 Preparation before building	7
3.1 Keystone(RISC-V Unleashed)	7
3.1.1 Required Packages	7
3.1.2 Build Keystone	8
3.1.3 Run Keystone examples	8
3.2 OPTEE (ARM64 RPI3)	8
3.2.1 Required Packages	8
3.2.2 Build OPTEE v3.9.0	9
3.2.3 Run OPTEE Examples	10
3.3 SGX (Intel NUC)	10
3.3.1 List of machines which are confirmed to work	10
3.3.2 BIOS Versions which are failed or succeeded in IAS Test	11
3.3.3 BIOS Settings	11
3.3.4 Required Packages	11
3.3.5 Build SGX	11
3.3.6 Run sgx-ra-sample	13
4 Building	15
4.1 Install Doxygen-1.9.2	15
4.2 Install Required Packages	15
4.3 Build and Install	15
4.4 ta-ref with Keystone	16
4.4.1 Cloning source and building	16
4.4.2 Check ta-ref by running test_gp, test_hello, on QEMU	16
4.5 ta-ref with OPTEE	18
4.5.1 Cloning source and building	18
4.5.2 Check ta-ref by running test_gp, test_hello, on QEMU	18

4.6 ta-ref with SGX	19
4.6.1 Cloning source and building	19
4.6.2 Check ta-ref by running test_gp, test_hello, simulation mode on any pc	20
5 Running on Dev Boards	21
5.1 Keystone, Unleashed	21
5.1.1 Preparation of rootfs on SD Card	21
5.1.2 Copying binaries of test_hello and test_gp	23
5.1.3 Check test_hello and test_gp on Unleashed	23
5.2 OPTEE, RPI3	24
5.2.1 Preparation of rootfs on SD Card	24
5.2.2 Copying binaries of test_hello and test_gp to rootfs partition	25
5.2.3 Check test_hello and test_gp	25
5.3 SGX, NUC	26
5.3.1 Copying binaries of test_hello and test_gp to NUC machine	27
5.3.2 Check test_hello and test_gp	27
6 API Compare With Full-Set of GP API	29
6.1 GP API	29
7 Class Index	31
7.1 Class List	31
8 File Index	32
8.1 File List	32
9 Class Documentation	32
9.1 __TEE_ObjectHandle Struct Reference	32
9.1.1 Member Data Documentation	33
9.2 __TEE_OperationHandle Struct Reference	33
9.2.1 Member Data Documentation	34
9.3 addrinfo Struct Reference	35
9.3.1 Member Data Documentation	35
9.4 enclave_report Struct Reference	36
9.4.1 Member Data Documentation	36
9.5 pollfd Struct Reference	37
9.5.1 Member Data Documentation	37
9.6 report Struct Reference	38
9.6.1 Member Data Documentation	38
9.7 sm_report Struct Reference	38
9.7.1 Member Data Documentation	39
9.8 TEE_Attribute Struct Reference	39
9.8.1 Member Data Documentation	39
9.9 TEE_Identity Struct Reference	40

9.9.1 Member Data Documentation	41
9.10 TEE_ObjectInfo Struct Reference	41
9.10.1 Member Data Documentation	41
9.11 TEE_OperationInfo Struct Reference	42
9.11.1 Member Data Documentation	43
9.12 TEE_OperationInfoKey Struct Reference	44
9.12.1 Member Data Documentation	44
9.13 TEE_OperationInfoMultiple Struct Reference	44
9.13.1 Member Data Documentation	45
9.14 TEE_Param Union Reference	46
9.14.1 Member Data Documentation	46
9.15 TEE_SEAID Struct Reference	47
9.15.1 Member Data Documentation	47
9.16 TEE_SEReaderProperties Struct Reference	47
9.16.1 Member Data Documentation	47
9.17 TEE_Time Struct Reference	48
9.17.1 Member Data Documentation	48
9.18 TEE_UUID Struct Reference	48
9.18.1 Member Data Documentation	49
9.19 TEEC_Context Struct Reference	49
9.19.1 Detailed Description	49
9.19.2 Member Data Documentation	49
9.20 TEEC_Operation Struct Reference	50
9.20.1 Detailed Description	50
9.20.2 Member Data Documentation	51
9.21 TEEC_Parameter Union Reference	51
9.21.1 Detailed Description	51
9.21.2 Member Data Documentation	52
9.22 TEEC_RegisteredMemoryReference Struct Reference	52
9.22.1 Detailed Description	53
9.22.2 Member Data Documentation	53
9.23 TEEC_Session Struct Reference	54
9.23.1 Detailed Description	54
9.23.2 Member Data Documentation	54
9.24 TEEC_SharedMemory Struct Reference	54
9.24.1 Detailed Description	55
9.24.2 Member Data Documentation	55
9.25 TEEC_TempMemoryReference Struct Reference	56
9.25.1 Detailed Description	56
9.25.2 Member Data Documentation	56
9.26 TEEC_UUID Struct Reference	57
9.26.1 Detailed Description	57

9.26.2 Member Data Documentation	57
9.27 TEEC.Value Struct Reference	58
9.27.1 Detailed Description	58
9.27.2 Member Data Documentation	58
10 File Documentation	59
10.1 ta-ref/api/include/compiler.h File Reference	59
10.1.1 Macro Definition Documentation	60
10.2 compiler.h	66
10.3 ta-ref/api/include/report.h File Reference	68
10.3.1 Macro Definition Documentation	69
10.4 report.h	69
10.5 ta-ref/api/include/tee-common.h File Reference	70
10.5.1 Detailed Description	70
10.5.2 Macro Definition Documentation	71
10.6 tee-common.h	71
10.7 ta-ref/api/include/tee-ta-internal.h File Reference	71
10.7.1 Detailed Description	74
10.7.2 Function Documentation	74
10.8 tee-ta-internal.h	96
10.9 ta-ref/api/include/tee_api_defines.h File Reference	99
10.9.1 Macro Definition Documentation	105
10.10 tee_api_defines.h	138
10.11 ta-ref/api/include/tee_api_defines_extensions.h File Reference	143
10.11.1 Macro Definition Documentation	144
10.12 tee_api_defines_extensions.h	147
10.13 ta-ref/api/include/tee_api_types.h File Reference	148
10.13.1 Macro Definition Documentation	150
10.13.2 Typedef Documentation	151
10.13.3 Enumeration Type Documentation	152
10.14 tee_api_types.h	153
10.15 ta-ref/api/include/tee_client_api.h File Reference	156
10.15.1 Macro Definition Documentation	158
10.15.2 Typedef Documentation	164
10.15.3 Function Documentation	164
10.16 tee_client_api.h	168
10.17 ta-ref/api/include/tee_internal_api.h File Reference	171
10.18 tee_internal_api.h	171
10.19 ta-ref/api/include/tee_internal_api_extensions.h File Reference	171
10.19.1 Macro Definition Documentation	172
10.19.2 Function Documentation	172
10.20 tee_internal_api_extensions.h	173

10.21 ta-ref/api/include/tee_ta_api.h File Reference	174
10.21.1 Macro Definition Documentation	175
10.21.2 Function Documentation	175
10.22 tee_ta_api.h	176
10.23 ta-ref/api/include/test_dev_key.h File Reference	178
10.23.1 Variable Documentation	178
10.24 test_dev_key.h	179
10.25 ta-ref/api/include/trace.h File Reference	179
10.25.1 Macro Definition Documentation	181
10.25.2 Function Documentation	183
10.25.3 Variable Documentation	184
10.26 trace.h	185
10.27 ta-ref/api/include/trace_levels.h File Reference	187
10.27.1 Macro Definition Documentation	188
10.28 trace_levels.h	189
10.29 ta-ref/api/keystone/tee-internal-api-machine.c File Reference	189
10.29.1 Function Documentation	190
10.30 ta-ref/api/keystone/tee-internal-api.c File Reference	190
10.30.1 Macro Definition Documentation	192
10.30.2 Function Documentation	192
10.31 ta-ref/api/sgx/tee-internal-api.c File Reference	201
10.31.1 Macro Definition Documentation	202
10.31.2 Function Documentation	203
10.32 ta-ref/api/keystone/tee_api_tee_types.h File Reference	210
10.32.1 Macro Definition Documentation	211
10.33 tee_api_tee_types.h	212
10.34 ta-ref/api/optee/tee_api_tee_types.h File Reference	213
10.35 tee_api_tee_types.h	213
10.36 ta-ref/api/sgx/tee_api_tee_types.h File Reference	213
10.36.1 Macro Definition Documentation	215
10.37 tee_api_tee_types.h	216
10.38 ta-ref/api/keystone/teec_stub.c File Reference	217
10.38.1 Function Documentation	218
10.39 ta-ref/api/keystone/trace.c File Reference	221
10.39.1 Function Documentation	221
10.40 ta-ref/api/tee-internal-api-cryptlib.c File Reference	223
10.40.1 Macro Definition Documentation	224
10.40.2 Function Documentation	225
10.41 ta-ref/docs/building.md File Reference	237
10.42 ta-ref/docs/gp_api.md File Reference	237
10.43 ta-ref/docs/how_to_program_on_ta-ref.md File Reference	237
10.44 ta-ref/docs/overview_of_ta-ref.md File Reference	237

10.45 ta-ref/docs/preparation.md File Reference	237
10.46 ta-ref/docs/running_on_dev_boards.md File Reference	237
Index	239

1 Overview of ta-ref

1.1 Features

1.1.1 What we did on RISC-V

- We designed the GP internal API library to be portable.
 - Keystone SDK is utilized because of runtime "Eyrie".
 - The library is ported to Intel SGX as well as RISC-V Keystone.
- Implementation Challenge
 - The combination of GP internal API and cipher suite is big.
 - * We pick up some important GP internal APIs.
 - Some APIs depend on CPU architecture.
 - * We separate APIs into CPU architecture dependent / independent.
 - Integrate GP TEE Internal API to Keystone SDK.
 - * Keystone SDK includes EDL (Enclave Definition Language) named "keedger".
 - * Keedger creates the code for OCALL (request from TEE to REE) to check the pointer and boundary.

1.1.2 Separate GP TEE Internal API

- CPU architecture dependent
 - Random Generator, Time, Secure Storage, Transient Object(TEE.GenerateKey)
- CPU architecture independent(Crypto)
 - Transient Object(exclude TEE.GenerateKey), Crypto Common, Authenticated Encryption, Symmetric/Asymmetric Cipher, Message Digest

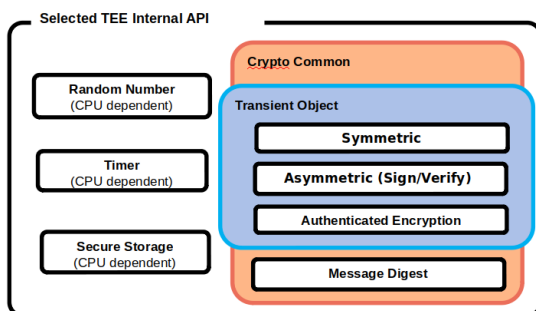
Category	CPU (In)Dependent	Functions
Random Number	Dependent	TEE_GenerateRandom
Time	Dependent	TEE_GetREETime, TEE_GetSystemTime
Secure Storage	Dependent	TEE_CreatePersistentObject, TEE_OpenPersistentObject, TEE_ReadObjectData, TEE_WriteObjectData, TEE_CloseObject
Transient Object	Dependent Independent	TEE_GenerateKey, TEE_AllocateTransientObject, TEE_FreeTransientObject, TEE_InitRefAttribute, TEE_InitValueAttribute, TEE_SetOperationKey
Crypto Common	Independent	TEE_AllocateOperation, TEE_FreeOperation
Authenticated Encryption	Independent	TEE_AEInit, TEE_AEUpdateAAD, TEE_AEUpdate, TEE_AEEncryptFinal, TEE_AEDecryptFinal
Symmetric Cipher	Independent	TEE_CipherInit, TEE_CipherUpdate, TEE_CipherDoFinal
Asymmetric Cipher	Independent	TEE_AsymmetricSignDigest, TEE_AsymmetricVerifyDigest
Message Digest	Independent	TEE_DigestUpdate, TEE_DigestDoFinal

1.2 Diagram

1.2.1 Dependency of category

Dependency of category

- Some categories have dependency.
 - Crypto Common
 - Cipher suite must be registered before use.
 - Transient Object
 - The space for a key must be prepared before use.



Sample Program

```

// Allocate a transient object for keypair
TEE_AllocateTransientObject(TEE_TYPE_ECDSA_KEYPAIR
,
    KEY_SIZE, &keypair);
// Assemble an attribute for ecc key
TEE_InitValueAttribute(&attr, TEE_ATTR_ECC_CURVE,
    TEE_ECC_CURVE_NIST_P256, KEY_SIZE);
// Generate a keypair having that attribute
TEE_GenerateKey(keypair, KEY_SIZE, &attr, 1);

// Allocate sign operation
TEE_AllocateOperation(&handle,
    TEE_ALG_ECDSA_P256,
    TEE_MODE_SIGN, KEY_SIZE);

// Set the generated key to the sign operation
TEE_SetOperationKey(handle, keypair);

// Sign
uint32 t_siglen = SIG_LENGTH;
TEE_AsymmetricSignDigest(handle, NULL, 0, hash,
    hashlen, sig, &siglen);
  
```

13

2 How to Program on ta-ref

2.1 Time Functions

This function retrieves the current time as seen from the point of view of the REE, which expressed in the number of seconds and prints the "GP REE second and millisecond".

```

--- Ree time ---
void gp_ree_time_test(void)
{
    TEE_Time time;
    /* REE time */
    TEE_GetREETime(&time);
    tee_printf ("GP REE time %u sec %u millis\n", time.seconds, time.millis);
}
--- end ---
  
```

This function retrieves the current system time as seen from the point of view of the TA, which expressed in the number of seconds and print the "GP System time second and millisecond".

```

--- start digest ---
void gp_trusted_time_test(void)
{
    TEE_Time time;
    /* System time */
    TEE_GetSystemTime(&time);
    tee_printf ("GP System time %u sec %u millis\n", time.seconds, time.millis);
}
--- end digest ---
  
```

2.2 Random Functions

This function generates the random data by invoking TEE_GenerateRandom function and it prints the generated random data.

```

--- random test ---
void gp_random_test(void)
  
```



```

{
    unsigned char rbuf[16];
    TEE.GenerateRandom(rbuf, sizeof(rbuf));
    tee_printf("@random: ");
    for (int i = 0; i < sizeof(rbuf); i++) {
        tee_printf ("%02x", rbuf[i]);
    }
    tee_printf("\n");
}
-----

```

2.3 Hash Functions

Pseudo code of how to use Message Digest Functions. Keystone uses sha3.c which is almost identical. Ultimate question is whether this should be done in 'Enclave (U-Mode) or Runtime (S-Mode) the library used in keystone.↔ The function performs many operations to achieve message data hash techniques to allocate the handle for a new cryptographic operation. And then finalize the message digest operation to produce the message hash. It prints the hash message.

```

--- start digest ---
void gp_message_digest_test(void)
{
    static unsigned char data[256] = {
        // 0x00,0x01,...,0xff
#include "test.dat"
    };
    unsigned char hash[SHA_LENGTH];
    TEE.OperationHandle handle;
    uint32_t hashlen = SHA_LENGTH;
    TEE.Result rv;
    // Take hash of test data
    /* sha3_init() in sha3.c */
    rv = TEE.AllocateOperation(&handle, TEE_ALG_SHA256, TEE_MODE_DIGEST, SHA_LENGTH);
    GP_ASSERT(rv, "TEE.AllocateOperation fails");
    /* sha3_update() in sha3.c */
    TEE.DigestUpdate(handle, data, sizeof(data));

    /* sha3_final() in sha3.c */
    rv = TEE.DigestDoFinal(handle, NULL, 0, hash, &hashlen);
    GP_ASSERT(rv, "TEE.DigestDoFinal fails");
    TEE.FreeOperation(handle);
    /* hash value is ready */
    // Dump hashed data
    tee_printf("hash: ");
    for (int i = 0; i < SHA_LENGTH; i++) {
        tee_printf ("%02x", hash[i]);
    }
    tee_printf("\n");
}
--- end digest ---

```

2.4 Symmetric Crypto Functions

Crypto, Authenticated Encryption with Symmetric Key Verification Functions. This function allocates an uninitialized transient object, i.e. a container for attributes. Transient objects are used to hold a cryptographic object (key or key-pair). With the generation of a key, a new cryptographic operation for encrypt and decrypt data is initiated with a symmetric cipher operation. The original data is compared with decrypted data by checking the data and its length.

```

--- AE encryption start ---
void gp_symmetric_key_enc_verify_test(void)
{
    TEE.OperationHandle handle;
    static unsigned char data[CIPHER_LENGTH] = {
        // 0x00,0x01,...,0xff
#include "test.dat"
    };
    uint8_t iv[16];
    unsigned char out[CIPHER_LENGTH];
    uint32_t outlen;
    TEE.ObjectHandle key;
    TEE.Result rv;
    // Generate key
    rv = TEE.AllocateTransientObject(TEE_TYPE_AES, 32*8, &key);
    GP_ASSERT(rv, "TEE.AllocateTransientObject fails");
    rv = TEE.GenerateKey(key, 256, NULL, 0);
    GP_ASSERT(rv, "TEE.GenerateKey fails");

```

```

// Encrypt test data
rv = TEE.AllocateOperation(&handle, TEE.ALG_AES_CBC_NOPAD, TEE.MODE_ENCRYPT, 256);
GP_ASSERT(rv, "TEE.AllocateOperation fails");
rv = TEE.SetOperationKey(handle, key);
GP_ASSERT(rv, "TEE.SetOperationKey fails");
TEE.GenerateRandom(iv, sizeof(iv));
TEE.CipherInit(handle, iv, sizeof(iv));
//GP_ASSERT(rv, "TEE.AEInit fails");
outlen = CIPHER_LENGTH;
rv = TEE.CipherUpdate(handle, data, CIPHER_LENGTH, out, &outlen);
GP_ASSERT(rv, "TEE.CipherUpdate fails");
TEE.FreeOperation(handle);
// Dump encrypted data
tee_printf("@cipher: ");
for (int i = 0; i < CIPHER_LENGTH; i++) {
    tee_printf ("%02x", out[i]);
}
tee_printf("\n");
// Decrypt it
rv= TEE.AllocateOperation(&handle, TEE.ALG_AES_CBC_NOPAD, TEE.MODE_DECRYPT, 256);
GP_ASSERT(rv, "TEE.AllocateOperation fails");
rv = TEE.SetOperationKey(handle, key);
GP_ASSERT(rv, "TEE.SetOperationKey fails");
TEE.CipherInit(handle, iv, sizeof(iv));
//GP_ASSERT(rv, "TEE.AEInit fails");
outlen = CIPHER_LENGTH;
rv = TEE.CipherUpdate(handle, out, CIPHER_LENGTH, out, &outlen);
GP_ASSERT(rv, "TEE.CipherUpdate fails");
TEE.FreeOperation(handle);
TEE.FreeTransientObject(key);
// Dump data
tee_printf("decrypted to: ");
for (int i = 0; i < CIPHER_LENGTH; i++) {
    tee_printf ("%02x", out[i]);
}
tee_printf("\n");
// Verify decrypted data against original one
int verify_ok;
verify_ok = !memcmp(out, data, CIPHER_LENGTH);
if (verify_ok) {
    tee_printf("verify ok\n");
} else {
    tee_printf("verify fails\n");
}
}
--- AE decrypt and verify end ---

```

2.5 Asymmetric Crypto Functions

Crypto, Sign and Verify with Asymmetric Key Verification Functions. Cryptographic Operations for API Message Digest Functions. The function performs cryptographic operation for API Message. To achieve this, the function allocates a handle for a new cryptographic operation, to finalize the message digest operation and to produce the message hash. The Hashed data is signed with signature key within an asymmetric operation. The original Hashed Data and Signed hashed data is compared for ok status.

```

--- Asymmetric Key sign start ---
void gp.asymmetric.key.sign.test(void)
{
    static unsigned char data[256] = {
        // 0x00,0x01,...,0xff
#include "test.dat"
    };
    unsigned char hash[SHA_LENGTH];
    unsigned char sig[SIG_LENGTH];
    TEE.OperationHandle handle;
    uint32_t hashlen = SHA_LENGTH;
    TEE.Result rv;

    // Take hash of test data
    /* Calculate hash */
    /* sha3.init() in sha3.c */
    rv = TEE.AllocateOperation(&handle, TEE.ALG_SHA256, TEE.MODE_DIGEST, SHA_LENGTH);
    GP_ASSERT(rv, "TEE.AllocateOperation fails");
    /* sha3.update() in sha3.c */
    TEE.DigestUpdate(handle, data, sizeof(data));

    /* sha3.final() in sha3.c */
    rv = TEE.DigestDoFinal(handle, NULL, 0, hash, &hashlen);
    GP_ASSERT(rv, "TEE.DigestDoFinal fails");
    /* free up */
    TEE.FreeOperation(handle);
}

```

```

/* Get the signature */
// Dump hashed data
tee_printf("@digest: ");
for (int i = 0; i < SHA_LENGTH; i++) {
    tee_printf ("%02x", hash[i]);
}
tee_printf("\n");
uint32_t siglen = SIG_LENGTH;
TEE_ObjectHandle keypair;
// Sign hashed data with the generated keys
/* set ecDSA_p256 key */
rv = TEE_AllocateOperation(&handle, TEE_ALG_ECDSA_P256, TEE_MODE_SIGN, 256);
GP_ASSERT(rv, "TEE_AllocateOperation fails");
// Generate keypair
rv = TEE_AllocateTransientObject(TEE_TYPE_ECDSA_KEYPAIR, 256, &keypair);
GP_ASSERT(rv, "TEE_AllocateTransientObject fails");
TEE_Attribute attr;
TEE_InitValueAttribute(&attr,
    TEE_ATTR_ECC_CURVE,
    TEE_ECC_CURVE_NIST_P256,
    256);
rv = TEE_GenerateKey(keypair, 256, &attr, 1);
GP_ASSERT(rv, "TEE_GenerateKey fails");
rv = TEE_SetOperationKey(handle, keypair);
GP_ASSERT(rv, "TEE_SetOperationKey fails");
/* Keystone has ecDSA_p256.sign() Equivalent in openssl is EVP_DigestSign() */
rv = TEE_AsymmetricSignDigest(handle, NULL, 0, hash, hashlen, sig, &siglen);
GP_ASSERT(rv, "TEE_AsymmetricSignDigest fails");

/* free up */
TEE_FreeOperation(handle);
/* Get the signature */
// Dump signature
tee_printf("@signature: ");
for (uint32_t i = 0; i < siglen; i++) {
    tee_printf ("%02x", sig[i]);
}
tee_printf("\n");
// Verify signature against hashed data
/* set ecDSA_p256 key */
rv = TEE_AllocateOperation(&handle, TEE_ALG_ECDSA_P256, TEE_MODE_VERIFY, 256);
GP_ASSERT(rv, "TEE_AllocateOperation fails");
rv = TEE_SetOperationKey(handle, keypair);
GP_ASSERT(rv, "TEE_SetOperationKey fails");
/* Keystone has ecDSA_p256.verify() Equivalent in openssl is EVP_DigestVerify() */
TEE_Result verify_ok;
verify_ok = TEE_AsymmetricVerifyDigest(handle, NULL, 0, hash, hashlen, sig, siglen);
/* free up */
TEE_FreeOperation(handle);
tee_printf("@@TEE_FreeOperation: \n");
TEE_FreeTransientObject(keypair);

if (verify_ok == TEE_SUCCESS) {
    tee_printf("verify ok\n");
} else {
    tee_printf("verify fails\n");
}
}
/* Check verify_ok for success of verification */
--- Asymmetric Key verify end ---

```

2.6 Asymmetric Crypto Gcm Functions

This function encrypt and decrypt the test data. The function allocates an uninitialized transient object, i.e. a container for attributes. Transient objects are used to hold a cryptographic object (key or key-pair). With the generation of a key, a new cryptographic operation for encrypt and decrypt data is initiated with a symmetric cipher operation. The data is also checked whether it is completely encrypted or decrypted. The original data is compared with decrypted data by checking the data and cipher length.

```

--- symmetric key gcm verification start ---
void gp_symmetric_key_gcm_verify_test(void)
{
    TEE_OperationHandle handle;
    static unsigned char data[CIPHER_LENGTH] = {
        // 0x00, 0x01, ..., 0xff
    };
#include "test.dat"
    };
    uint8_t iv[16];
    unsigned char out[CIPHER_LENGTH];
    uint32_t outlen;
    unsigned char tag[16];

```

```

TEE_ObjectHandle key;
TEE_Result rv;
// Generate key
rv = TEE_AllocateTransientObject(TEE_TYPE_AES, 256, &key);
GP_ASSERT(rv, "TEE_AllocateTransientObject fails");
rv = TEE_GenerateKey(key, 256, NULL, 0);
GP_ASSERT(rv, "TEE_GenerateKey fails");
// Encrypt test data
rv = TEE_AllocateOperation(&handle, TEE_ALG_AES_GCM, TEE_MODE_ENCRYPT, 256);
GP_ASSERT(rv, "TEE_AllocateOperation fails");
rv = TEE_SetOperationKey(handle, key);
GP_ASSERT(rv, "TEE_SetOperationKey fails");
TEE_GenerateRandom(iv, sizeof(iv));
/* Equivalent in openssl is EVP_EncryptInit_ex() */
rv = TEE_AEInit(handle, iv, sizeof(iv), 16*8, 16, 16);
GP_ASSERT(rv, "TEE_AEInit fails");
/* Equivalent in openssl is EVP_EncryptUpdate() */
// rv = TEE_AEUpdateAAD(handle, aad, 16);
// GP_ASSERT(rv, "TEE_AEUpdateAAD fails");
unsigned int taglen = 16;
memset(tag, 0, 16);
outlen = CIPHER_LENGTH;
/* Equivalent in openssl is EVP_EncryptFinal() */
rv = TEE_AEEncryptFinal(handle, data, 256, out, &outlen, tag, &taglen);
TEE_FreeOperation(handle);
/* Get the auth.tag */
// Dump encrypted data and tag
tee_printf("@cipher: ");
for (int i = 0; i < CIPHER_LENGTH; i++) {
    tee_printf(" %02x", out[i]);
}
tee_printf("\n");
tee_printf("@tag: ");
for (int i = 0; i < 16; i++) {
    tee_printf(" %02x", tag[i]);
}
tee_printf("\n");
// Decrypt it
rv = TEE_AllocateOperation(&handle, TEE_ALG_AES_GCM, TEE_MODE_DECRYPT, 256);
GP_ASSERT(rv, "TEE_AllocateOperation fails");
rv = TEE_SetOperationKey(handle, key);
GP_ASSERT(rv, "TEE_SetOperationKey fails");
/* Equivalent in openssl is EVP_DecryptInit_ex() */
rv = TEE_AEInit(handle, iv, sizeof(iv), 16*8, 16, 16);
GP_ASSERT(rv, "TEE_AEInit fails");
// rv = TEE_AEUpdateAAD(handle, aad, 16);
// GP_ASSERT(rv, "TEE_AEUpdateAAD fails");
unsigned char decode[CIPHER_LENGTH];
outlen = 256;
/* Equivalent in openssl require two functions
   EVP_CIPHER_CTX_ctrl(tag) and EVP_DecryptFinal(others) */
rv = TEE_AEDecryptFinal(handle, out, 256, decode, &outlen, tag, 16);
GP_ASSERT(rv, "TEE_AEDecryptFinal fails");
TEE_FreeOperation(handle);
TEE_FreeTransientObject(key);
// Dump data and tag
tee_printf("decrypted to: ");
for (int i = 0; i < CIPHER_LENGTH; i++) {
    tee_printf(" %02x", decode[i]);
}
tee_printf("\n");

// Verify decrypted data against original one
/* Check verify_ok for success of decrypting and authentication */
int verify_ok;
verify_ok = !memcmp(decode, data, CIPHER_LENGTH);
if (verify_ok) {
    tee_printf("verify ok\n");
} else {
    tee_printf("verify fails\n");
}
}
--- symmetric key gcm verification end ---

```

2.7 Open, Read, Write, Close On Secure Storage

Core Functions, Secure Storage Functions. Pseudo code of how to use Secure Storage. These could be implemented using ocall on Keystone. Almost identical to `open()`, `clone()`, `read()`, `write()` in POSIX API. The function creates a persistent object for reading and writing the data. The created data individually for read and write are compared for data length. If the length of both the objects are same, the function prints "verify ok" and prints "verify fails" if it is not the same.

```

--- write file start ---
void gp_secure_storage_test(void)
{
    static unsigned char data[] = {
        // 0x00,0x01,...,0xff
#include "test.dat"
    };
    static unsigned char buf[DATA_LENGTH];
    TEE_Result rv;
    /* write */
    TEE_ObjectHandle object;
    rv = TEE_CreatePersistentObject(TEE_STORAGE_PRIVATE,
                                   "FileOne", strlen("FileOne"),
                                   (TEE_DATA_FLAG_ACCESS_WRITE
                                    | TEE_DATA_FLAG_OVERWRITE),
                                   TEE_HANDLE_NULL,
                                   NULL, 0,
                                   &object);
    GP_ASSERT(rv, "TEE_CreatePersistentObject fails");
    memcpy(buf, data, DATA_LENGTH);
    /* fill the date in buffer */
    rv = TEE_WriteObjectData(object, (const char *)data, DATA_LENGTH);
    GP_ASSERT(rv, "TEE_WriteObjectData fails");
    TEE_CloseObject(object);
--- write file end ---
    /* clear buf */
    memset(buf, 0, DATA_LENGTH);
--- read file start ---
    /* read */
    rv = TEE_OpenPersistentObject(TEE_STORAGE_PRIVATE,
                                   "FileOne", strlen("FileOne"),
                                   TEE_DATA_FLAG_ACCESS_READ,
                                   &object);
    GP_ASSERT(rv, "TEE_OpenPersistentObject fails");
    uint32_t count;
    rv = TEE_ReadObjectData(object, (char *)buf, DATA_LENGTH, &count);

    GP_ASSERT(rv, "TEE_ReadObjectData fails");
    TEE_CloseObject(object);
    /* use the date in buffer */
    tee_printf("%d bytes read: ", count);
    for (uint32_t i = 0; i < count; i++) {
        tee_printf ("%02x", buf[i]);
    }
    tee_printf("\n");
    /* Compare read data with written data */
    int verify_ok;
    verify_ok = !memcmp(buf, data, DATA_LENGTH);
    if (verify_ok) {
        tee_printf("verify ok\n");
    } else {
        tee_printf("verify fails\n");
    }
}
--- read file end ---

```

3 Preparation before building

3.1 Keystone(RISC-V Unleashed)

Keystone is an open-source TEE framework for RISC-V processors. For more details check,

- <http://docs.keystone-enclave.org/en/latest>

3.1.1 Required Packages

Install following Packages

```

apt-get update
apt-get install -y autoconf automake autotools-dev bc bison build-essential curl expat libexpat1-dev flex
gawk gcc git gperf libgmp-dev libmpc-dev libmpfr-dev libtool texinfo tmux patchutils zlibg-dev wget
bzip2 patch vim-common lbzip2 python pkg-config libglib2.0-dev libpixmap-1-dev libssl-dev screen
device-tree-compiler expect makeunzip unzip cpio rsync cmake

```

3.1.2 Build Keystone

Download the keystone sources

```
git clone https://github.com/keystone-enclave/keystone.git
cd keystone
git checkout v0.3
./fast-setup.sh
make
source source.sh
./sdk/scripts/init.sh
./sdk/examples/hello/vault.sh
./sdk/examples/hello-native/vault.sh
./tests/tests/vault.sh
make image
```

RISC-V Toolchain:

- When you execute `./fast-setup.sh`, the toolchain for RISC-V has been installed at `$KEYSTONE_DIR/riscv/bin` and it adds to your `PATH`.

3.1.3 Run Keystone examples

Launch QEMU console

```
./scripts/run-qemu.sh
Welcome to Buildroot
```

Login to console with user=root, passwd=sifive

```
buildroot login: root
Password:
$
```

Run hello example

```
$ insmod keystone-driver.ko
[ 365.354299] keystone.driver: loading out-of-tree module taints kernel.
[ 365.364279] keystone.enclave: keystone enclave v0.2
$
$ ./hello/hello.ke
Verifying archive integrity... 100% All good.
Uncompressing Keystone vault archive 100%
hello, world!
```

Poweroff the console incase, if you want to exit.

```
$ poweroff
```

3.2 OPTEE (ARM64 RPI3)

OP-TEE is a Trusted Execution Environment (TEE) designed as companion to a non-secure Linux kernel running on Arm. Lets build OPTEE for QEMU and Raspberry Pi3 Model B development board. For more details check,

- <https://optee.readthedocs.io/en/latest/>

3.2.1 Required Packages

Install following packages on Ubuntu 18.04

```
sudo dpkg --add-architecture i386
sudo apt-get update -y
sudo apt-get install -y android-tools-adb android-tools-fastboot autoconf \
    automake bc bison build-essential ccache cscope curl device-tree-compiler \
    expect flex ftp-upload gdisk iasl libattr1-dev libc6:i386 libcap-dev \
    libfdt-dev libftdi-dev libglib2.0-dev libhidapi-dev libncurses5-dev \
    libpixman-1-dev libssl-dev libstdc++6:i386 libtool libz1:i386 make \
    mtools netcat python python-crypto python3-crypto python-pyelftools \
    python3-pycryptodome python3-pyelftools python3-serial vim-common \
    rsync unzip uuid-dev xdg-utils xterm xz-utils zlib1g-dev \
    git python3-pip wget cpio \
    texlive texinfo \
sudo pip3 install pycryptodomex
```

3.2.2 Build OPTEE v3.9.0

Configure git

```
git config --global user.name "dummy"
git config --global user.email "dummy@gmail.com"
git config --global color.ui false
mkdir ~/bin
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo && \
chmod a+x ~/bin/repo
```

3.2.2.1 Download Toolchains

```
export TOOLCHAIN_DIR=${HOME}/toolchains
sudo apt-get install -y wget xz-utils
mkdir -p ${TOOLCHAIN_DIR}/aarch64 ${TOOLCHAIN_DIR}/aarch32
wget http://192.168.100.100:2000/gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabi.tar.xz -o /dev/null -O
aarch32.tar.xz && \
tar xf aarch32.tar.xz --strip-components=1 -C ${TOOLCHAIN_DIR}/aarch32
wget http://192.168.100.100:2000/gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu.tar.xz -o /dev/null -O
aarch64.tar.xz && \
tar xf aarch64.tar.xz --strip-components=1 -C ${TOOLCHAIN_DIR}/aarch64
export PATH=${TOOLCHAIN_DIR}/aarch64/bin:${TOOLCHAIN_DIR}/aarch32/bin:${PATH}
```

3.2.2.2 Clone and Build OPTEE v3.9.0 for QEMU

Clone optee version 3.9.0 for QEMU

```
mkdir optee-3.9.0-qemu
cd optee-3.9.0-qemu
~/bin/repo init -u https://github.com/knknkn1162/manifest.git -m qemu.v8.xml -b 3.9.0
~/bin/repo sync -j4 --no-clone-bundle
ln -s ~/toolchains toolchains
cd build
make
```

If build is successful, the rootfs can be found as follows

```
ls -l ../out-br/images/rootfs.cpio.gz
```

3.2.2.3 Clone and Build OPTEE v3.9.0 for RPI3

Copy the following lines into "optee-rpi3.sh" script

```
#!/bin/bash -u
export OPTEE_VER=$1
export OPTEE_DIR=${PWD}/optee-${OPTEE_VER}-rpi3
mkdir ${OPTEE_DIR} || true
cd ${OPTEE_DIR}
~/bin/repo init -u https://github.com/knknkn1162/manifest.git -m rpi3.xml -b ${OPTEE_VER}
~/bin/repo sync -j4 --no-clone-bundle
ln -s ~/toolchains ${OPTEE_DIR}/. || true
echo 'CONFIG_CMDLINE="console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 root=/dev/mmcblk0p2 rootfstype=ext4
noinitrd rw rootwait init=/lib/systemd/systemd"' > build/defconfig-cmdline.txt
cd build
make OPTEE_CLIENT_BIN_ARCH_EXCLUDE=/boot
LINUX_DEFCONFIG_COMMON_FILES=${OPTEE_DIR}/linux/arch/arm64/configs/bcmrpi3_defconfig
${OPTEE_DIR}/build/kconfigs/rpi3.conf ${OPTEE_DIR}/build/defconfig-cmdline.txt"
BR2_PACKAGE_OPTEE_OS_EXT=n BR2_PACKAGE_OPTEE_TEST_EXT=n BR2_PACKAGE_OPTEE_EXAMPLES_EXT=n
BR2_TOOLCHAIN_EXTERNAL_GCC_8=y BR2_TOOLCHAIN_EXTERNAL_HEADERS_4_19=y BR2_HOST_GCC_AT_LEAST_8=y
BR2_TOOLCHAIN_HEADERS_AT_LEAST="4.19" -j`nproc`
```

Run the script as follows

```
chmod +x optee-rpi3.sh
./optee-rpi3.sh 3.9.0
```

If build is successful, the rootfs can be found as follows

```
ls -l ../out-br/images/rootfs.cpio.gz
```

3.2.3 Run OPTEE Examples

3.2.3.1 Launching QEMU Console

Run following commands from OPTEE build directory

```
cd $OPTEE_DIR/build
make run
```

Once above command is success, QEMU is ready

```
* QEMU is now waiting to start the execution
* Start execution with either a 'c' followed by <enter> in the QEMU console or
* attach a debugger and continue from there.
*
* To run OP-TEE tests, use the xtest command in the 'Normal World' terminal
* Enter 'xtest -h' for help.
cd /TEE/demo/rpi3/optee.3.9.0.qemu/build/./out/bin &&
/TEE/demo/rpi3/optee.3.9.0.qemu/build/./qemu/aarch64-softmmu/qemu-system-aarch64 \
-nographic \
-serial tcp:localhost:54320 -serial tcp:localhost:54321 \
-smp 2 \
-s -S -machine virt,secure=on -cpu cortex-a57 \
-d unimp -semihosting-config enable,target=native \
-m 1057 \
-bios bl1.bin \
-initrd rootfs.cpio.gz \
-kernel Image -no-acpi \
-append 'console=ttyAMA0,38400 keepbootcon root=/dev/vda2' \
-object rng-random,filename=/dev/urandom,id=rng0 -device
virtio-rng-pci,rng=rng0,max-bytes=1024,period=1000 -netdev user,id=vmnic -device
virtio-net-device,netdev=vmnic
QEMU 3.0.93 monitor - type 'help' for more information
(qemu) c
Now Optee started to boot from another tab on the Terminal
```

3.2.3.2 Run hello world example

Once boot completed it displays following message, then enter "root" to login to the shell

```
Welcome to Buildroot, type root or test to login
buildroot login: root
$
$ optee.example.hello.world
Invoking TA to increment 42
TA incremented value to 43
```

Poweroff the console in case, if you want to exit.

```
$ poweroff
```

3.3 SGX (Intel NUC)

Intel(R) Software Guard Extensions (Intel(R) SGX) is an Intel technology for application developers who is seeking to protect selected code and data from disclosure or modification. For more details check,

- <https://github.com/intel/linux-sgx/blob/master/README.md>

3.3.1 List of machines which are confirmed to work

1. Intel NUC7PJYH - Intel(R) Celeron(R) J4005 CPU @ 2.00GHz
2. Intel NUC7PJYH - Intel(R) Pentium(R) Silver J5005 CPU @ 1.50GHz
3. Intel NUC9VXQNX - Intel(R) Xeon(R) E-2286M CPU @ 2.40GHz (Partially working)

3.3.2 BIOS Versions which are failed or succeeded in IAS Test

1. BIOS Version JYGLKCPX.86A.0050.2019.0418.1441 - IAS Test was Failed
2. BIOS Version JYGLKCPX.86A.0053.2019.1015.1510 - IAS Test was Failed
3. BIOS Version JYGLKCPX.86A.0057.2020.1020.1637 - IAS Test was Success
4. BIOS Version QNCFLX70.0034.2019.1125.1424 - IAS Test was Failed
5. BIOS Version QNCFLX70.0059.2020.1130.2122 - IAS Test was Success

Update BIOS from:

- <https://downloadcenter.intel.com/download/29987/BIOS-Update-JYGLKCPX->
- <https://downloadcenter.intel.com/download/30069/BIOS-Update-QNCFLX70->

3.3.3 BIOS Settings

1. Make sure you are running with latest version BIOS
2. Make sure you enabled SGX support in BIOS
3. Make sure Secure Boot disabled in BIOS

Refer: <https://github.com/intel/sgx-software-enable/blob/master/README.md>

3.3.4 Required Packages

Intall following packages on Ubuntu 18.04

```
sudo apt-get install build-essential ocaml ocamlbuild automake autoconf libtool wget python libssl-dev git  
cmake perl libssl-dev libcurl4-openssl-dev protobuf-compiler libprotobuf-dev debhelper cmake reprepro  
expect unzip sshpass
```

3.3.5 Build SGX

There are 3 components which need to be build for SGX

1. linux-sgx
2. linux-sgx-driver
3. sgx-ra-sample

3.3.5.1 SGX SDK

Clone and build

```
git clone https://github.com/intel/linux-sgx.git -b sgx.2.10
cd linux-sgx
git checkout sgx.2.10
./download_prebuilt.sh
sudo cp external/toolset/ubuntu18.04/{as,ld,ld.gold,objdump} /usr/local/bin/
make -j`nproc` sdk_install_pkg DEBUG=1
```

Install SGX SDK

```
sudo ./linux/installer/bin//sgx_linux_x64_sdk.${version}.bin
```

where `${version}` is a string something similar to 2.10.100.2.

Answer the question with `no` and input the install dir as `/opt/intel`

Build and Install SGX PSW packages

See here: <https://github.com/intel/linux-sgx#install-the-intelr-sgx-psw>

```
source /opt/intel/sgxsdk/environment
make deb_psw_pkg DEBUG=1
rm ./linux/installer/deb/*/*sgx-dcap-pccs*.deb
sudo dpkg -i ./linux/installer/deb/*/*.deb
```

Install SGX PSW packages from Intel Repository

See here: <https://github.com/intel/linux-sgx#install-the-intelr-sgx-psw-1>

Using the local repo is recommended, since the system will resolve the dependencies automatically.

Check at page no.7, https://download.01.org/intel-sgx/sgx-linux/2.9/docs/Intel_SGX_Installation_Guide_Linux_2.9_Open_Source.pdf

```
sudo apt install libsgx-enclave-common libsgx-epid libsgx-launch libsgx-urts libsgx-uae-service
libsgx-quote-ex
```

If you see below error,

```
Errors were encountered while processing:
/tmp/apt-dpkg-install-pCB0cR/04-libsgx-headers.2.12.100.3-bionic1.amd64.deb
```

Here is the fix

```
sudo apt -o Dpkg::Options::="--force-overwrite" --fix-broken install
```

3.3.5.2 Build and Install SGX Driver

See [linux-sgx-driver](#).

Caveat: Whenever updating kernel, don't forget rebuilding this driver with new version of the kernel header. (There are a few linux-sgx-driver-dkms repo, though I've experienced troubles with them.)

Clone and build

```
$ git clone https://github.com/intel/linux-sgx-driver.git
$ cd linux-sgx-driver
$ make
```

Install SGX driver

```
$ sudo mkdir -p "/lib/modules/$(uname -r)"/kernel/drivers/intel/sgx"
$ sudo cp isgx.ko "/lib/modules/$(uname -r)"/kernel/drivers/intel/sgx"
$ sudo sh -c "cat /etc/modules | grep -Fxq isgx || echo isgx >> /etc/modules"
$ sudo /sbin/depmod
$ sudo /sbin/modprobe isgx
```

When modprobe fails with "Operation is not permitted", disable secure boot in BIOS. So that the unsigned kernel driver can be installed. If it is success, reboot your machine and verify `sudo lsmod | grep isgx` if it shows `isgx.ko`

3.3.6 Run sgx-ra-sample

3.3.6.1 Build sgx-ra-sample

Clone and build OpenSSL 1.1.c

```
wget https://www.openssl.org/source/openssl-1.1.1c.tar.gz
tar xf openssl-1.1.1c.tar.gz
cd openssl-1.1.1c/
./config --prefix=/opt/openssl/1.1.1c --openssldir=/opt/openssl/1.1.1c
make
sudo make install
cd ..
```

Clone and build sgx-ra-sample

```
git clone https://github.com/intel/sgx-ra-sample.git
cd sgx-ra-sample/
./bootstrap
./configure --with-openssldir=/opt/openssl/1.1.1c
make
```

3.3.6.2 Prepare for IAS Test

1. Obtain a subscription key for the Intel SGX Attestation Service Utilizing Enhanced Privacy ID (EPID). See here: <https://api.portal.trustedservices.intel.com/EPID-attestation>
2. Download Intel.SGX.Attestation.RootCA.pem from above portal.
3. Edit settings file and update the file with your own values obtained from portal.

```
@ -15,14 +15,14 @@ QUERY_IAS_PRODUCTION=0
# Your Service Provider ID. This should be a 32-character hex string.
# [REQUIRED]

-SPID=0123456789ABCDEF0123456789ABCDEF
+SPID=EF9AE4A8635825B88751C8698CB370B4

# Set to a non-zero value if this SPID is associated with linkable
# quotes. If you change this, you'll need to change SPID,
# IAS_PRIMARY_SUBSCRIPTION_KEY and IAS_SECONDARY_SUBSCRIPTION_KEY too.

-LINKABLE=0
+LINKABLE=1

#=====
@@ -50,18 +50,18 @@ USE_PLATFORM_SERVICES=0
# More Info: https://api.portal.trustedservices.intel.com/EPID-attestation
# Associated SPID above is required

-IAS_PRIMARY_SUBSCRIPTION_KEY=
+IAS_PRIMARY_SUBSCRIPTION_KEY=b6da4c9c41464924a14954ad8c03e8cf

# Intel Attestation Service Secondary Subscription Key
# This will be used in case the primary subscription key does not work

-IAS_SECONDARY_SUBSCRIPTION_KEY=
+IAS_SECONDARY_SUBSCRIPTION_KEY=188d91f86c064deb97e7472175ae1e79

# The Intel IAS SGX Report Signing CA file. You are sent this certificate
# when you apply for access to SGX Developer Services at
# http://software.intel.com/sgx [REQUIRED]

-IAS_REPORT_SIGNING_CA_FILE=
+IAS_REPORT_SIGNING_CA_FILE=./Intel.SGX.Attestation.RootCA.pem

# Debugging options
@@ -82,7 +82,7 @@ IAS_REPORT_SIGNING_CA_FILE=

# Set to non-zero for verbose output

-VERBOSE=0
+VERBOSE=1
```

3.3.6.3 Run IAS Test

Run "run-server"

[illegible]

Open another terminal and run "run-client"

[illegible]

3.3.6.4 Possible wget Error

Server may invoke wget command to get some files from intel servers. If the server side fails with following error

```
Connecting to api.trustedservices.intel.com (api.trustedservices.intel.com)|40.87.90.88|:443... connected.
ERROR: cannot verify api.trustedservices.intel.com's certificate, issued by 'CN=COMODO RSA Organization
Validation Secure Server CA,O=COMODO CA Limited,L=Salford,ST=Greater Manchester,C=GB':
Unable to locally verify the issuer's authority.
To connect to api.trustedservices.intel.com insecurely, use '--no-check-certificate'.
```

then add a line

```
ca-certificate = /etc/ssl/certs/ca-certificates.crt
```

to /etc/wgetrc file as super user, then test again.

3.3.6.5 BIOS Updating

If BIOS version is outdated, IAS may not succeed. So when you are done with BIOS update, the sgx driver would be required to make and install again.

Update BIOS from:

- <https://downloadcenter.intel.com/download/29987/BIOS-Update-JYGLKCPX->
- <https://downloadcenter.intel.com/download/30069/BIOS-Update-QNCFLX70->

3.3.6.6 Run LocalAttestation

Running SDK code samples in simulation mode

```
source /opt/intel/sgxsdk/environment
cd linux-sgx/SampleCode/LocalAttestation
make SGX_MODE=SIM
cd bin
./app
succeed to load enclaves.
succeed to establish secure channel.
Succeed to exchange secure message...
Succeed to close Session...
```

Running in hardware mode (It works when you have latest BIOS and SGX support is enabled in BIOS)

```
source /opt/intel/sgxsdk/environment
cd linux-sgx/SampleCode/LocalAttestation
make SGX_MODE=HW
cd bin
./app
succeed to load enclaves.
succeed to establish secure channel.
Succeed to exchange secure message...
Succeed to close Session...
```

4 Building

4.1 Install Doxygen-1.9.2

This PDF was generated using Doxygen version 1.9.2. To install doxygen-1.9.2 following procedure is necessary.

4.2 Install Required Packages

Install following packages on Ubuntu 18.04

```
sudo apt install doxygen-latex graphviz texlive-full texlive-latex-base latex-cjk-all
```

Above packages required to generate PDF using doxygen.

4.3 Build and Install

```
git clone https://github.com/doxygen/doxygen.git
cd doxygen
mkdir build
cd build
cmake -G "Unix Makefiles" ..
make
sudo make install
```

4.4 ta-ref with Keystone

Make sure Keystone and other dependant sources have been built

4.4.1 Cloning source and building

Install required packages

```
sudo apt-get update
sudo apt-get install -y clang-tools-6.0 libclang-6.0-dev cmake ocaml expect screen sshpass
```

Setup Env

```
export KEYSTONE_DIR=<path to your keystone directory>
export PATH=$PATH:$KEYSTONE_DIR/riscv/bin
```

Clone and Build KEYEDGE

```
GIT_SSL_NO_VERIFY=1 git clone --recursive https://192.168.100.100/rinkai/keyedge.git
cd keyedge
git checkout f9406aba2117147cc54462ede4766e26f028ced9
make
```

Clone and Build KEEDGER8R

```
GIT_SSL_NO_VERIFY=1 git clone --recursive https://192.168.100.100/rinkai/keedger8r.git
cd keedger8r
make
sed -i 's/MAX_EDGE_CALL 10$/MAX_EDGE_CALL 1000/' ${KEYSTONE_DIR}/sdk/lib/edge/include/edge.common.h
make -C ${KEYSTONE_DIR}/sdk/lib clean all
```

Clone the source

```
git clone https://192.168.100.100/rinkai/ta-ref.git
cd ta-ref
git checkout teep-device-tb-slim
git submodule sync --recursive
git submodule update --init --recursive
```

Build

```
export KEYSTONE_DIR=<path to keystone directory>
export KEYSTONE_SDK_DIR=$KEYSTONE_DIR/sdk
export KEYEDGE_DIR=<path to keyedge directory>
export KEEDGER8R_DIR=<path to keedger8r directory>
source env/keystone.sh
make build test-bin MACHINE=HIFIVE TEST_DIR=test_hello
make build test-bin MACHINE=HIFIVE TEST_DIR=test_gp
```

4.4.2 Check ta-ref by running test_gp, test_hello, on QEMU

Copy the test_hello and test_gp programs to QEMU.

4.4.2.1 Launch QEMU Console

```
cd $KEYSTONE_DIR
./scripts/run-qemu.sh
Welcome to Buildroot
```

4.4.2.2 test_hello

Run test_hello

```
cp test_hello/keystone/Enclave/Enclave.eapp.riscv $KEYSTONE_DIR/buildroot.overlay/root/test_hello/
cp test_hello/keystone/Enclave/App.client $KEYSTONE_DIR/buildroot.overlay/root/test_hello/
cp $KEYSTONE_SDK_DIR/rts/eyrie/eyrie-rt $KEYSTONE_DIR/buildroot.overlay/root/test_hello/
insmod keystone-driver.ko
./App.client Enclave.eapp.riscv eyrie-rt
hello world!
```

4.4.2.3 test_gp

Run test_gp

```

cp test_gp/keystone/Enclave/Enclave.eapp.riscv $KEYSTONE_DIR/buildroot_overlay/root/test_gp/
cp test_gp/keystone/Enclave/App.client $KEYSTONE_DIR/buildroot_overlay/root/test_gp/
cp $KEYSTONE_SDK_DIR/rts/eyrie/eyrie-rt $KEYSTONE_DIR/buildroot_overlay/root/test_gp/
insmod keystone-driver.ko
./App.client Enclave.eapp.riscv eyrie-rt
main start
TEE.GenerateRandom(0x000000003FFFFEE0, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@random: 5ea8741bd8a3b298cf53d214eca693fb
TEE.GetREETime(): start
@[SE] gettimeofday 77 sec 865873 usec -> 0
@GP REE time 77 sec 865 millis
TEE.GetSystemTime(): start
@GP System time 100063195 sec 609 millis
TEE.CreatePersistentObject(): start
@[SE] open file FileOne flags 241 -> 3 (0)
TEE.WriteObjectData(): start
@[SE] write desc 3 buf 480d0 len 256-> 256
TEE.CloseObject(): start
@[SE] close desc 3 -> 0
TEE.OpenPersistentObject(): start
@[SE] open file FileOne flags 0 -> 3 (0)
TEE.ReadObjectData(): start
@[SE] read desc 3 buf fff41664 len 256-> 256
TEE.CloseObject(): start
@[SE] close desc 3 -> 0
256 bytes read: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20212223242526272829
2a2b2c2d2e2f303132333435363738393a3b3c3d3f
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
hash: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(0x000000003FFFFD88, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AllocateOperation(): start
TEE.GenerateRandom(0x000000003FFFFED0, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
@cipher: e94431cd22a6029185d0dbb1a17b5d62843bfeef25591583d2d668ec6fed1c692f88ce4754d690c346c8d9f2726
630e0386abf4e45699a2ca2b34b344eaa454bc489c
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f
verify ok
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(0x000000003FFFFC68, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AllocateOperation(): start
TEE.GenerateRandom(0x000000003FFFFEC8, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AEInit(): start
TEE.AEEncryptFinal(): start
TEE.FreeOperation(): start
@cipher: c23e9ce04589e80a66debe23a788ae5393bdcd8e875e87e1bcf2b2d998f6418ccc6ee4ab112fdbfc5175868691e
fb40781a318ff439d30b49cc9f726886ad42d5be15
@tag: a551f999317b3fbd1eea7b622ce2caee
TEE.AllocateOperation(): start
TEE.AEInit(): start
TEE.AEDecryptFinal(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f

```

```

TEE.AllocateOperation(): start
TEE.AllocateTransientObject(): start
TEE.InitValueAttribute(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(0x000000003FFFFFFE28, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AsymmetricSignDigest(): start
TEE.FreeOperation(): start
@signature: d6e6b6e54db8b6a62fc1927886938bead27f4813f19ce77182e3016b5426bcad067ca98cd75f9dfddafe9eb0
655c48df992d3ad674db69d831f26ae63caf1405
TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation:
TEE.FreeTransientObject(): start
verify ok
main end

```

4.5 ta-ref with OPTEE

Make sure optee_3.9.0_rpi3 has been built already.

4.5.1 Cloning source and building

Clone the source

```

git clone https://192.168.100.100/rinkai/ta-ref.git
cd ta-ref
git checkout teep-device-tb-slim
git submodule sync --recursive
git submodule update --init --recursive

```

Build

```

export OPTEE_DIR=<path to optee_3.9.0_rpi3>
source env/optee_rpi3.sh
make build test-bin MACHINE=RPI3 TEST_DIR=test.hello
make build test-bin MACHINE=RPI3 TEST_DIR=test_gp

```

4.5.2 Check ta-ref by running test_gp, test_hello, on QEMU

Copy the test.hello and test_gp programs to QEMU buildroot directory

```

mkdir -p optee_3.9.0_qemu/out-br/target/home/gitlab/out/{test.hello,test_gp}
cp ta-ref/test.hello/optee/App/optee_ref.ta optee_3.9.0_qemu/out-br/target/home/gitlab/out/test.hello/
cp ta-ref/test.hello/optee/Enclave/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
  optee_3.9.0_qemu/out-br/target/home/gitlab/out/test.hello/
cp ta-ref/test_gp/optee/App/optee_ref.ta optee_3.9.0_qemu/out-br/target/home/gitlab/out/test_gp/
cp ta-ref/test_gp/optee/Enclave/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
  optee_3.9.0_qemu/out-br/target/home/gitlab/out/test_gp/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
cp ./test_gp/optee/Enclave/Enclave.nm /TEE/demo/rpi3/optee_3.9.0_qemu/out-br/target/home/gitlab/out/test_gp/

```

4.5.2.1 test.hello

Run test_hello

```

cp /home/gitlab/out/test.hello/
cp a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta /home/gitlab/out/
ln -s /home/gitlab/out/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
  /lib64/optee.armtz/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
./optee_ref.ta
--- enclave log start---
ecall_ta_main() start
hello world!
ecall_ta_main() end
--- enclave log end---

```

If executed successfully, you see above messages

4.5.2.2 test_gp

Run test_gp

```
cd /home/gitlab/out/test_gp/
cp a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta /home/gitlab/out/
ln -s /home/gitlab/out/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
    /lib64/optee.armtz/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
./optee_ref.ta
start TEEC.InvokeCommand
--- enclave log start---
ecall.ta_main() start
@random: fe0c7d3eebf9bd5e63b8a0cce29af7eb
@GP REE time 1612156259 sec 390 millis
@GP System time 249187 sec 954 millis
256 bytes read: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20212223242526272829
2a2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b
5c5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d
8e8f909192939495969798999a9b9c9d9e9f9a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc
c0c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
hash: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@cipher: 30a558176172c53be4a2ac320776de105da79c29726879fe67d06b629f065731285f8a90f8a521ce34ecee51e1
5e928d157ea10d149bb687dd78be79469c28696506283edcda527fcd86f6a47e852bbc3488df3fc651b46b034faf4ab5f12f
51a285478ea01e58d40e8177d415be243df93b23cdf889feb91fa3be8906fe190d836fe61168aed047340be1054dd88a381
ef25381d906ea3780ba74fb1cfe1434cbd168de8386dce2e2b92ee0fc432f3c0514f462cbeaf96753b174a4a673f323e671
61272fe932ead4bc95770fcc130dd5877b521d6a79f961eeadd1680042f69257ccf9368927aa170176af8ac211dd22161997
7224837232dad970220f4
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9f9a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc0
c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
@cipher: ff409d8fe203bf0d81de36832b86c702f07edd343f408d3a2fb5ab347b4f72b10031efff0c17b7e0bc56c3f2f95
f53c0d731ed87eb3e1187b6714a25cfc65082284682b44450941654e7edc99af0f7b037c3ba9ea731036070aa9496e34cfeb
db6845e8aa9955416ba227970d3dd1f8207b5743e1490a7f5fd78d81fce0a24576de06a2f528d49c5b11e79a5cab015806ba
d73f118e205a3645a95b2b330ff9da12e00c693e7ee8cfd04eb0f08c3c657c4fa0ae384ed2d5ab1e15ffc835c3e4cc116cd
1049611f896cf445ab36dc8b393a6fe75d20d45b2273a5d8c2d3b935e3f22b2c8b2b4c952812d66a902155d288d5f26ac6722
fe72498bd72ea523c914c
@tag: 9b357baf76d2632fa7d16231640d6324
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9f9a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc0
c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
@digest: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@signature: 719fa9898f3423b754675b835268f9b2368b77a429eeabf7369d60d135dee08158c3902fd2ed3c1bf17cb34e
76f2ba25da915fa3970c757962f7533c8d8bad7d
@@TEE.FreeOperation:
verify ok
ecall.ta_main() end
--- enclave log end---
res = TEEC.SUCCESS; TEEC.InvokeCommand succeeded!
```

If executed successfully, you see above messages

4.6 ta-ref with SGX

Build ta-ref for Intel SGX platforms

4.6.1 Cloning source and building

Clone the source

```
git clone https://192.168.100.100/rinkai/ta-ref.git
cd ta-ref
git checkout teep-device-tb-slim
git submodule sync --recursive
git submodule update --init --recursive
```

Build

```
source /opt/intel/sgxsdk/environment
source env/sgx.x64.sh
make build test-bin MACHINE=NUC TEST_DIR=test.hello
make build test-bin MACHINE=NUC TEST_DIR=test_gp
```

4.6.2 Check ta-ref by running test_gp, test_hello, simulation mode on any pc

Copy the ta-ref's test_hello & test_gp executables to test directory

4.6.2.1 test_hello

Run test_hello

```
cp test_hello/sgx/Enclave/enclave.signed.so <test directory>
cp test_hello/sgx/App/sgx_app <test directory>
<test directory>/sgx_app
hello world!
Info: Enclave successfully returned.
```

4.6.2.2 test_gp

Run test_gp

```
cp test_gp/sgx/Enclave/enclave.signed.so <test directory>
cp test_gp/sgx/App/sgx_app <test directory>
<test directory>/sgx_app
main start
TEE.GenerateRandom(): start
@random: f35c1d1e4bbf6641c5511c9dc5aaf638
TEE.GetREETime(): start
request to get unix time 1612257364, 199
@GP REE time 1612257364 sec 199 millis
TEE.GetSystemTime(): start
@GP System time 727941859 sec 984 millis
TEE.CreatePersistentObject(): start
request to open FileOne flags 241 -> 3
TEE.WriteObjectData(): start
request to write 256 bytes to descriptor 3
TEE.CloseObject(): start
request to close descriptor 3
TEE.OpenPersistentObject(): start
request to open FileOne flags 0 -> 3
TEE.ReadObjectData(): start
request to read 256 bytes from descriptor 3
TEE.CloseObject(): start
request to close descriptor 3
256 bytes read: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20212223242526272829
2a2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b
5c5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d
8e8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbef
c0c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9daddbdcdddedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1
f2f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
hash: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(): start
TEE.AllocateOperation(): start
TEE.GenerateRandom(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
@cipher: 7427bff21e729a824a239e25332ebd455d18fa6aec1ec6618b77c252f768e0a9345608b0135727568867ce5b0fa
c872f6647787861b88220840281f3944eeaa456a2769081e6598079b52edc541e2201ffd2e96a6c3e485be25a0ce4f5c07544
aa0c67b3e34bd069b293843daf66db51b751b3c09f2a9c6912c22a6062c8ecbd0effd4698081660e218f6f0c1249e3691a33
e91836953953513040eb29ce709efe50f96e67f07d6alb00f08beacebc5950f9744b0049cb76ec5ba17a49d7270b60034c47
23bb79dc61d465062b039ae8d93f98c2391ee2b02b7b537b375e01cc5eeb8eb2e62df839048db0f1fdbdd1b7f5c6ef2faa1
a5b305ef045936c9146f8
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc0
c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9daddbdcdddedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
```

```

f3f4f5f6f7f8f9fafbfcfdfeff
verify ok
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(): start
TEE.AllocateOperation(): start
TEE.GenerateRandom(): start
TEE.AEInit(): start
TEE.AEEncryptFinal(): start
TEE.FreeOperation(): start
@cipher: e33f34122c80b9a10002725e4e21542256da7c7cd3f6dd1b62b71cf8308f9e4a0daa50b29880a8f76707c4ed432
549c4da9e68e7930189d2127fdd7aa2379106090814b5deed9a9e161ef0886da03a2a94c3fb9e0faadfd1ce8bb09fb5388bb
23a042944f8e269d486aa4f21a91a41968184122520dfc308850059efce660a52adb17361bd52f570bfba05ccad32ffa9ea
c94914725ded073355f28eb3dc30d60f00cfd2de76c3a05df8bef32f302bb4d14b493a3a90b1dee4eba64e625695c4d58ec4
febf8436d62e4cac82fcb00e60c8138af7176995a742b08a572f64e539e9f9850a9f6f33907a829108ca6540332aab53f3f
6a4fd2c3de35c5556a427
@tag: 4c920ce2aef079e468ab24e25730d9d2
TEE.AllocateOperation(): start
TEE.AEInit(): start
TEE.AEDecryptFinal(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbbfbfc0
c1c2c3c4c5c6c7c8c9cacbcccdcecfcd0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfeff
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateOperation(): start
TEE.AllocateTransientObject(): start
TEE.InitValueAttribute(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(): start
TEE.AsymmetricSignDigest(): start
TEE.FreeOperation(): start
@signature: 100b392ce043e9b8dc703088f505dd3083ec47bfcb8d59d968a66b54e80464d684d56dc9c44336f08fd96309
79863a2d8fb7cd672a819ef609357e9ac6a3d80e
TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation():
TEE.FreeTransientObject(): start
verify ok
main end
Info: Enclave successfully returned.

```

5 Running on Dev Boards

5.1 Keystone, Unleashed

Make sure Keystone and other dependant sources have been built

5.1.1 Preparation of rootfs on SD Card

Build a modified gdisk which can handle the sifive specific partition types.

Prerequisites: libncursesw5-dev, libpopt-dev

```

$ cd ..
$ sudo apt install libncursesw5-dev lib64ncurses5-dev uuid-dev libpopt-dev build-essential
$ git clone https://192.168.100.100/rinkai/gptfdisk.git
$ cd gptfdisk
$ git checkout -b risc-v-sd 3d6a15873f582803aa8ad3288b3e32d3daff9fde
$ make

```

5.1.1.1 Create SD-card partition manually

```

sudo ./gdisk /dev/mmcblk0
GPT fdisk (gdisk) version 1.0.4
Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present
Found valid GPT with protective MBR; using GPT.
Command (? for help): n
Partition number (1-128, default 1): 1
First sector (34-15523806, default = 2048) or {+}-size{KMGTP}:
Last sector (2048-15523806, default = 15523806) or {+}-size{KMGTP}: 67583
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300): 5202
Changed type of partition to 'SiFive bare-metal (or stage 2 loader)'
Command (? for help): n
Partition number (2-128, default 2): 4
First sector (34-15523806, default = 67584) or {+}-size{KMGTP}:
Last sector (67584-15523806, default = 15523806) or {+}-size{KMGTP}: 67839
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300): 5201
Changed type of partition to 'SiFive FSBL (first-stage bootloader)'
Command (? for help): n
Partition number (2-128, default 2):
First sector (34-15523806, default = 69632) or {+}-size{KMGTP}: 264192
Last sector (264192-15523806, default = 15523806) or {+}-size{KMGTP}:
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300): 8300
Changed type of partition to 'Linux filesystem'
Command (? for help): p
Disk /dev/mmcblk0: 15523840 sectors, 7.4 GiB
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 11A0F8F6-D5DE-4993-8C0D-D543DFBA17AD
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 15523806
Partitions will be aligned on 2048-sector boundaries
Total free space is 198366 sectors (96.9 MiB)
Number  Start (sector)    End (sector)  Size      Code  Name
   1            2048             67583     32.0 MiB   5202   SiFive bare-metal (...)
   2          264192          15523806     7.3 GiB   8300   Linux filesystem
   4            67584          67839     128.0 KiB  5201   SiFive FSBL (first-...
Command (? for help): i
Partition number (1-4): 4
Partition GUID code: 5B193300-FC78-40CD-8002-E86C45580B47 (SiFive FSBL (first-stage bootloader))
Partition unique GUID: FC1FBC7C-EC94-4B0A-9DAF-0ED85452B885
First sector: 67584 (at 33.0 MiB)
Last sector: 67839 (at 33.1 MiB)
Partition size: 256 sectors (128.0 KiB)
Attribute flags: 0000000000000000
Partition name: 'SiFive FSBL (first-stage bootloader)'
Command (? for help): i
Partition number (1-4): 1
Partition GUID code: 2E54B353-1271-4842-806F-E436D6AF6985 (SiFive bare-metal (or stage 2 loader))
Partition unique GUID: 2FFF07EF-E44A-4278-A16D-C29697C6653D
First sector: 2048 (at 1024.0 KiB)
Last sector: 67583 (at 33.0 MiB)
Partition size: 65536 sectors (32.0 MiB)
Attribute flags: 0000000000000000
Partition name: 'SiFive bare-metal (or stage 2 loader)'
Command (? for help): wq
Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!
Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/mmcblk1.
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.

```

5.1.1.2 Write boot and rootfs files into SD-card

Build FSBL for hifive-Unleashed board

```

$ git clone https://github.com/keystone-enclave/freedom-u540-c000-bootloader.git
$ cd freedom-u540-c000-bootloader
$ git checkout -b dev-unleashed bbfcc288fb438312af51adef420aa444a0833452
$# Make sure riscv64 compiler set to PATH (export PATH=$KEYSTONE_DIR/riscv/bin:$PATH)
$ make

```

Writing fsbl.bin and bbl.bin

```

sudo dd if=freedom-u540-c000-bootloader/fsbl.bin of=/dev/mmcblk0p4 bs=4096 conv=fsync

```

```
sudo dd if=$KEYSTONE_DIR/hifive-work/bbl.bin of=/dev/mmcblk0p1 bs=4096 conv=fsync
```

Once files written, insert the SD-card into unleashed

5.1.2 Copying binaries of test_hello and test_gp

```
sudo mount /dev/mmcblk0p1 /media/rootfs/
sudo mkdir /media/rootfs/root/{test_hello,test_gp}
Copy test_hello
sudo cp ta-ref/test_hello/keystone/Enclave/Enclave.eapp_riscv /media/rootfs/root/test_hello/
sudo cp ta-ref/test_hello/keystone/Enclave/App.client /media/rootfs/root/test_hello/
sudo cp $KEYSTONE.SDK_DIR/rts/eyrie/eyrie-rt /media/rootfs/root/test_hello/
Copy test_gp
sudo cp ta-ref/test_gp/keystone/Enclave/Enclave.eapp_riscv /media/rootfs/root/test_gp/
sudo cp ta-ref/test_gp/keystone/Enclave/App.client /media/rootfs/root/test_gp/
sudo cp $KEYSTONE.SDK_DIR/rts/eyrie/eyrie-rt /media/rootfs/root/test_gp/
```

Now, we are ready to test on unleashed board.

5.1.3 Check test_hello and test_gp on Unleased

1. Insert SD-card into unleashed board
2. Boot Hifive-Unleased board
3. Connect Unleased board with your development machine over USB-Serial cable (/dev/ttyUSB1)
4. Checking on Unleased

Login to serial console with user=root, passwd=sifive

```
buildroot login: root
Password:
$
```

test_hello:

```
insmod keystone-driver.ko
./App.client Enclave.eapp_riscv eyrie-rt
hello world!
```

test_gp:

```
insmod keystone-driver.ko
./App.client Enclave.eapp_riscv eyrie-rt
main start
TEE.GenerateRandom(0x000000003FFFFFFE0, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@random: 5ea8741bd8a3b298cf53d214eca693fb
TEE.GetREETime(): start
@[SE] gettimeofday 77 sec 865873 usec -> 0
@GP REE time 77 sec 865 millis
TEE.GetSystemTime(): start
@GP System time 100063195 sec 609 millis
TEE.CreatePersistentObject(): start
@[SE] open file FileOne flags 241 -> 3 (0)
TEE.WriteObjectData(): start
@[SE] write desc 3 buf 480d0 len 256-> 256
TEE.CloseObject(): start
@[SE] close desc 3 -> 0
TEE.OpenPersistentObject(): start
@[SE] open file FileOne flags 0 -> 3 (0)
TEE.ReadObjectData(): start
@[SE] read desc 3 buf fff41664 len 256-> 256
TEE.CloseObject(): start
@[SE] close desc 3 -> 0
256 bytes read: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20212223242526272829
2a2b2c2d2e2f303132333435363738393a3b3c3d3f
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
hash: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(0x000000003FFFFD88, 32): start
```

```

@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AllocateOperation(): start
TEE.GenerateRandom(0x000000003FFFFED0, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
@cipher: e94431cd22a6029185d0dbb1a17b5d62843bfeef25591583d2d668ec6fed1c692f88ce4754d690c346c8d9f2726
630e0386abf4e45699a2ca2b34b344eaa454bc489c
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f
verify ok
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(0x000000003FFFFC68, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AllocateOperation(): start
TEE.GenerateRandom(0x000000003FFFFEC8, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AEInit(): start
TEE.AEEncryptFinal(): start
TEE.FreeOperation(): start
@cipher: c23e9ce04589e80a66debe23a788ae5393bdcd8e875e87e1bcf2b2d998f6418ccc6ee4ab112fdbfc5175868691e
fb40781a318ff439d30b49cc9f726886ad42d5be15
@tag: a551f999317b3fbdleaa7b622ce2caee
TEE.AllocateOperation(): start
TEE.AEInit(): start
TEE.AEDecryptFinal(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateOperation(): start
TEE.AllocateTransientObject(): start
TEE.InitValueAttribute(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(0x000000003FFFFE28, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AsymmetricSignDigest(): start
TEE.FreeOperation(): start
@signature: d6e6b6e54db8b6a62fc1927886938bead27f4813f19ce77182e3016b5426bcad067ca98cd75f9dfddafe9eb0
655c48df992d3ad674db69d831f26ae63caf1405
TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation:
TEE.FreeTransientObject(): start
verify ok
main end

```

Test is successful.

5.2 OPTEE, RPI3

Make sure OPTEE v3.9.0 and other dependant sources have been built

5.2.1 Preparation of rootfs on SD Card

Use following examples to create partitions of boot and roots on SD-card

```

make img-help
$ fdisk /dev/sdx # where sdx is the name of your sd-card
> p # prints partition table
> d # repeat until all partitions are deleted

```

```

> n          # create a new partition
> p          # create primary
> 1          # make it the first partition
> <enter>    # use the default sector
> +32M      # create a boot partition with 32MB of space
> n          # create rootfs partition
> p
> 2
> <enter>
> <enter>    # fill the remaining disk, adjust size to fit your needs
> t          # change partition type
> 1          # select first partition
> e          # use type 'e' (FAT16)
> a          # make partition bootable
> 1          # select first partition
> p          # double check everything looks right
> w          # write partition table to disk.

```

Usually your SD-card detected as `/dev/mmcblk0`. After partition it looks like below BOOT partition = `/dev/mmcblk0p1` rootfs partition = `/dev/mmcblk0p2`

Write boot file

```

$ mkfs.vfat -F16 -n BOOT /dev/mmcblk0p1
$ mkdir -p /media/boot
$ sudo mount /dev/mmcblk0p1 /media/boot
$ cd /media
$ gunzip -cd optee_3.9.0_rpi3/out-br/images/rootfs.cpio.gz | sudo cpio -idmv "boot/*"
$ umount boot

```

Write rootfs

```

$ mkfs.ext4 -L rootfs /dev/mmcblk0p2
$ mkdir -p /media/rootfs
$ sudo mount /dev/mmcblk0p2 /media/rootfs
$ cd rootfs
$ gunzip -cd <your-base-dir>/optee_3.9.0_rpi3/build/./out-br/images/rootfs.cpio.gz | sudo cpio -idmv
$ rm -rf /media/rootfs/boot/*
$ cd .. && sudo umount rootfs

```

If you use CI from AIST, download `rpi3_sdimage` as follows

```

$ wget http://192.168.100.100:2000/optee_rpi3_sdimage.tar.xz
$ tar xf optee_rpi3_sdimage.tar.xz
$ dd if=rpi3_sdimage.bin of=/dev/mmcblk0p2 conv=fsync bs=4096

```

Now SD-card is ready to boot RPI3.

5.2.2 Copying binaries of `test.hello` and `test_gp` to rootfs partition

Copying `test.hello` & `test_gp`

```

$ sudo mount /dev/mmcblk0p2 /media/rootfs
$ sudo mkdir -p /media/rootfs/home/gitlab/out/{test.hello,test_gp}
$ sudo cp ta-ref/test.hello/optee/App/optee.ref.ta /media/rootfs/home/gitlab/out/test.hello/
$ sudo cp ta-ref/test.hello/optee/Enclave/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
/media/rootfs/home/gitlab/out/test.hello/
$ sudo cp ta-ref/test_gp/optee/App/optee.ref.ta /media/rootfs/home/gitlab/out/test_gp/
$ sudo cp ta-ref/test_gp/optee/Enclave/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
/media/rootfs/home/gitlab/out/test_gp/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
$ sudo cp ta-ref/test_gp/optee/Enclave/Enclave.nm /media/rootfs/home/gitlab/out/test_gp/

```

5.2.3 Check `test.hello` and `test_gp`

1. Insert SD-card into RPI3 board, then power-on
2. Connect RPI3 board Serial console to your laptop (`/dev/ttyUSB0` over minicom)
3. Checking on RPI3

Login to Serial console and enter "root" as username

```
buildroot login: root
Password:
$
```

test.hello:

```
cp /home/gitlab/out/test.hello/
cp a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta /home/gitlab/out/
ln -s /home/gitlab/out/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
    /lib64/optee.armtz/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
./optee.ref.ta
--- enclave log start---
ecall.ta.main() start
hello world!
ecall.ta.main() end
--- enclave log end---
```

If executed successfully, you see above messages

test.gp:

```
cd /home/gitlab/out/test.gp/
cp a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta /home/gitlab/out/
ln -s /home/gitlab/out/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
    /lib64/optee.armtz/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
./optee.ref.ta
start TEEC.InvokeCommand
--- enclave log start---
ecall.ta.main() start
@random: fe0c7d3eefb9bd5e63b8a0cce29af7eb
@GP REE time 1612156259 sec 390 millis
@GP System time 249187 sec 954 millis
256 bytes read: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20212223242526272829
2a2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b
5c5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d
8e8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc
c0c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9dadbdcdddedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
hash: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@cipher: 30a558176172c53be4a2ac320776de105da79c29726879fe67d06b629f065731285f8a90f8a521ce34ecee51e1
5e928d157ea10d149bb687dd78be79469c28696506283edcda527fcd86f6a47e852bbc3488df3fc651b46b034faf4ab5f12f
51a285478ea01e58d40e8177d415be243df93b23cdf889feb91fa3be8906fe190d836fe61168aed0473406be1054dd88a381
ef25381d920ea3780ba74fb1cfe1434cbd168de8386dccc2e2b92ee0fc432f3c0514f462cbeaf96753b174a4a673f323e671
61272fe932ead4bc95770fcc130dd5877b521d6a79f961eeadd1680042f69257ccf9368927aa170176faf8ac211dd22161997
7224837232dad970220f4
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc0
c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9dadbdcdddedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
@cipher: ff409d8fe203bf0d81de36832b86c702f07edd343f408d3a2fb5ab347b4f72b10031efff0c17b7e0bc56c3f2f95
f53c0d731ed87eb3e1187b6714a25cfc65082284682b44450941654e7edc99af0f7b037c3ba9ea731036070aa9496e34cfefb
db6845e8aa9955416ba227970d3dd1f8207b5743e1490a7f5fd78d81fce0a24576de06a2f528d49c5b11e79a5cab015806ba
d73f118e205a3645a95b2b330ff9da12e00c693e7ee8cfd04eb0f08c3c657c4fa0ae384ed2d5able15fffc835c3e4cc116cd
1049611f896cf445ab36dc8b393a6fe75d20d45b2273a5d8c2d3b3935e3f22bc82b24c952812d66a902155d288d5f26ac6722
fe72498bd72ea523c914c
@tag: 9b357baf76d2632fa7d16231640d6324
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc0
c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9dadbdcdddedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
@digest: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@signature: 719fa9898f3423b754675b835268f9b2368b77a429eeabf7369d60d135dee08158c3902fd2ed3c1bf17cb34e
76f2ba25da915fa3970c757962f7533c8d8bad7d
@@TEE.FreeOperation:
verify ok
ecall.ta.main() end
--- enclave log end---
res = TEEC.SUCCESS; TEEC.InvokeCommand succeeded!
```

If executed successfully, you see above messages

5.3 SGX, NUC

Make sure SGX SDK, sgx driver and other dependant sources have been built and installed on NUC machine

5.3.1 Copying binaries of test_hello and test_gp to NUC machine

Login to NUC machine over SSH (Assuming that SSH enabled on NIC machine). Assuming that ta-ref was natively built on NUC machine at ~/ta-ref

```
ssh <ssh-user>@<IP-Address> 'mkdir -p ~/test_hello,test_gp'
scp ta-ref/test_hello/sgx/Enclave/enclave.signed.so <ssh-user>@<IP-Address>:~/test_hello
scp ta-ref/test_hello/sgx/App/sgx_app <ssh-user>@<IP-Address>:~/test_hello
scp ta-ref/test_gp/sgx/Enclave/enclave.signed.so <ssh-user>@<IP-Address>:~/test_gp
scp ta-ref/test_gp/sgx/App/sgx_app <ssh-user>@<IP-Address>:~/test_gp
```

Now can login to NUC machine for further testing.

5.3.2 Check test_hello and test_gp

Checking test_hello

```
cd ~/test_hello
./sgx_app
hello world!
Info: Enclave successfully returned.
```

Checking test_gp

```
cd ~/test_gp
./sgx_app
main start
TEE.GenerateRandom(): start
@random: f35c1d1e4bbf6641c5511c9dc5aaf638
TEE.GetREETime(): start
request to get unix time 1612257364, 199
@GP REE time 1612257364 sec 199 millis
TEE.GetSystemTime(): start
@GP System time 727941859 sec 984 millis
TEE.CreatePersistentObject(): start
request to open FileOne flags 241 -> 3
TEE.WriteObjectData(): start
request to write 256 bytes to descriptor 3
TEE.CloseObject(): start
request to close descriptor 3
TEE.OpenPersistentObject(): start
request to open FileOne flags 0 -> 3
TEE.ReadObjectData(): start
request to read 256 bytes from descriptor 3
TEE.CloseObject(): start
request to close descriptor 3
256 bytes read: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20212223242526272829
2a2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b
5c5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d
8e8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbef
c0c1c2c3c4c5c6c7c8c9cacbcccdcecfcd0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1
f2f3f4f5f6f7f8f9fafbfcfdfeff
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
hash: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(): start
TEE.AllocateOperation(): start
TEE.GenerateRandom(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
@cipher: 7427bff21e729a824a239e25332ebd455d18fa6aec1ec6618b77c252f768e0a9345608b0135727568867ce5b0fa
c872f6647787861b88220840281f3944eea456a2769081e6598079b52edc541e2201ffd2e96a6c3e485be25a0ce4f5c07544
aa0c67b3e34bd069b293843daf66db51b751b3c09f2a9c6912c22a6062c8ecbd0effd4698081660e218f6f0c1249e3691a33
e91836953953513040eb29ce709efe50f96e67f07d6a1b00f08beacebc5950f9744b0049cb76ec5ba17a49d7270b60034c47
23bb79dc61d465062b0394e8d93f98c2391ee2b02b7b537b375e0elcc5eeb8eb2e62df839048db0f1fddbdlb7f5c6ef2faa1
a5b305ef045936c9146f8
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc0
```

```

c1c2c3c4c5c6c7c8c9cacbcccdcecfcd0d1d2d3d4d5d6d7d8d9dadbdcdddedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfeff
verify ok
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(): start
TEE.AllocateOperation(): start
TEE.GenerateRandom(): start
TEE.AEInit(): start
TEE.AEEncryptFinal(): start
TEE.FreeOperation(): start
@cipher: e33f34122c80b9a10002725e4e21542256da7c7cd3f6dd1b62b71cf8308f9e4a0daa50b29880a8f76707c4ed432
549c4da9e68e7930189d2127fdd7aa2379106090814b5deed9a9e161ef0886da03a2a94c3fb9e0faadfd1ce8bb09fb5388bb
23a042944fbeb269d486aa4f21a91a41968184122520dfc308850059efce660a52adb17361bd52f570bfba05cccad32ffa9ea
c94914725ded073355f28eb3dc30d60f00cfd2de76c3a05df8bef32f302bb4d14b493a3a90b1dee4eba64e625695c4d58ec4
febfb8436d62e4cac82fcbcd00e60c8138af7176995a742b08a572f64e539e9f9850a9f6f33907a829108ca6540332aab53f3f
6a4fd2c3de35c5556a427
@tag: 4c920ce2aef079e468ab24e25730d9d2
TEE.AllocateOperation(): start
TEE.AEInit(): start
TEE.AEDecryptFinal(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9f9a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbbfc0
c1c2c3c4c5c6c7c8c9cacbcccdcecfcd0d1d2d3d4d5d6d7d8d9dadbdcdddedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfeff
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateOperation(): start
TEE.AllocateTransientObject(): start
TEE.InitValueAttribute(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(): start
TEE.AsymmetricSignDigest(): start
TEE.FreeOperation(): start
@signature: 100b392ce043e9b8dc703088f505dd3083ec47bfcb8d59d968a66b54e80464d684d56dc9c44336f08fd96309
79863a2d8fb7cd672a819ef609357e9ac6a3d80e
TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation:
TEE.FreeTransientObject(): start
verify ok
main end
Info: Enclave successfully returned.

```

6 API Compare With Full-Set of GP API

6.1 GP API

API Functions by Category

APIs supported by both GP and AIST-GP are in Blue

API list from TEE Internal Core API Specification documentation, GlobalPlatform Technology

Asymmetric	TEE_FreeOperation
TEE_AsymmetricDecrypt	TEE_GetOperationInfo
TEE_AsymmetricEncrypt	TEE_GetOperationInfoMultiple
TEE_AsymmetricSignDigest	TEE_IsAlgorithmSupported
TEE_AsymmetricVerifyDigest	TEE_ResetOperation
Authenticated Encryption	TEE_SetOperationKey
TEE_AEDecryptFinal	TEE_SetOperationKey2
TEE_AEEncryptFinal	Initialization
TEE_AEInit	TEE_BigIntInit
TEE_AEUpdate	TEE_BigIntInitFMM
TEE_AEUpdateAAD	TEE_BigIntInitFMMContext
Basic Arithmetic	Internal Client API
TEE_BigIntAdd	TEE_CloseTASession
TEE_BigIntDiv	TEE_InvokeTACommand
TEE_BigIntMul	TEE_OpenTASession
TEE_BigIntNeg	Key Derivation
TEE_BigIntSquare	TEE_DeriveKey
TEE_BigIntSub	Logical Operation
Cancellation	TEE_BigIntCmp
TEE_GetCancellationFlag	TEE_BigIntCmpS32
TEE_MaskCancellation	TEE_BigIntGetBit
TEE_UnmaskCancellation	TEE_BigIntGetBitCount
Converter	TEE_BigIntShiftRight
TEE_BigIntConvertFromOctetString	MAC
TEE_BigIntConvertFromS32	TEE_MACCompareFinal
TEE_BigIntConvertToOctetString	TEE_MACComputeFinal
TEE_BigIntConvertToS32	TEE_MACInit
Data Stream Access	TEE_MACUpdate
TEE_ReadObjectData	Memory Allocation and Size of Objects
TEE_SeekObjectData	TEE_BigIntFMMContextSizeInU32
TEE_TruncateObjectData	TEE_BigIntFMMSizeInU32
TEE_WriteObjectData	TEE_BigIntSizeInU32 (macro)
Deprecated	Memory Management
TEE_CloseAndDeletePersistentObject	TEE_CheckMemoryAccessRights
TEE_CopyObjectAttributes	TEE_Free
TEE_GetObjectInfo	TEE_GetInstanceData
TEE_RestrictObjectUsage	TEE_Malloc
Fast Modular Multiplication	TEE_MemCompare
TEE_BigIntComputeFMM	TEE_MemFill
TEE_BigIntConvertFromFMM	TEE_MemMove
TEE_BigIntConvertToFMM	TEE_Realloc
Generic Object	TEE_SetInstanceData
TEE_CloseObject	Message Digest
TEE_GetObjectBufferAttribute	TEE_DigestDoFinal
TEE_GetObjectInfo (deprecated)	TEE_DigestUpdate
TEE_GetObjectInfo1	Modular Arithmetic
TEE_GetObjectValueAttribute	TEE_BigIntAddMod
TEE_RestrictObjectUsage (deprecated)	TEE_BigIntInvMod
TEE_RestrictObjectUsage1	TEE_BigIntMod
Generic Operation	TEE_BigIntMulMod
TEE_AllocateOperation	TEE_BigIntSquareMod
TEE_CopyOperation	TEE_BigIntSubMod

Other Arithmetic

- TEE_BigIntComputeExtendedGcd
- TEE_BigIntIsProbablePrime
- TEE_BigIntRelativePrime

Panic Function

- TEE_Panic

Persistent Object

- TEE_CloseAndDeletePersistentObject
(deprecated)
- TEE_CloseAndDeletePersistentObject1
- TEE_CreatePersistentObject
- TEE_OpenPersistentObject
- TEE_RenamePersistentObject

Persistent Object Enumeration *

- TEE_AllocatePersistentObjectEnumerator
- TEE_FreePersistentObjectEnumerator
- TEE_GetNextPersistentObject
- TEE_ResetPersistentObjectEnumerator
- TEE_StartPersistentObjectEnumerator

Property Access

- TEE_AllocatePropertyEnumerator
- TEE_FreePropertyEnumerator
- TEE_GetNextProperty
- TEE_GetPropertyAsBinaryBlock
- TEE_GetPropertyAsBool
- TEE_GetPropertyAsIdentity
- TEE_GetPropertyAsString
- TEE_GetPropertyAsU32
- TEE_GetPropertyAsU64
- TEE_GetPropertyAsUUID
- TEE_GetPropertyName

- TEE_ResetPropertyEnumerator

- TEE_StartPropertyEnumerator

Random Data Generation

- TEE_GenerateRandom

Symmetric Cipher

- TEE_CipherDoFinal
- TEE_CipherInit
- TEE_CipherUpdate

TA Interface

- TA_CloseSessionEntryPoint
- TA_CreateEntryPoint
- TA_DestroyEntryPoint
- TA_InvokeCommandEntryPoint
- TA_OpenSessionEntryPoint

Time

- TEE_GetREETime
- TEE_GetSystemTime
- TEE_GetTAPersistentTime
- TEE_SetTAPersistentTime
- TEE_Wait

Transient Object

- TEE_AllocateTransientObject
- TEE_CopyObjectAttributes (deprecated)
- TEE_CopyObjectAttributes1
- TEE_FreeTransientObject
- TEE_GenerateKey
- TEE_InitRefAttribute
- TEE_InitValueAttribute
- TEE_PopulateTransientObject
- TEE_ResetTransientObject

7 Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

_TEE_ObjectHandle	32
_TEE_OperationHandle	33
addrinfo	35
enclave_report	36
pollfd	37
report	38
sm_report	38
TEE_Attribute	39
TEE_Identity	40
TEE_ObjectInfo	41
TEE_OperationInfo	42
TEE_OperationInfoKey	44
TEE_OperationInfoMultiple	44
TEE_Param	46
TEE_SEAID	47
TEE_SEReaderProperties	47
TEE_Time	48
TEE_UUID	48
TEEC_Context	49
TEEC_Operation	50
TEEC_Parameter	51
TEEC_RegisteredMemoryReference	52
TEEC_Session	54
TEEC_SharedMemory	54
TEEC_TempMemoryReference	56
TEEC_UUID	57
TEEC_Value	58

8 File Index

8.1 File List

Here is a list of all files with brief descriptions:

ta-ref/api/tee-internal-api-cryptlib.c	223
ta-ref/api/include/compiler.h	59
ta-ref/api/include/report.h	68
ta-ref/api/include/tee-common.h	
Common type and definitions of RISC-V TEE	70
ta-ref/api/include/tee-ta-internal.h	
Candidate API list for Global Platform like RISC-V TEE	71
ta-ref/api/include/tee_api_defines.h	99
ta-ref/api/include/tee_api_defines_extensions.h	143
ta-ref/api/include/tee_api_types.h	148
ta-ref/api/include/tee_client_api.h	156
ta-ref/api/include/tee_internal_api.h	171
ta-ref/api/include/tee_internal_api_extensions.h	171
ta-ref/api/include/tee_ta_api.h	174
ta-ref/api/include/test_dev_key.h	178
ta-ref/api/include/trace.h	179
ta-ref/api/include/trace_levels.h	187
ta-ref/api/keystone/tee-internal-api-machine.c	189
ta-ref/api/keystone/tee-internal-api.c	190
ta-ref/api/keystone/tee_api_tee_types.h	210
ta-ref/api/keystone/teec_stub.c	217
ta-ref/api/keystone/trace.c	221
ta-ref/api/optee/tee_api_tee_types.h	213
ta-ref/api/sgx/tee-internal-api.c	201
ta-ref/api/sgx/tee_api_tee_types.h	213

9 Class Documentation

9.1 __TEE_ObjectHandle Struct Reference

```
#include <tee_api_tee_types.h>
```

Public Attributes

- unsigned int [type](#)
- int [flags](#)
- int [desc](#)
- struct AES_ctx [persist_ctx](#)
- unsigned char [public_key](#) [TEE_OBJECT_KEY_SIZE]
- unsigned char [private_key](#) [TEE_OBJECT_SKEY_SIZE]

9.1.1 Member Data Documentation

9.1.1.1 desc int __TEE_ObjectHandle::desc

9.1.1.2 flags int __TEE_ObjectHandle::flags

9.1.1.3 persist_ctx struct AES_ctx __TEE_ObjectHandle::persist_ctx

9.1.1.4 private_key unsigned char __TEE_ObjectHandle::private_key

9.1.1.5 public_key unsigned char __TEE_ObjectHandle::public_key

9.1.1.6 type unsigned int __TEE_ObjectHandle::type

The documentation for this struct was generated from the following files:

- ta-ref/api/keystone/[tee_api_tee_types.h](#)
- ta-ref/api/sgx/[tee_api_tee_types.h](#)

9.2 __TEE_OperationHandle Struct Reference

```
#include <tee_api_tee_types.h>
```

Public Attributes

- int [mode](#)
- int [flags](#)
- int [alg](#)
- sha3_ctx_t [ctx](#)
- struct AES_ctx [aectx](#)
- int [aegcm_state](#)
- unsigned char [aeiv](#) [TEE_OBJECT_NONCE_SIZE]
- unsigned char [aekey](#) [32]
- unsigned char [pubkey](#) [TEE_OBJECT_KEY_SIZE]
- unsigned char [prikey](#) [TEE_OBJECT_SKEY_SIZE]

9.2.1 Member Data Documentation

9.2.1.1 aectx struct AES_ctx __TEE_OperationHandle::aectx

9.2.1.2 aegcm_state int __TEE_OperationHandle::aegcm_state

9.2.1.3 aeiv unsigned char __TEE_OperationHandle::aeiv

9.2.1.4 aekey unsigned char __TEE_OperationHandle::aekey

9.2.1.5 alg int __TEE_OperationHandle::alg

9.2.1.6 ctx sha3_ctx_t __TEE_OperationHandle::ctx

9.2.1.7 flags int __TEE_OperationHandle::flags

9.2.1.8 mode `int __TEE_OperationHandle::mode`

9.2.1.9 prikey `unsigned char __TEE_OperationHandle::prikey`

9.2.1.10 pubkey `unsigned char __TEE_OperationHandle::pubkey`

The documentation for this struct was generated from the following files:

- [ta-ref/api/keystone/tee_api_tee_types.h](#)
- [ta-ref/api/sgx/tee_api_tee_types.h](#)

9.3 addrinfo Struct Reference

```
#include <tee_api_types.h>
```

Collaboration diagram for addrinfo:



Public Attributes

- `int` [ai_flags](#)
- `int` [ai_family](#)
- `int` [ai_socktype](#)
- `int` [ai_protocol](#)
- `socklen_t` [ai_addrlen](#)
- `struct sockaddr *` [ai_addr](#)
- `char *` [ai_canonname](#)
- `struct addrinfo *` [ai_next](#)

9.3.1 Member Data Documentation

9.3.1.1 ai_addr `struct sockaddr* addrinfo::ai_addr`

9.3.1.2 ai_addrlen `socklen_t` `addrinfo::ai_addrlen`

9.3.1.3 ai_canonname `char*` `addrinfo::ai_canonname`

9.3.1.4 ai_family `int` `addrinfo::ai_family`

9.3.1.5 ai_flags `int` `addrinfo::ai_flags`

9.3.1.6 ai_next `struct addrinfo*` `addrinfo::ai_next`

9.3.1.7 ai_protocol `int` `addrinfo::ai_protocol`

9.3.1.8 ai_socktype `int` `addrinfo::ai_socktype`

The documentation for this struct was generated from the following file:

- `ta-ref/api/include/tee_api_types.h`

9.4 `enclave_report` Struct Reference

```
#include <report.h>
```

Public Attributes

- `uint8_t` `hash` [`MDSIZE`]
- `uint64_t` `data_len`
- `uint8_t` `data` [`ATTEST_DATA_MAXLEN`]
- `uint8_t` `signature` [`SIGNATURE_SIZE`]

9.4.1 Member Data Documentation

9.4.1.1 data `uint8_t enclave_report::data[ATTEST_DATA_MAXLEN]`

9.4.1.2 data.len `uint64_t enclave_report::data.len`

9.4.1.3 hash `uint8_t enclave_report::hash[MDSIZE]`

9.4.1.4 signature `uint8_t enclave_report::signature[SIGNATURE_SIZE]`

The documentation for this struct was generated from the following file:

- [ta-ref/api/include/report.h](#)

9.5 pollfd Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- `int fd`
- `short int events`
- `short int revents`

9.5.1 Member Data Documentation

9.5.1.1 events `short int pollfd::events`

9.5.1.2 fd `int pollfd::fd`

9.5.1.3 revents `short int pollfd::revents`

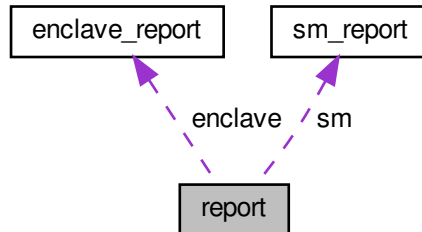
The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_api_types.h](#)

9.6 report Struct Reference

```
#include <report.h>
```

Collaboration diagram for report:



Public Attributes

- struct [enclave_report](#) `enclave`
- struct [sm_report](#) `sm`
- `uint8_t` [dev_public_key](#) [`PUBLIC_KEY_SIZE`]

9.6.1 Member Data Documentation

9.6.1.1 dev_public_key `uint8_t` `report::dev_public_key` [`PUBLIC_KEY_SIZE`]

9.6.1.2 enclave `struct` [enclave_report](#) `report::enclave`

9.6.1.3 sm `struct` [sm_report](#) `report::sm`

The documentation for this struct was generated from the following file:

- `ta-ref/api/include/report.h`

9.7 sm_report Struct Reference

```
#include <report.h>
```

Public Attributes

- uint8_t [hash](#) [[MDSIZE](#)]
- uint8_t [public_key](#) [[PUBLIC_KEY_SIZE](#)]
- uint8_t [signature](#) [[SIGNATURE_SIZE](#)]

9.7.1 Member Data Documentation

9.7.1.1 hash uint8_t sm_report::hash [[MDSIZE](#)]

9.7.1.2 public_key uint8_t sm_report::public_key [[PUBLIC_KEY_SIZE](#)]

9.7.1.3 signature uint8_t sm_report::signature [[SIGNATURE_SIZE](#)]

The documentation for this struct was generated from the following file:

- [ta-ref/api/include/report.h](#)

9.8 TEE_Attribute Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- uint32_t [attributeID](#)
- union {
 - struct {
 - void * [buffer](#)
 - uint32_t [length](#)
 - [ref](#)
 - struct {
 - uint32_t [a](#)
 - uint32_t [b](#)
 - [value](#)
- [content](#)

9.8.1 Member Data Documentation

9.8.1.1 a `uint32_t TEE_Attribute::a`

9.8.1.2 attributeID `uint32_t TEE_Attribute::attributeID`

9.8.1.3 b `uint32_t TEE_Attribute::b`

9.8.1.4 buffer `void* TEE_Attribute::buffer`

9.8.1.5 `union { ... } TEE_Attribute::content`

9.8.1.6 length `uint32_t TEE_Attribute::length`

9.8.1.7 `struct { ... } TEE_Attribute::ref`

9.8.1.8 `struct { ... } TEE_Attribute::value`

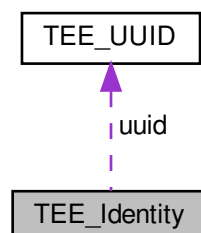
The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_api_types.h](#)

9.9 TEE_Identity Struct Reference

```
#include <tee_api_types.h>
```

Collaboration diagram for TEE_Identity:



Public Attributes

- uint32_t [login](#)
- [TEE_UUID](#) [uuid](#)

9.9.1 Member Data Documentation

9.9.1.1 login uint32_t TEE_Identity::login

9.9.1.2 uuid [TEE_UUID](#) TEE_Identity::uuid

The documentation for this struct was generated from the following file:

- ta-ref/api/include/[tee_api_types.h](#)

9.10 TEE_ObjectInfo Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- uint32_t [objectType](#)
- union {
 - uint32_t [keySize](#)
 - uint32_t [objectSize](#)
};
- union {
 - uint32_t [maxKeySize](#)
 - uint32_t [maxObjectSize](#)
};
- uint32_t [objectUsage](#)
- uint32_t [dataSize](#)
- uint32_t [dataPosition](#)
- uint32_t [handleFlags](#)

9.10.1 Member Data Documentation

9.10.1.1 __extension__ union { ... } TEE_ObjectInfo::@3

9.10.1.2 `__extension__ union { ... } TEE_ObjectInfo::@5`

9.10.1.3 dataPosition `uint32_t TEE_ObjectInfo::dataPosition`

9.10.1.4 dataSize `uint32_t TEE_ObjectInfo::dataSize`

9.10.1.5 handleFlags `uint32_t TEE_ObjectInfo::handleFlags`

9.10.1.6 keySize `uint32_t TEE_ObjectInfo::keySize`

9.10.1.7 maxKeySize `uint32_t TEE_ObjectInfo::maxKeySize`

9.10.1.8 maxObjectSize `uint32_t TEE_ObjectInfo::maxObjectSize`

9.10.1.9 objectSize `uint32_t TEE_ObjectInfo::objectSize`

9.10.1.10 objectType `uint32_t TEE_ObjectInfo::objectType`

9.10.1.11 objectUsage `uint32_t TEE_ObjectInfo::objectUsage`

The documentation for this struct was generated from the following file:

- `ta-ref/api/include/tee_api_types.h`

9.11 TEE_OperationInfo Struct Reference

```
#include <tee_api_types.h>
```


Public Attributes

- uint32_t [algorithm](#)
- uint32_t [operationClass](#)
- uint32_t [mode](#)
- uint32_t [digestLength](#)
- uint32_t [maxKeySize](#)
- uint32_t [keySize](#)
- uint32_t [requiredKeyUsage](#)
- uint32_t [handleState](#)

9.11.1 Member Data Documentation

9.11.1.1 algorithm uint32_t TEE_OperationInfo::algorithm

9.11.1.2 digestLength uint32_t TEE_OperationInfo::digestLength

9.11.1.3 handleState uint32_t TEE_OperationInfo::handleState

9.11.1.4 keySize uint32_t TEE_OperationInfo::keySize

9.11.1.5 maxKeySize uint32_t TEE_OperationInfo::maxKeySize

9.11.1.6 mode uint32_t TEE_OperationInfo::mode

9.11.1.7 operationClass uint32_t TEE_OperationInfo::operationClass

9.11.1.8 **requiredKeyUsage** `uint32_t TEE_OperationInfo::requiredKeyUsage`

The documentation for this struct was generated from the following file:

- ta-ref/api/include/[tee_api_types.h](#)

9.12 **TEE_OperationInfoKey Struct Reference**

```
#include <tee_api_types.h>
```

Public Attributes

- `uint32_t` [keySize](#)
- `uint32_t` [requiredKeyUsage](#)

9.12.1 Member Data Documentation

9.12.1.1 **keySize** `uint32_t TEE_OperationInfoKey::keySize`

9.12.1.2 **requiredKeyUsage** `uint32_t TEE_OperationInfoKey::requiredKeyUsage`

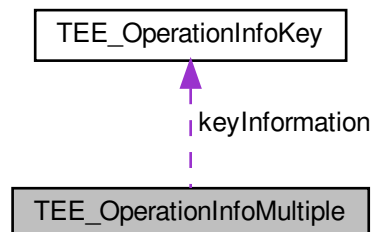
The documentation for this struct was generated from the following file:

- ta-ref/api/include/[tee_api_types.h](#)

9.13 **TEE_OperationInfoMultiple Struct Reference**

```
#include <tee_api_types.h>
```

Collaboration diagram for TEE_OperationInfoMultiple:



Public Attributes

- uint32_t [algorithm](#)
- uint32_t [operationClass](#)
- uint32_t [mode](#)
- uint32_t [digestLength](#)
- uint32_t [maxKeySize](#)
- uint32_t [handleState](#)
- uint32_t [operationState](#)
- uint32_t [numberOfKeys](#)
- [TEE_OperationInfoKey](#) [keyInformation](#) []

9.13.1 Member Data Documentation

9.13.1.1 algorithm uint32_t TEE_OperationInfoMultiple::algorithm

9.13.1.2 digestLength uint32_t TEE_OperationInfoMultiple::digestLength

9.13.1.3 handleState uint32_t TEE_OperationInfoMultiple::handleState

9.13.1.4 keyInformation [TEE_OperationInfoKey](#) TEE_OperationInfoMultiple::keyInformation[]

9.13.1.5 maxKeySize uint32_t TEE_OperationInfoMultiple::maxKeySize

9.13.1.6 mode uint32_t TEE_OperationInfoMultiple::mode

9.13.1.7 numberOfKeys uint32_t TEE_OperationInfoMultiple::numberOfKeys

9.13.1.8 operationClass uint32_t TEE_OperationInfoMultiple::operationClass

9.13.1.9 **operationState** `uint32_t TEE_OperationInfoMultiple::operationState`

The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_api_types.h](#)

9.14 TEE_Param Union Reference

```
#include <tee_api_types.h>
```

Public Attributes

- struct {
 void * [buffer](#)
 uint32_t [size](#)
} [memref](#)
- struct {
 uint32_t [a](#)
 uint32_t [b](#)
} [value](#)

9.14.1 Member Data Documentation

9.14.1.1 **a** `uint32_t TEE_Param::a`

9.14.1.2 **b** `uint32_t TEE_Param::b`

9.14.1.3 **buffer** `void* TEE_Param::buffer`

9.14.1.4 `struct { ... } TEE_Param::memref`

9.14.1.5 **size** `uint32_t TEE_Param::size`

9.14.1.6 `struct { ... } TEE_Param::value`

The documentation for this union was generated from the following file:

- [ta-ref/api/include/tee_api_types.h](#)

9.15 TEE_SEAID Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- `uint8_t *` [buffer](#)
- `size_t` [bufferLen](#)

9.15.1 Member Data Documentation

9.15.1.1 **buffer** `uint8_t* TEE_SEAID::buffer`

9.15.1.2 **bufferLen** `size_t TEE_SEAID::bufferLen`

The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_api_types.h](#)

9.16 TEE_SEReaderProperties Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- `bool` [sePresent](#)
- `bool` [teeOnly](#)
- `bool` [selectResponseEnable](#)

9.16.1 Member Data Documentation

9.16.1.1 selectResponseEnable `bool TEE_SEReadProperties::selectResponseEnable`

9.16.1.2 sePresent `bool TEE_SEReadProperties::sePresent`

9.16.1.3 teeOnly `bool TEE_SEReadProperties::teeOnly`

The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_api_types.h](#)

9.17 TEE_Time Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- `uint32_t` [seconds](#)
- `uint32_t` [millis](#)

9.17.1 Member Data Documentation

9.17.1.1 millis `uint32_t TEE_Time::millis`

9.17.1.2 seconds `uint32_t TEE_Time::seconds`

The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_api_types.h](#)

9.18 TEE_UUID Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- uint32_t [timeLow](#)
- uint16_t [timeMid](#)
- uint16_t [timeHiAndVersion](#)
- uint8_t [clockSeqAndNode](#) [8]

9.18.1 Member Data Documentation

9.18.1.1 clockSeqAndNode uint8_t TEE_UUID::clockSeqAndNode[8]

9.18.1.2 timeHiAndVersion uint16_t TEE_UUID::timeHiAndVersion

9.18.1.3 timeLow uint32_t TEE_UUID::timeLow

9.18.1.4 timeMid uint16_t TEE_UUID::timeMid

The documentation for this struct was generated from the following file:

- ta-ref/api/include/[tee_api_types.h](#)

9.19 TEEC_Context Struct Reference

```
#include <tee_client_api.h>
```

Public Attributes

- int [fd](#)
- bool [reg_mem](#)

9.19.1 Detailed Description

struct [TEEC_Context](#) - Represents a connection between a client application and a TEE.

9.19.2 Member Data Documentation

9.19.2.1 fd `int TEEC_Context::fd`

9.19.2.2 reg_mem `bool TEEC_Context::reg_mem`

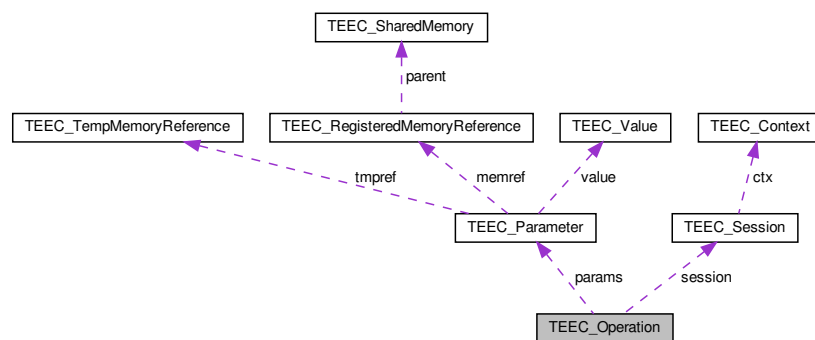
The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_client_api.h](#)

9.20 TEEC_Operation Struct Reference

```
#include <tee_client_api.h>
```

Collaboration diagram for TEEC_Operation:



Public Attributes

- `uint32_t started`
- `uint32_t paramTypes`
- `TEEC_Parameter params [TEEC_CONFIG_PAYLOAD_REF_COUNT]`
- `TEEC_Session * session`

9.20.1 Detailed Description

struct [TEEC_Operation](#) - Holds information and memory references used in [TEEC_InvokeCommand\(\)](#).

Parameters

<i>started</i>	Client must initialize to zero if it needs to cancel an operation about to be performed.
<i>paramTypes</i>	Type of data passed. Use <code>TEEC_PARAMS_TYPE</code> macro to create the correct flags. 0 means <code>TEEC_NONE</code> is passed for all params.
<i>params</i>	Array of parameters of type TEEC_Parameter .
<i>session</i>	Internal pointer to the last session used by <code>TEEC_InvokeCommand</code> with this operation.

9.20.2 Member Data Documentation

9.20.2.1 params `TEEC_Parameter` `TEEC_Operation::params[TEEC_CONFIG_PAYLOAD_REF_COUNT]`

9.20.2.2 paramTypes `uint32_t` `TEEC_Operation::paramTypes`

9.20.2.3 session `TEEC_Session*` `TEEC_Operation::session`

9.20.2.4 started `uint32_t` `TEEC_Operation::started`

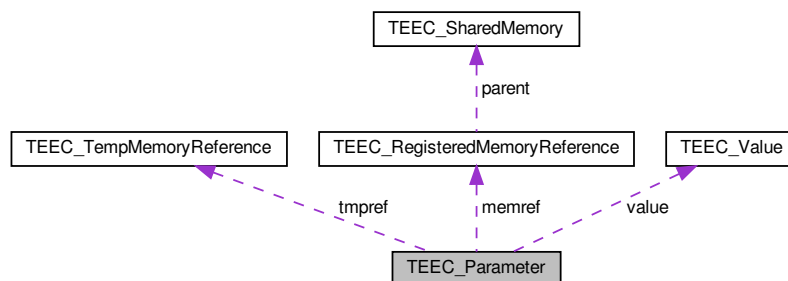
The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_client_api.h](#)

9.21 TEEC_Parameter Union Reference

```
#include <tee_client_api.h>
```

Collaboration diagram for TEEC_Parameter:



Public Attributes

- `TEEC_TempMemoryReference` `tmpref`
- `TEEC_RegisteredMemoryReference` `memref`
- `TEEC_Value` `value`

9.21.1 Detailed Description

union `TEEC_Parameter` - Memory container to be used when passing data between client application and trusted code.

Either the client uses a shared memory reference, parts of it or a small raw data container.

Parameters

<i>tmpref</i>	A temporary memory reference only valid for the duration of the operation.
<i>memref</i>	The entire shared memory or parts of it.
<i>value</i>	The small raw data container to use

9.21.2 Member Data Documentation

9.21.2.1 memref [TEEC.RegisteredMemoryReference](#) `TEEC.Parameter::memref`

9.21.2.2 tmpref [TEEC.TempMemoryReference](#) `TEEC.Parameter::tmpref`

9.21.2.3 value [TEEC.Value](#) `TEEC.Parameter::value`

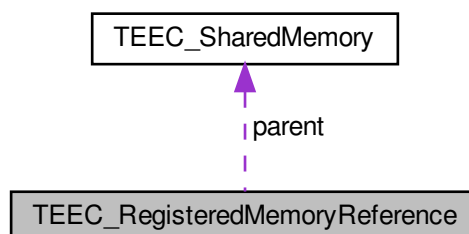
The documentation for this union was generated from the following file:

- [ta-ref/api/include/tee_client_api.h](#)

9.22 TEEC_RegisterMemoryReference Struct Reference

```
#include <tee_client_api.h>
```

Collaboration diagram for TEEC_RegisterMemoryReference:



Public Attributes

- [TEEC_SharedMemory](#) * [parent](#)
- [size_t](#) [size](#)
- [size_t](#) [offset](#)

9.22.1 Detailed Description

struct [TEEC_RegisteredMemoryReference](#) - use a pre-registered or pre-allocated shared memory block of memory to transfer data between a client application and trusted code.

Parameters

<i>parent</i>	Points to a shared memory structure. The memory reference may utilize the whole shared memory or only a part of it. Must not be NULL
<i>size</i>	The size, in bytes, of the memory buffer.
<i>offset</i>	The offset, in bytes, of the referenced memory region from the start of the shared memory block.

9.22.2 Member Data Documentation

9.22.2.1 offset `size_t TEEC_RegisteredMemoryReference::offset`

9.22.2.2 parent `TEEC_SharedMemory* TEEC_RegisteredMemoryReference::parent`

9.22.2.3 size `size_t TEEC_RegisteredMemoryReference::size`

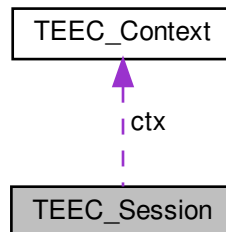
The documentation for this struct was generated from the following file:

- `ta-ref/api/include/tee_client_api.h`

9.23 TEEC_Session Struct Reference

```
#include <tee_client_api.h>
```

Collaboration diagram for TEEC_Session:



Public Attributes

- `TEEC_Context * ctx`
- `uint32_t session_id`

9.23.1 Detailed Description

struct `TEEC_Session` - Represents a connection between a client application and a trusted application.

9.23.2 Member Data Documentation

9.23.2.1 `ctx` `TEEC_Context*` `TEEC_Session::ctx`

9.23.2.2 `session_id` `uint32_t` `TEEC_Session::session_id`

The documentation for this struct was generated from the following file:

- `ta-ref/api/include/tee_client_api.h`

9.24 TEEC_SharedMemory Struct Reference

```
#include <tee_client_api.h>
```

Public Attributes

- void * [buffer](#)
- size_t [size](#)
- uint32_t [flags](#)
- int [id](#)
- size_t [allocated_size](#)
- void * [shadow_buffer](#)
- int [registered_fd](#)
- bool [buffer_allocated](#)

9.24.1 Detailed Description

struct [TEEC_SharedMemory](#) - Memory to transfer data between a client application and trusted code.

Parameters

<i>buffer</i>	The memory buffer which is to be, or has been, shared with the TEE.
<i>size</i>	The size, in bytes, of the memory buffer.
<i>flags</i>	Bit-vector which holds properties of buffer. The bit-vector can contain either or both of the TEEC_MEM_INPUT and TEEC_MEM_OUTPUT flags.

A shared memory block is a region of memory allocated in the context of the client application memory space that can be used to transfer data between that client application and a trusted application. The user of this struct is responsible to populate the buffer pointer.

9.24.2 Member Data Documentation

9.24.2.1 [allocated_size](#) `size_t TEEC_SharedMemory::allocated_size`

9.24.2.2 [buffer](#) `void* TEEC_SharedMemory::buffer`

9.24.2.3 [buffer_allocated](#) `bool TEEC_SharedMemory::buffer_allocated`

9.24.2.4 [flags](#) `uint32_t TEEC_SharedMemory::flags`

9.24.2.5 id `int TEEC_SharedMemory::id`

9.24.2.6 registered_fd `int TEEC_SharedMemory::registered_fd`

9.24.2.7 shadow_buffer `void* TEEC_SharedMemory::shadow_buffer`

9.24.2.8 size `size_t TEEC_SharedMemory::size`

The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_client_api.h](#)

9.25 TEEC_TempMemoryReference Struct Reference

```
#include <tee_client_api.h>
```

Public Attributes

- `void *` [buffer](#)
- `size_t` [size](#)

9.25.1 Detailed Description

struct [TEEC_TempMemoryReference](#) - Temporary memory to transfer data between a client application and trusted code, only used for the duration of the operation.

Parameters

<i>buffer</i>	The memory buffer which is to be, or has been shared with the TEE.
<i>size</i>	The size, in bytes, of the memory buffer.

A memory buffer that is registered temporarily for the duration of the operation to be called.

9.25.2 Member Data Documentation

9.25.2.1 buffer void* TEEC_TempMemoryReference::buffer

9.25.2.2 size size_t TEEC_TempMemoryReference::size

The documentation for this struct was generated from the following file:

- ta-ref/api/include/[tee_client_api.h](#)

9.26 TEEC_UUID Struct Reference

```
#include <tee_client_api.h>
```

Public Attributes

- uint32_t [timeLow](#)
- uint16_t [timeMid](#)
- uint16_t [timeHiAndVersion](#)
- uint8_t [clockSeqAndNode](#) [8]

9.26.1 Detailed Description

This type contains a Universally Unique Resource Identifier (UUID) type as defined in RFC4122. These UUID values are used to identify Trusted Applications.

9.26.2 Member Data Documentation

9.26.2.1 clockSeqAndNode uint8_t TEEC_UUID::clockSeqAndNode[8]

9.26.2.2 timeHiAndVersion uint16_t TEEC_UUID::timeHiAndVersion

9.26.2.3 timeLow uint32_t TEEC_UUID::timeLow

9.26.2.4 timeMid `uint16_t TEEC_UUID::timeMid`

The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_client_api.h](#)

9.27 TEEC_Value Struct Reference

```
#include <tee_client_api.h>
```

Public Attributes

- `uint32_t a`
- `uint32_t b`

9.27.1 Detailed Description

struct [TEEC_Value](#) - Small raw data container

Instead of allocating a shared memory buffer this structure can be used to pass small raw data between a client application and trusted code.

Parameters

<i>a</i>	The first integer value.
<i>b</i>	The second second value.

9.27.2 Member Data Documentation

9.27.2.1 **a** `uint32_t TEEC_Value::a`

9.27.2.2 **b** `uint32_t TEEC_Value::b`

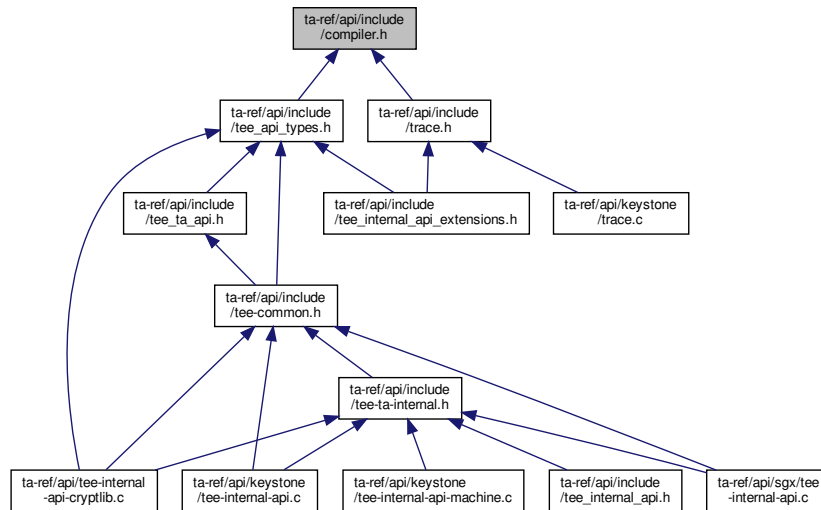
The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_client_api.h](#)

10 File Documentation

10.1 ta-ref/api/include/compiler.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- `#define __deprecated __attribute__((deprecated))`
- `#define __packed __attribute__((packed))`
- `#define __weak __attribute__((weak))`
- `#define __noreturn __attribute__((noreturn))`
- `#define __pure __attribute__((pure))`
- `#define __aligned(x) __attribute__((aligned(x)))`
- `#define __printf(a, b) __attribute__((format(printf, a, b)))`
- `#define __noinline __attribute__((noinline))`
- `#define __attr_const __attribute__((__const__))`
- `#define __unused __attribute__((unused))`
- `#define __maybe_unused __attribute__((unused))`
- `#define __used __attribute__((__used__))`
- `#define __must_check __attribute__((warn_unused_result))`
- `#define __cold __attribute__((__cold__))`
- `#define __section(x) __attribute__((section(x)))`
- `#define __data __section(".data")`
- `#define __bss __section(".bss")`
- `#define __rodata __section(".rodata")`
- `#define __rodata_unpaged __section(".rodata.unpaged")`
- `#define __early_ta __section(".rodata.early_ta")`
- `#define __noprof __attribute__((no_instrument_function))`
- `#define __compiler_bswap64(x) __builtin_bswap64(x)`
- `#define __compiler_bswap32(x) __builtin_bswap32(x)`
- `#define __compiler_bswap16(x) __builtin_bswap16(x)`

- `#define __GCC_VERSION`
- `#define __INTOF_HALF_MAX_SIGNED(type) ((type)1 << (sizeof(type)*8-2))`
- `#define __INTOF_MAX_SIGNED(type)`
- `#define __INTOF_MIN_SIGNED(type) (-1 - __INTOF_MAX_SIGNED(type))`
- `#define __INTOF_MIN(type) ((type)-1 < 1? __INTOF_MIN_SIGNED(type):(type)0)`
- `#define __INTOF_MAX(type) ((type)~__INTOF_MIN(type))`
- `#define __INTOF_ASSIGN(dest, src)`
- `#define __INTOF_ADD(c, a, b)`
- `#define __INTOF_SUB(c, a, b)`
- `#define __intof_mul_negate ((__intof_oa < 1) != (__intof_ob < 1))`
- `#define __intof_mul_hshift (sizeof(uintmax_t) * 8 / 2)`
- `#define __intof_mul_hmask (UINTMAX_MAX >> __intof_mul_hshift)`
- `#define __intof_mul_a0 ((uintmax_t)(__intof_a) >> __intof_mul_hshift)`
- `#define __intof_mul_b0 ((uintmax_t)(__intof_b) >> __intof_mul_hshift)`
- `#define __intof_mul_a1 ((uintmax_t)(__intof_a) & __intof_mul_hmask)`
- `#define __intof_mul_b1 ((uintmax_t)(__intof_b) & __intof_mul_hmask)`
- `#define __intof_mul_t`
- `#define __INTOF_MUL(c, a, b)`
- `#define __compiler_add_overflow(a, b, res) __INTOF_ADD(*(res), (a), (b))`
- `#define __compiler_sub_overflow(a, b, res) __INTOF_SUB(*(res), (a), (b))`
- `#define __compiler_mul_overflow(a, b, res) __INTOF_MUL(*(res), (a), (b))`
- `#define __compiler_compare_and_swap(p, oval, nval)`
- `#define __compiler_atomic_load(p) __atomic_load_n((p), __ATOMIC_RELAXED)`
- `#define __compiler_atomic_store(p, val) __atomic_store_n((p), (val), __ATOMIC_RELAXED)`

10.1.1 Macro Definition Documentation

10.1.1.1 `__aligned` `#define __aligned(
x) __attribute__((aligned(x)))`

10.1.1.2 `__attr_const` `#define __attr_const __attribute__((__const__))`

10.1.1.3 `__bss` `#define __bss __section(".bss")`

10.1.1.4 `__cold` `#define __cold __attribute__((__cold__))`

10.1.1.5 __compiler_add_overflow #define __compiler_add_overflow(
a,
b,
res) __INTOF_ADD(*(res), (a), (b))

10.1.1.6 __compiler_atomic_load #define __compiler_atomic_load(
p) __atomic_load_n((p), __ATOMIC_RELAXED)

10.1.1.7 __compiler_atomic_store #define __compiler_atomic_store(
p,
val) __atomic_store_n((p), (val), __ATOMIC_RELAXED)

10.1.1.8 __compiler_bswap16 #define __compiler_bswap16(
x) __builtin_bswap16((x))

10.1.1.9 __compiler_bswap32 #define __compiler_bswap32(
x) __builtin_bswap32((x))

10.1.1.10 __compiler_bswap64 #define __compiler_bswap64(
x) __builtin_bswap64((x))

10.1.1.11 __compiler_compare_and_swap #define __compiler_compare_and_swap(
p,
oval,
nval)

Value:

__atomic_compare_exchange_n((p), (oval), (nval), true, \
__ATOMIC_ACQUIRE, __ATOMIC_RELAXED) \

__HAVE_BUILTIN_OVERFLOW

10.1.1.12 __compiler_mul_overflow #define __compiler_mul_overflow(
a,
b,
res) __INTOF_MUL(*(res), (a), (b))

10.1.1.13 `__compiler_sub_overflow` `#define __compiler_sub_overflow(
a,
b,
res) __INTOF_SUB(*(res), (a), (b))`

10.1.1.14 `__data` `#define __data __section(".data")`

10.1.1.15 `__deprecated` `#define __deprecated __attribute__((deprecated))`

10.1.1.16 `__early_ta` `#define __early_ta __section(".rodata.early-ta")`

10.1.1.17 `__GCC_VERSION` `#define __GCC_VERSION`

Value:

```
(__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + \
__GNUC_PATCHLEVEL__)
```

10.1.1.18 `__INTOF_ADD` `#define __INTOF_ADD(
c,
a,
b)`

Value:

```
(__extension__({ \
  typeof(a) __intofa_a = (a); \
  typeof(b) __intofa_b = (b); \
  \
  __intofa_b < 1 ? \
    ((__INTOF_MIN(typeof(c)) - __intofa_b <= __intofa_a) ? \
      __INTOF_ASSIGN((c), __intofa_a + __intofa_b) : 1) : \
    ((__INTOF_MAX(typeof(c)) - __intofa_b >= __intofa_a) ? \
      __INTOF_ASSIGN((c), __intofa_a + __intofa_b) : 1); \
}))
```

10.1.1.19 `__INTOF_ASSIGN` `#define __INTOF_ASSIGN(
dest,
src)`

Value:

```
(__extension__({ \
  typeof(src) __intof_x = (src); \
  typeof(dest) __intof_y = __intof_x; \
  (((uintmax_t)__intof_x == (uintmax_t)__intof_y) && \
    ((__intof_x < 1) == (__intof_y < 1))) ? \
    (void)((dest) = __intof_y , 0 : 1); \
}))
```

10.1.1.20 `__INTOF_HALF_MAX_SIGNED` `#define __INTOF_HALF_MAX_SIGNED(`
`type) ((type)1 << (sizeof(type)*8-2))`

`__HAVE_BUILTIN_OVERFLOW`

10.1.1.21 `__INTOF_MAX` `#define __INTOF_MAX(`
`type) ((type)~__INTOF_MIN(type))`

10.1.1.22 `__INTOF_MAX_SIGNED` `#define __INTOF_MAX_SIGNED(`
`type)`

Value:

```
(__INTOF_HALF_MAX_SIGNED(type) - 1 + \
__INTOF_HALF_MAX_SIGNED(type))
```

10.1.1.23 `__INTOF_MIN` `#define __INTOF_MIN(`
`type) ((type)-1 < 1?__INTOF_MIN_SIGNED(type):(type)0)`

10.1.1.24 `__INTOF_MIN_SIGNED` `#define __INTOF_MIN_SIGNED(`
`type) (-1 - __INTOF_MAX_SIGNED(type))`

10.1.1.25 `__INTOF_MUL` `#define __INTOF_MUL(`
`c,`
`a,`
`b)`

Value:

```
(__extension__({ \
    typeof(a) __intof_oa = (a); \
    typeof(a) __intof_a = __intof_oa < 1 ? -__intof_oa : __intof_oa; \
    typeof(b) __intof_ob = (b); \
    typeof(b) __intof_b = __intof_ob < 1 ? -__intof_ob : __intof_ob; \
    typeof(c) __intof_c; \
    \
    __intof_oa == 0 || __intof_ob == 0 || \
    __intof_oa == 1 || __intof_ob == 1 ? \
        __INTOF_ASSIGN((c), __intof_oa * __intof_ob) : \
    (__intof_mul_a0 && __intof_mul_b0) || \
    __intof_mul_t > __intof_mul_hmask ? 1 : \
    __INTOF_ADD((__intof_c), __intof_mul_t << __intof_mul_hshift, \
        __intof_mul_a1 * __intof_mul_b1) ? 1 : \
    __intof_mul_negate ? __INTOF_ASSIGN((c), -__intof_c) : \
        __INTOF_ASSIGN((c), __intof_c); \
}))
```

10.1.1.26 `__intof_mul_a0` `#define __intof_mul_a0 ((uintmax_t)(__intof_a) >> __intof_mul_hshift)`

10.1.1.27 `__intof_mul_a1` `#define __intof_mul_a1 ((uintmax_t)(__intof_a) & __intof_mul_hmask)`

10.1.1.28 `__intof_mul_b0` `#define __intof_mul_b0 ((uintmax_t)(__intof_b) >> __intof_mul_hshift)`

10.1.1.29 `__intof_mul_b1` `#define __intof_mul_b1 ((uintmax_t)(__intof_b) & __intof_mul_hmask)`

10.1.1.30 `__intof_mul_hmask` `#define __intof_mul_hmask (UINTMAX_MAX >> __intof_mul_hshift)`

10.1.1.31 `__intof_mul_hshift` `#define __intof_mul_hshift (sizeof(uintmax_t) * 8 / 2)`

10.1.1.32 `__intof_mul_negate` `#define __intof_mul_negate ((__intof_oa < 1) != (__intof_ob < 1))`

10.1.1.33 `__intof_mul_t` `#define __intof_mul_t`

Value:

```
(__intof_mul_a1 * __intof_mul_b0 + \
__intof_mul_a0 * __intof_mul_b1)
```

10.1.1.34 `__INTOF_SUB` `#define __INTOF_SUB(`

```
    c,
    a,
    b )
```

Value:

```
(__extension__({ \
    typeof(a) __intofs_a = a; \
    typeof(b) __intofs_b = b; \
    \
    __intofs_b < 1 ? \
        ((__INTOF_MAX(typeof(c)) + __intofs_b >= __intofs_a) ? \
            __INTOF_ASSIGN((c), __intofs_a - __intofs_b) : 1) : \
        ((__INTOF_MIN(typeof(c)) + __intofs_b <= __intofs_a) ? \
            __INTOF_ASSIGN((c), __intofs_a - __intofs_b) : 1); \
}))
```

10.1.1.35 `__maybe_unused` `#define __maybe_unused __attribute__((unused))`

10.1.1.36 `__must_check` `#define __must_check __attribute__((warn_unused_result))`

10.1.1.37 `__noinline` `#define __noinline __attribute__((noinline))`

10.1.1.38 `__noprof` `#define __noprof __attribute__((no_instrument_function))`

10.1.1.39 `__noreturn` `#define __noreturn __attribute__((noreturn))`

10.1.1.40 `__packed` `#define __packed __attribute__((packed))`

10.1.1.41 `__printf` `#define __printf(
 a,
 b) __attribute__((format(printf, a, b)))`

10.1.1.42 `__pure` `#define __pure __attribute__((pure))`

10.1.1.43 `__rodata` `#define __rodata __section(".rodata")`

10.1.1.44 `__rodata_unpaged` `#define __rodata_unpaged __section(".rodata.__unpaged")`

10.1.1.45 `__section` `#define __section(
 x) __attribute__((section(x)))`

10.1.1.46 `__unused` `#define __unused __attribute__((unused))`

10.1.1.47 `__used` `#define __used __attribute__((__used__))`

10.1.1.48 `__weak` `#define __weak __attribute__((weak))`

10.2 compiler.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27
28 #ifndef COMPILER_H
29 #define COMPILER_H
30
31 /*
32  * Macros that should be used instead of using __attribute__ directly to
33  * ease portability and make the code easier to read.
34  */
35
36 #define __deprecated __attribute__((deprecated))
37 #define __packed __attribute__((packed))
38 #define __weak __attribute__((weak))
39 #define __noreturn __attribute__((noreturn))
40 #define __pure __attribute__((pure))
41 #define __aligned(x) __attribute__((aligned(x)))
42 #define __printf(a, b) __attribute__((format(printf, a, b)))
43 #define __noinline __attribute__((noinline))
44 #define __attr_const __attribute__((__const__))
45 #define __unused __attribute__((unused))
46 #define __maybe_unused __attribute__((unused))
47 #define __used __attribute__((__used__))
48 #define __must_check __attribute__((warn_unused_result))
49 #define __cold __attribute__((__cold__))
50 #define __section(x) __attribute__((section(x)))
51 #define __data __section(".data")
52 #define __bss __section(".bss")
53 #define __rodata __section(".rodata")
54 #define __rodata_unpaged __section(".rodata.unpaged")
55 #define __early_ta __section(".rodata.early.ta")
56 #define __noprof __attribute__((no_instrument_function))
57
58 #define __compiler_bswap64(x) __builtin_bswap64((x))
59 #define __compiler_bswap32(x) __builtin_bswap32((x))
60 #define __compiler_bswap16(x) __builtin_bswap16((x))
61
62 #define __GCC_VERSION (__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + \

```



```

63         __GNUC_PATCHLEVEL__)
64
65 #if __GCC_VERSION >= 50100 && !defined(__CHECKER__)
66 #define __HAVE_BUILTIN_OVERFLOW 1
67 #endif
68
69 #ifdef __HAVE_BUILTIN_OVERFLOW
70 #define __compiler_add_overflow(a, b, res) \
71     __builtin_add_overflow((a), (b), (res))
72
73 #define __compiler_sub_overflow(a, b, res) \
74     __builtin_sub_overflow((a), (b), (res))
75
76 #define __compiler_mul_overflow(a, b, res) \
77     __builtin_mul_overflow((a), (b), (res))
78 #else
79 /*
80  * Copied/inspired from https://www.fefe.de/intof.html
81  */
82 #define __INTOF_HALF_MAX_SIGNED(type) ((type)1 << (sizeof(type)*8-2))
83 #define __INTOF_MAX_SIGNED(type) (__INTOF_HALF_MAX_SIGNED(type) - 1 + \
84     __INTOF_HALF_MAX_SIGNED(type))
85 #define __INTOF_MIN_SIGNED(type) (-1 - __INTOF_MAX_SIGNED(type))
86
87 #define __INTOF_MIN(type) ((type)-1 < 1?__INTOF_MIN_SIGNED(type):(type)0)
88 #define __INTOF_MAX(type) ((type)~__INTOF_MIN(type))
89
90 #define __INTOF_ASSIGN(dest, src) (__extension__({ \
91     typeof(src) __intof_x = (src); \
92     typeof(dest) __intof_y = __intof_x; \
93     (((uintmax_t)__intof_x == (uintmax_t)__intof_y) && \
94     ((__intof_x < 1) == (__intof_y < 1))) ? \
95     (void)((dest) = __intof_y), 0 : 1); \
96 }))
97
98 #define __INTOF_ADD(c, a, b) (__extension__({ \
99     typeof(a) __intofa_a = (a); \
100     typeof(b) __intofa_b = (b); \
101     \
102     __intofa_b < 1 ? \
103     ((__INTOF_MIN(typeof(c)) - __intofa_b <= __intofa_a) ? \
104     __INTOF_ASSIGN((c), __intofa_a + __intofa_b) : 1) : \
105     ((__INTOF_MAX(typeof(c)) - __intofa_b >= __intofa_a) ? \
106     __INTOF_ASSIGN((c), __intofa_a + __intofa_b) : 1); \
107 }))
108
109 #define __INTOF_SUB(c, a, b) (__extension__({ \
110     typeof(a) __intofs_a = a; \
111     typeof(b) __intofs_b = b; \
112     \
113     __intofs_b < 1 ? \
114     ((__INTOF_MAX(typeof(c)) + __intofs_b >= __intofs_a) ? \
115     __INTOF_ASSIGN((c), __intofs_a - __intofs_b) : 1) : \
116     ((__INTOF_MIN(typeof(c)) + __intofs_b <= __intofs_a) ? \
117     __INTOF_ASSIGN((c), __intofs_a - __intofs_b) : 1); \
118 }))
119
120 /*
121  * Dealing with detecting overflow in multiplication of integers.
122  *
123  * First step is to remove two corner cases with the minum signed integer
124  * which can't be represented as a positive integer + sign.
125  * Multiply with 0 or 1 can't overflow, no checking needed of the operation,
126  * only if it can be assigned to the result.
127  *
128  * After the corner cases are eliminated we convert the two factors to
129  * positive unsigned values, keeping track of the original in another
130  * variable which is used at the end to determine the sign of the product.
131  *
132  * The two terms (a and b) are divided into upper and lower half (x1 upper
133  * and x0 lower), so the product is:
134  * ((a1 << hshift) + a0) * ((b1 << hshift) + b0)
135  * which also is:
136  * ((a1 * b1) << (hshift * 2)) + (T1)
137  * ((a1 * b0 + a0 * b1) << hshift) + (T2)
138  * (a0 * b0) (T3)
139  *
140  * From this we can tell and (a1 * b1) has to be 0 or we'll overflow, that
141  * is, at least one of a1 or b1 has to be 0. Once this has been checked the
142  * addition: ((a1 * b0) << hshift) + ((a0 * b1) << hshift)
143  * isn't an addition as one of the terms will be 0.
144  *
145  * Since each factor in: (a0 * b0)
146  * only uses half the capacity of the underlying type it can't overflow
147  *
148  * The addition of T2 and T3 can overflow so we use __INTOF_ADD() to
149  * perform that addition. If the addition succeeds without overflow the

```

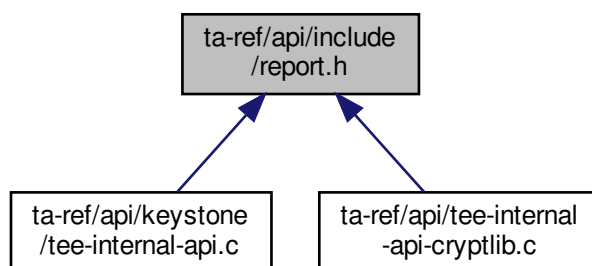
```

151 * result is assigned the required sign and checked for overflow again.
152 */
153
154 #define __intof_mul_negate ((__intof_oa < 1) != (__intof_ob < 1))
155 #define __intof_mul_hshift (sizeof(uintmax_t) * 8 / 2)
156 #define __intof_mul_hmask (UINTMAX_MAX >> __intof_mul_hshift)
157 #define __intof_mul_a0 ((uintmax_t)(__intof_a) >> __intof_mul_hshift)
158 #define __intof_mul_b0 ((uintmax_t)(__intof_b) >> __intof_mul_hshift)
159 #define __intof_mul_a1 ((uintmax_t)(__intof_a) & __intof_mul_hmask)
160 #define __intof_mul_b1 ((uintmax_t)(__intof_b) & __intof_mul_hmask)
161 #define __intof_mul_t (__intof_mul_a1 * __intof_mul_b0 + \
162     __intof_mul_a0 * __intof_mul_b1)
163
164 #define __INTOF_MUL(c, a, b) (__extension__({ \
165     typeof(a) __intof_oa = (a); \
166     typeof(a) __intof_a = __intof_oa < 1 ? __intof_oa : __intof_oa; \
167     typeof(b) __intof_ob = (b); \
168     typeof(b) __intof_b = __intof_ob < 1 ? __intof_ob : __intof_ob; \
169     typeof(c) __intof_c; \
170     \
171     __intof_oa == 0 || __intof_ob == 0 || \
172     __intof_oa == 1 || __intof_ob == 1 ? \
173     __INTOF_ASSIGN((c), __intof_oa * __intof_ob) : \
174     (__intof_mul_a0 && __intof_mul_b0) || \
175     __intof_mul_t > __intof_mul_hmask ? 1 : \
176     __INTOF_ADD((__intof_c), __intof_mul_t << __intof_mul_hshift, \
177         __intof_mul_a1 * __intof_mul_b1) ? 1 : \
178     __intof_mul_negate ? __INTOF_ASSIGN((c), -__intof_c) : \
179     __INTOF_ASSIGN((c), __intof_c); \
180 })))
181
182 #define __compiler_add_overflow(a, b, res) __INTOF_ADD(*res, (a), (b))
183 #define __compiler_sub_overflow(a, b, res) __INTOF_SUB(*res, (a), (b))
184 #define __compiler_mul_overflow(a, b, res) __INTOF_MUL(*res, (a), (b))
185
186 #endif
187 #define __compiler_compare_and_swap(p, oval, nval) \
188     __atomic_compare_exchange_n((p), (oval), (nval), true, \
189         __ATOMIC_ACQUIRE, __ATOMIC_RELAXED) \
190
191 #define __compiler_atomic_load(p) __atomic_load_n((p), __ATOMIC_RELAXED)
192 #define __compiler_atomic_store(p, val) \
193     __atomic_store_n((p), (val), __ATOMIC_RELAXED)
194
195 #endif /*COMPILER_H*/

```

10.3 ta-ref/api/include/report.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct [enclave_report](#)
- struct [sm_report](#)
- struct [report](#)

Macros

- #define MDSIZE 64
- #define SIGNATURE_SIZE 64
- #define PUBLIC_KEY_SIZE 32
- #define ATTEST_DATA_MAXLEN 1024

10.3.1 Macro Definition Documentation

10.3.1.1 ATTEST_DATA_MAXLEN #define ATTEST_DATA_MAXLEN 1024

10.3.1.2 MDSIZE #define MDSIZE 64

10.3.1.3 PUBLIC_KEY_SIZE #define PUBLIC_KEY_SIZE 32

10.3.1.4 SIGNATURE_SIZE #define SIGNATURE_SIZE 64

10.4 report.h

[Go to the documentation of this file.](#)

```

1 #ifndef _REPORT_H
2 #define _REPORT_H
3
4 #define MDSIZE 64
5 #define SIGNATURE_SIZE 64
6 #define PUBLIC_KEY_SIZE 32
7 #define ATTEST_DATA_MAXLEN 1024
8
9 /* attestation reports */
10 struct enclave_report
11 {
12     uint8_t hash[MDSIZE];
13     uint64_t data_len;
14     uint8_t data[ATTEST_DATA_MAXLEN];
15     uint8_t signature[SIGNATURE_SIZE];
16 };
17
18 struct sm_report
19 {
20     uint8_t hash[MDSIZE];
21     uint8_t public_key[PUBLIC_KEY_SIZE];
22     uint8_t signature[SIGNATURE_SIZE];
23 };
24
25 struct report
26 {
27     struct enclave_report enclave;
28     struct sm_report sm;
29     uint8_t dev_public_key[PUBLIC_KEY_SIZE];
30 };
31
32 #endif // _REPORT_H

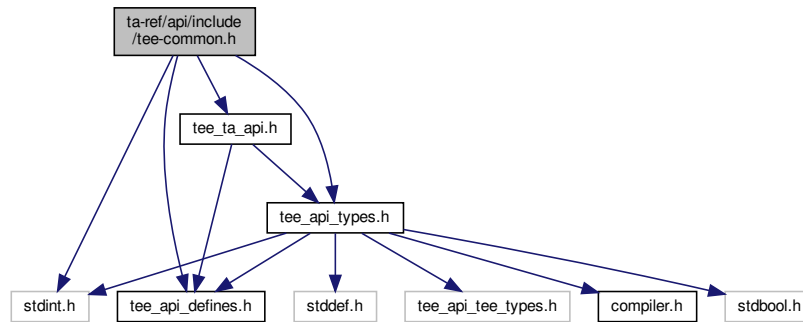
```

10.5 ta-ref/api/include/tee-common.h File Reference

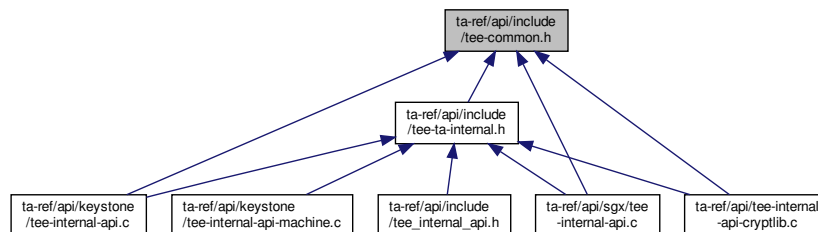
Common type and definitions of RISC-V TEE.

```
#include <stdint.h>
#include <tee_api_defines.h>
#include <tee_api_types.h>
#include <tee_ta_api.h>
```

Include dependency graph for tee-common.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define `pr_deb(...)` do { } while (0)

10.5.1 Detailed Description

Common type and definitions of RISC-V TEE.

draft RISC-V Internal TEE API

Author

Akira Tsukamoto, AIST

Date

2019/09/25

10.5.2 Macro Definition Documentation

10.5.2.1 pr_deb #define pr_deb(
...) do { } while (0)

10.6 tee-common.h

[Go to the documentation of this file.](#)

```

1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                               and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10 *
11 * 1. Redistributions of source code must retain the above copyright notice,
12 * this list of conditions and the following disclaimer.
13 *
14 * 2. Redistributions in binary form must reproduce the above copyright notice,
15 * this list of conditions and the following disclaimer in the documentation
16 * and/or other materials provided with the distribution.
17 *
18 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28 * POSSIBILITY OF SUCH DAMAGE.
29 */
30
31 #ifndef TEE_COMMON_H
32 #define TEE_COMMON_H
33
34 #include <stdint.h>
35
36 #ifdef __cplusplus
37 extern "C" {
38 #endif
39
40 #ifdef DEBUG
41 #define pr_deb(...) do { printf(__VA_ARGS__); } while (0)
42 #else
43 #define pr_deb(...) do { } while (0)
44 #endif /* DEBUG */
45
46 // #include <tee_api.h>
47 #include <tee_api_defines.h>
48 #include <tee_api_types.h>
49 #include <tee_ta_api.h>
50
51 // typedef uint32_t TEE_Result;
52
53 #ifdef __cplusplus
54 }
55 #endif
56 #endif /* TEE_COMMON_H */

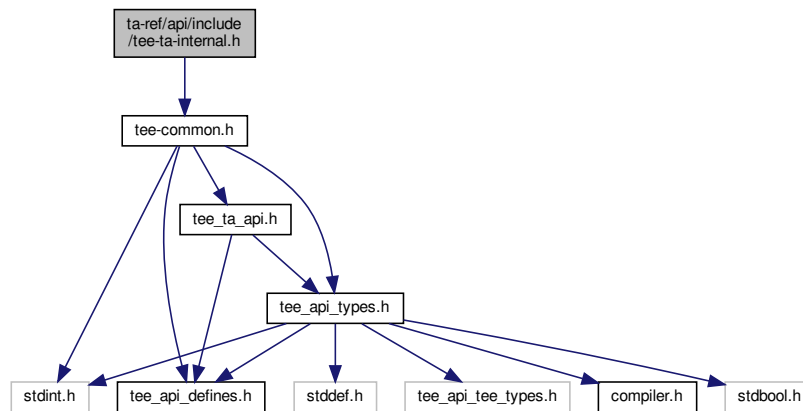
```

10.7 ta-ref/api/include/tee-ta-internal.h File Reference

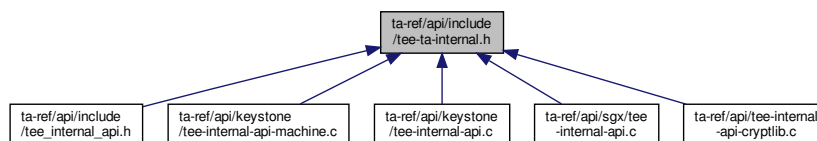
Candidate API list for Global Platform like RISC-V TEE.

```
#include "tee-common.h"
```

Include dependency graph for tee-ta-internal.h:



This graph shows which files directly or indirectly include this file:



Functions

- void `__attribute__((noreturn)) TEE_Panic(unsigned long code)`
Core Functions, Time Functions.
- void `TEE_GetREETime (TEE_Time *time)`
Core Functions, Time Functions.
- void `TEE_GetSystemTime (TEE_Time *time)`
Core Functions, Time Functions.
- `TEE_Result GetRelTimeStart (uint64_t start)`
Core Functions, Time Functions.
- `TEE_Result GetRelTimeEnd (uint64_t end)`
Core Functions, Time Functions.
- `TEE_Result TEE_CreatePersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle attributes, const void *initialData, uint32_t initialDataLen, TEE_ObjectHandle *object)`
Core Functions, Secure Storage Functions (data is isolated for each TA)
- `TEE_Result TEE_OpenPersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle *object)`
Core Functions, Secure Storage Functions (data is isolated for each TA)
- `TEE_Result TEE_GetObjectInfo1 (TEE_ObjectHandle object, TEE_ObjectInfo *objectInfo)`
Core Functions, Secure Storage Functions (data is isolated for each TA)

- **TEE_Result TEE_WriteObjectData** (**TEE_ObjectHandle** object, const void *buffer, uint32_t size)
Core Functions, Secure Storage Functions (data is isolated for each TA)
- **TEE_Result TEE_ReadObjectData** (**TEE_ObjectHandle** object, void *buffer, uint32_t size, uint32_t *count)
Core Functions, Secure Storage Functions (data is isolated for each TA)
- void **TEE_CloseObject** (**TEE_ObjectHandle** object)
Core Functions, Secure Storage Functions (data is isolated for each TA)
- void **TEE_GenerateRandom** (void *randomBuffer, uint32_t randomBufferLen)
Crypto, common.
- **TEE_Result TEE_AllocateOperation** (**TEE_OperationHandle** *operation, uint32_t algorithm, uint32_t mode, uint32_t maxKeySize)
Crypto, for all Crypto Functions.
- void **TEE_FreeOperation** (**TEE_OperationHandle** operation)
Crypto, for all Crypto Functions.
- void **TEE_DigestUpdate** (**TEE_OperationHandle** operation, const void *chunk, uint32_t chunkSize)
Crypto, Message Digest Functions.
- **TEE_Result TEE_DigestDoFinal** (**TEE_OperationHandle** operation, const void *chunk, uint32_t chunkLen, void *hash, uint32_t *hashLen)
- **TEE_Result TEE_SetOperationKey** (**TEE_OperationHandle** operation, **TEE_ObjectHandle** key)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- **TEE_Result TEE_AEInit** (**TEE_OperationHandle** operation, const void *nonce, uint32_t nonceLen, uint32_t tagLen, uint32_t AADLen, uint32_t payloadLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- **TEE_Result TEE_AEUpdate** (**TEE_OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- void **TEE_AEUpdateAAD** (**TEE_OperationHandle** operation, const void *AADdata, uint32_t AADdataLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- **TEE_Result TEE_AEEncryptFinal** (**TEE_OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen, void *tag, uint32_t *tagLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- **TEE_Result TEE_AEDecryptFinal** (**TEE_OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen, void *tag, uint32_t tagLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- void **TEE_CipherInit** (**TEE_OperationHandle** operation, const void *nonce, uint32_t nonceLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- **TEE_Result TEE_CipherUpdate** (**TEE_OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- **TEE_Result TEE_GenerateKey** (**TEE_ObjectHandle** object, uint32_t keySize, const **TEE_Attribute** *params, uint32_t paramCount)
Crypto, Asymmetric key Verification Functions.
- **TEE_Result TEE_AllocateTransientObject** (**TEE_ObjectType** objectType, uint32_t maxKeySize, **TEE_ObjectHandle** *object)
Crypto, Asymmetric key Verification Functions.
- void **TEE_InitRefAttribute** (**TEE_Attribute** *attr, uint32_t attributeID, const void *buffer, uint32_t length)
Crypto, Asymmetric key Verification Functions.
- void **TEE_InitValueAttribute** (**TEE_Attribute** *attr, uint32_t attributeID, uint32_t a, uint32_t b)
Crypto, Asymmetric key Verification Functions.
- void **TEE_FreeTransientObject** (**TEE_ObjectHandle** object)
Crypto, Asymmetric key Verification Functions.
- **TEE_Result TEE_AsymmetricSignDigest** (**TEE_OperationHandle** operation, const **TEE_Attribute** *params, uint32_t paramCount, const void *digest, uint32_t digestLen, void *signature, uint32_t *signatureLen)

Crypto, Asymmetric key Verification Functions.

- `TEE_Result TEE_AsymmetricVerifyDigest (TEE_OperationHandle operation, const TEE_Attribute *params, uint32_t paramCount, const void *digest, uint32_t digestLen, const void *signature, uint32_t signatureLen)`

Crypto, Asymmetric key Verification Functions.

10.7.1 Detailed Description

Candidate API list for Global Platform like RISC-V TEE.

draft RISC-V Internal TEE API

Author

Akira Tsukamoto, AIST

Date

2019/09/25

10.7.2 Function Documentation

10.7.2.1 `__attribute__((noret)) void __attribute__((noret))`

TEE.Panic() - Raises a panic in the Trusted Application instance.

When a Trusted Application calls the TEE.Panic function, the current instance shall be destroyed and all the resources opened by the instance shall be reclaimed. All sessions opened from the panicking instance on another TA shall be gracefully closed and all cryptographic objects and operations shall be closed properly.

Parameters

<i>code</i>	An informative panic code defined by the TA.
-------------	--

Returns

panic code will be returned.

TEE.Panic() - Raises a Panic in the Trusted Application instance

When a Trusted Application calls the TEE.Panic function, the current instance shall be destroyed and all the resources opened by the instance shall be reclaimed.

Parameters

<i>ec</i>	An informative panic code defined by the TA. May be displayed in traces if traces are available.
-----------	--

10.7.2.2 GetRelTimeEnd()

```
TEE_Result GetRelTimeEnd (
    uint64_t end )
```

Core Functions, Time Functions.

Return the elapsed.

[GetRelTimeEnd\(\)](#) - finds the real time of the end timing.

This function prints the ending time.

Parameters

<i>end</i>	End timing
------------	------------

Returns

0 If success

[GetRelTimeStart\(\)](#) - find the real time of the end timing.

This function prints the End timing.

Parameters

<i>end</i>	End timing
------------	------------

Returns

0 if success else error occurred

10.7.2.3 GetRelTimeStart()

```
TEE_Result GetRelTimeStart (
    uint64_t start )
```

Core Functions, Time Functions.

Fast relative Time function which guarantees no hart switch or context switch between Trusted and Untrusted sides.

Most of the time ending up writing similar functions when only measuring the relative time in usec resolution which do not require the quality of the time itself but the distance of the two points.
For the usage above, the function does not have to return wall clock time.
Not prepared in both Keystone and GP.

[GetRelTimeStart\(\)](#) - Gets the real time of the start timing.

This function prints the starting time.

Parameters

<i>start</i>	Start timing
--------------	--------------

Returns

0 on success

[GetRelTimeStart\(\)](#) - Gets the real time of the start timing.

This function prints the start timing.

Parameters

<i>start</i>	start timing
--------------	--------------

Returns

0 if success else error occurred.

10.7.2.4 TEE_AEDecryptFinal() `TEE_Result TEE_AEDecryptFinal (`
 `TEE_OperationHandle operation,`
 `const void * srcData,`
 `uint32_t srcLen,`
 `void * destData,`
 `uint32_t * destLen,`
 `void * tag,`
 `uint32_t tagLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

[TEE_AEDecryptFinal\(\)](#) - Processes data that has not been processed by previous calls to TEE_AEUpdate as well as data supplied in srcData.

This function completes the AE operation and compares the computed tag with the tag supplied in the parameter tag. The operation handle can be reused or newly initialized. The buffers srcData and destData shall be either completely disjoint or equal in their starting positions. The operation may be in either initial or active state and enters initial state afterwards.

Parameters

<i>operation</i>	Handle of a running AE operation
<i>srcData</i>	Reference to final chunk of input data to be encrypted
<i>srcLen</i>	length of the input data
<i>destData</i>	Output buffer. Can be omitted if the output is to be discarded.
<i>destLen</i>	length of the buffer.
<i>tag</i>	Output buffer filled with the computed tag
<i>tagLen</i>	length of the tag.

Returns

0 on success.

TEE_ERROR_SHORT_BUFFER If the output buffer is not large enough to contain the output

TEE_ERROR_MAC_INVALID If the computed tag does not match the supplied tag

10.7.2.5 TEE_AEEncryptFinal() `TEE_Result TEE_AEEncryptFinal (`
`TEE_OperationHandle operation,`
`const void * srcData,`
`uint32_t srcLen,`
`void * destData,`
`uint32_t * destLen,`
`void * tag,`
`uint32_t * tagLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

[TEE_AEEncryptFinal\(\)](#) - processes data that has not been processed by previous calls to TEE_AEUpdate as well as data supplied in srcData .

TEE_AEEncryptFinal completes the AE operation and computes the tag. The operation handle can be reused or newly initialized. The buffers srcData and destData SHALL be either completely disjoint or equal in their starting positions. The operation may be in either initial or active state and enters initial state afterwards.

Parameters

<i>operation</i>	Handle of a running AE operation
<i>srcData</i>	Reference to final chunk of input data to be encrypted
<i>srcLen</i>	length of the input data
<i>destData</i>	Output buffer. Can be omitted if the output is to be discarded.
<i>destLen</i>	length of the buffer.
<i>tag</i>	Output buffer filled with the computed tag
<i>tagLen</i>	length of the tag.

Returns

0 on success.

TEE_ERROR_SHORT_BUFFER If the output or tag buffer is not large enough to contain the output.

10.7.2.6 TEE_AEInit() `TEE_Result TEE_AEInit (`
 `TEE_OperationHandle operation,`
 `const void * nonce,`
 `uint32_t nonceLen,`
 `uint32_t tagLen,`
 `uint32_t AADLen,`
 `uint32_t payloadLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

[TEE_AEInit\(\)](#) - Initializes an Authentication Encryption operation.

The operation must be in initial state and remains in the initial state afterwards.

Parameters

<i>operation</i>	A handle on the operation.
<i>nonce</i>	The operation nonce or IV
<i>nonceLen</i>	length of nonce
<i>tagLen</i>	Size in bits of the tag
<i>AADLen</i>	Length in bytes of the AAD
<i>payloadLen</i>	Length in bytes of the payload.

Returns

0 on success.

TEE_ERROR_NOT_SUPPORTED If the tag length is not supported by the algorithm.

10.7.2.7 TEE_AEUpdate() `TEE_Result TEE_AEUpdate (`
 `TEE_OperationHandle operation,`
 `const void * srcData,`
 `uint32_t srcLen,`
 `void * destData,`
 `uint32_t * destLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

[TEE_AEUpdate\(\)](#) - Accumulates data for an Authentication Encryption operation

This function describes Input data does not have to be a multiple of block size. Subsequent calls to this function are possible. Unless one or more calls of this function have supplied sufficient input data, no output is generated. when using this routine to decrypt the returned data may be corrupt since the integrity check is not performed until all the data has been processed. If this is a concern then only use the TEE_AEDecryptFinal routine.

Parameters

<i>operation</i>	Handle of a running AE operation.
<i>srcData</i>	Input data buffer to be encrypted or decrypted
<i>srcLen</i>	length of the input buffer.
<i>destData</i>	Output buffer
<i>destLen</i>	length of the out put buffer.

Returns

0 on success.

TEE_ERROR_SHORT_BUFFER if the output buffer is not large enough to contain the output.

10.7.2.8 TEE_AEUpdateAAD() void TEE_AEUpdateAAD (
 TEE_OperationHandle operation,
 const void * AADdata,
 uint32_t AADdataLen)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

[TEE_AEUpdateAAD\(\)](#) - Feeds a new chunk of Additional Authentication Data (AAD) to the AE operation. Subsequent calls to this function are possible.

The TEE_AEUpdateAAD function feeds a new chunk of Additional Authentication Data (AAD) to the AE operation. Subsequent calls to this function are possible. The buffers srcData and destData shall be either completely disjoint or equal in their starting positions. The operation SHALL be in initial state and remains in initial state afterwards.

Parameters

<i>operation</i>	Handle on the AE operation
<i>AADdata</i>	Input buffer containing the chunk of AAD
<i>AADdataLen</i>	length of the chunk of AAD.

10.7.2.9 TEE_AllocateOperation() TEE_Result TEE_AllocateOperation (
 TEE_OperationHandle * operation,
 uint32_t algorithm,
 uint32_t mode,
 uint32_t maxKeySize)

Crypto, for all Crypto Functions.

All Crypto Functions use TEE_OperationHandle* operation instances.
 Create Crypto instance.

[TEE_AllocateOperation\(\)](#) - Allocates a handle for a new cryptographic operation and sets the mode and algorithm type.

If this function does not return with TEE_SUCCESS then there is no valid handle value. Once a cryptographic operation has been created, the implementation shall guarantee that all resources necessary for the operation are allocated and that any operation with a key of at most maxKeySize bits can be performed. For algorithms that take multiple keys, for example the AES XTS algorithm, the maxKeySize parameter specifies the size of the largest key. It is up to the implementation to properly allocate space for multiple keys if the algorithm so requires.

Parameters

<i>operation</i>	reference to generated operation handle.
<i>algorithm</i>	One of the cipher algorithms.
<i>mode</i>	The operation mode.
<i>maxKeySize</i>	Maximum key size in bits for the operation.

Returns

0 in case of success

TEE_ERROR_OUT_OF_MEMORY If there are not enough resources to allocate the operation.

TEE_ERROR_NOT_SUPPORTED If the mode is not compatible with the algorithm or key size or if the algorithm is not one of the listed algorithms or if maxKeySize is not appropriate for the algorithm.

10.7.2.10 TEE_AllocateTransientObject() `TEE_Result TEE_AllocateTransientObject (`
`TEE_ObjectType objectType,`
`uint32_t maxKeySize,`
`TEE_ObjectHandle * object)`

Crypto, Asymmetric key Verification Functions.

Create object storing asymmetric key.

[TEE_AllocateTransientObject\(\)](#) - Allocates an uninitialized transient object. Transient objects are used to hold a cryptographic object (key or key-pair).

The value TEE.KEYSIZE.NO_KEY should be used for maxObjectSize for object types that do not require a key so that all the container resources can be pre-allocated. As allocated, the container is uninitialized. It can be initialized by subsequently importing the object material, generating an object, deriving an object, or loading an object from the Trusted Storage.

Parameters

<i>objectType</i>	Type of uninitialized object container to be created
<i>maxKeySize</i>	Key Size of the object.
<i>object</i>	Filled with a handle on the newly created key container.

Returns

0 on success

TEE_ERROR_OUT_OF_MEMORY If not enough resources are available to allocate the object handle.

TEE_ERROR_NOT_SUPPORTED If the key size is not supported or the object type is not supported.

10.7.2.11 TEE_AsymmetricSignDigest() `TEE_Result` TEE_AsymmetricSignDigest (
 `TEE.OperationHandle` operation,
 const `TEE.Attribute` * params,
 uint32_t paramCount,
 const void * digest,
 uint32_t digestLen,
 void * signature,
 uint32_t * signatureLen)

Crypto, Asymmetric key Verification Functions.

Sign a message digest within an asymmetric key operation.

Keystone has ed25519_sign().

Equivalent in openssl is EVP_DigestSign().

[TEE_AsymmetricSignDigest\(\)](#) - Signs a message digest within an asymmetric operation.

Parameters

<i>operation</i>	Handle on the operation, which SHALL have been suitably set up with an operation key.
<i>params</i>	Optional operation parameters
<i>paramCount</i>	size of param
<i>digest</i>	Input buffer containing the input message digest
<i>digestLen</i>	length of input buffer.
<i>signature</i>	Output buffer written with the signature of the digest
<i>signatureLen</i>	length of output buffer.

Returns

0 on success

TEE_ERROR_SHORT_BUFFER If the signature buffer is not large enough to hold the result

10.7.2.12 TEE_AsymmetricVerifyDigest() `TEE_Result` TEE_AsymmetricVerifyDigest (
 `TEE.OperationHandle` operation,
 const `TEE.Attribute` * params,
 uint32_t paramCount,
 const void * digest,
 uint32_t digestLen,


```
const void * signature,
uint32_t signatureLen )
```

Crypto, Asymmetric key Verification Functions.

Verifies a message digest signature within an asymmetric key operation.

Keystone has `ed25519_verify()`.

Equivalent in openssl is `EVP_DigestVerify()`.

[TEE_AsymmetricVerifyDigest\(\)](#) - verifies a message digest signature within an asymmetric operation.

This function describes the message digest signature verify by calling `ed25519_verify()`.

Parameters

<i>operation</i>	Handle on the operation, which SHALL have been suitably set up with an operation key.
<i>params</i>	Optional operation parameters
<i>paramCount</i>	size of param.
<i>digest</i>	Input buffer containing the input message digest
<i>digestLen</i>	length of input buffer.
<i>signature</i>	Output buffer written with the signature of the digest
<i>signatureLen</i>	length of output buffer.

Returns

TEE_SUCCESS on success

TEE_ERROR_SIGNATURE_INVALID if the signature is invalid.

10.7.2.13 TEE.CipherInit() `void TEE_CipherInit (`
`TEE.OperationHandle operation,`
`const void * nonce,`
`uint32_t nonceLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CBC.

[TEE.CipherInit\(\)](#) - starts the symmetric cipher operation.

The operation shall have been associated with a key. If the operation is in active state, it is reset and then initialized. If the operation is in initial state, it is moved to active state.

Parameters

<i>operation</i>	A handle on an opened cipher operation setup with a key
<i>nonce</i>	Buffer containing the operation Initialization Vector as appropriate.
<i>nonceLen</i>	length of the buffer

10.7.2.14 TEE_CipherUpdate() `TEE.Result` TEE_CipherUpdate (
 `TEE.OperationHandle` operation,
 const void * srcData,
 uint32_t srcLen,
 void * destData,
 uint32_t * destLen)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CBC.

`TEE_CipherUpdate()` - encrypts or decrypts input data.

Input data does not have to be a multiple of block size. Subsequent calls to this function are possible. Unless one or more calls of this function have supplied sufficient input data, no output is generated. The cipher operation is finalized with a call to `TEE_CipherDoFinal`. The buffers `srcData` and `destData` SHALL be either completely disjoint or equal in their starting positions. The operation SHALL be in active state.

Parameters

<i>operation</i>	Handle of a running Cipher operation
<i>srcData</i>	Input data buffer to be encrypted or decrypted
<i>srcLen</i>	length of input buffer
<i>destData</i>	output buffer
<i>destLen</i>	output buffer length.

Returns

0 on success else

TEE_ERROR_SHORT_BUFFER If the output buffer is not large enough to contain the output. In this case, the input is not fed into the algorithm.

10.7.2.15 TEE_CloseObject() void TEE_CloseObject (
 `TEE.ObjectHandle` object)

Core Functions, Secure Storage Functions (data is isolated for each TA)

Destroy object (key, key-pair or Data).

`TEE_CloseObject()` - Closes an opened object handle.

The object can be persistent or transient. For transient objects, `TEE_CloseObject` is equivalent to `TEE_FreeTransientObject`.

Parameters

<i>object</i>	Handle of the object.
---------------	-----------------------

Returns

TEE_SUCCESS if success else error occurred.

[TEE_CloseObject\(\)](#) - Function closes an opened object handle.

The object can be persistent or transient. For transient objects, TEE_CloseObject is equivalent to TEE_Free↔ TransientObject.

Parameters

<i>object</i>	Handle of the object
---------------	----------------------

Returns

TEE_SUCCESS if success else error occurred.

10.7.2.16 TEE_CreatePersistentObject() `TEE_Result TEE_CreatePersistentObject (`
 uint32_t *storageID*,
 const void * *objectID*,
 uint32_t *objectIDLen*,
 uint32_t *flags*,
 [TEE_ObjectHandle](#) *attributes*,
 const void * *initialData*,
 uint32_t *initialDataLen*,
 [TEE_ObjectHandle](#) * *object*)

Core Functions, Secure Storage Functions (data is isolated for each TA)

Create persistent object (key, key-pair or Data).

For the people who have not written code on GP then probably do not need to care the meaning of what is Persistent Object is, since the following are enough to use secure storage feature.

[TEE_CreatePersistentObject\(\)](#) - Creates a persistent object with initial attributes.

In this function an initial data stream content returns either a handle on the created object or TEE_HANDLE_NULL upon failure.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
Paramter list continued on next page	

<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>attributes</i>	A handle on a persistent object or an initialized transient object from which to take the persistent object attributes
<i>initialData</i>	The initial data content of the persistent object
<i>initialDataLen</i>	The initial data content of the persistent object
<i>object</i>	A pointer to the handle which contains the opened handle upon successful completion

Returns

0 if success else error occurred.

[TEE.CreatePersistentObject\(\)](#) - Creates a persistent object with initial attributes.

An initial data stream content, and optionally returns either a handle on the created object, or TEE_HANDLE_NULL upon failure.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>attributes</i>	A handle on a persistent object or an initialized transient object from which to take the persistent object attributes
<i>initialData</i>	The initial data content of the persistent object
<i>initialDataLen</i>	The initial data content of the persistent object
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion

Returns

0 if success, else error occurred.

```

10.7.2.17 TEE_DigestDoFinal() TEE_Result TEE_DigestDoFinal (
    TEE_OperationHandle operation,
    const void * chunk,
    uint32_t chunkLen,
    void * hash,
    uint32_t * hashLen )

```

Function accumulates message data for hashing.

[TEE.DigestDoFinal\(\)](#) - Finalizes the message digest operation and produces the message hash.

This function finalizes the message digest operation and produces the message hash. Afterwards the Message Digest operation is reset to initial state and can be reused.

Parameters

<i>operation</i>	Handle of a running Message Digest operation.
<i>chunk</i>	Chunk of data to be hashed.
<i>chunkLen</i>	size of the chunk.
<i>hash</i>	Output buffer filled with the message hash.
<i>hashLen</i>	length of the message hash.

Returns

0 on success

TEE_ERROR_SHORT_BUFFER If the output buffer is too small. In this case, the operation is not finalized.

10.7.2.18 TEE.DigestUpdate() void TEE.DigestUpdate (
 TEE.OperationHandle operation,
 const void * chunk,
 uint32_t chunkSize)

Crypto, Message Digest Functions.

Function accumulates message data for hashing.

[TEE.DigestUpdate\(\)](#)- Accumulates message data for hashing.

This function describes the message does not have to be block aligned. Subsequent calls to this function are possible. The operation may be in either initial or active state and becomes active.

Parameters

<i>operation</i>	Handle of a running Message Digest operation.
<i>chunk</i>	Chunk of data to be hashed
<i>chunkSize</i>	size of the chunk.

10.7.2.19 TEE.FreeOperation() void TEE.FreeOperation (
 TEE.OperationHandle operation)

Crypto, for all Crypto Functions.

All Crypto Functions use TEE.OperationHandle* operation instances.

Destroy Crypto instance.

[TEE.FreeOperation\(\)](#) - Deallocates all resources associated with an operation handle.

This function deallocates all resources associated with an operation handle. After this function is called, the operation handle is no longer valid. All cryptographic material in the operation is destroyed. The function does nothing if operation is TEE_HANDLE_NULL.

Parameters

<i>operation</i>	Reference to operation handle.
------------------	--------------------------------

Returns

nothing after the operation free.

10.7.2.20 TEE_FreeTransientObject() `void TEE_FreeTransientObject (TEE_ObjectHandle object)`

Crypto, Asymmetric key Verification Functions.

Destroy object storing asymmetric key.

[TEE_FreeTransientObject\(\)](#) - Deallocates a transient object previously allocated with [TEE_AllocateTransientObject](#) .

this function describes the object handle is no longer valid and all resources associated with the transient object shall have been reclaimed after the [TEE_AllocateTransientObject\(\)](#) call.

Parameters

<i>object</i>	Handle on the object to free.
---------------	-------------------------------

10.7.2.21 TEE_GenerateKey() `TEE_Result TEE_GenerateKey (TEE_ObjectHandle object, uint32_t keySize, const TEE_Attribute * params, uint32_t paramCount)`

Crypto, Asymmetric key Verification Functions.

Generate asymmetric keypair.

[TEE_GenerateKey \(\)](#) - Generates a random key or a key-pair and populates a transient key object with the generated key material.

The size of the desired key is passed in the keySize parameter and shall be less than or equal to the maximum key size specified when the transient object was created.

Parameters

<i>object</i>	Handle on an uninitialized transient key to populate with the generated key.
Paramter list continued on next page	

<i>keySize</i>	Requested key size shall be less than or equal to the maximum key size specified when the object container was created
<i>params</i>	Parameters for the key generation.
<i>paramCount</i>	The values of all parameters are copied into the object so that the params array and all the memory buffers it points to may be freed after this routine returns without affecting the object.

Returns

0 on success

TEE_ERROR_BAD_PARAMETERS If an incorrect or inconsistent attribute is detected. The checks that are performed depend on the implementation.

10.7.2.22 TEE_GenerateRandom() void TEE_GenerateRandom (
 void * *randomBuffer*,
 uint32_t *randomBufferLen*)

Crypto, common.

Random Data Generation Function. The quality of the random is implementation dependent.

I am not sure this should be in Keystone or not, but it is very handy.

Good to have adding a way to check the quality of the random implementation.

ocall_getrandom() - For getting random data.

This function describes that the return value is returned based on the size of buffer by calling the functions ocall_getrandom196 and ocall_getrandom16

Parameters

<i>buf</i>	character type buffer
<i>len</i>	size of the buffer
<i>flags</i>	unassigned integer flag

Returns

return value will be returned based on length of buffer. [TEE_GenerateRandom\(\)](#) - Function generates random data.

This function generates random data of random buffer length and is stored in to randomBuffer by calling ocall_getrandom(). If return is not equal to randomBufferLen then TEE_Panic function is called.

Parameters

<i>randomBuffer</i>	Reference to generated random data
<i>randomBufferLen</i>	Byte length of requested random data

Returns

ocall version random data

[TEE.GenerateRandom\(\)](#) - Generates random data.

This function generates random data of random bufferlength and is stored in to randomBuffer by calling `sgx_read_rand()`.

Parameters

<i>randomBuffer</i>	Reference to generated random data
<i>randomBufferLen</i>	Byte length of requested random data

10.7.2.23 TEE.GetObjectInfo1() `TEE_Result TEEGetObjectInfo1 (TEE_ObjectHandle object, TEE_ObjectInfo * objectInfo)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

Get length of object required before reading the object.

[TEE.GetObjectInfo1\(\)](#) - Returns the characteristics of an object.

This function returns a handle which can be used to access the object's attributes and data stream.

Parameters

<i>objectInfo</i>	Pointer to a structure filled with the object information
<i>object</i>	Handle of the object

Returns

0 if success else error occurred.

[TEE.GetObjectInfo1\(\)](#) - Function returns the characteristics of an object.

It returns a handle that can be used to access the object's attributes and data stream.

Parameters

<i>objectInfo</i>	Pointer to a structure filled with the object information
<i>object</i>	Handle of the object

Returns

0 if success else error occurred.

10.7.2.24 TEE_GetREETime() `void TEE_GetREETime (`
`TEE_Time * time)`

Core Functions, Time Functions.

Wall clock time of host OS, expressed in the number of seconds since 1970-01-01 UTC. This could be implemented on Keystone using ocall.

[TEE_GetREETime\(\)](#) - Retrieves the current REE system time.

This function retrieves the current time as seen from the point of view of the REE.

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds
-------------	--

[TEE_GetREETime\(\)](#) - Function retrieves the current REE system time.

This function retrieves the current time as seen from the point of view of the REE.

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds.
-------------	---

10.7.2.25 TEE_GetSystemTime() `void TEE_GetSystemTime (`
`TEE_Time * time)`

Core Functions, Time Functions.

Time of TEE-controlled secure timer or Host OS time, implementation dependent.

[TEE_GetSystemTime\(\)](#) - Retrieves the current system time.

This function describes the system time has an arbitrary implementation defined origin that can vary across TA instances. The minimum guarantee is that the system time shall be monotonic for a given TA instance.

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds
-------------	--

[TEE_GetSystemTime\(\)](#) - Retrieves the current system time.

The system time has an arbitrary implementation-defined origin that can vary across TA instances

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds.
-------------	---

10.7.2.26 TEE_InitRefAttribute() `void TEE_InitRefAttribute (`
 `TEE_Attribute * attr,`
 `uint32_t attributeID,`
 `const void * buffer,`
 `uint32_t length)`

Crypto, Asymmetric key Verification Functions.

Storing asymmetric key.

[TEE_InitRefAttribute\(\)](#) - The helper function can be used to populate a single attribute either with a reference to a buffer or with integer values.

In `TEE_InitRefAttribute ()` only the buffer pointer is copied, not the content of the buffer. This means that the attribute structure maintains a pointer back to the supplied buffer. It is the responsibility of the TA author to ensure that the contents of the buffer maintain their value until the attributes array is no longer in use.

Parameters

<i>attr</i>	attribute structure to initialize.
<i>attributeID</i>	Identifier of the attribute to populate.
<i>buffer</i>	input buffer that holds the content of the attribute.
<i>length</i>	buffer length.

10.7.2.27 TEE_InitValueAttribute() `void TEE_InitValueAttribute (`
 `TEE_Attribute * attr,`
 `uint32_t attributeID,`
 `uint32_t a,`
 `uint32_t b)`

Crypto, Asymmetric key Verification Functions.

Storing asymmetric key.

[TEE.InitValueAttribute\(\)](#) - The helper function can be used to populate a single attribute either with a reference to a buffer or with integer values.

Parameters

<i>attr</i>	attribute structure to initialize.
<i>attributeID</i>	Identifier of the attribute to populate.
<i>a</i>	unsigned integer value to assign to the a member of the attribute structure.
<i>b</i>	unsigned integer value to assign to the b member of the attribute structure

10.7.2.28 TEE_OpenPersistentObject() `TEE.Result TEE_OpenPersistentObject (`
 uint32_t *storageID*,
 const void * *objectID*,
 uint32_t *objectIDLen*,
 uint32_t *flags*,
 TEE.ObjectHandle * *object*)

Core Functions, Secure Storage Functions (data is isolated for each TA)

Open persistent object.

[TEE.OpenPersistentObject\(\)](#) - Opens a handle on an existing persistent object.

This function returns a handle which can be used to access the object's attributes and data stream.

Parameters

<i>storageID</i>	The storage to use
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion

Returns

0 if success else error occurred.

[TEE.OpenPersistentObject\(\)](#) - Opens a handle on an existing persistent object.

This function returns a handle that can be used to access the object's attributes and data stream.

Parameters

<i>storageID</i>	The storage to use.
Paramter list continued on next page	

<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion

Returns

0 if success, else error occurred.

10.7.2.29 TEE_ReadObjectData() `TEE_Result TEE_ReadObjectData (`
`TEE_ObjectHandle object,`
`void * buffer,`
`uint32_t size,`
`uint32_t * count)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

Read object.

[TEE_ReadObjectData\(\)](#) - Attempts to read size bytes from the data stream associated with the object into the buffer pointed to by buffer.

The bytes are read starting at the position in the data stream currently stored in the object handle. The handle's position is incremented by the number of bytes actually read. On completion of TEE_ReadObjectData sets the number of bytes actually read in the "uint32_t" pointed to by count. The value written to *count may be less than size if the number of bytes until the end-of-3067 stream is less than size. It is set to 0 if the position at the start of the read operation is at or beyond the end-of-stream. These are the only cases where *count may be less than size.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write
<i>count</i>	size of the buffer.

Returns

TEE_SUCCESS if success else error occurred.

[TEE_ReadObjectData\(\)](#) - Attempts to read size bytes from the data stream associated with the object object into the buffer pointed to by buffer.

The bytes are read starting at the position in the data stream currently stored in the object handle. The handle's position is incremented by the number of bytes actually read. On completion TEE_ReadObjectData sets the number of bytes actually read in the uint32_t pointed to by count. The value written to *count may be less than size if the number of bytes until the end-of-3067 stream is less than size. It is set to 0 if the position at the start of the read operation is at or beyond the end-of-stream. These are the only cases where *count may be less than size.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write
<i>count</i>	size of the buffer.

Returns

TEE_SUCCESS if success, else error occurred.

10.7.2.30 TEE_SetOperationKey() `TEE_Result` TEE_SetOperationKey (
`TEE_OperationHandle` operation,
`TEE_ObjectHandle` key)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Set symmetric key used in operation.

[TEE_SetOperationKey\(\)](#) - Programs the key of an operation; that is, it associates an operation with a key.

The key material is copied from the key object handle into the operation. After the key has been set, there is no longer any link between the operation and the key object. The object handle can be closed or reset and this will not affect the operation. This copied material exists until the operation is freed using TEE_FreeOperation or another key is set into the operation.

Parameters

<i>operation</i>	Operation handle.
<i>key</i>	A handle on a key object.

Returns

0 on success return

TEE_ERROR_CORRUPT_OBJECT If the object is corrupt. The object handle is closed.

TEE_ERROR_STORAGE_NOT_AVAILABLE If the persistent object is stored in a storage area which is currently inaccessible.

10.7.2.31 TEE_WriteObjectData() `TEE_Result` TEE_WriteObjectData (
`TEE_ObjectHandle` object,
const void * *buffer*,
uint32_t *size*)

Core Functions, Secure Storage Functions (data is isolated for each TA)

Write object.

[TEE_WriteObjectData\(\)](#) - Writes the buffer data in to persistent objects.

In this function it checks if object is present or not, the encryption/ decryption buffer is taken by calling `mbedtls_aes_crypt_cbc()` then that buffer data is encrypted and mapped to object. On the base of object creation `TEE_SUCCESS` appears else `TEE_ERROR_GENERIC` appears.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write

Returns

`TEE_SUCCESS` if success else error occurred.

[TEE_WriteObjectData\(\)](#) - writes size bytes from the buffer pointed to by buffer to the data stream associated with the open object handle object.

If the current data position points before the end-of-stream, then size bytes are written to the data stream, overwriting bytes starting at the current data position. If the current data position points beyond the stream's end, then the data stream is first extended with zero bytes until the length indicated by the data position indicator is reached, and then size bytes are written to the stream.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write

Returns

`TEE_SUCCESS` if success else error occurred.

10.8 tee-ta-internal.h

[Go to the documentation of this file.](#)

```

1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                               and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10 *
11 * 1. Redistributions of source code must retain the above copyright notice,
12 * this list of conditions and the following disclaimer.
13 *
14 * 2. Redistributions in binary form must reproduce the above copyright notice,
15 * this list of conditions and the following disclaimer in the documentation
16 * and/or other materials provided with the distribution.

```

```

17 *
18 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28 * POSSIBILITY OF SUCH DAMAGE.
29 */
30
31 #ifndef TA_INTERNAL_TEE_H
32 #define TA_INTERNAL_TEE_H
33
34 #include "tee-common.h"
35
36 #ifdef __cplusplus
37 extern "C" {
38 #endif
39
40 void __attribute__((noreturn)) TEE_Panic(unsigned long code);
41
42 void TEE_GetREETime(TEE_Time *time);
43
44 /* Wall clock time is important for verifying certificates. */
45 void TEE_GetSystemTime(TEE_Time *time);
46
47 /* Start timer */
48 TEE_Result GetRelTimeStart(uint64_t start);
49
50 TEE_Result GetRelTimeEnd(uint64_t end);
51
52 TEE_Result TEE_CreatePersistentObject(uint32_t storageID, const void *objectID,
53                                     uint32_t objectIDLen, uint32_t flags,
54                                     TEE_ObjectHandle attributes,
55                                     const void *initialData,
56                                     uint32_t initialDataLen,
57                                     TEE_ObjectHandle *object);
58
59 TEE_Result TEE_OpenPersistentObject(uint32_t storageID, const void *objectID,
60                                    uint32_t objectIDLen, uint32_t flags,
61                                    TEE_ObjectHandle *object);
62
63 TEE_Result TEE_GetObjectInfo1(TEE_ObjectHandle object, TEE_ObjectInfo *objectInfo);
64
65 TEE_Result TEE_WriteObjectData(TEE_ObjectHandle object, const void *buffer,
66                               uint32_t size);
67
68 TEE_Result TEE_ReadObjectData(TEE_ObjectHandle object, void *buffer,
69                              uint32_t size, uint32_t *count);
70
71 void TEE_CloseObject(TEE_ObjectHandle object);
72
73 void TEE_GenerateRandom(void *randomBuffer, uint32_t randomBufferLen);
74
75 TEE_Result TEE_AllocateOperation(TEE_OperationHandle *operation,
76                                 uint32_t algorithm, uint32_t mode,
77                                 uint32_t maxKeySize);
78
79 void TEE_FreeOperation(TEE_OperationHandle operation);
80
81 void TEE_DigestUpdate(TEE_OperationHandle operation,
82                      const void *chunk, uint32_t chunkSize);
83
84 TEE_Result TEE_DigestDoFinal(TEE_OperationHandle operation, const void *chunk,
85                             uint32_t chunkLen, void *hash, uint32_t *hashLen);
86
87 TEE_Result TEE_SetOperationKey(TEE_OperationHandle operation,
88                               TEE_ObjectHandle key);
89
90 TEE_Result TEE_AEInit(TEE_OperationHandle operation, const void *nonce,
91                      uint32_t nonceLen, uint32_t tagLen, uint32_t AADLen,
92                      uint32_t payloadLen);
93
94 TEE_Result TEE_AEUpdate(TEE_OperationHandle operation, const void *srcData,
95                        uint32_t srcLen, void *destData, uint32_t *destLen);

```

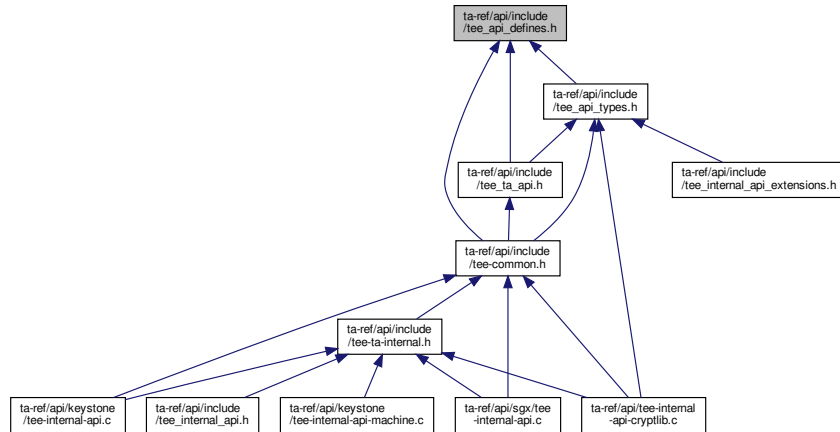
```

155
156 void TEE_AEUpdateAAD(TEE_OperationHandle operation, const void *AADdata,
157                     uint32_t AADdataLen);
159
160 TEE_Result TEE_AEEncryptFinal(TEE_OperationHandle operation,
161                               const void *srcData, uint32_t srcLen,
162                               void *destData, uint32_t *destLen, void *tag,
163                               uint32_t *tagLen);
165
166 TEE_Result TEE_AEDecryptFinal(TEE_OperationHandle operation,
167                               const void *srcData, uint32_t srcLen,
168                               void *destData, uint32_t *destLen, void *tag,
169                               uint32_t tagLen);
170
172
173 void TEE_CipherInit(TEE_OperationHandle operation, const void *nonce,
174                    uint32_t nonceLen);
176
177 TEE_Result TEE_CipherUpdate(TEE_OperationHandle operation, const void *srcData,
178                             uint32_t srcLen, void *destData, uint32_t *destLen);
179
181
182 TEE_Result TEE_GenerateKey(TEE_ObjectHandle object, uint32_t keySize,
183                             const TEE_Attribute *params, uint32_t paramCount);
185
186 TEE_Result TEE_AllocateTransientObject(TEE_ObjectType objectType,
187                                       uint32_t maxKeySize,
188                                       TEE_ObjectHandle *object);
190
191 void TEE_InitRefAttribute(TEE_Attribute *attr, uint32_t attributeID,
192                           const void *buffer, uint32_t length);
194
195 void TEE_InitValueAttribute(TEE_Attribute *attr, uint32_t attributeID,
196                             uint32_t a, uint32_t b);
198
199 void TEE_FreeTransientObject(TEE_ObjectHandle object);
200
202
206 TEE_Result TEE_AsymmetricSignDigest(TEE_OperationHandle operation,
207                                     const TEE_Attribute *params,
208                                     uint32_t paramCount, const void *digest,
209                                     uint32_t digestLen, void *signature,
210                                     uint32_t *signatureLen);
212
216 TEE_Result TEE_AsymmetricVerifyDigest(TEE_OperationHandle operation,
217                                       const TEE_Attribute *params,
218                                       uint32_t paramCount, const void *digest,
219                                       uint32_t digestLen, const void *signature,
220                                       uint32_t signatureLen);
221
222 #ifdef __cplusplus
223 }
224 #endif
225
226 #endif /* TA_INTERNAL_TEE_H */

```


10.9 ta-ref/api/include/tee_api_defines.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define [TEE_INT_CORE_API_SPEC_VERSION](#) 0x0000000A
- #define [TEE_HANDLE_NULL](#) 0
- #define [TEE_TIMEOUT_INFINITE](#) 0xFFFFFFFF
- #define [TEE_SUCCESS](#) 0x00000000
- #define [TEE_ERROR_CORRUPT_OBJECT](#) 0xF0100001
- #define [TEE_ERROR_CORRUPT_OBJECT_2](#) 0xF0100002
- #define [TEE_ERROR_STORAGE_NOT_AVAILABLE](#) 0xF0100003
- #define [TEE_ERROR_STORAGE_NOT_AVAILABLE_2](#) 0xF0100004
- #define [TEE_ERROR_GENERIC](#) 0xFFFF0000
- #define [TEE_ERROR_ACCESS_DENIED](#) 0xFFFF0001
- #define [TEE_ERROR_CANCEL](#) 0xFFFF0002
- #define [TEE_ERROR_ACCESS_CONFLICT](#) 0xFFFF0003
- #define [TEE_ERROR_EXCESS_DATA](#) 0xFFFF0004
- #define [TEE_ERROR_BAD_FORMAT](#) 0xFFFF0005
- #define [TEE_ERROR_BAD_PARAMETERS](#) 0xFFFF0006
- #define [TEE_ERROR_BAD_STATE](#) 0xFFFF0007
- #define [TEE_ERROR_ITEM_NOT_FOUND](#) 0xFFFF0008
- #define [TEE_ERROR_NOT_IMPLEMENTED](#) 0xFFFF0009
- #define [TEE_ERROR_NOT_SUPPORTED](#) 0xFFFF000A
- #define [TEE_ERROR_NO_DATA](#) 0xFFFF000B
- #define [TEE_ERROR_OUT_OF_MEMORY](#) 0xFFFF000C
- #define [TEE_ERROR_BUSY](#) 0xFFFF000D
- #define [TEE_ERROR_COMMUNICATION](#) 0xFFFF000E
- #define [TEE_ERROR_SECURITY](#) 0xFFFF000F
- #define [TEE_ERROR_SHORT_BUFFER](#) 0xFFFF0010
- #define [TEE_ERROR_EXTERNAL_CANCEL](#) 0xFFFF0011
- #define [TEE_ERROR_OVERFLOW](#) 0xFFFF300F
- #define [TEE_ERROR_TARGET_DEAD](#) 0xFFFF3024
- #define [TEE_ERROR_STORAGE_NO_SPACE](#) 0xFFFF3041
- #define [TEE_ERROR_MAC_INVALID](#) 0xFFFF3071

- #define TEE_ERROR_SIGNATURE_INVALID 0xFFFF3072
- #define TEE_ERROR_TIME_NOT_SET 0xFFFF5000
- #define TEE_ERROR_TIME_NEEDS_RESET 0xFFFF5001
- #define TEE_PARAM_TYPE_NONE 0
- #define TEE_PARAM_TYPE_VALUE_INPUT 1
- #define TEE_PARAM_TYPE_VALUE_OUTPUT 2
- #define TEE_PARAM_TYPE_VALUE_INOUT 3
- #define TEE_PARAM_TYPE_MEMREF_INPUT 5
- #define TEE_PARAM_TYPE_MEMREF_OUTPUT 6
- #define TEE_PARAM_TYPE_MEMREF_INOUT 7
- #define TEE_LOGIN_PUBLIC 0x00000000
- #define TEE_LOGIN_USER 0x00000001
- #define TEE_LOGIN_GROUP 0x00000002
- #define TEE_LOGIN_APPLICATION 0x00000004
- #define TEE_LOGIN_APPLICATION_USER 0x00000005
- #define TEE_LOGIN_APPLICATION_GROUP 0x00000006
- #define TEE_LOGIN_TRUSTED_APP 0xF0000000
- #define TEE_ORIGIN_API 0x00000001
- #define TEE_ORIGIN_COMMS 0x00000002
- #define TEE_ORIGIN_TEE 0x00000003
- #define TEE_ORIGIN_TRUSTED_APP 0x00000004
- #define TEE_PROPSET_TEE_IMPLEMENTATION (TEE_PropSetHandle)0xFFFFFFFFD
- #define TEE_PROPSET_CURRENT_CLIENT (TEE_PropSetHandle)0xFFFFFFFFE
- #define TEE_PROPSET_CURRENT_TA (TEE_PropSetHandle)0xFFFFFFFFF
- #define TEE_MEMORY_ACCESS_READ 0x00000001
- #define TEE_MEMORY_ACCESS_WRITE 0x00000002
- #define TEE_MEMORY_ACCESS_ANY_OWNER 0x00000004
- #define TEE_MALLOC_FILL_ZERO 0x00000000
- #define TEE_STORAGE_PRIVATE 0x00000001
- #define TEE_DATA_FLAG_ACCESS_READ 0x00000001
- #define TEE_DATA_FLAG_ACCESS_WRITE 0x00000002
- #define TEE_DATA_FLAG_ACCESS_WRITE_META 0x00000004
- #define TEE_DATA_FLAG_SHARE_READ 0x00000010
- #define TEE_DATA_FLAG_SHARE_WRITE 0x00000020
- #define TEE_DATA_FLAG_OVERWRITE 0x00000400
- #define TEE_DATA_MAX_POSITION 0xFFFFFFFF
- #define TEE_OBJECT_ID_MAX_LEN 64
- #define TEE_USAGE_EXTRACTABLE 0x00000001
- #define TEE_USAGE_ENCRYPT 0x00000002
- #define TEE_USAGE_DECRYPT 0x00000004
- #define TEE_USAGE_MAC 0x00000008
- #define TEE_USAGE_SIGN 0x00000010
- #define TEE_USAGE_VERIFY 0x00000020
- #define TEE_USAGE_DERIVE 0x00000040
- #define TEE_HANDLE_FLAG_PERSISTENT 0x00010000
- #define TEE_HANDLE_FLAG_INITIALIZED 0x00020000
- #define TEE_HANDLE_FLAG_KEY_SET 0x00040000
- #define TEE_HANDLE_FLAG_EXPECT_TWO_KEYS 0x00080000
- #define TEE_OPERATION_CIPHER 1
- #define TEE_OPERATION_MAC 3
- #define TEE_OPERATION_AE 4
- #define TEE_OPERATION_DIGEST 5
- #define TEE_OPERATION_ASYMMETRIC_CIPHER 6
- #define TEE_OPERATION_ASYMMETRIC_SIGNATURE 7
- #define TEE_OPERATION_KEY_DERIVATION 8

- #define TEE_OPERATION_STATE_INITIAL 0x00000000
- #define TEE_OPERATION_STATE_ACTIVE 0x00000001
- #define TEE_ALG_AES_ECB_NOPAD 0x10000010
- #define TEE_ALG_AES_CBC_NOPAD 0x10000110
- #define TEE_ALG_AES_CTR 0x10000210
- #define TEE_ALG_AES_CTS 0x10000310
- #define TEE_ALG_AES_XTS 0x10000410
- #define TEE_ALG_AES_CBC_MAC_NOPAD 0x30000110
- #define TEE_ALG_AES_CBC_MAC_PKCS5 0x30000510
- #define TEE_ALG_AES_CMAC 0x30000610
- #define TEE_ALG_AES_CCM 0x40000710
- #define TEE_ALG_AES_GCM 0x40000810
- #define TEE_ALG_DES_ECB_NOPAD 0x10000011
- #define TEE_ALG_DES_CBC_NOPAD 0x10000111
- #define TEE_ALG_DES_CBC_MAC_NOPAD 0x30000111
- #define TEE_ALG_DES_CBC_MAC_PKCS5 0x30000511
- #define TEE_ALG_DES3_ECB_NOPAD 0x10000013
- #define TEE_ALG_DES3_CBC_NOPAD 0x10000113
- #define TEE_ALG_DES3_CBC_MAC_NOPAD 0x30000113
- #define TEE_ALG_DES3_CBC_MAC_PKCS5 0x30000513
- #define TEE_ALG_RSASSA_PKCS1_V1_5_MD5 0x70001830
- #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA1 0x70002830
- #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA224 0x70003830
- #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA256 0x70004830
- #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA384 0x70005830
- #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA512 0x70006830
- #define TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1 0x7000F830
- #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1 0x70212930
- #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224 0x70313930
- #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256 0x70414930
- #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384 0x70515930
- #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512 0x70616930
- #define TEE_ALG_RSAES_PKCS1_V1_5 0x60000130
- #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1 0x60210230
- #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224 0x60310230
- #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256 0x60410230
- #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384 0x60510230
- #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512 0x60610230
- #define TEE_ALG_RSA_NOPAD 0x60000030
- #define TEE_ALG_DSA_SHA1 0x70002131
- #define TEE_ALG_DSA_SHA224 0x70003131
- #define TEE_ALG_DSA_SHA256 0x70004131
- #define TEE_ALG_DH_DERIVE_SHARED_SECRET 0x80000032
- #define TEE_ALG_MD5 0x50000001
- #define TEE_ALG_SHA1 0x50000002
- #define TEE_ALG_SHA224 0x50000003
- #define TEE_ALG_SHA256 0x50000004
- #define TEE_ALG_SHA384 0x50000005
- #define TEE_ALG_SHA512 0x50000006
- #define TEE_ALG_MD5SHA1 0x5000000F
- #define TEE_ALG_HMAC_MD5 0x30000001
- #define TEE_ALG_HMAC_SHA1 0x30000002
- #define TEE_ALG_HMAC_SHA224 0x30000003
- #define TEE_ALG_HMAC_SHA256 0x30000004
- #define TEE_ALG_HMAC_SHA384 0x30000005

- #define TEE_ALG_HMAC_SHA512 0x30000006
- #define TEE_ALG_ECDSA_P192 0x70001041
- #define TEE_ALG_ECDSA_P224 0x70002041
- #define TEE_ALG_ECDSA_P256 0x70003041
- #define TEE_ALG_ECDSA_P384 0x70004041
- #define TEE_ALG_ECDSA_P521 0x70005041
- #define TEE_ALG_ECDH_P192 0x80001042
- #define TEE_ALG_ECDH_P224 0x80002042
- #define TEE_ALG_ECDH_P256 0x80003042
- #define TEE_ALG_ECDH_P384 0x80004042
- #define TEE_ALG_ECDH_P521 0x80005042
- #define TEE_TYPE_AES 0xA0000010
- #define TEE_TYPE_DES 0xA0000011
- #define TEE_TYPE_DES3 0xA0000013
- #define TEE_TYPE_HMAC_MD5 0xA0000001
- #define TEE_TYPE_HMAC_SHA1 0xA0000002
- #define TEE_TYPE_HMAC_SHA224 0xA0000003
- #define TEE_TYPE_HMAC_SHA256 0xA0000004
- #define TEE_TYPE_HMAC_SHA384 0xA0000005
- #define TEE_TYPE_HMAC_SHA512 0xA0000006
- #define TEE_TYPE_RSA_PUBLIC_KEY 0xA0000030
- #define TEE_TYPE_RSA_KEYPAIR 0xA1000030
- #define TEE_TYPE_DSA_PUBLIC_KEY 0xA0000031
- #define TEE_TYPE_DSA_KEYPAIR 0xA1000031
- #define TEE_TYPE_DH_KEYPAIR 0xA1000032
- #define TEE_TYPE_ECDSA_PUBLIC_KEY 0xA0000041
- #define TEE_TYPE_ECDSA_KEYPAIR 0xA1000041
- #define TEE_TYPE_ECDH_PUBLIC_KEY 0xA0000042
- #define TEE_TYPE_ECDH_KEYPAIR 0xA1000042
- #define TEE_TYPE_GENERIC_SECRET 0xA0000000
- #define TEE_TYPE_CORRUPTED_OBJECT 0xA00000BE
- #define TEE_TYPE_DATA 0xA00000BF
- #define TEE_ATTR_SECRET_VALUE 0xC0000000
- #define TEE_ATTR_RSA_MODULUS 0xD0000130
- #define TEE_ATTR_RSA_PUBLIC_EXPONENT 0xD0000230
- #define TEE_ATTR_RSA_PRIVATE_EXPONENT 0xC0000330
- #define TEE_ATTR_RSA_PRIME1 0xC0000430
- #define TEE_ATTR_RSA_PRIME2 0xC0000530
- #define TEE_ATTR_RSA_EXPONENT1 0xC0000630
- #define TEE_ATTR_RSA_EXPONENT2 0xC0000730
- #define TEE_ATTR_RSA_COEFFICIENT 0xC0000830
- #define TEE_ATTR_DSA_PRIME 0xD0001031
- #define TEE_ATTR_DSA_SUBPRIME 0xD0001131
- #define TEE_ATTR_DSA_BASE 0xD0001231
- #define TEE_ATTR_DSA_PUBLIC_VALUE 0xD0000131
- #define TEE_ATTR_DSA_PRIVATE_VALUE 0xC0000231
- #define TEE_ATTR_DH_PRIME 0xD0001032
- #define TEE_ATTR_DH_SUBPRIME 0xD0001132
- #define TEE_ATTR_DH_BASE 0xD0001232
- #define TEE_ATTR_DH_X_BITS 0xF0001332
- #define TEE_ATTR_DH_PUBLIC_VALUE 0xD0000132
- #define TEE_ATTR_DH_PRIVATE_VALUE 0xC0000232
- #define TEE_ATTR_RSA_OAEP_LABEL 0xD0000930
- #define TEE_ATTR_RSA_PSS_SALT_LENGTH 0xF0000A30
- #define TEE_ATTR_ECC_PUBLIC_VALUE_X 0xD0000141

- #define TEE_ATTR_ECC_PUBLIC_VALUE_Y 0xD0000241
- #define TEE_ATTR_ECC_PRIVATE_VALUE 0xC0000341
- #define TEE_ATTR_ECC_CURVE 0xF0000441
- #define TEE_ATTR_BIT_PROTECTED (1 << 28)
- #define TEE_ATTR_BIT_VALUE (1 << 29)
- #define TEE_ECC_CURVE_NIST_P192 0x00000001
- #define TEE_ECC_CURVE_NIST_P224 0x00000002
- #define TEE_ECC_CURVE_NIST_P256 0x00000003
- #define TEE_ECC_CURVE_NIST_P384 0x00000004
- #define TEE_ECC_CURVE_NIST_P521 0x00000005
- #define TEE_PANIC_ID_TA_CLOSESESSIONENTRYPOINT 0x00000101
- #define TEE_PANIC_ID_TA_CREATEENTRYPOINT 0x00000102
- #define TEE_PANIC_ID_TA_DESTROYENTRYPOINT 0x00000103
- #define TEE_PANIC_ID_TA_INVOKECOMMANDENTRYPOINT 0x00000104
- #define TEE_PANIC_ID_TA_OPENSESSIONENTRYPOINT 0x00000105
- #define TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERATOR 0x00000201
- #define TEE_PANIC_ID_TEE_FREEPROPERTYENUMERATOR 0x00000202
- #define TEE_PANIC_ID_TEE_GETNEXTPROPERTY 0x00000203
- #define TEE_PANIC_ID_TEE_GETPROPERTYASBINARYBLOCK 0x00000204
- #define TEE_PANIC_ID_TEE_GETPROPERTYASBOOL 0x00000205
- #define TEE_PANIC_ID_TEE_GETPROPERTYASIDENTITY 0x00000206
- #define TEE_PANIC_ID_TEE_GETPROPERTYASSTRING 0x00000207
- #define TEE_PANIC_ID_TEE_GETPROPERTYASU32 0x00000208
- #define TEE_PANIC_ID_TEE_GETPROPERTYASUUID 0x00000209
- #define TEE_PANIC_ID_TEE_GETPROPERTYNAME 0x0000020A
- #define TEE_PANIC_ID_TEE_RESETPROPERTYENUMERATOR 0x0000020B
- #define TEE_PANIC_ID_TEE_STARTPROPERTYENUMERATOR 0x0000020C
- #define TEE_PANIC_ID_TEE_PANIC 0x00000301
- #define TEE_PANIC_ID_TEE_CLOSETASESSION 0x00000401
- #define TEE_PANIC_ID_TEE_INVOKETACOMMAND 0x00000402
- #define TEE_PANIC_ID_TEE_OPENTASESSION 0x00000403
- #define TEE_PANIC_ID_TEE_GETCANCELLATIONFLAG 0x00000501
- #define TEE_PANIC_ID_TEE_MASKCANCELLATION 0x00000502
- #define TEE_PANIC_ID_TEE_UNMASKCANCELLATION 0x00000503
- #define TEE_PANIC_ID_TEE_CHECKMEMORYACCESSRIGHTS 0x00000601
- #define TEE_PANIC_ID_TEE_FREE 0x00000602
- #define TEE_PANIC_ID_TEE_GETINSTANCEDATA 0x00000603
- #define TEE_PANIC_ID_TEE_MALLOC 0x00000604
- #define TEE_PANIC_ID_TEE_MEMCOMPARE 0x00000605
- #define TEE_PANIC_ID_TEE_MEMFILL 0x00000606
- #define TEE_PANIC_ID_TEE_MEMMOVE 0x00000607
- #define TEE_PANIC_ID_TEE_REALLOC 0x00000608
- #define TEE_PANIC_ID_TEE_SETINSTANCEDATA 0x00000609
- #define TEE_PANIC_ID_TEE_CLOSEOBJECT 0x00000701
- #define TEE_PANIC_ID_TEE_GETOBJECTBUFFERATTRIBUTE 0x00000702
- #define TEE_PANIC_ID_TEE_GETOBJECTINFO 0x00000703
- #define TEE_PANIC_ID_TEE_GETOBJECTVALUEATTRIBUTE 0x00000704
- #define TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE 0x00000705
- #define TEE_PANIC_ID_TEE_GETOBJECTINFO1 0x00000706
- #define TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE1 0x00000707
- #define TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT 0x00000801
- #define TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES 0x00000802
- #define TEE_PANIC_ID_TEE_FREETRANSIENTOBJECT 0x00000803
- #define TEE_PANIC_ID_TEE_GENERATEKEY 0x00000804
- #define TEE_PANIC_ID_TEE_INITREFATTRIBUTE 0x00000805

- #define TEE_PANIC_ID_TEE_INITVALUEATTRIBUTE 0x00000806
- #define TEE_PANIC_ID_TEE_POPULATETRANSIENTOBJECT 0x00000807
- #define TEE_PANIC_ID_TEE_RESETTRANSIENTOBJECT 0x00000808
- #define TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES1 0x00000809
- #define TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT 0x00000901
- #define TEE_PANIC_ID_TEE_CREATEPERSISTENTOBJECT 0x00000902
- #define TEE_PANIC_ID_TEE_OPENPERSISTENTOBJECT 0x00000903
- #define TEE_PANIC_ID_TEE_RENAMEPERSISTENTOBJECT 0x00000904
- #define TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT1 0x00000905
- #define TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR 0x00000A01
- #define TEE_PANIC_ID_TEE_FREEPERSISTENTOBJECTENUMERATOR 0x00000A02
- #define TEE_PANIC_ID_TEE_GETNEXTPERSISTENTOBJECT 0x00000A03
- #define TEE_PANIC_ID_TEE_RESETPERSISTENTOBJECTENUMERATOR 0x00000A04
- #define TEE_PANIC_ID_TEE_STARTPERSISTENTOBJECTENUMERATOR 0x00000A05
- #define TEE_PANIC_ID_TEE_READOBJECTDATA 0x00000B01
- #define TEE_PANIC_ID_TEE_SEEKOBJECTDATA 0x00000B02
- #define TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA 0x00000B03
- #define TEE_PANIC_ID_TEE_WRITEOBJECTDATA 0x00000B04
- #define TEE_PANIC_ID_TEE_ALLOCATEOPERATION 0x00000C01
- #define TEE_PANIC_ID_TEE_COPYOPERATION 0x00000C02
- #define TEE_PANIC_ID_TEE_FREEOPERATION 0x00000C03
- #define TEE_PANIC_ID_TEE_GETOPERATIONINFO 0x00000C04
- #define TEE_PANIC_ID_TEE_RESETOPERATION 0x00000C05
- #define TEE_PANIC_ID_TEE_SETOPERATIONKEY 0x00000C06
- #define TEE_PANIC_ID_TEE_SETOPERATIONKEY2 0x00000C07
- #define TEE_PANIC_ID_TEE_GETOPERATIONINFOMULTIPLE 0x00000C08
- #define TEE_PANIC_ID_TEE_DIGESTDOFINAL 0x00000D01
- #define TEE_PANIC_ID_TEE_DIGESTUPDATE 0x00000D02
- #define TEE_PANIC_ID_TEE_CIPHERDOFINAL 0x00000E01
- #define TEE_PANIC_ID_TEE_CIPHERINIT 0x00000E02
- #define TEE_PANIC_ID_TEE_CIPHERUPDATE 0x00000E03
- #define TEE_PANIC_ID_TEE_MACCOMPAREFINAL 0x00000F01
- #define TEE_PANIC_ID_TEE_MACCOMPUTEFINAL 0x00000F02
- #define TEE_PANIC_ID_TEE_MACINIT 0x00000F03
- #define TEE_PANIC_ID_TEE_MACUPDATE 0x00000F04
- #define TEE_PANIC_ID_TEE_AEDECRIPTFINAL 0x00001001
- #define TEE_PANIC_ID_TEE_AEENCRYPTFINAL 0x00001002
- #define TEE_PANIC_ID_TEE_AEINIT 0x00001003
- #define TEE_PANIC_ID_TEE_AEUPDATE 0x00001004
- #define TEE_PANIC_ID_TEE_AEUPDATEAAD 0x00001005
- #define TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT 0x00001101
- #define TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT 0x00001102
- #define TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST 0x00001103
- #define TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST 0x00001104
- #define TEE_PANIC_ID_TEE_DERIVEKEY 0x00001201
- #define TEE_PANIC_ID_TEE_GENERATERANDOM 0x00001301
- #define TEE_PANIC_ID_TEE_GETREETIME 0x00001401
- #define TEE_PANIC_ID_TEE_GETSYSTEMTIME 0x00001402
- #define TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME 0x00001403
- #define TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME 0x00001404
- #define TEE_PANIC_ID_TEE_WAIT 0x00001405
- #define TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32 0x00001501
- #define TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32 0x00001502
- #define TEE_PANIC_ID_TEE_BIGINTINIT 0x00001601
- #define TEE_PANIC_ID_TEE_BIGINTINITFMM 0x00001602

- #define [TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT](#) 0x00001603
- #define [TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING](#) 0x00001701
- #define [TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32](#) 0x00001702
- #define [TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING](#) 0x00001703
- #define [TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32](#) 0x00001704
- #define [TEE_PANIC_ID_TEE_BIGINTCMP](#) 0x00001801
- #define [TEE_PANIC_ID_TEE_BIGINTCMPS32](#) 0x00001802
- #define [TEE_PANIC_ID_TEE_BIGINTGETBIT](#) 0x00001803
- #define [TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT](#) 0x00001804
- #define [TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT](#) 0x00001805
- #define [TEE_PANIC_ID_TEE_BIGINTADD](#) 0x00001901
- #define [TEE_PANIC_ID_TEE_BIGINTDIV](#) 0x00001902
- #define [TEE_PANIC_ID_TEE_BIGINTMUL](#) 0x00001903
- #define [TEE_PANIC_ID_TEE_BIGINTNEG](#) 0x00001904
- #define [TEE_PANIC_ID_TEE_BIGINTSQUARE](#) 0x00001905
- #define [TEE_PANIC_ID_TEE_BIGINTSUB](#) 0x00001906
- #define [TEE_PANIC_ID_TEE_BIGINTADDMOD](#) 0x00001A01
- #define [TEE_PANIC_ID_TEE_BIGINTINVMOD](#) 0x00001A02
- #define [TEE_PANIC_ID_TEE_BIGINTMOD](#) 0x00001A03
- #define [TEE_PANIC_ID_TEE_BIGINTMULMOD](#) 0x00001A04
- #define [TEE_PANIC_ID_TEE_BIGINTSQUAREMOD](#) 0x00001A05
- #define [TEE_PANIC_ID_TEE_BIGINTSUBMOD](#) 0x00001A06
- #define [TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD](#) 0x00001B01
- #define [TEE_PANIC_ID_TEE_BIGINTISPROBABLEPRIME](#) 0x00001B02
- #define [TEE_PANIC_ID_TEE_BIGINTRELATIVEPRIME](#) 0x00001B03
- #define [TEE_PANIC_ID_TEE_BIGINTCOMPUTEFMM](#) 0x00001C01
- #define [TEE_PANIC_ID_TEE_BIGINTCONVERTFROMFMM](#) 0x00001C02
- #define [TEE_PANIC_ID_TEE_BIGINTCONVERTTOFMM](#) 0x00001C03
- #define [TEE_PARAM_TYPES](#)(t0, t1, t2, t3) (((t0) | ((t1) << 4) | ((t2) << 8) | ((t3) << 12))
- #define [TEE_PARAM_TYPE_GET](#)(t, i) (((uint32_t)t) >> ((i)*4)) & 0xF
- #define [TEE_PARAM_TYPE_SET](#)(t, i) (((uint32_t)t) & 0xF) << ((i)*4)
- #define [TEE_NUM_PARAMS](#) 4
- #define [TEE_BigIntSizeInU32](#)(n) (((n)+31)/32)+2

10.9.1 Macro Definition Documentation

10.9.1.1 TEE_ALG_AES_CBC_MAC_NOPAD #define TEE_ALG_AES_CBC_MAC_NOPAD 0x30000110

10.9.1.2 TEE_ALG_AES_CBC_MAC_PKCS5 #define TEE_ALG_AES_CBC_MAC_PKCS5 0x30000510

10.9.1.3 TEE_ALG_AES_CBC_NOPAD #define TEE_ALG_AES_CBC_NOPAD 0x10000110

10.9.1.4 TEE_ALG_AES_CCM #define TEE_ALG_AES_CCM 0x40000710

10.9.1.5 TEE_ALG_AES_CMAC #define TEE_ALG_AES_CMAC 0x30000610

10.9.1.6 TEE_ALG_AES_CTR #define TEE_ALG_AES_CTR 0x10000210

10.9.1.7 TEE_ALG_AES_CTS #define TEE_ALG_AES_CTS 0x10000310

10.9.1.8 TEE_ALG_AES_ECB_NOPAD #define TEE_ALG_AES_ECB_NOPAD 0x10000010

10.9.1.9 TEE_ALG_AES_GCM #define TEE_ALG_AES_GCM 0x40000810

10.9.1.10 TEE_ALG_AES_XTS #define TEE_ALG_AES_XTS 0x10000410

10.9.1.11 TEE_ALG_DES3_CBC_MAC_NOPAD #define TEE_ALG_DES3_CBC_MAC_NOPAD 0x30000113

10.9.1.12 TEE_ALG_DES3_CBC_MAC_PKCS5 #define TEE_ALG_DES3_CBC_MAC_PKCS5 0x30000513

10.9.1.13 TEE_ALG_DES3_CBC_NOPAD #define TEE_ALG_DES3_CBC_NOPAD 0x10000113

10.9.1.14 TEE_ALG_DES3_ECB_NOPAD #define TEE_ALG_DES3_ECB_NOPAD 0x10000013

10.9.1.15 TEE_ALG_DES_CBC_MAC_NOPAD #define TEE_ALG_DES_CBC_MAC_NOPAD 0x30000111

10.9.1.16 TEE_ALG_DES_CBC_MAC_PKCS5 #define TEE_ALG_DES_CBC_MAC_PKCS5 0x30000511

10.9.1.17 TEE_ALG_DES_CBC_NOPAD #define TEE_ALG_DES_CBC_NOPAD 0x10000111

10.9.1.18 TEE_ALG_DES_ECB_NOPAD #define TEE_ALG_DES_ECB_NOPAD 0x10000011

10.9.1.19 TEE_ALG_DH_DERIVE_SHARED_SECRET #define
TEE_ALG_DH_DERIVE_SHARED_SECRET 0x80000032

10.9.1.20 TEE_ALG_DSA_SHA1 #define TEE_ALG_DSA_SHA1 0x70002131

10.9.1.21 TEE_ALG_DSA_SHA224 #define TEE_ALG_DSA_SHA224 0x70003131

10.9.1.22 TEE_ALG_DSA_SHA256 #define TEE_ALG_DSA_SHA256 0x70004131

10.9.1.23 TEE_ALG_ECDH_P192 #define TEE_ALG_ECDH_P192 0x80001042

10.9.1.24 TEE_ALG_ECDH_P224 #define TEE_ALG_ECDH_P224 0x80002042

10.9.1.25 TEE_ALG_ECDH_P256 #define TEE_ALG_ECDH_P256 0x80003042

10.9.1.26 TEE_ALG_ECDH_P384 #define TEE_ALG_ECDH_P384 0x80004042

10.9.1.27 TEE_ALG_ECDH_P521 #define TEE_ALG_ECDH_P521 0x80005042

10.9.1.28 TEE_ALG_ECDSA_P192 #define TEE_ALG_ECDSA_P192 0x70001041

10.9.1.29 TEE_ALG_ECDSA_P224 #define TEE_ALG_ECDSA_P224 0x70002041

10.9.1.30 TEE_ALG_ECDSA_P256 #define TEE_ALG_ECDSA_P256 0x70003041

10.9.1.31 TEE_ALG_ECDSA_P384 #define TEE_ALG_ECDSA_P384 0x70004041

10.9.1.32 TEE_ALG_ECDSA_P521 #define TEE_ALG_ECDSA_P521 0x70005041

10.9.1.33 TEE_ALG_HMAC_MD5 #define TEE_ALG_HMAC_MD5 0x30000001

10.9.1.34 TEE_ALG_HMAC_SHA1 #define TEE_ALG_HMAC_SHA1 0x30000002

10.9.1.35 TEE_ALG_HMAC_SHA224 #define TEE_ALG_HMAC_SHA224 0x30000003

10.9.1.36 TEE_ALG_HMAC_SHA256 #define TEE_ALG_HMAC_SHA256 0x30000004

10.9.1.37 TEE_ALG_HMAC_SHA384 #define TEE_ALG_HMAC_SHA384 0x30000005

10.9.1.38 TEE_ALG_HMAC_SHA512 #define TEE_ALG_HMAC_SHA512 0x30000006

10.9.1.39 TEE_ALG_MD5 #define TEE_ALG_MD5 0x50000001

10.9.1.40 TEE_ALG_MD5SHA1 #define TEE_ALG_MD5SHA1 0x5000000F

10.9.1.41 TEE_ALG_RSA_NOPAD #define TEE_ALG_RSA_NOPAD 0x60000030

10.9.1.42 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1 #define
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1 0x60210230

10.9.1.43 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224 #define
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224 0x60310230

10.9.1.44 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256 #define
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256 0x60410230

10.9.1.45 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384 #define
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384 0x60510230

10.9.1.46 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512 #define
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512 0x60610230

10.9.1.47 TEE_ALG_RSAES_PKCS1_V1_5 #define TEE_ALG_RSAES_PKCS1_V1_5 0x60000130

10.9.1.48 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1 #define
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1 0x70212930

10.9.1.49 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224 #define
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224 0x70313930

10.9.1.50 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256 #define
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256 0x70414930

10.9.1.51 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384 #define
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384 0x70515930

10.9.1.52 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512 #define
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512 0x70616930

10.9.1.53 TEE_ALG_RSASSA_PKCS1_V1_5_MD5 #define TEE_ALG_RSASSA_PKCS1_V1_5_MD5 0x70001830

10.9.1.54 TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1 #define
TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1 0x7000F830

10.9.1.55 TEE_ALG_RSASSA_PKCS1_V1_5_SHA1 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA1 0x70002830

10.9.1.56 TEE_ALG_RSASSA_PKCS1_V1_5_SHA224 #define
TEE_ALG_RSASSA_PKCS1_V1_5_SHA224 0x70003830

10.9.1.57 TEE_ALG_RSASSA_PKCS1_V1_5_SHA256 #define
TEE_ALG_RSASSA_PKCS1_V1_5_SHA256 0x70004830

10.9.1.58 TEE_ALG_RSASSA_PKCS1_V1_5_SHA384 #define
TEE_ALG_RSASSA_PKCS1_V1_5_SHA384 0x70005830

10.9.1.59 TEE_ALG_RSASSA_PKCS1_V1_5_SHA512 #define
TEE_ALG_RSASSA_PKCS1_V1_5_SHA512 0x70006830

10.9.1.60 TEE_ALG_SHA1 #define TEE_ALG_SHA1 0x50000002

10.9.1.61 TEE_ALG_SHA224 #define TEE_ALG_SHA224 0x50000003

10.9.1.62 TEE_ALG_SHA256 #define TEE_ALG_SHA256 0x50000004

10.9.1.63 TEE_ALG_SHA384 #define TEE_ALG_SHA384 0x50000005

10.9.1.64 TEE_ALG_SHA512 #define TEE_ALG_SHA512 0x50000006

10.9.1.65 TEE_ATTR_BIT_PROTECTED #define TEE_ATTR_BIT_PROTECTED (1 << 28)

10.9.1.66 TEE_ATTR_BIT_VALUE #define TEE_ATTR_BIT_VALUE (1 << 29)

10.9.1.67 TEE_ATTR_DH_BASE #define TEE_ATTR_DH_BASE 0xD0001232

10.9.1.68 TEE_ATTR_DH_PRIME #define TEE_ATTR_DH_PRIME 0xD0001032

10.9.1.69 TEE_ATTR_DH_PRIVATE_VALUE #define TEE_ATTR_DH_PRIVATE_VALUE 0xC0000232

10.9.1.70 TEE_ATTR_DH_PUBLIC_VALUE #define TEE_ATTR_DH_PUBLIC_VALUE 0xD0000132

10.9.1.71 TEE_ATTR_DH_SUBPRIME #define TEE_ATTR_DH_SUBPRIME 0xD0001132

10.9.1.72 TEE_ATTR_DH_X_BITS #define TEE_ATTR_DH_X_BITS 0xF0001332

10.9.1.73 TEE_ATTR_DSA_BASE #define TEE_ATTR_DSA_BASE 0xD0001231

10.9.1.74 TEE_ATTR_DSA_PRIME #define TEE_ATTR_DSA_PRIME 0xD0001031

10.9.1.75 TEE_ATTR_DSA_PRIVATE_VALUE #define TEE_ATTR_DSA_PRIVATE_VALUE 0xC0000231

10.9.1.76 TEE_ATTR_DSA_PUBLIC_VALUE #define TEE_ATTR_DSA_PUBLIC_VALUE 0xD0000131

10.9.1.77 TEE_ATTR_DSA_SUBPRIME #define TEE_ATTR_DSA_SUBPRIME 0xD0001131

10.9.1.78 TEE_ATTR_ECC_CURVE #define TEE_ATTR_ECC_CURVE 0xF0000441

10.9.1.79 TEE_ATTR_ECC_PRIVATE_VALUE #define TEE_ATTR_ECC_PRIVATE_VALUE 0xC0000341

10.9.1.80 TEE_ATTR_ECC_PUBLIC_VALUE_X #define TEE_ATTR_ECC_PUBLIC_VALUE_X 0xD0000141

10.9.1.81 TEE_ATTR_ECC_PUBLIC_VALUE_Y #define TEE_ATTR_ECC_PUBLIC_VALUE_Y 0xD0000241

10.9.1.82 TEE_ATTR_RSA_COEFFICIENT #define TEE_ATTR_RSA_COEFFICIENT 0xC0000830

10.9.1.83 TEE_ATTR_RSA_EXPONENT1 #define TEE_ATTR_RSA_EXPONENT1 0xC0000630

10.9.1.84 TEE_ATTR_RSA_EXPONENT2 #define TEE_ATTR_RSA_EXPONENT2 0xC0000730

10.9.1.85 TEE_ATTR_RSA_MODULUS #define TEE_ATTR_RSA_MODULUS 0xD0000130

10.9.1.86 TEE_ATTR_RSA_OAEP_LABEL #define TEE_ATTR_RSA_OAEP_LABEL 0xD0000930

10.9.1.87 TEE_ATTR_RSA_PRIME1 #define TEE_ATTR_RSA_PRIME1 0xC0000430

10.9.1.88 TEE_ATTR_RSA_PRIME2 #define TEE_ATTR_RSA_PRIME2 0xC0000530

10.9.1.89 TEE_ATTR_RSA_PRIVATE_EXPONENT #define TEE_ATTR_RSA_PRIVATE_EXPONENT 0xC0000330

10.9.1.90 TEE_ATTR_RSA_PSS_SALT_LENGTH #define TEE_ATTR_RSA_PSS_SALT_LENGTH 0xF0000A30

10.9.1.91 TEE_ATTR_RSA_PUBLIC_EXPONENT #define TEE_ATTR_RSA_PUBLIC_EXPONENT 0xD0000230

10.9.1.92 TEE_ATTR_SECRET_VALUE #define TEE_ATTR_SECRET_VALUE 0xC0000000

10.9.1.93 TEE_BigIntSizeInU32 #define TEE_BigIntSizeInU32(
 n) (((n)+31)/32)+2)

10.9.1.94 TEE_DATA_FLAG_ACCESS_READ #define TEE_DATA_FLAG_ACCESS_READ 0x00000001

10.9.1.95 TEE_DATA_FLAG_ACCESS_WRITE #define TEE_DATA_FLAG_ACCESS_WRITE 0x00000002

10.9.1.96 TEE_DATA_FLAG_ACCESS_WRITE_META #define
TEE_DATA_FLAG_ACCESS_WRITE_META 0x00000004

10.9.1.97 TEE_DATA_FLAG_OVERWRITE #define TEE_DATA_FLAG_OVERWRITE 0x00000400

10.9.1.98 TEE_DATA_FLAG_SHARE_READ #define TEE_DATA_FLAG_SHARE_READ 0x00000010

10.9.1.99 TEE_DATA_FLAG_SHARE_WRITE #define TEE_DATA_FLAG_SHARE_WRITE 0x00000020

10.9.1.100 TEE_DATA_MAX_POSITION #define TEE_DATA_MAX_POSITION 0xFFFFFFFF

10.9.1.101 TEE_ECC_CURVE_NIST_P192 #define TEE_ECC_CURVE_NIST_P192 0x00000001

10.9.1.102 TEE_ECC_CURVE_NIST_P224 #define TEE_ECC_CURVE_NIST_P224 0x00000002

10.9.1.103 TEE_ECC_CURVE_NIST_P256 #define TEE_ECC_CURVE_NIST_P256 0x00000003

10.9.1.104 TEE_ECC_CURVE_NIST_P384 #define TEE_ECC_CURVE_NIST_P384 0x00000004

10.9.1.105 TEE_ECC_CURVE_NIST_P521 #define TEE_ECC_CURVE_NIST_P521 0x00000005

10.9.1.106 TEE_ERROR_ACCESS_CONFLICT #define TEE_ERROR_ACCESS_CONFLICT 0xFFFF0003

10.9.1.107 TEE_ERROR_ACCESS_DENIED #define TEE_ERROR_ACCESS_DENIED 0xFFFF0001

10.9.1.108 TEE_ERROR_BAD_FORMAT #define TEE_ERROR_BAD_FORMAT 0xFFFF0005

10.9.1.109 TEE_ERROR_BAD_PARAMETERS #define TEE_ERROR_BAD_PARAMETERS 0xFFFF0006

10.9.1.110 TEE_ERROR_BAD_STATE #define TEE_ERROR_BAD_STATE 0xFFFF0007

10.9.1.111 TEE_ERROR_BUSY #define TEE_ERROR_BUSY 0xFFFF000D

10.9.1.112 TEE_ERROR_CANCEL #define TEE_ERROR_CANCEL 0xFFFF0002

10.9.1.113 TEE_ERROR_COMMUNICATION #define TEE_ERROR_COMMUNICATION 0xFFFF000E

10.9.1.114 TEE_ERROR_CORRUPT_OBJECT #define TEE_ERROR_CORRUPT_OBJECT 0xF0100001

10.9.1.115 TEE_ERROR_CORRUPT_OBJECT_2 #define TEE_ERROR_CORRUPT_OBJECT_2 0xF0100002

10.9.1.116 TEE_ERROR_EXCESS_DATA #define TEE_ERROR_EXCESS_DATA 0xFFFF0004

10.9.1.117 TEE_ERROR_EXTERNAL_CANCEL #define TEE_ERROR_EXTERNAL_CANCEL 0xFFFF0011

10.9.1.118 TEE_ERROR_GENERIC #define TEE_ERROR_GENERIC 0xFFFF0000

10.9.1.119 TEE_ERROR_ITEM_NOT_FOUND #define TEE_ERROR_ITEM_NOT_FOUND 0xFFFF0008

10.9.1.120 TEE_ERROR_MAC_INVALID #define TEE_ERROR_MAC_INVALID 0xFFFF3071

10.9.1.121 TEE_ERROR_NO_DATA #define TEE_ERROR_NO_DATA 0xFFFF000B

10.9.1.122 TEE_ERROR_NOT_IMPLEMENTED #define TEE_ERROR_NOT_IMPLEMENTED 0xFFFF0009

10.9.1.123 TEE_ERROR_NOT_SUPPORTED #define TEE_ERROR_NOT_SUPPORTED 0xFFFF000A

10.9.1.124 TEE_ERROR_OUT_OF_MEMORY #define TEE_ERROR_OUT_OF_MEMORY 0xFFFF000C

10.9.1.125 TEE_ERROR_OVERFLOW #define TEE_ERROR_OVERFLOW 0xFFFF300F

10.9.1.126 TEE_ERROR_SECURITY #define TEE_ERROR_SECURITY 0xFFFF000F

10.9.1.127 TEE_ERROR_SHORT_BUFFER #define TEE_ERROR_SHORT_BUFFER 0xFFFF0010

10.9.1.128 TEE_ERROR_SIGNATURE_INVALID #define TEE_ERROR_SIGNATURE_INVALID 0xFFFF3072

10.9.1.129 TEE_ERROR_STORAGE_NO_SPACE #define TEE_ERROR_STORAGE_NO_SPACE 0xFFFF3041

10.9.1.130 TEE_ERROR_STORAGE_NOT_AVAILABLE #define
TEE_ERROR_STORAGE_NOT_AVAILABLE 0xF0100003

10.9.1.131 TEE_ERROR_STORAGE_NOT_AVAILABLE 2 #define
TEE_ERROR_STORAGE_NOT_AVAILABLE_2 0xF0100004

10.9.1.132 TEE_ERROR_TARGET_DEAD #define TEE_ERROR_TARGET_DEAD 0xFFFF3024

10.9.1.133 TEE_ERROR_TIME_NEEDS_RESET #define TEE_ERROR_TIME_NEEDS_RESET 0xFFFF5001

10.9.1.134 TEE_ERROR_TIME_NOT_SET #define TEE_ERROR_TIME_NOT_SET 0xFFFF5000

10.9.1.135 TEE_HANDLE_FLAG_EXPECT_TWO_KEYS #define
TEE_HANDLE_FLAG_EXPECT_TWO_KEYS 0x00080000

10.9.1.136 TEE_HANDLE_FLAG_INITIALIZED #define TEE_HANDLE_FLAG_INITIALIZED 0x00020000

10.9.1.137 TEE_HANDLE_FLAG_KEY_SET #define TEE_HANDLE_FLAG_KEY_SET 0x00040000

10.9.1.138 TEE_HANDLE_FLAG_PERSISTENT #define TEE_HANDLE_FLAG_PERSISTENT 0x00010000

10.9.1.139 TEE_HANDLE_NULL #define TEE_HANDLE_NULL 0

10.9.1.140 TEE_INT_CORE_API_SPEC_VERSION #define TEE_INT_CORE_API_SPEC_VERSION 0x0000000A

10.9.1.141 TEE_LOGIN_APPLICATION #define TEE_LOGIN_APPLICATION 0x00000004

10.9.1.142 TEE_LOGIN_APPLICATION_GROUP #define TEE_LOGIN_APPLICATION_GROUP 0x00000006

10.9.1.143 TEE_LOGIN_APPLICATION_USER #define TEE_LOGIN_APPLICATION_USER 0x00000005

10.9.1.144 TEE_LOGIN_GROUP #define TEE_LOGIN_GROUP 0x00000002

10.9.1.145 TEE_LOGIN_PUBLIC `#define TEE_LOGIN_PUBLIC 0x00000000`

10.9.1.146 TEE_LOGIN_TRUSTED_APP `#define TEE_LOGIN_TRUSTED_APP 0xF0000000`

10.9.1.147 TEE_LOGIN_USER `#define TEE_LOGIN_USER 0x00000001`

10.9.1.148 TEE_MALLOC_FILL_ZERO `#define TEE_MALLOC_FILL_ZERO 0x00000000`

10.9.1.149 TEE_MEMORY_ACCESS_ANY_OWNER `#define TEE_MEMORY_ACCESS_ANY_OWNER 0x00000004`

10.9.1.150 TEE_MEMORY_ACCESS_READ `#define TEE_MEMORY_ACCESS_READ 0x00000001`

10.9.1.151 TEE_MEMORY_ACCESS_WRITE `#define TEE_MEMORY_ACCESS_WRITE 0x00000002`

10.9.1.152 TEE_NUM_PARAMS `#define TEE_NUM_PARAMS 4`

10.9.1.153 TEE_OBJECT_ID_MAX_LEN `#define TEE_OBJECT_ID_MAX_LEN 64`

10.9.1.154 TEE_OPERATION_AE `#define TEE_OPERATION_AE 4`

10.9.1.155 TEE_OPERATION_ASYMMETRIC_CIPHER `#define TEE_OPERATION_ASYMMETRIC_CIPHER 6`

10.9.1.156 TEE_OPERATION_ASYMMETRIC_SIGNATURE #define
TEE_OPERATION_ASYMMETRIC_SIGNATURE 7

10.9.1.157 TEE_OPERATION_CIPHER #define TEE_OPERATION_CIPHER 1

10.9.1.158 TEE_OPERATION_DIGEST #define TEE_OPERATION_DIGEST 5

10.9.1.159 TEE_OPERATION_KEY_DERIVATION #define TEE_OPERATION_KEY_DERIVATION 8

10.9.1.160 TEE_OPERATION_MAC #define TEE_OPERATION_MAC 3

10.9.1.161 TEE_OPERATION_STATE_ACTIVE #define TEE_OPERATION_STATE_ACTIVE 0x00000001

10.9.1.162 TEE_OPERATION_STATE_INITIAL #define TEE_OPERATION_STATE_INITIAL 0x00000000

10.9.1.163 TEE_ORIGIN_API #define TEE_ORIGIN_API 0x00000001

10.9.1.164 TEE_ORIGIN_COMMS #define TEE_ORIGIN_COMMS 0x00000002

10.9.1.165 TEE_ORIGIN_TEE #define TEE_ORIGIN_TEE 0x00000003

10.9.1.166 TEE_ORIGIN_TRUSTED_APP #define TEE_ORIGIN_TRUSTED_APP 0x00000004

10.9.1.167 TEE_PANIC_ID_TA_CLOSESESSIONENTRYPOINT #define
TEE_PANIC_ID_TA_CLOSESESSIONENTRYPOINT 0x00000101

10.9.1.168 TEE_PANIC_ID_TA_CREATEENTRYPOINT #define
TEE_PANIC_ID_TA_CREATEENTRYPOINT 0x00000102

10.9.1.169 TEE_PANIC_ID_TA_DESTROYENTRYPOINT #define
TEE_PANIC_ID_TA_DESTROYENTRYPOINT 0x00000103

10.9.1.170 TEE_PANIC_ID_TA_INVOKECOMMANDENTRYPOINT #define
TEE_PANIC_ID_TA_INVOKECOMMANDENTRYPOINT 0x00000104

10.9.1.171 TEE_PANIC_ID_TA_OPENSESSIONENTRYPOINT #define
TEE_PANIC_ID_TA_OPENSESSIONENTRYPOINT 0x00000105

10.9.1.172 TEE_PANIC_ID_TEE_AEDECRIPTFINAL #define
TEE_PANIC_ID_TEE_AEDECRIPTFINAL 0x00001001

10.9.1.173 TEE_PANIC_ID_TEE_AEENCRYPTFINAL #define
TEE_PANIC_ID_TEE_AEENCRYPTFINAL 0x00001002

10.9.1.174 TEE_PANIC_ID_TEE_AEINIT #define TEE_PANIC_ID_TEE_AEINIT 0x00001003

10.9.1.175 TEE_PANIC_ID_TEE_AEUPDATE #define TEE_PANIC_ID_TEE_AEUPDATE 0x00001004

10.9.1.176 TEE_PANIC_ID_TEE_AEUPDATEAAD #define TEE_PANIC_ID_TEE_AEUPDATEAAD 0x00001005

10.9.1.177 TEE_PANIC_ID_TEE_ALLOCATEOPERATION #define
TEE_PANIC_ID_TEE_ALLOCATEOPERATION 0x00000C01

10.9.1.178 TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR #define
TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR 0x00000A01

10.9.1.179 TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERATOR #define
TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERATOR 0x00000201

10.9.1.180 TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT #define
TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT 0x00000801

10.9.1.181 TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT #define
TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT 0x00001101

10.9.1.182 TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT #define
TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT 0x00001102

10.9.1.183 TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST #define
TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST 0x00001103

10.9.1.184 TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST #define
TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST 0x00001104

10.9.1.185 TEE_PANIC_ID_TEE_BIGINTADD #define TEE_PANIC_ID_TEE_BIGINTADD 0x00001901

10.9.1.186 TEE_PANIC_ID_TEE_BIGINTADDMOD #define TEE_PANIC_ID_TEE_BIGINTADDMOD 0x00001A01

10.9.1.187 TEE_PANIC_ID_TEE_BIGINTCMP #define TEE_PANIC_ID_TEE_BIGINTCMP 0x00001801

10.9.1.188 TEE_PANIC_ID_TEE_BIGINTCMPS32 #define TEE_PANIC_ID_TEE_BIGINTCMPS32 0x00001802

10.9.1.189 TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD #define
TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD 0x00001B01

10.9.1.190 TEE_PANIC_ID_TEE_BIGINTCOMPUTEFMM #define
TEE_PANIC_ID_TEE_BIGINTCOMPUTEFMM 0x00001C01

10.9.1.191 TEE_PANIC_ID_TEE_BIGINTCONVERTFROMFMM #define
TEE_PANIC_ID_TEE_BIGINTCONVERTFROMFMM 0x00001C02

10.9.1.192 TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING #define
TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING 0x00001701

10.9.1.193 TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32 #define
TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32 0x00001702

10.9.1.194 TEE_PANIC_ID_TEE_BIGINTCONVERTTOFMM #define
TEE_PANIC_ID_TEE_BIGINTCONVERTTOFMM 0x00001C03

10.9.1.195 TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING #define
TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING 0x00001703

10.9.1.196 TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32 #define
TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32 0x00001704

10.9.1.197 TEE_PANIC_ID_TEE_BIGINTDIV #define TEE_PANIC_ID_TEE_BIGINTDIV 0x00001902

10.9.1.198 TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32 #define
TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32 0x00001501

10.9.1.199 TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32 #define
TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32 0x00001502

10.9.1.200 TEE_PANIC_ID_TEE_BIGINTGETBIT #define TEE_PANIC_ID_TEE_BIGINTGETBIT 0x00001803

10.9.1.201 TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT #define
TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT 0x00001804

10.9.1.202 TEE_PANIC_ID_TEE_BIGINTINIT #define TEE_PANIC_ID_TEE_BIGINTINIT 0x00001601

10.9.1.203 TEE_PANIC_ID_TEE_BIGINTINITFMM #define TEE_PANIC_ID_TEE_BIGINTINITFMM 0x00001602

10.9.1.204 TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT #define
TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT 0x00001603

10.9.1.205 TEE_PANIC_ID_TEE_BIGINTINVMOD #define TEE_PANIC_ID_TEE_BIGINTINVMOD 0x00001A02

10.9.1.206 TEE_PANIC_ID_TEE_BIGINTISPROBABLEPRIME #define
TEE_PANIC_ID_TEE_BIGINTISPROBABLEPRIME 0x00001B02

10.9.1.207 TEE_PANIC_ID_TEE_BIGINTMOD #define TEE_PANIC_ID_TEE_BIGINTMOD 0x00001A03

10.9.1.208 TEE_PANIC_ID_TEE_BIGINTMUL #define TEE_PANIC_ID_TEE_BIGINTMUL 0x00001903

10.9.1.209 TEE_PANIC_ID_TEE_BIGINTMULMOD #define TEE_PANIC_ID_TEE_BIGINTMULMOD 0x00001A04

10.9.1.210 TEE_PANIC_ID_TEE_BIGINTNEG #define TEE_PANIC_ID_TEE_BIGINTNEG 0x00001904

10.9.1.211 TEE_PANIC_ID_TEE_BIGINTRELATIVEPRIME #define
TEE_PANIC_ID_TEE_BIGINTRELATIVEPRIME 0x00001B03

10.9.1.212 TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT #define
TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT 0x00001805

10.9.1.213 TEE_PANIC_ID_TEE_BIGINTSQUARE #define TEE_PANIC_ID_TEE_BIGINTSQUARE 0x00001905

10.9.1.214 TEE_PANIC_ID_TEE_BIGINTSQUAREMOD #define
TEE_PANIC_ID_TEE_BIGINTSQUAREMOD 0x00001A05

10.9.1.215 TEE_PANIC_ID_TEE_BIGINTSUB #define TEE_PANIC_ID_TEE_BIGINTSUB 0x00001906

10.9.1.216 TEE_PANIC_ID_TEE_BIGINTSUBMOD #define TEE_PANIC_ID_TEE_BIGINTSUBMOD 0x00001A06

10.9.1.217 TEE_PANIC_ID_TEE_CHECKMEMORYACCESSRIGHTS #define
TEE_PANIC_ID_TEE_CHECKMEMORYACCESSRIGHTS 0x00000601

10.9.1.218 TEE_PANIC_ID_TEE_CIPHERDOFINAL #define TEE_PANIC_ID_TEE_CIPHERDOFINAL 0x00000E01

10.9.1.219 TEE_PANIC_ID_TEE_CIPHERINIT #define TEE_PANIC_ID_TEE_CIPHERINIT 0x00000E02

10.9.1.220 TEE_PANIC_ID_TEE_CIPHERUPDATE #define TEE_PANIC_ID_TEE_CIPHERUPDATE 0x00000E03

10.9.1.221 TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT #define
TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT 0x00000901

10.9.1.222 TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT1 #define
TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT1 0x00000905

10.9.1.223 TEE_PANIC_ID_TEE_CLOSEOBJECT #define TEE_PANIC_ID_TEE_CLOSEOBJECT 0x00000701

10.9.1.224 TEE_PANIC_ID_TEE_CLOSETASESSION #define
TEE_PANIC_ID_TEE_CLOSETASESSION 0x00000401

10.9.1.225 TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES #define
TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES 0x00000802

10.9.1.226 TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES1 #define
TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES1 0x00000809

10.9.1.227 TEE_PANIC_ID_TEE_COPYOPERATION #define TEE_PANIC_ID_TEE_COPYOPERATION 0x00000C02

10.9.1.228 TEE_PANIC_ID_TEE_CREATEPERSISTENTOBJECT #define
TEE_PANIC_ID_TEE_CREATEPERSISTENTOBJECT 0x00000902

10.9.1.229 TEE_PANIC_ID_TEE_DERIVEKEY #define TEE_PANIC_ID_TEE_DERIVEKEY 0x00001201

10.9.1.230 TEE_PANIC_ID_TEE_DIGESTDOFINAL #define TEE_PANIC_ID_TEE_DIGESTDOFINAL 0x00000D01

10.9.1.231 TEE_PANIC_ID_TEE_DIGESTUPDATE #define TEE_PANIC_ID_TEE_DIGESTUPDATE 0x00000D02

10.9.1.232 TEE_PANIC_ID_TEE_FREE #define TEE_PANIC_ID_TEE_FREE 0x00000602

10.9.1.233 TEE_PANIC_ID_TEE_FREEOPERATION #define TEE_PANIC_ID_TEE_FREEOPERATION 0x00000C03

10.9.1.234 TEE_PANIC_ID_TEE_FREEPERSISTENTOBJECTENUMERATOR #define
TEE_PANIC_ID_TEE_FREEPERSISTENTOBJECTENUMERATOR 0x00000A02

10.9.1.235 TEE_PANIC_ID_TEE_FREEPROPERTYENUMERATOR #define
TEE_PANIC_ID_TEE_FREEPROPERTYENUMERATOR 0x00000202

10.9.1.236 TEE_PANIC_ID_TEE_FREETRANSIENTOBJECT #define
TEE_PANIC_ID_TEE_FREETRANSIENTOBJECT 0x00000803

10.9.1.237 TEE_PANIC_ID_TEE_GENERATEKEY #define TEE_PANIC_ID_TEE_GENERATEKEY 0x00000804

10.9.1.238 TEE_PANIC_ID_TEE_GENERATERANDOM #define
TEE_PANIC_ID_TEE_GENERATERANDOM 0x00001301

10.9.1.239 TEE_PANIC_ID_TEE_GETCANCELLATIONFLAG #define
TEE_PANIC_ID_TEE_GETCANCELLATIONFLAG 0x00000501

10.9.1.240 TEE_PANIC_ID_TEE_GETINSTANCEDATA #define
TEE_PANIC_ID_TEE_GETINSTANCEDATA 0x00000603

10.9.1.241 TEE_PANIC_ID_TEE_GETNEXTPERSISTENTOBJECT #define
TEE_PANIC_ID_TEE_GETNEXTPERSISTENTOBJECT 0x00000A03

10.9.1.242 TEE_PANIC_ID_TEE_GETNEXTPROPERTY #define
TEE_PANIC_ID_TEE_GETNEXTPROPERTY 0x00000203

10.9.1.243 TEE_PANIC_ID_TEE_GETOBJECTBUFFERATTRIBUTE #define
TEE_PANIC_ID_TEE_GETOBJECTBUFFERATTRIBUTE 0x00000702

10.9.1.244 TEE_PANIC_ID_TEE_GETOBJECTINFO #define TEE_PANIC_ID_TEE_GETOBJECTINFO 0x00000703

10.9.1.245 TEE_PANIC_ID_TEE_GETOBJECTINFO1 #define
TEE_PANIC_ID_TEE_GETOBJECTINFO1 0x00000706

10.9.1.246 TEE_PANIC_ID_TEE_GETOBJECTVALUEATTRIBUTE #define
TEE_PANIC_ID_TEE_GETOBJECTVALUEATTRIBUTE 0x00000704

10.9.1.247 TEE_PANIC_ID_TEE_GETOPERATIONINFO #define
TEE_PANIC_ID_TEE_GETOPERATIONINFO 0x00000C04

10.9.1.248 TEE_PANIC_ID_TEE_GETOPERATIONINFOMULTIPLE #define
TEE_PANIC_ID_TEE_GETOPERATIONINFOMULTIPLE 0x00000C08

10.9.1.249 TEE_PANIC_ID_TEE_GETPROPERTYASBINARYBLOCK #define
TEE_PANIC_ID_TEE_GETPROPERTYASBINARYBLOCK 0x00000204

10.9.1.250 TEE_PANIC_ID_TEE_GETPROPERTYASBOOL #define
TEE_PANIC_ID_TEE_GETPROPERTYASBOOL 0x00000205

10.9.1.251 TEE_PANIC_ID_TEE_GETPROPERTYASIDENTITY #define
TEE_PANIC_ID_TEE_GETPROPERTYASIDENTITY 0x00000206

10.9.1.252 TEE_PANIC_ID_TEE_GETPROPERTYASSTRING #define
TEE_PANIC_ID_TEE_GETPROPERTYASSTRING 0x00000207

10.9.1.253 TEE_PANIC_ID_TEE_GETPROPERTYASU32 #define
TEE_PANIC_ID_TEE_GETPROPERTYASU32 0x00000208

10.9.1.254 TEE_PANIC_ID_TEE_GETPROPERTYASUUID #define
TEE_PANIC_ID_TEE_GETPROPERTYASUUID 0x00000209

10.9.1.255 TEE_PANIC_ID_TEE_GETPROPERTYNAME #define
TEE_PANIC_ID_TEE_GETPROPERTYNAME 0x0000020A

10.9.1.256 TEE_PANIC_ID_TEE_GETREETIME #define TEE_PANIC_ID_TEE_GETREETIME 0x00001401

10.9.1.257 TEE_PANIC_ID_TEE_GETSYSTEMTIME #define TEE_PANIC_ID_TEE_GETSYSTEMTIME 0x00001402

10.9.1.258 TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME #define
TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME 0x00001403

10.9.1.259 TEE_PANIC_ID_TEE_INITREFATTRIBUTE #define
TEE_PANIC_ID_TEE_INITREFATTRIBUTE 0x00000805

10.9.1.260 TEE_PANIC_ID_TEE_INITVALUEATTRIBUTE #define
TEE_PANIC_ID_TEE_INITVALUEATTRIBUTE 0x00000806

10.9.1.261 TEE_PANIC_ID_TEE_INVOKETACOMMAND #define
TEE_PANIC_ID_TEE_INVOKETACOMMAND 0x00000402

10.9.1.262 TEE_PANIC_ID_TEE_MACCOMPAREFINAL #define
TEE_PANIC_ID_TEE_MACCOMPAREFINAL 0x00000F01

10.9.1.263 TEE_PANIC_ID_TEE_MACCOMPUTEFINAL #define
TEE_PANIC_ID_TEE_MACCOMPUTEFINAL 0x00000F02

10.9.1.264 TEE_PANIC_ID_TEE_MACINIT #define TEE_PANIC_ID_TEE_MACINIT 0x00000F03

10.9.1.265 TEE_PANIC_ID_TEE_MACUPDATE #define TEE_PANIC_ID_TEE_MACUPDATE 0x00000F04

10.9.1.266 TEE_PANIC_ID_TEE_MALLOC #define TEE_PANIC_ID_TEE_MALLOC 0x00000604

10.9.1.267 TEE_PANIC_ID_TEE_MASKANCELLATION #define
TEE_PANIC_ID_TEE_MASKANCELLATION 0x00000502

10.9.1.268 TEE_PANIC_ID_TEE_MEMCOMPARE #define TEE_PANIC_ID_TEE_MEMCOMPARE 0x00000605

10.9.1.269 TEE_PANIC_ID_TEE_MEMFILL #define TEE_PANIC_ID_TEE_MEMFILL 0x00000606

10.9.1.270 TEE_PANIC_ID_TEE_MEMMOVE #define TEE_PANIC_ID_TEE_MEMMOVE 0x00000607

10.9.1.271 TEE_PANIC_ID_TEE_OPENPERSISTENTOBJECT #define
TEE_PANIC_ID_TEE_OPENPERSISTENTOBJECT 0x00000903

10.9.1.272 TEE_PANIC_ID_TEE_OPENTASESSION #define TEE_PANIC_ID_TEE_OPENTASESSION 0x00000403

10.9.1.273 TEE_PANIC_ID_TEE_PANIC #define TEE_PANIC_ID_TEE_PANIC 0x00000301

10.9.1.274 TEE_PANIC_ID_TEE_POPULATETRANSIENTOBJECT #define
TEE_PANIC_ID_TEE_POPULATETRANSIENTOBJECT 0x00000807

10.9.1.275 TEE_PANIC_ID_TEE_READOBJECTDATA #define
TEE_PANIC_ID_TEE_READOBJECTDATA 0x00000B01

10.9.1.276 TEE_PANIC_ID_TEE_REALLOC #define TEE_PANIC_ID_TEE_REALLOC 0x00000608

10.9.1.277 TEE_PANIC_ID_TEE_RENAMEPERSISTENTOBJECT #define
TEE_PANIC_ID_TEE_RENAMEPERSISTENTOBJECT 0x00000904

10.9.1.278 TEE_PANIC_ID_TEE_RESETOperation #define
TEE_PANIC_ID_TEE_RESETOperation 0x00000C05

10.9.1.279 TEE_PANIC_ID_TEE_RESETPersistentObjectEnumerator #define
TEE_PANIC_ID_TEE_RESETPersistentObjectEnumerator 0x00000A04

10.9.1.280 TEE_PANIC_ID_TEE_RESETPropertyEnumerator #define
TEE_PANIC_ID_TEE_RESETPropertyEnumerator 0x0000020B

10.9.1.281 TEE_PANIC_ID_TEE_RESETTransientObject #define
TEE_PANIC_ID_TEE_RESETTransientObject 0x00000808

10.9.1.282 TEE_PANIC_ID_TEE_RESTRICTObjectUsage #define
TEE_PANIC_ID_TEE_RESTRICTObjectUsage 0x00000705

10.9.1.283 TEE_PANIC_ID_TEE_RESTRICTObjectUsage1 #define
TEE_PANIC_ID_TEE_RESTRICTObjectUsage1 0x00000707

10.9.1.284 TEE_PANIC_ID_TEE_SEEKObjectData #define
TEE_PANIC_ID_TEE_SEEKObjectData 0x00000B02

10.9.1.285 TEE_PANIC_ID_TEE_SETInstancedata #define
TEE_PANIC_ID_TEE_SETInstancedata 0x00000609

10.9.1.286 TEE_PANIC_ID_TEE_SetOperationKey #define
TEE_PANIC_ID_TEE_SetOperationKey 0x00000C06

10.9.1.287 TEE_PANIC_ID_TEE_SetOperationKey2 #define
TEE_PANIC_ID_TEE_SetOperationKey2 0x00000C07

10.9.1.288 TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME #define
TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME 0x00001404

10.9.1.289 TEE_PANIC_ID_TEE_STARTPERSISTENTOBJECTENUMERATOR #define
TEE_PANIC_ID_TEE_STARTPERSISTENTOBJECTENUMERATOR 0x00000A05

10.9.1.290 TEE_PANIC_ID_TEE_STARTPROPERTYENUMERATOR #define
TEE_PANIC_ID_TEE_STARTPROPERTYENUMERATOR 0x0000020C

10.9.1.291 TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA #define
TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA 0x00000B03

10.9.1.292 TEE_PANIC_ID_TEE_UNMASKCANCELLATION #define
TEE_PANIC_ID_TEE_UNMASKCANCELLATION 0x00000503

10.9.1.293 TEE_PANIC_ID_TEE_WAIT #define TEE_PANIC_ID_TEE_WAIT 0x00001405

10.9.1.294 TEE_PANIC_ID_TEE_WRITEOBJECTDATA #define
TEE_PANIC_ID_TEE_WRITEOBJECTDATA 0x00000B04

10.9.1.295 TEE_PARAM_TYPE_GET #define TEE_PARAM_TYPE_GET(
 t,
 i) (((uint32_t)*t*) >> ((*i*)*4)) & 0xF)

10.9.1.296 TEE_PARAM_TYPE_MEMREF_INOUT #define TEE_PARAM_TYPE_MEMREF_INOUT 7

10.9.1.297 TEE_PARAM_TYPE_MEMREF_INPUT #define TEE_PARAM_TYPE_MEMREF_INPUT 5

10.9.1.298 TEE_PARAM_TYPE_MEMREF_OUTPUT `#define TEE_PARAM_TYPE_MEMREF_OUTPUT 6`

10.9.1.299 TEE_PARAM_TYPE_NONE `#define TEE_PARAM_TYPE_NONE 0`

10.9.1.300 TEE_PARAM_TYPE_SET `#define TEE_PARAM_TYPE_SET(
 t,
 i) (((uint32_t)(t) & 0xF) << ((i)*4))`

10.9.1.301 TEE_PARAM_TYPE_VALUE_INOUT `#define TEE_PARAM_TYPE_VALUE_INOUT 3`

10.9.1.302 TEE_PARAM_TYPE_VALUE_INPUT `#define TEE_PARAM_TYPE_VALUE_INPUT 1`

10.9.1.303 TEE_PARAM_TYPE_VALUE_OUTPUT `#define TEE_PARAM_TYPE_VALUE_OUTPUT 2`

10.9.1.304 TEE_PARAM_TYPES `#define TEE_PARAM_TYPES(
 t0,
 t1,
 t2,
 t3) (((t0) | ((t1) << 4) | ((t2) << 8) | ((t3) << 12))`

10.9.1.305 TEE_PROPSET_CURRENT_CLIENT `#define
TEE_PROPSET_CURRENT_CLIENT (TEE_PropSetHandle) 0xFFFFFFFF`

10.9.1.306 TEE_PROPSET_CURRENT_TA `#define
TEE_PROPSET_CURRENT_TA (TEE_PropSetHandle) 0xFFFFFFFF`

10.9.1.307 TEE_PROPSET_TEE_IMPLEMENTATION `#define
TEE_PROPSET_TEE_IMPLEMENTATION (TEE_PropSetHandle) 0xFFFFFFFFD`

10.9.1.308 TEE_STORAGE_PRIVATE `#define TEE_STORAGE_PRIVATE 0x00000001`

10.9.1.309 TEE_SUCCESS `#define TEE_SUCCESS 0x00000000`

10.9.1.310 TEE_TIMEOUT_INFINITE `#define TEE_TIMEOUT_INFINITE 0xFFFFFFFF`

10.9.1.311 TEE_TYPE_AES `#define TEE_TYPE_AES 0xA0000010`

10.9.1.312 TEE_TYPE_CORRUPTED_OBJECT `#define TEE_TYPE_CORRUPTED_OBJECT 0xA00000BE`

10.9.1.313 TEE_TYPE_DATA `#define TEE_TYPE_DATA 0xA00000BF`

10.9.1.314 TEE_TYPE_DES `#define TEE_TYPE_DES 0xA0000011`

10.9.1.315 TEE_TYPE_DES3 `#define TEE_TYPE_DES3 0xA0000013`

10.9.1.316 TEE_TYPE_DH_KEYPAIR `#define TEE_TYPE_DH_KEYPAIR 0xA1000032`

10.9.1.317 TEE_TYPE_DSA_KEYPAIR `#define TEE_TYPE_DSA_KEYPAIR 0xA1000031`

10.9.1.318 TEE_TYPE_DSA_PUBLIC_KEY `#define TEE_TYPE_DSA_PUBLIC_KEY 0xA0000031`

10.9.1.319 TEE_TYPE_ECDH_KEYPAIR #define TEE_TYPE_ECDH_KEYPAIR 0xA1000042

10.9.1.320 TEE_TYPE_ECDH_PUBLIC_KEY #define TEE_TYPE_ECDH_PUBLIC_KEY 0xA0000042

10.9.1.321 TEE_TYPE_ECDSA_KEYPAIR #define TEE_TYPE_ECDSA_KEYPAIR 0xA1000041

10.9.1.322 TEE_TYPE_ECDSA_PUBLIC_KEY #define TEE_TYPE_ECDSA_PUBLIC_KEY 0xA0000041

10.9.1.323 TEE_TYPE_GENERIC_SECRET #define TEE_TYPE_GENERIC_SECRET 0xA0000000

10.9.1.324 TEE_TYPE_HMAC_MD5 #define TEE_TYPE_HMAC_MD5 0xA0000001

10.9.1.325 TEE_TYPE_HMAC_SHA1 #define TEE_TYPE_HMAC_SHA1 0xA0000002

10.9.1.326 TEE_TYPE_HMAC_SHA224 #define TEE_TYPE_HMAC_SHA224 0xA0000003

10.9.1.327 TEE_TYPE_HMAC_SHA256 #define TEE_TYPE_HMAC_SHA256 0xA0000004

10.9.1.328 TEE_TYPE_HMAC_SHA384 #define TEE_TYPE_HMAC_SHA384 0xA0000005

10.9.1.329 TEE_TYPE_HMAC_SHA512 #define TEE_TYPE_HMAC_SHA512 0xA0000006

10.9.1.330 TEE_TYPE_RSA_KEYPAIR #define TEE_TYPE_RSA_KEYPAIR 0xA1000030

10.9.1.331 TEE_TYPE_RSA_PUBLIC_KEY #define TEE_TYPE_RSA_PUBLIC_KEY 0xA0000030

10.9.1.332 TEE_USAGE_DECRYPT #define TEE_USAGE_DECRYPT 0x00000004

10.9.1.333 TEE_USAGE_DERIVE #define TEE_USAGE_DERIVE 0x00000040

10.9.1.334 TEE_USAGE_ENCRYPT #define TEE_USAGE_ENCRYPT 0x00000002

10.9.1.335 TEE_USAGE_EXTRACTABLE #define TEE_USAGE_EXTRACTABLE 0x00000001

10.9.1.336 TEE_USAGE_MAC #define TEE_USAGE_MAC 0x00000008

10.9.1.337 TEE_USAGE_SIGN #define TEE_USAGE_SIGN 0x00000010

10.9.1.338 TEE_USAGE_VERIFY #define TEE_USAGE_VERIFY 0x00000020

10.10 tee_api_defines.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27
28 /* Based on GP TEE Internal Core API Specification Version 1.1 */
29
30 #ifndef TEE_API_DEFINES_H
31 #define TEE_API_DEFINES_H
32
33 #define TEE_INT_CORE_API_SPEC_VERSION    0x0000000A
34
35 #define TEE_HANDLE_NULL                  0
36
37 #define TEE_TIMEOUT_INFINITE              0xFFFFFFFF
38
39 /* API Error Codes */
40 #define TEE_SUCCESS                      0x00000000
41 #define TEE_ERROR_CORRUPT_OBJECT          0xF0100001
42 #define TEE_ERROR_CORRUPT_OBJECT_2       0xF0100002
43 #define TEE_ERROR_STORAGE_NOT_AVAILABLE  0xF0100003
44 #define TEE_ERROR_STORAGE_NOT_AVAILABLE_2 0xF0100004
45 #define TEE_ERROR_GENERIC                 0xFFFF0000
46 #define TEE_ERROR_ACCESS_DENIED           0xFFFF0001
47 #define TEE_ERROR_CANCEL                  0xFFFF0002
48 #define TEE_ERROR_ACCESS_CONFLICT         0xFFFF0003
49 #define TEE_ERROR_EXCESS_DATA             0xFFFF0004
50 #define TEE_ERROR_BAD_FORMAT              0xFFFF0005
51 #define TEE_ERROR_BAD_PARAMETERS          0xFFFF0006
52 #define TEE_ERROR_BAD_STATE               0xFFFF0007
53 #define TEE_ERROR_ITEM_NOT_FOUND          0xFFFF0008
54 #define TEE_ERROR_NOT_IMPLEMENTED         0xFFFF0009
55 #define TEE_ERROR_NOT_SUPPORTED           0xFFFF000A
56 #define TEE_ERROR_NO_DATA                 0xFFFF000B
57 #define TEE_ERROR_OUT_OF_MEMORY           0xFFFF000C
58 #define TEE_ERROR_BUSY                    0xFFFF000D
59 #define TEE_ERROR_COMMUNICATION           0xFFFF000E
60 #define TEE_ERROR_SECURITY                0xFFFF000F
61 #define TEE_ERROR_SHORT_BUFFER            0xFFFF0010
62 #define TEE_ERROR_EXTERNAL_CANCEL         0xFFFF0011
63 #define TEE_ERROR_OVERFLOW                0xFFFF300F
64 #define TEE_ERROR_TARGET_DEAD             0xFFFF3024
65 #define TEE_ERROR_STORAGE_NO_SPACE        0xFFFF3041
66 #define TEE_ERROR_MAC_INVALID             0xFFFF3071
67 #define TEE_ERROR_SIGNATURE_INVALID        0xFFFF3072
68 #define TEE_ERROR_TIME_NOT_SET            0xFFFF5000
69 #define TEE_ERROR_TIME_NEEDS_RESET        0xFFFF5001
70
71 /* Parameter Type Constants */
72 #define TEE_PARAM_TYPE_NONE               0
73 #define TEE_PARAM_TYPE_VALUE_INPUT        1
74 #define TEE_PARAM_TYPE_VALUE_OUTPUT      2
75 #define TEE_PARAM_TYPE_VALUE_INOUT       3
76 #define TEE_PARAM_TYPE_MEMREF_INPUT      5
77 #define TEE_PARAM_TYPE_MEMREF_OUTPUT     6
78 #define TEE_PARAM_TYPE_MEMREF_INOUT      7
79
80 /* Login Type Constants */
81 #define TEE_LOGIN_PUBLIC                   0x00000000
82 #define TEE_LOGIN_USER                     0x00000001
83 #define TEE_LOGIN_GROUP                    0x00000002

```



```

84 #define TEE.LOGIN.APPLICATION          0x00000004
85 #define TEE.LOGIN.APPLICATION.USER     0x00000005
86 #define TEE.LOGIN.APPLICATION.GROUP    0x00000006
87 #define TEE.LOGIN.TRUSTED_APP          0xF0000000
88
89 /* Origin Code Constants */
90 #define TEE.ORIGIN_API                  0x00000001
91 #define TEE.ORIGIN_COMMS                0x00000002
92 #define TEE.ORIGIN_TEE                  0x00000003
93 #define TEE.ORIGIN_TRUSTED_APP          0x00000004
94
95 /* Property Sets pseudo handles */
96 #define TEE.PROPSET.TEE.IMPLEMENTATION (TEE.PropSetHandle) 0xFFFFFFFF
97 #define TEE.PROPSET.CURRENT_CLIENT     (TEE.PropSetHandle) 0xFFFFFFFF
98 #define TEE.PROPSET.CURRENT_TA         (TEE.PropSetHandle) 0xFFFFFFFF
99
100 /* Memory Access Rights Constants */
101 #define TEE.MEMORY.ACCESS.READ          0x00000001
102 #define TEE.MEMORY.ACCESS.WRITE         0x00000002
103 #define TEE.MEMORY.ACCESS.ANY_OWNER     0x00000004
104
105 /* Memory Management Constant */
106 #define TEE.MALLOC.FILL_ZERO            0x00000000
107
108 /* Other constants */
109 #define TEE.STORAGE.PRIVATE              0x00000001
110
111 #define TEE.DATA.FLAG.ACCESS.READ        0x00000001
112 #define TEE.DATA.FLAG.ACCESS.WRITE      0x00000002
113 #define TEE.DATA.FLAG.ACCESS.WRITE.META 0x00000004
114 #define TEE.DATA.FLAG.SHARE.READ        0x00000010
115 #define TEE.DATA.FLAG.SHARE.WRITE       0x00000020
116 #define TEE.DATA.FLAG.OVERWRITE         0x00000400
117 #define TEE.DATA.MAX.POSITION           0xFFFFFFFF
118 #define TEE.OBJECT.ID.MAX.LEN            64
119 #define TEE.USAGE.EXTRACTABLE            0x00000001
120 #define TEE.USAGE.ENCRYPT                 0x00000002
121 #define TEE.USAGE.DECRYPT                0x00000004
122 #define TEE.USAGE.MAC                    0x00000008
123 #define TEE.USAGE.SIGN                   0x00000010
124 #define TEE.USAGE.VERIFY                  0x00000020
125 #define TEE.USAGE.DERIVE                  0x00000040
126 #define TEE.HANDLE.FLAG.PERSISTENT       0x00010000
127 #define TEE.HANDLE.FLAG.INITIALIZED      0x00020000
128 #define TEE.HANDLE.FLAG.KEY.SET          0x00040000
129 #define TEE.HANDLE.FLAG.EXPECT.TWO.KEYS 0x00080000
130 #define TEE.OPERATION.CIPHER              1
131 #define TEE.OPERATION.MAC                 3
132 #define TEE.OPERATION.AE                  4
133 #define TEE.OPERATION.DIGEST              5
134 #define TEE.OPERATION.ASYMMETRIC.CIPHER  6
135 #define TEE.OPERATION.ASYMMETRIC.SIGNATURE 7
136 #define TEE.OPERATION.KEY.DERIVATION      8
137 #define TEE.OPERATION.STATE.INITIAL       0x00000000
138 #define TEE.OPERATION.STATE.ACTIVE        0x00000001
139
140 /* Algorithm Identifiers */
141 #define TEE.ALG.AES.ECB.NOPAD             0x10000010
142 #define TEE.ALG.AES.CBC.NOPAD             0x10000110
143 #define TEE.ALG.AES.CTR                   0x10000210
144 #define TEE.ALG.AES.CTS                   0x10000310
145 #define TEE.ALG.AES.XTS                   0x10000410
146 #define TEE.ALG.AES.CBC.MAC.NOPAD         0x30000110
147 #define TEE.ALG.AES.CBC.MAC.PKCS5         0x30000510
148 #define TEE.ALG.AES.CMAC                  0x30000610
149 #define TEE.ALG.AES.CCM                    0x40000710
150 #define TEE.ALG.AES.GCM                    0x40000810
151 #define TEE.ALG.DES.ECB.NOPAD             0x10000011
152 #define TEE.ALG.DES.CBC.NOPAD             0x10000111
153 #define TEE.ALG.DES.CBC.MAC.NOPAD         0x30000111
154 #define TEE.ALG.DES.CBC.MAC.PKCS5         0x30000511
155 #define TEE.ALG.DES3.ECB.NOPAD            0x10000013
156 #define TEE.ALG.DES3.CBC.NOPAD            0x10000113
157 #define TEE.ALG.DES3.CBC.MAC.NOPAD        0x30000113
158 #define TEE.ALG.DES3.CBC.MAC.PKCS5        0x30000513
159 #define TEE.ALG.RSASSA.PKCS1.V1.5.MD5     0x70001830
160 #define TEE.ALG.RSASSA.PKCS1.V1.5.SHA1    0x70002830
161 #define TEE.ALG.RSASSA.PKCS1.V1.5.SHA224  0x70003830
162 #define TEE.ALG.RSASSA.PKCS1.V1.5.SHA256  0x70004830
163 #define TEE.ALG.RSASSA.PKCS1.V1.5.SHA384  0x70005830
164 #define TEE.ALG.RSASSA.PKCS1.V1.5.SHA512  0x70006830
165 #define TEE.ALG.RSASSA.PKCS1.V1.5.MD5SHA1 0x7000F830
166 #define TEE.ALG.RSASSA.PKCS1.PSS.MGF1.SHA1 0x70212930
167 #define TEE.ALG.RSASSA.PKCS1.PSS.MGF1.SHA224 0x70313930
168 #define TEE.ALG.RSASSA.PKCS1.PSS.MGF1.SHA256 0x70414930
169 #define TEE.ALG.RSASSA.PKCS1.PSS.MGF1.SHA384 0x70515930
170 #define TEE.ALG.RSASSA.PKCS1.PSS.MGF1.SHA512 0x70616930

```

```

171 #define TEE_ALG_RSAES_PKCS1_V1_5 0x60000130
172 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1 0x60210230
173 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224 0x60310230
174 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256 0x60410230
175 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384 0x60510230
176 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512 0x60610230
177 #define TEE_ALG_RSA_NOPAD 0x60000030
178 #define TEE_ALG_DSA_SHA1 0x70002131
179 #define TEE_ALG_DSA_SHA224 0x70003131
180 #define TEE_ALG_DSA_SHA256 0x70004131
181 #define TEE_ALG_DH_DERIVE_SHARED_SECRET 0x80000032
182 #define TEE_ALG_MD5 0x50000001
183 #define TEE_ALG_SHA1 0x50000002
184 #define TEE_ALG_SHA224 0x50000003
185 #define TEE_ALG_SHA256 0x50000004
186 #define TEE_ALG_SHA384 0x50000005
187 #define TEE_ALG_SHA512 0x50000006
188 #define TEE_ALG_MD5_SHA1 0x5000000F
189 #define TEE_ALG_HMAC_MD5 0x30000001
190 #define TEE_ALG_HMAC_SHA1 0x30000002
191 #define TEE_ALG_HMAC_SHA224 0x30000003
192 #define TEE_ALG_HMAC_SHA256 0x30000004
193 #define TEE_ALG_HMAC_SHA384 0x30000005
194 #define TEE_ALG_HMAC_SHA512 0x30000006
195 /*
196  * Fix GP Internal Core API v1.1
197  * "Table 6-12: Structure of Algorithm Identifier"
198  * indicates ECDSA have the algorithm "0x41" and ECDH "0x42"
199  * whereas
200  * "Table 6-11: List of Algorithm Identifiers" defines
201  * TEE_ALG_ECDSA_P192 as 0x70001042
202  *
203  * We chose to define TEE_ALG_ECDSA_P192 as 0x70001041 (conform to table 6-12)
204  */
205 #define TEE_ALG_ECDSA_P192 0x70001041
206 #define TEE_ALG_ECDSA_P224 0x70002041
207 #define TEE_ALG_ECDSA_P256 0x70003041
208 #define TEE_ALG_ECDSA_P384 0x70004041
209 #define TEE_ALG_ECDSA_P521 0x70005041
210 #define TEE_ALG_ECDH_P192 0x80001042
211 #define TEE_ALG_ECDH_P224 0x80002042
212 #define TEE_ALG_ECDH_P256 0x80003042
213 #define TEE_ALG_ECDH_P384 0x80004042
214 #define TEE_ALG_ECDH_P521 0x80005042
215
216 /* Object Types */
217
218 #define TEE_TYPE_AES 0xA0000010
219 #define TEE_TYPE_DES 0xA0000011
220 #define TEE_TYPE_DES3 0xA0000013
221 #define TEE_TYPE_HMAC_MD5 0xA0000001
222 #define TEE_TYPE_HMAC_SHA1 0xA0000002
223 #define TEE_TYPE_HMAC_SHA224 0xA0000003
224 #define TEE_TYPE_HMAC_SHA256 0xA0000004
225 #define TEE_TYPE_HMAC_SHA384 0xA0000005
226 #define TEE_TYPE_HMAC_SHA512 0xA0000006
227 #define TEE_TYPE_RSA_PUBLIC_KEY 0xA0000030
228 #define TEE_TYPE_RSA_KEYPAIR 0xA1000030
229 #define TEE_TYPE_DSA_PUBLIC_KEY 0xA0000031
230 #define TEE_TYPE_DSA_KEYPAIR 0xA1000031
231 #define TEE_TYPE_DH_KEYPAIR 0xA1000032
232 #define TEE_TYPE_ECDSA_PUBLIC_KEY 0xA0000041
233 #define TEE_TYPE_ECDSA_KEYPAIR 0xA1000041
234 #define TEE_TYPE_ECDH_PUBLIC_KEY 0xA0000042
235 #define TEE_TYPE_ECDH_KEYPAIR 0xA1000042
236 #define TEE_TYPE_GENERIC_SECRET 0xA0000000
237 #define TEE_TYPE_CORRUPTED_OBJECT 0xA00000BE
238 #define TEE_TYPE_DATA 0xA00000BF
239
240 /* List of Object or Operation Attributes */
241
242 #define TEE_ATTR_SECRET_VALUE 0xC0000000
243 #define TEE_ATTR_RSA_MODULUS 0xD0000130
244 #define TEE_ATTR_RSA_PUBLIC_EXPONENT 0xD0000230
245 #define TEE_ATTR_RSA_PRIVATE_EXPONENT 0xC0000330
246 #define TEE_ATTR_RSA_PRIME1 0xC0000430
247 #define TEE_ATTR_RSA_PRIME2 0xC0000530
248 #define TEE_ATTR_RSA_EXPONENT1 0xC0000630
249 #define TEE_ATTR_RSA_EXPONENT2 0xC0000730
250 #define TEE_ATTR_RSA_COEFFICIENT 0xC0000830
251 #define TEE_ATTR_DSA_PRIME 0xD0001031
252 #define TEE_ATTR_DSA_SUBPRIME 0xD0001131
253 #define TEE_ATTR_DSA_BASE 0xD0001231
254 #define TEE_ATTR_DSA_PUBLIC_VALUE 0xD0001331
255 #define TEE_ATTR_DSA_PRIVATE_VALUE 0xC0000231
256 #define TEE_ATTR_DH_PRIME 0xD0001032
257 #define TEE_ATTR_DH_SUBPRIME 0xD0001132

```

```

258 #define TEE_ATTR_DH_BASE 0xD0001232
259 #define TEE_ATTR_DH_X_BITS 0xF0001332
260 #define TEE_ATTR_DH_PUBLIC_VALUE 0xD0000132
261 #define TEE_ATTR_DH_PRIVATE_VALUE 0xC0000232
262 #define TEE_ATTR_RSA_OAEP_LABEL 0xD0000930
263 #define TEE_ATTR_RSA_PSS_SALT_LENGTH 0xF0000A30
264 #define TEE_ATTR_ECC_PUBLIC_VALUE_X 0xD0000141
265 #define TEE_ATTR_ECC_PUBLIC_VALUE_Y 0xD0000241
266 #define TEE_ATTR_ECC_PRIVATE_VALUE 0xC0000341
267 #define TEE_ATTR_ECC_CURVE 0xF0000441
268
269 #define TEE_ATTR_BIT_PROTECTED (1 << 28)
270 #define TEE_ATTR_BIT_VALUE (1 << 29)
271
272 /* List of Supported ECC Curves */
273 #define TEE_ECC_CURVE_NIST_P192 0x00000001
274 #define TEE_ECC_CURVE_NIST_P224 0x00000002
275 #define TEE_ECC_CURVE_NIST_P256 0x00000003
276 #define TEE_ECC_CURVE_NIST_P384 0x00000004
277 #define TEE_ECC_CURVE_NIST_P521 0x00000005
278
279
280 /* Panicked Functions Identification */
281 /* TA Interface */
282 #define TEE_PANIC_ID_TA_CLOSESESSIONENTRYPOINT 0x00000101
283 #define TEE_PANIC_ID_TA_CREATEENTRYPOINT 0x00000102
284 #define TEE_PANIC_ID_TA_DESTROYENTRYPOINT 0x00000103
285 #define TEE_PANIC_ID_TA_INVOKECOMMANDENTRYPOINT 0x00000104
286 #define TEE_PANIC_ID_TA_OPENSESSIONENTRYPOINT 0x00000105
287 /* Property Access */
288 #define TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERATOR 0x00000201
289 #define TEE_PANIC_ID_TEE_FREEPROPERTYENUMERATOR 0x00000202
290 #define TEE_PANIC_ID_TEE_GETNEXTPROPERTY 0x00000203
291 #define TEE_PANIC_ID_TEE_GETPROPERTYASBINARYBLOCK 0x00000204
292 #define TEE_PANIC_ID_TEE_GETPROPERTYASBOOL 0x00000205
293 #define TEE_PANIC_ID_TEE_GETPROPERTYASIDENTITY 0x00000206
294 #define TEE_PANIC_ID_TEE_GETPROPERTYASSTRING 0x00000207
295 #define TEE_PANIC_ID_TEE_GETPROPERTYASU32 0x00000208
296 #define TEE_PANIC_ID_TEE_GETPROPERTYASUUID 0x00000209
297 #define TEE_PANIC_ID_TEE_GETPROPERTYNAME 0x0000020A
298 #define TEE_PANIC_ID_TEE_RESETPROPERTYENUMERATOR 0x0000020B
299 #define TEE_PANIC_ID_TEE_STARTPROPERTYENUMERATOR 0x0000020C
300 /* Panic Function */
301 #define TEE_PANIC_ID_TEE_PANIC 0x00000301
302 /* Internal Client API */
303 #define TEE_PANIC_ID_TEE_CLOSETASESSION 0x00000401
304 #define TEE_PANIC_ID_TEE_INVOKETACOMMAND 0x00000402
305 #define TEE_PANIC_ID_TEE_OPENTASESSION 0x00000403
306 /* Cancellation */
307 #define TEE_PANIC_ID_TEE_GETCANCELLATIONFLAG 0x00000501
308 #define TEE_PANIC_ID_TEE_MASKCANCELLATION 0x00000502
309 #define TEE_PANIC_ID_TEE_UNMASKCANCELLATION 0x00000503
310 /* Memory Management */
311 #define TEE_PANIC_ID_TEE_CHECKMEMORYACCESSRIGHTS 0x00000601
312 #define TEE_PANIC_ID_TEE_FREE 0x00000602
313 #define TEE_PANIC_ID_TEE_GETINSTANCEDATA 0x00000603
314 #define TEE_PANIC_ID_TEE_MALLOC 0x00000604
315 #define TEE_PANIC_ID_TEE_MEMCOMPARE 0x00000605
316 #define TEE_PANIC_ID_TEE_MEMFILL 0x00000606
317 #define TEE_PANIC_ID_TEE_MEMMOVE 0x00000607
318 #define TEE_PANIC_ID_TEE_REALLOC 0x00000608
319 #define TEE_PANIC_ID_TEE_SETINSTANCEDATA 0x00000609
320 /* Generic Object */
321 #define TEE_PANIC_ID_TEE_CLOSEOBJECT 0x00000701
322 #define TEE_PANIC_ID_TEE_GETOBJECTBUFFERATTRIBUTE 0x00000702
323 /* deprecated */
324 #define TEE_PANIC_ID_TEE_GETOBJECTINFO 0x00000703
325 #define TEE_PANIC_ID_TEE_GETOBJECTVALUEATTRIBUTE 0x00000704
326 /* deprecated */
327 #define TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE 0x00000705
328 #define TEE_PANIC_ID_TEE_GETOBJECTINFO1 0x00000706
329 #define TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE1 0x00000707
330 /* Transient Object */
331 #define TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT 0x00000801
332 /* deprecated */
333 #define TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES 0x00000802
334 #define TEE_PANIC_ID_TEE_FREETRANSIENTOBJECT 0x00000803
335 #define TEE_PANIC_ID_TEE_GENERATEKEY 0x00000804
336 #define TEE_PANIC_ID_TEE_INITREFATTRIBUTE 0x00000805
337 #define TEE_PANIC_ID_TEE_INITVALUEATTRIBUTE 0x00000806
338 #define TEE_PANIC_ID_TEE_POPULATETRANSIENTOBJECT 0x00000807
339 #define TEE_PANIC_ID_TEE_RESETTRANSIENTOBJECT 0x00000808
340 #define TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES1 0x00000809
341 /* Persistent Object */
342 /* deprecated */
343 #define TEE_PANIC_ID_TEE_CLOSEANDDELETERPERSISTENTOBJECT 0x00000901
344 #define TEE_PANIC_ID_TEE_CREATEPERSISTENTOBJECT 0x00000902

```

```

345 #define TEE_PANIC_ID_TEE_OPENPERSISTENTOBJECT 0x00000903
346 #define TEE_PANIC_ID_TEE_RENAMEPERSISTENTOBJECT 0x00000904
347 #define TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT1 0x00000905
348 /* Persistent Object Enumeration */
349 #define TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR 0x00000A01
350 #define TEE_PANIC_ID_TEE_FREEPERSISTENTOBJECTENUMERATOR 0x00000A02
351 #define TEE_PANIC_ID_TEE_GETNEXTPERSISTENTOBJECT 0x00000A03
352 #define TEE_PANIC_ID_TEE_RESETPERSISTENTOBJECTENUMERATOR 0x00000A04
353 #define TEE_PANIC_ID_TEE_STARTPERSISTENTOBJECTENUMERATOR 0x00000A05
354 /* Data Stream Access */
355 #define TEE_PANIC_ID_TEE_READOBJECTDATA 0x00000B01
356 #define TEE_PANIC_ID_TEE_SEEKOBJECTDATA 0x00000B02
357 #define TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA 0x00000B03
358 #define TEE_PANIC_ID_TEE_WRITEOBJECTDATA 0x00000B04
359 /* Generic Operation */
360 #define TEE_PANIC_ID_TEE_ALLOCATEOPERATION 0x00000C01
361 #define TEE_PANIC_ID_TEE_COPYOPERATION 0x00000C02
362 #define TEE_PANIC_ID_TEE_FREEOPERATION 0x00000C03
363 #define TEE_PANIC_ID_TEE_GETOPERATIONINFO 0x00000C04
364 #define TEE_PANIC_ID_TEE_RESETOPERATION 0x00000C05
365 #define TEE_PANIC_ID_TEE_SETOPERATIONKEY 0x00000C06
366 #define TEE_PANIC_ID_TEE_SETOPERATIONKEY2 0x00000C07
367 #define TEE_PANIC_ID_TEE_GETOPERATIONINFOMULTIPLE 0x00000C08
368 /* Message Digest */
369 #define TEE_PANIC_ID_TEE_DIGESTDOFINAL 0x00000D01
370 #define TEE_PANIC_ID_TEE_DIGESTUPDATE 0x00000D02
371 /* Symmetric Cipher */
372 #define TEE_PANIC_ID_TEE_CIPHERDOFINAL 0x00000E01
373 #define TEE_PANIC_ID_TEE_CIPHERINIT 0x00000E02
374 #define TEE_PANIC_ID_TEE_CIPHERUPDATE 0x00000E03
375 /* MAC */
376 #define TEE_PANIC_ID_TEE_MACCOMPAREFINAL 0x00000F01
377 #define TEE_PANIC_ID_TEE_MACCOMPUTEFINAL 0x00000F02
378 #define TEE_PANIC_ID_TEE_MACINIT 0x00000F03
379 #define TEE_PANIC_ID_TEE_MACUPDATE 0x00000F04
380 /* Authenticated Encryption */
381 #define TEE_PANIC_ID_TEE_AEDECRYPTFINAL 0x00001001
382 #define TEE_PANIC_ID_TEE_AEENCRYPTFINAL 0x00001002
383 #define TEE_PANIC_ID_TEE_AEINIT 0x00001003
384 #define TEE_PANIC_ID_TEE_AEUPDATE 0x00001004
385 #define TEE_PANIC_ID_TEE_AEUPDATEAAD 0x00001005
386 /* Asymmetric */
387 #define TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT 0x00001101
388 #define TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT 0x00001102
389 #define TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST 0x00001103
390 #define TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST 0x00001104
391 /* Key Derivation */
392 #define TEE_PANIC_ID_TEE_DERIVEKEY 0x00001201
393 /* Random Data Generation */
394 #define TEE_PANIC_ID_TEE_GENERATERANDOM 0x00001301
395 /* Time */
396 #define TEE_PANIC_ID_TEE_GETREETIME 0x00001401
397 #define TEE_PANIC_ID_TEE_GETSYSTEMTIME 0x00001402
398 #define TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME 0x00001403
399 #define TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME 0x00001404
400 #define TEE_PANIC_ID_TEE_WAIT 0x00001405
401 /* Memory Allocation and Size of Objects */
402 #define TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32 0x00001501
403 #define TEE_PANIC_ID_TEE_BIGINTFMMMSIZEINU32 0x00001502
404 /* Initialization */
405 #define TEE_PANIC_ID_TEE_BIGINTINIT 0x00001601
406 #define TEE_PANIC_ID_TEE_BIGINTINITFMM 0x00001602
407 #define TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT 0x00001603
408 /* Converter */
409 #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING 0x00001701
410 #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32 0x00001702
411 #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING 0x00001703
412 #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32 0x00001704
413 /* Logical Operation */
414 #define TEE_PANIC_ID_TEE_BIGINTCMP 0x00001801
415 #define TEE_PANIC_ID_TEE_BIGINTCMPS32 0x00001802
416 #define TEE_PANIC_ID_TEE_BIGINTGETBIT 0x00001803
417 #define TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT 0x00001804
418 #define TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT 0x00001805
419 /* Basic Arithmetic */
420 #define TEE_PANIC_ID_TEE_BIGINTADD 0x00001901
421 #define TEE_PANIC_ID_TEE_BIGINTDIV 0x00001902
422 #define TEE_PANIC_ID_TEE_BIGINTMUL 0x00001903
423 #define TEE_PANIC_ID_TEE_BIGINTNEG 0x00001904
424 #define TEE_PANIC_ID_TEE_BIGINTSQUARE 0x00001905
425 #define TEE_PANIC_ID_TEE_BIGINTSUB 0x00001906
426 /* Modular Arithmetic */
427 #define TEE_PANIC_ID_TEE_BIGINTADDMOD 0x00001A01
428 #define TEE_PANIC_ID_TEE_BIGINTINVMOD 0x00001A02
429 #define TEE_PANIC_ID_TEE_BIGINTMOD 0x00001A03
430 #define TEE_PANIC_ID_TEE_BIGINTMULMOD 0x00001A04
431 #define TEE_PANIC_ID_TEE_BIGINTSQUAREMOD 0x00001A05

```

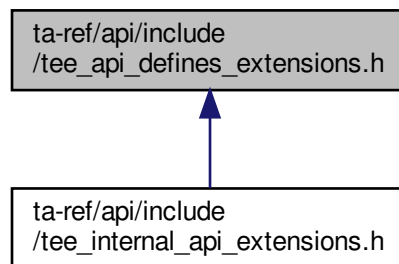
```

432 #define TEE_PANIC_ID_TEE_BIGINTSUBMOD 0x00001A06
433 /* Other Arithmetic */
434 #define TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD 0x00001B01
435 #define TEE_PANIC_ID_TEE_BIGINTISPROBABLEPRIME 0x00001B02
436 #define TEE_PANIC_ID_TEE_BIGINTRELATIVEPRIME 0x00001B03
437 /* Fast Modular Multiplication */
438 #define TEE_PANIC_ID_TEE_BIGINTCOMPUTEFCMM 0x00001C01
439 #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMFMM 0x00001C02
440 #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOFMM 0x00001C03
441
442 /*
443  * The macro TEE_PARAM_TYPES can be used to construct a value that you can
444  * compare against an incoming paramTypes to check the type of all the
445  * parameters in one comparison, like in the following example:
446  * if (paramTypes != TEE_PARAM_TYPES(TEE_PARAM_TYPE_MEMREF_INPUT,
447  *                                   TEE_PARAM_TYPE_MEMREF_OUTPUT,
448  *                                   TEE_PARAM_TYPE_NONE, TEE_PARAM_TYPE_NONE)) {
449  *     return TEE_ERROR_BAD_PARAMETERS;
450  * }
451  */
452 #define TEE_PARAM_TYPES(t0,t1,t2,t3) \
453     ((t0) | ((t1) << 4) | ((t2) << 8) | ((t3) << 12))
454
455 /*
456  * The macro TEE_PARAM_TYPE_GET can be used to extract the type of a given
457  * parameter from paramTypes if you need more fine-grained type checking.
458  */
459 #define TEE_PARAM_TYPE_GET(t, i) (((uint32_t)t) >> ((i)*4) & 0xF)
460
461 /*
462  * The macro TEE_PARAM_TYPE_SET can be used to load the type of a given
463  * parameter from paramTypes without specifying all types (TEE_PARAM_TYPES)
464  */
465 #define TEE_PARAM_TYPE_SET(t, i) (((uint32_t)t) & 0xF << ((i)*4))
466
467 /* Not specified in the standard */
468 #define TEE_NUM_PARAMS 4
469
470 /* TEE Arithmetical APIs */
471
472 #define TEE_BigIntSizeInU32(n) (((n)+31)/32)+2
473
474 #endif /* TEE_API_DEFINES_H */

```

10.11 ta-ref/api/include/tee_api_defines_extensions.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define [TEE_ALG_HKDF_MD5_DERIVE_KEY](#) 0x800010C0
- #define [TEE_ALG_HKDF_SHA1_DERIVE_KEY](#) 0x800020C0

- `#define TEE_ALG_HKDF_SHA224_DERIVE_KEY 0x800030C0`
- `#define TEE_ALG_HKDF_SHA256_DERIVE_KEY 0x800040C0`
- `#define TEE_ALG_HKDF_SHA384_DERIVE_KEY 0x800050C0`
- `#define TEE_ALG_HKDF_SHA512_DERIVE_KEY 0x800060C0`
- `#define TEE_TYPE_HKDF_IKM 0xA10000C0`
- `#define TEE_ATTR_HKDF_IKM 0xC00001C0`
- `#define TEE_ATTR_HKDF_SALT 0xD00002C0`
- `#define TEE_ATTR_HKDF_INFO 0xD00003C0`
- `#define TEE_ATTR_HKDF_OKM_LENGTH 0xF00004C0`
- `#define TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY 0x800020C1`
- `#define TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY 0x800030C1`
- `#define TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY 0x800040C1`
- `#define TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY 0x800050C1`
- `#define TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY 0x800060C1`
- `#define TEE_TYPE_CONCAT_KDF_Z 0xA10000C1`
- `#define TEE_ATTR_CONCAT_KDF_Z 0xC00001C1`
- `#define TEE_ATTR_CONCAT_KDF_OTHER_INFO 0xD00002C1`
- `#define TEE_ATTR_CONCAT_KDF_DKM_LENGTH 0xF00003C1`
- `#define TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY 0x800020C2`
- `#define TEE_TYPE_PBKDF2_PASSWORD 0xA10000C2`
- `#define TEE_ATTR_PBKDF2_PASSWORD 0xC00001C2`
- `#define TEE_ATTR_PBKDF2_SALT 0xD00002C2`
- `#define TEE_ATTR_PBKDF2_ITERATION_COUNT 0xF00003C2`
- `#define TEE_ATTR_PBKDF2_DKM_LENGTH 0xF00004C2`
- `#define TEE_STORAGE_PRIVATE_REE 0x80000000`
- `#define TEE_STORAGE_PRIVATE_RPMB 0x80000100`
- `#define TEE_STORAGE_PRIVATE_SQL_RESERVED 0x80000200`
- `#define TEE_MEMORY_ACCESS_NONSECURE 0x10000000`
- `#define TEE_MEMORY_ACCESS_SECURE 0x20000000`

10.11.1 Macro Definition Documentation

10.11.1.1 TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY `#define`
`TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY 0x800020C1`

10.11.1.2 TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY `#define`
`TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY 0x800030C1`

10.11.1.3 TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY `#define`
`TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY 0x800040C1`

10.11.1.4 TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY #define
TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY 0x800050C1

10.11.1.5 TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY #define
TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY 0x800060C1

10.11.1.6 TEE_ALG_HKDF_MD5_DERIVE_KEY #define TEE_ALG_HKDF_MD5_DERIVE_KEY 0x800010C0

10.11.1.7 TEE_ALG_HKDF_SHA1_DERIVE_KEY #define TEE_ALG_HKDF_SHA1_DERIVE_KEY 0x800020C0

10.11.1.8 TEE_ALG_HKDF_SHA224_DERIVE_KEY #define TEE_ALG_HKDF_SHA224_DERIVE_KEY 0x800030C0

10.11.1.9 TEE_ALG_HKDF_SHA256_DERIVE_KEY #define TEE_ALG_HKDF_SHA256_DERIVE_KEY 0x800040C0

10.11.1.10 TEE_ALG_HKDF_SHA384_DERIVE_KEY #define TEE_ALG_HKDF_SHA384_DERIVE_KEY 0x800050C0

10.11.1.11 TEE_ALG_HKDF_SHA512_DERIVE_KEY #define TEE_ALG_HKDF_SHA512_DERIVE_KEY 0x800060C0

10.11.1.12 TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY #define
TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY 0x800020C2

10.11.1.13 TEE_ATTR_CONCAT_KDF_DKM_LENGTH #define
TEE_ATTR_CONCAT_KDF_DKM_LENGTH 0xF00003C1

10.11.1.14 TEE_ATTR_CONCAT_KDF_OTHER_INFO #define TEE_ATTR_CONCAT_KDF_OTHER_INFO 0xD00002C1

10.11.1.15 TEE_ATTR_CONCAT_KDF_Z #define TEE_ATTR_CONCAT_KDF_Z 0xC00001C1

10.11.1.16 TEE_ATTR_HKDF_IKM #define TEE_ATTR_HKDF_IKM 0xC00001C0

10.11.1.17 TEE_ATTR_HKDF_INFO #define TEE_ATTR_HKDF_INFO 0xD00003C0

10.11.1.18 TEE_ATTR_HKDF_OKM_LENGTH #define TEE_ATTR_HKDF_OKM_LENGTH 0xF00004C0

10.11.1.19 TEE_ATTR_HKDF_SALT #define TEE_ATTR_HKDF_SALT 0xD00002C0

10.11.1.20 TEE_ATTR_PBKDF2_DKM_LENGTH #define TEE_ATTR_PBKDF2_DKM_LENGTH 0xF00004C2

10.11.1.21 TEE_ATTR_PBKDF2_ITERATION_COUNT #define
TEE_ATTR_PBKDF2_ITERATION_COUNT 0xF00003C2

10.11.1.22 TEE_ATTR_PBKDF2_PASSWORD #define TEE_ATTR_PBKDF2_PASSWORD 0xC00001C2

10.11.1.23 TEE_ATTR_PBKDF2_SALT #define TEE_ATTR_PBKDF2_SALT 0xD00002C2

10.11.1.24 TEE_MEMORY_ACCESS_NONSECURE #define TEE_MEMORY_ACCESS_NONSECURE 0x10000000

10.11.1.25 TEE_MEMORY_ACCESS_SECURE #define TEE_MEMORY_ACCESS_SECURE 0x20000000

10.11.1.26 TEE_STORAGE_PRIVATE_REE #define TEE_STORAGE_PRIVATE_REE 0x80000000

10.11.1.27 TEE_STORAGE_PRIVATE_RPMB #define TEE_STORAGE_PRIVATE_RPMB 0x80000100

10.11.1.28 TEE_STORAGE_PRIVATE_SQL_RESERVED #define
TEE_STORAGE_PRIVATE_SQL_RESERVED 0x80000200

10.11.1.29 TEE_TYPE_CONCAT_KDF_Z #define TEE_TYPE_CONCAT_KDF_Z 0xA10000C1

10.11.1.30 TEE_TYPE_HKDF_IKM #define TEE_TYPE_HKDF_IKM 0xA10000C0

10.11.1.31 TEE_TYPE_PBKDF2_PASSWORD #define TEE_TYPE_PBKDF2_PASSWORD 0xA10000C2

10.12 tee_api_defines_extensions.h

[Go to the documentation of this file.](#)

```
1 /*
2  * Copyright (c) 2014, Linaro Limited
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27
28 #ifndef TEE_API_DEFINES_EXTENSIONS_H
29 #define TEE_API_DEFINES_EXTENSIONS_H
30
31 /*
32  * HMAC-based Extract-and-Expand Key Derivation Function (HKDF)
33  */
34
35 #define TEE_ALG_HKDF_MD5_DERIVE_KEY      0x800010C0
36 #define TEE_ALG_HKDF_SHA1_DERIVE_KEY    0x800020C0
37 #define TEE_ALG_HKDF_SHA224_DERIVE_KEY  0x800030C0
```

```

38 #define TEE_ALG_HKDF_SHA256_DERIVE_KEY 0x800040C0
39 #define TEE_ALG_HKDF_SHA384_DERIVE_KEY 0x800050C0
40 #define TEE_ALG_HKDF_SHA512_DERIVE_KEY 0x800060C0
41
42 #define TEE_TYPE_HKDF_IKM 0xA10000C0
43
44 #define TEE_ATTR_HKDF_IKM 0xC00001C0
45 #define TEE_ATTR_HKDF_SALT 0xD00002C0
46 #define TEE_ATTR_HKDF_INFO 0xD00003C0
47 #define TEE_ATTR_HKDF_OKM_LENGTH 0xF00004C0
48
49 /*
50  * Concatenation Key Derivation Function (Concat KDF)
51  * NIST SP 800-56A section 5.8.1
52  */
53
54 #define TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY 0x800020C1
55 #define TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY 0x800030C1
56 #define TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY 0x800040C1
57 #define TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY 0x800050C1
58 #define TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY 0x800060C1
59
60 #define TEE_TYPE_CONCAT_KDF_Z 0xA10000C1
61
62 #define TEE_ATTR_CONCAT_KDF_Z 0xC00001C1
63 #define TEE_ATTR_CONCAT_KDF_OTHER_INFO 0xD00002C1
64 #define TEE_ATTR_CONCAT_KDF_DKM_LENGTH 0xF00003C1
65
66 /*
67  * PKCS #5 v2.0 Key Derivation Function 2 (PBKDF2)
68  * RFC 2898 section 5.2
69  * https://www.ietf.org/rfc/rfc2898.txt
70  */
71
72 #define TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY 0x800020C2
73
74 #define TEE_TYPE_PBKDF2_PASSWORD 0xA10000C2
75
76 #define TEE_ATTR_PBKDF2_PASSWORD 0xC00001C2
77 #define TEE_ATTR_PBKDF2_SALT 0xD00002C2
78 #define TEE_ATTR_PBKDF2_ITERATION_COUNT 0xF00003C2
79 #define TEE_ATTR_PBKDF2_DKM_LENGTH 0xF00004C2
80
81 /*
82  * Implementation-specific object storage constants
83  */
84
85 /* Storage is provided by the Rich Execution Environment (REE) */
86 #define TEE_STORAGE_PRIVATE_REE 0x80000000
87 /* Storage is the Replay Protected Memory Block partition of an eMMC device */
88 #define TEE_STORAGE_PRIVATE_RPMB 0x80000100
89 /* Was TEE_STORAGE_PRIVATE_SQL, which isn't supported any longer */
90 #define TEE_STORAGE_PRIVATE_SQL_RESERVED 0x80000200
91
92 /*
93  * Extension of "Memory Access Rights Constants"
94  * #define TEE_MEMORY_ACCESS_READ 0x00000001
95  * #define TEE_MEMORY_ACCESS_WRITE 0x00000002
96  * #define TEE_MEMORY_ACCESS_ANY_OWNER 0x00000004
97  *
98  * TEE_MEMORY_ACCESS_NONSECURE : if set TEE_CheckMemoryAccessRights()
99  * successfully returns only if target vmem range is mapped non-secure.
100  *
101  * TEE_MEMORY_ACCESS_SECURE : if set TEE_CheckMemoryAccessRights()
102  * successfully returns only if target vmem range is mapped secure.
103  */
104 /*
105 #define TEE_MEMORY_ACCESS_NONSECURE 0x10000000
106 #define TEE_MEMORY_ACCESS_SECURE 0x20000000
107
108 #endif /* TEE_API_DEFINES_EXTENSIONS_H */

```

10.13 ta-ref/api/include/tee_api_types.h File Reference

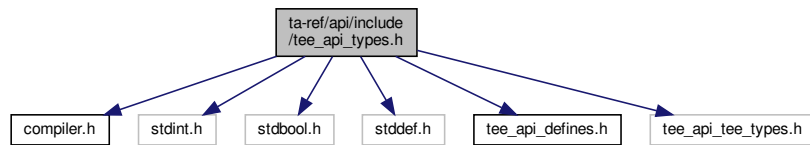
```

#include <compiler.h>
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <tee_api_defines.h>

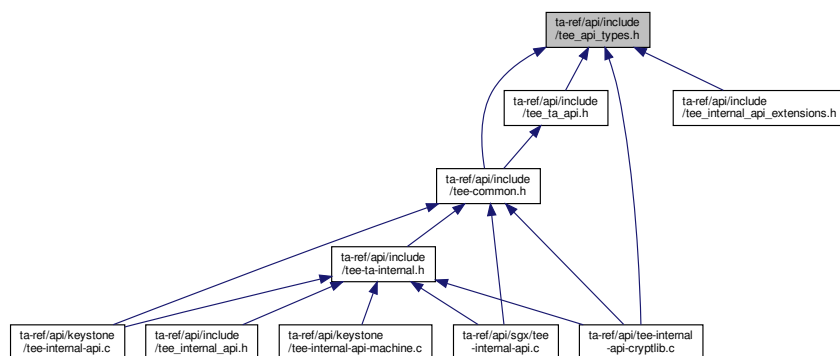
```

```
#include "tee_api_types.h"
```

Include dependency graph for tee_api_types.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct [TEE_UUID](#)
- struct [TEE_Identity](#)
- union [TEE_Param](#)
- struct [TEE_ObjectInfo](#)
- struct [TEE_Attribute](#)
- struct [TEE.OperationInfo](#)
- struct [TEE.OperationInfoKey](#)
- struct [TEE.OperationInfoMultiple](#)
- struct [TEE_Time](#)
- struct [TEE.SERReaderProperties](#)
- struct [TEE_SEAID](#)
- struct [pollfd](#)
- struct [addrinfo](#)

Macros

- [#define DMREQ_FINISH 0](#)
- [#define DMREQ_WRITE 1](#)
- [#define TEE_MEM_INPUT 0x00000001](#)
- [#define TEE_MEM_OUTPUT 0x00000002](#)
- [#define TEE_MEMREF_0_USED 0x00000001](#)
- [#define TEE_MEMREF_1_USED 0x00000002](#)
- [#define TEE_MEMREF_2_USED 0x00000004](#)
- [#define TEE_MEMREF_3_USED 0x00000008](#)
- [#define TEE_SE_READER_NAME_MAX 20](#)

Typedefs

- typedef uint32_t [TEE_Result](#)
- typedef struct __TEE_TASessionHandle * [TEE_TASessionHandle](#)
- typedef struct __TEE_PropSetHandle * [TEE_PropSetHandle](#)
- typedef struct __TEE_ObjectHandle * [TEE_ObjectHandle](#)
- typedef struct __TEE_ObjectEnumHandle * [TEE_ObjectEnumHandle](#)
- typedef struct __TEE_OperationHandle * [TEE_OperationHandle](#)
- typedef uint32_t [TEE_ObjectType](#)
- typedef uint32_t [TEE_BigInt](#)
- typedef uint32_t [TEE_BigIntFMM](#)
- typedef uint32_t TEE_BigIntFMMContext [__aligned](#)([__alignof](#)__(void *))
- typedef struct __TEE_SEServiceHandle * [TEE_SEServiceHandle](#)
- typedef struct __TEE_SEReaderHandle * [TEE_SEReaderHandle](#)
- typedef struct __TEE_SESessionHandle * [TEE_SESessionHandle](#)
- typedef struct __TEE_SEChannelHandle * [TEE_SEChannelHandle](#)
- typedef uint32_t [TEE_ErrorOrigin](#)
- typedef void * [TEE_Session](#)
- typedef unsigned long int [nfds_t](#)
- typedef unsigned int [socklen_t](#)

Enumerations

- enum [TEE_Whence](#) { [TEE_DATA_SEEK_SET](#) = 0 , [TEE_DATA_SEEK_CUR](#) = 1 , [TEE_DATA_SEEK_END](#) = 2 }
- enum [TEE_OperationMode](#) {
 [TEE_MODE_ENCRYPT](#) = 0 , [TEE_MODE_DECRYPT](#) = 1 , [TEE_MODE_SIGN](#) = 2 , [TEE_MODE_VERIFY](#) = 3 ,
 [TEE_MODE_MAC](#) = 4 , [TEE_MODE_DIGEST](#) = 5 , [TEE_MODE_DERIVE](#) = 6 }

10.13.1 Macro Definition Documentation

10.13.1.1 DMREQ_FINISH `#define DMREQ_FINISH 0`

10.13.1.2 DMREQ_WRITE `#define DMREQ_WRITE 1`

10.13.1.3 TEE_MEM_INPUT `#define TEE_MEM_INPUT 0x00000001`

10.13.1.4 TEE_MEM_OUTPUT `#define TEE_MEM_OUTPUT 0x00000002`

10.13.1.5 TEE_MEMREF_0_USED `#define TEE_MEMREF_0_USED 0x00000001`

10.13.1.6 TEE_MEMREF_1_USED `#define TEE_MEMREF_1_USED 0x00000002`

10.13.1.7 TEE_MEMREF_2_USED `#define TEE_MEMREF_2_USED 0x00000004`

10.13.1.8 TEE_MEMREF_3_USED `#define TEE_MEMREF_3_USED 0x00000008`

10.13.1.9 TEE_SE_READER_NAME_MAX `#define TEE_SE_READER_NAME_MAX 20`

10.13.2 Typedef Documentation

10.13.2.1 __aligned `typedef uint32_t TEE_BigIntFMMContext __aligned(__alignof__(void *))`

10.13.2.2 nfds_t `typedef unsigned long int nfds_t`

10.13.2.3 socklen_t `typedef unsigned int socklen_t`

10.13.2.4 TEE_BigInt `typedef uint32_t TEE_BigInt`

10.13.2.5 TEE_BigIntFMM `typedef uint32_t TEE_BigIntFMM`

10.13.2.6 TEE_ErrorOrigin `typedef uint32_t TEE_ErrorOrigin`

10.13.2.7 TEE_ObjectEnumHandle typedef struct __TEE_ObjectEnumHandle* [TEE_ObjectEnumHandle](#)

10.13.2.8 TEE_ObjectHandle typedef struct __TEE_ObjectHandle* [TEE_ObjectHandle](#)

10.13.2.9 TEE_ObjectType typedef uint32_t [TEE_ObjectType](#)

10.13.2.10 TEE_OperationHandle typedef struct __TEE_OperationHandle* [TEE_OperationHandle](#)

10.13.2.11 TEE_PropSetHandle typedef struct __TEE_PropSetHandle* [TEE_PropSetHandle](#)

10.13.2.12 TEE_Result typedef uint32_t [TEE_Result](#)

10.13.2.13 TEE_SEChannelHandle typedef struct __TEE_SEChannelHandle* [TEE_SEChannelHandle](#)

10.13.2.14 TEE_SEReaderHandle typedef struct __TEE_SEReaderHandle* [TEE_SEReaderHandle](#)

10.13.2.15 TEE_SEServiceHandle typedef struct __TEE_SEServiceHandle* [TEE_SEServiceHandle](#)

10.13.2.16 TEE_SESessionHandle typedef struct __TEE_SESessionHandle* [TEE_SESessionHandle](#)

10.13.2.17 TEE_Session typedef void* [TEE_Session](#)

10.13.2.18 TEE_TASessionHandle typedef struct __TEE_TASessionHandle* [TEE_TASessionHandle](#)

10.13.3 Enumeration Type Documentation

10.13.3.1 TEE_OperationMode enum [TEE_OperationMode](#)

Enumerator

TEE_MODE_ENCRYPT	
TEE_MODE_DECRYPT	
TEE_MODE_SIGN	
TEE_MODE_VERIFY	
TEE_MODE_MAC	
TEE_MODE_DIGEST	
TEE_MODE_DERIVE	

10.13.3.2 TEE_Whence enum TEE_Whence

Enumerator

TEE_DATA_SEEK_SET	
TEE_DATA_SEEK_CUR	
TEE_DATA_SEEK_END	

10.14 tee_api.types.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27
28 /* Based on GP TEE Internal API Specification Version 0.11 */
29 #ifndef TEE_API_TYPES_H
30 #define TEE_API_TYPES_H
31
32 #include <compiler.h>
33 #include <stdint.h>
34 #include <stdbool.h>
35 #include <stddef.h>
36 #include <tee_api_defines.h>
37 #include "tee_api_tee_types.h"
38
39 /*
40  * Common Definitions
41  */
42
43 typedef uint32_t TEE_Result;
```

```

44
45 typedef struct {
46     uint32_t timeLow;
47     uint16_t timeMid;
48     uint16_t timeHiAndVersion;
49     uint8_t clockSeqAndNode[8];
50 } TEE_UUID;
51
52 /*
53 * The TEE.Identity structure defines the full identity of a Client:
54 * - login is one of the TEE.LOGIN.XXX constants
55 * - uuid contains the client UUID or Nil if not applicable
56 */
57 typedef struct {
58     uint32_t login;
59     TEE_UUID uuid;
60 } TEE.Identity;
61
62 /*
63 * This union describes one parameter passed by the Trusted Core Framework
64 * to the entry points TA.OpenSessionEntryPoint or
65 * TA.InvokeCommandEntryPoint or by the TA to the functions
66 * TEE.OpenTASession or TEE.InvokeTACommand.
67 *
68 * Which of the field value or memref to select is determined by the
69 * parameter type specified in the argument paramTypes passed to the entry
70 * point.
71 */
72 typedef union {
73     struct {
74         void *buffer;
75         uint32_t size;
76     } memref;
77     struct {
78         uint32_t a;
79         uint32_t b;
80     } value;
81 } TEE_Param;
82
83 /*
84 * The type of opaque handles on TA Session. These handles are returned by
85 * the function TEE.OpenTASession.
86 */
87 typedef struct __TEE_TASessionHandle *TEE_TASessionHandle;
88
89 /*
90 * The type of opaque handles on property sets or enumerators. These
91 * handles are either one of the pseudo handles TEE.PROPSET.XXX or are
92 * returned by the function TEE.AllocatePropertyEnumerator.
93 */
94 typedef struct __TEE_PropSetHandle *TEE_PropSetHandle;
95
96 typedef struct __TEE_ObjectHandle *TEE_ObjectHandle;
97 typedef struct __TEE_ObjectEnumHandle *TEE_ObjectEnumHandle;
98 typedef struct __TEE_OperationHandle *TEE_OperationHandle;
99
100 /*
101 * Storage Definitions
102 */
103
104 typedef uint32_t TEE_ObjectType;
105
106 typedef struct {
107     uint32_t objectType;
108     __extension__ union {
109         uint32_t keySize; /* used in 1.1 spec */
110         uint32_t objectSize; /* used in 1.1.1 spec */
111     };
112     __extension__ union {
113         uint32_t maxKeySize; /* used in 1.1 spec */
114         uint32_t maxObjectSize; /* used in 1.1.1 spec */
115     };
116     uint32_t objectUsage;
117     uint32_t dataSize;
118     uint32_t dataPosition;
119     uint32_t handleFlags;
120 } TEE_ObjectInfo;
121
122 typedef enum {
123     TEE_DATA_SEEK_SET = 0,
124     TEE_DATA_SEEK_CUR = 1,
125     TEE_DATA_SEEK_END = 2
126 } TEE_Whence;
127
128 typedef struct {
129     uint32_t attributeID;
130     union {

```



```

131     struct {
132         void *buffer;
133         uint32_t length;
134     } ref;
135     struct {
136         uint32_t a, b;
137     } value;
138     } content;
139 } TEEAttribute;
140
141 #define DMREQ_FINISH 0
142 #define DMREQ_WRITE 1
143
144 /* Cryptographic Operations API */
145
146 typedef enum {
147     TEE_MODE_ENCRYPT = 0,
148     TEE_MODE_DECRYPT = 1,
149     TEE_MODE_SIGN = 2,
150     TEE_MODE_VERIFY = 3,
151     TEE_MODE_MAC = 4,
152     TEE_MODE_DIGEST = 5,
153     TEE_MODE_DERIVE = 6
154 } TEEOperationMode;
155
156 typedef struct {
157     uint32_t algorithm;
158     uint32_t operationClass;
159     uint32_t mode;
160     uint32_t digestLength;
161     uint32_t maxKeySize;
162     uint32_t keySize;
163     uint32_t requiredKeyUsage;
164     uint32_t handleState;
165 } TEEOperationInfo;
166
167 typedef struct {
168     uint32_t keySize;
169     uint32_t requiredKeyUsage;
170 } TEEOperationInfoKey;
171
172 typedef struct {
173     uint32_t algorithm;
174     uint32_t operationClass;
175     uint32_t mode;
176     uint32_t digestLength;
177     uint32_t maxKeySize;
178     uint32_t handleState;
179     uint32_t operationState;
180     uint32_t numberOfKeys;
181     TEEOperationInfoKey keyInformation[];
182 } TEEOperationInfoMultiple;
183
184 /* Time & Date API */
185
186 typedef struct {
187     uint32_t seconds;
188     uint32_t millis;
189 } TEETime;
190
191 /* TEE Arithmetical APIs */
192
193 typedef uint32_t TEEBigInt;
194
195 typedef uint32_t TEEBigIntFMM;
196
197 typedef uint32_t TEEBigIntFMMContext __aligned(__alignof__(void *));
198
199 /* Tee Secure Element APIs */
200
201 typedef struct __TEE_SEServiceHandle *TEE_SEServiceHandle;
202 typedef struct __TEE_SEReaderHandle *TEE_SEReaderHandle;
203 typedef struct __TEE_SESessionHandle *TEE_SESessionHandle;
204 typedef struct __TEE_SEChannelHandle *TEE_SEChannelHandle;
205
206 typedef struct {
207     bool sePresent;
208     bool teeOnly;
209     bool selectResponseEnable;
210 } TEE_SEReaderProperties;
211
212 typedef struct {
213     uint8_t *buffer;
214     size_t bufferLen;
215 } TEE_SEAID;
216
217 /* Other definitions */

```

```

218 typedef uint32_t TEE_ErrorOrigin;
219 typedef void *TEE_Session;
220
221 #define TEE_MEM_INPUT    0x00000001
222 #define TEE_MEM_OUTPUT   0x00000002
223
224 #define TEE_MEMREF_0_USED 0x00000001
225 #define TEE_MEMREF_1_USED 0x00000002
226 #define TEE_MEMREF_2_USED 0x00000004
227 #define TEE_MEMREF_3_USED 0x00000008
228
229 #define TEE_SE_READER_NAME_MAX 20
230
231 typedef unsigned long int nfds_t;
232
233 struct pollfd
234 {
235     int fd; /* File descriptor to poll. */
236     short int events; /* Types of events poller cares about. */
237     short int revents; /* Types of events that actually occurred. */
238 };
239
240 typedef unsigned int socklen_t;
241
242 struct addrinfo {
243     int ai_flags;
244     int ai_family;
245     int ai_socktype;
246     int ai_protocol;
247     socklen_t ai_addrlen;
248     struct sockaddr *ai_addr;
249     char *ai_canonname;
250     struct addrinfo *ai_next;
251 };
252
253
254
255 #endif /* TEE_API_TYPES_H */

```

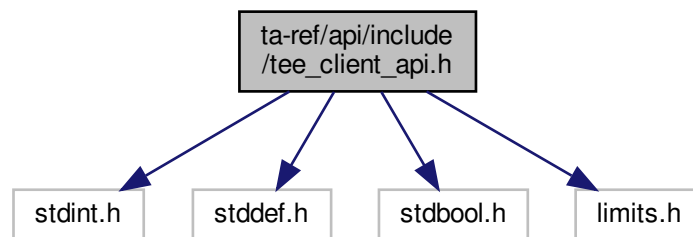
10.15 ta-ref/api/include/tee_client_api.h File Reference

```

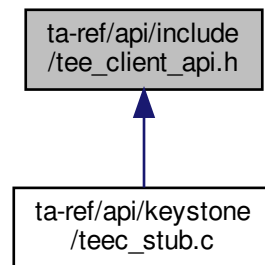
#include <stdint.h>
#include <stddef.h>
#include <stdbool.h>
#include <limits.h>

```

Include dependency graph for tee_client_api.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct [TEEC_Context](#)
- struct [TEEC_UUID](#)
- struct [TEEC_SharedMemory](#)
- struct [TEEC_TempMemoryReference](#)
- struct [TEEC_RegisteredMemoryReference](#)
- struct [TEEC_Value](#)
- union [TEEC_Parameter](#)
- struct [TEEC_Session](#)
- struct [TEEC_Operation](#)

Macros

- #define [TEEC_CONFIG_PAYLOAD_REF_COUNT](#) 4
- #define [TEEC_CONFIG_SHAREDMEM_MAX_SIZE](#) ULONG_MAX
- #define [TEEC_NONE](#) 0x00000000
- #define [TEEC_VALUE_INPUT](#) 0x00000001
- #define [TEEC_VALUE_OUTPUT](#) 0x00000002
- #define [TEEC_VALUE_INOUT](#) 0x00000003
- #define [TEEC_MEMREF_TEMP_INPUT](#) 0x00000005
- #define [TEEC_MEMREF_TEMP_OUTPUT](#) 0x00000006
- #define [TEEC_MEMREF_TEMP_INOUT](#) 0x00000007
- #define [TEEC_MEMREF_WHOLE](#) 0x0000000C
- #define [TEEC_MEMREF_PARTIAL_INPUT](#) 0x0000000D
- #define [TEEC_MEMREF_PARTIAL_OUTPUT](#) 0x0000000E
- #define [TEEC_MEMREF_PARTIAL_INOUT](#) 0x0000000F
- #define [TEEC_MEM_INPUT](#) 0x00000001
- #define [TEEC_MEM_OUTPUT](#) 0x00000002
- #define [TEEC_SUCCESS](#) 0x00000000
- #define [TEEC_ERROR_GENERIC](#) 0xFFFF0000
- #define [TEEC_ERROR_ACCESS_DENIED](#) 0xFFFF0001
- #define [TEEC_ERROR_CANCEL](#) 0xFFFF0002
- #define [TEEC_ERROR_ACCESS_CONFLICT](#) 0xFFFF0003
- #define [TEEC_ERROR_EXCESS_DATA](#) 0xFFFF0004

- #define `TEEC_ERROR_BAD_FORMAT` 0xFFFF0005
- #define `TEEC_ERROR_BAD_PARAMETERS` 0xFFFF0006
- #define `TEEC_ERROR_BAD_STATE` 0xFFFF0007
- #define `TEEC_ERROR_ITEM_NOT_FOUND` 0xFFFF0008
- #define `TEEC_ERROR_NOT_IMPLEMENTED` 0xFFFF0009
- #define `TEEC_ERROR_NOT_SUPPORTED` 0xFFFF000A
- #define `TEEC_ERROR_NO_DATA` 0xFFFF000B
- #define `TEEC_ERROR_OUT_OF_MEMORY` 0xFFFF000C
- #define `TEEC_ERROR_BUSY` 0xFFFF000D
- #define `TEEC_ERROR_COMMUNICATION` 0xFFFF000E
- #define `TEEC_ERROR_SECURITY` 0xFFFF000F
- #define `TEEC_ERROR_SHORT_BUFFER` 0xFFFF0010
- #define `TEEC_ERROR_EXTERNAL_CANCEL` 0xFFFF0011
- #define `TEEC_ERROR_TARGET_DEAD` 0xFFFF3024
- #define `TEEC_ORIGIN_API` 0x00000001
- #define `TEEC_ORIGIN_COMMS` 0x00000002
- #define `TEEC_ORIGIN_TEE` 0x00000003
- #define `TEEC_ORIGIN_TRUSTED_APP` 0x00000004
- #define `TEEC_LOGIN_PUBLIC` 0x00000000
- #define `TEEC_LOGIN_USER` 0x00000001
- #define `TEEC_LOGIN_GROUP` 0x00000002
- #define `TEEC_LOGIN_APPLICATION` 0x00000004
- #define `TEEC_LOGIN_USER_APPLICATION` 0x00000005
- #define `TEEC_LOGIN_GROUP_APPLICATION` 0x00000006
- #define `TEEC_PARAM_TYPES`(p0, p1, p2, p3) ((p0) | ((p1) << 4) | ((p2) << 8) | ((p3) << 12))
- #define `TEEC_PARAM_TYPE_GET`(p, i) (((p) >> (i * 4)) & 0xF)

Typedefs

- typedef uint32_t `TEEC_Result`

Functions

- `TEEC_Result TEEC_InitializeContext` (const char *name, `TEEC_Context` *context)
- void `TEEC_FinalizeContext` (`TEEC_Context` *context)
- `TEEC_Result TEEC_OpenSession` (`TEEC_Context` *context, `TEEC_Session` *session, const `TEEC_UUID` *destination, uint32_t connectionMethod, const void *connectionData, `TEEC_Operation` *operation, uint32_t *returnOrigin)
- void `TEEC_CloseSession` (`TEEC_Session` *session)
- `TEEC_Result TEEC_InvokeCommand` (`TEEC_Session` *session, uint32_t commandID, `TEEC_Operation` *operation, uint32_t *returnOrigin)
- `TEEC_Result TEEC_RegisterSharedMemory` (`TEEC_Context` *context, `TEEC_SharedMemory` *sharedMem)
- `TEEC_Result TEEC_AllocateSharedMemory` (`TEEC_Context` *context, `TEEC_SharedMemory` *sharedMem)
- void `TEEC_ReleaseSharedMemory` (`TEEC_SharedMemory` *sharedMemory)
- void `TEEC_RequestCancellation` (`TEEC_Operation` *operation)

10.15.1 Macro Definition Documentation

10.15.1.1 TEEC_CONFIG_PAYLOAD_REF_COUNT #define TEEC_CONFIG_PAYLOAD_REF_COUNT 4

10.15.1.2 TEEC_CONFIG_SHARED_MEM_MAX_SIZE #define TEEC_CONFIG_SHARED_MEM_MAX_SIZE ULONG_MAX

Defines the maximum size of a single shared memory block, in bytes, of both API allocated and API registered memory. There is no good value to put here (limits depend on specific config used), so this define does not provide any restriction in this implementation.

10.15.1.3 TEEC_ERROR_ACCESS_CONFLICT #define TEEC_ERROR_ACCESS_CONFLICT 0xFFFF0003

10.15.1.4 TEEC_ERROR_ACCESS_DENIED #define TEEC_ERROR_ACCESS_DENIED 0xFFFF0001

10.15.1.5 TEEC_ERROR_BAD_FORMAT #define TEEC_ERROR_BAD_FORMAT 0xFFFF0005

10.15.1.6 TEEC_ERROR_BAD_PARAMETERS #define TEEC_ERROR_BAD_PARAMETERS 0xFFFF0006

10.15.1.7 TEEC_ERROR_BAD_STATE #define TEEC_ERROR_BAD_STATE 0xFFFF0007

10.15.1.8 TEEC_ERROR_BUSY #define TEEC_ERROR_BUSY 0xFFFF000D

10.15.1.9 TEEC_ERROR_CANCEL #define TEEC_ERROR_CANCEL 0xFFFF0002

10.15.1.10 TEEC_ERROR_COMMUNICATION #define TEEC_ERROR_COMMUNICATION 0xFFFF000E

10.15.1.11 TEEC_ERROR_EXCESS_DATA #define TEEC_ERROR_EXCESS_DATA 0xFFFF0004

10.15.1.12 TEEC_ERROR_EXTERNAL_CANCEL #define TEEC_ERROR_EXTERNAL_CANCEL 0xFFFF0011

10.15.1.13 TEEC_ERROR_GENERIC #define TEEC_ERROR_GENERIC 0xFFFF0000

10.15.1.14 TEEC_ERROR_ITEM_NOT_FOUND #define TEEC_ERROR_ITEM_NOT_FOUND 0xFFFF0008

10.15.1.15 TEEC_ERROR_NO_DATA #define TEEC_ERROR_NO_DATA 0xFFFF000B

10.15.1.16 TEEC_ERROR_NOT_IMPLEMENTED #define TEEC_ERROR_NOT_IMPLEMENTED 0xFFFF0009

10.15.1.17 TEEC_ERROR_NOT_SUPPORTED #define TEEC_ERROR_NOT_SUPPORTED 0xFFFF000A

10.15.1.18 TEEC_ERROR_OUT_OF_MEMORY #define TEEC_ERROR_OUT_OF_MEMORY 0xFFFF000C

10.15.1.19 TEEC_ERROR_SECURITY #define TEEC_ERROR_SECURITY 0xFFFF000F

10.15.1.20 TEEC_ERROR_SHORT_BUFFER #define TEEC_ERROR_SHORT_BUFFER 0xFFFF0010

10.15.1.21 TEEC_ERROR_TARGET_DEAD #define TEEC_ERROR_TARGET_DEAD 0xFFFF3024

10.15.1.22 TEEC_LOGIN_APPLICATION #define TEEC_LOGIN_APPLICATION 0x00000004

10.15.1.23 TEEC_LOGIN_GROUP `#define TEEC_LOGIN_GROUP 0x00000002`

10.15.1.24 TEEC_LOGIN_GROUP_APPLICATION `#define TEEC_LOGIN_GROUP_APPLICATION 0x00000006`

10.15.1.25 TEEC_LOGIN_PUBLIC `#define TEEC_LOGIN_PUBLIC 0x00000000`

Session login methods, for use in [TEEC_OpenSession\(\)](#) as parameter connectionMethod. Type is uint32_t.

TEEC_LOGIN_PUBLIC No login data is provided. TEEC_LOGIN_USER Login data about the user running the Client Application process is provided. TEEC_LOGIN_GROUP Login data about the group running the Client Application process is provided. TEEC_LOGIN_APPLICATION Login data about the running Client Application itself is provided. TEEC_LOGIN_USER_APPLICATION Login data about the user and the running Client Application itself is provided. TEEC_LOGIN_GROUP_APPLICATION Login data about the group and the running Client Application itself is provided.

10.15.1.26 TEEC_LOGIN_USER `#define TEEC_LOGIN_USER 0x00000001`

10.15.1.27 TEEC_LOGIN_USER_APPLICATION `#define TEEC_LOGIN_USER_APPLICATION 0x00000005`

10.15.1.28 TEEC_MEM_INPUT `#define TEEC_MEM_INPUT 0x00000001`

Flag constants indicating the data transfer direction of memory in [TEEC_Parameter](#). TEEC_MEM_INPUT signifies data transfer direction from the client application to the TEE. TEEC_MEM_OUTPUT signifies data transfer direction from the TEE to the client application. Type is uint32_t.

TEEC_MEM_INPUT The Shared Memory can carry data from the client application to the Trusted Application. TEEC_MEM_OUTPUT The Shared Memory can carry data from the Trusted Application to the client application.

10.15.1.29 TEEC_MEM_OUTPUT `#define TEEC_MEM_OUTPUT 0x00000002`

10.15.1.30 TEEC_MEMREF_PARTIAL_INOUT `#define TEEC_MEMREF_PARTIAL_INOUT 0x0000000F`

10.15.1.31 TEEC_MEMREF_PARTIAL_INPUT `#define TEEC_MEMREF_PARTIAL_INPUT 0x0000000D`

10.15.1.32 TEEC_MEMREF_PARTIAL_OUTPUT `#define TEEC_MEMREF_PARTIAL_OUTPUT 0x0000000E`

10.15.1.33 TEEC_MEMREF_TEMP_INOUT `#define TEEC_MEMREF_TEMP_INOUT 0x00000007`

10.15.1.34 TEEC_MEMREF_TEMP_INPUT `#define TEEC_MEMREF_TEMP_INPUT 0x00000005`

10.15.1.35 TEEC_MEMREF_TEMP_OUTPUT `#define TEEC_MEMREF_TEMP_OUTPUT 0x00000006`

10.15.1.36 TEEC_MEMREF_WHOLE `#define TEEC_MEMREF_WHOLE 0x0000000C`

10.15.1.37 TEEC_NONE `#define TEEC_NONE 0x00000000`

Flag constants indicating the type of parameters encoded inside the operation payload ([TEEC.Operation](#)), Type is `uint32_t`.

TEEC_NONE The Parameter is not used

TEEC.VALUE_INPUT The Parameter is a [TEEC.Value](#) tagged as input.

TEEC.VALUE_OUTPUT The Parameter is a [TEEC.Value](#) tagged as output.

TEEC.VALUE_INOUT The Parameter is a [TEEC.Value](#) tagged as both as input and output, i.e., for which both the behaviors of **TEEC.VALUE_INPUT** and **TEEC.VALUE_OUTPUT** apply.

TEEC.MEMREF_TEMP_INPUT The Parameter is a [TEEC.TempMemoryReference](#) describing a region of memory which needs to be temporarily registered for the duration of the Operation and is tagged as input.

TEEC.MEMREF_TEMP_OUTPUT Same as **TEEC.MEMREF_TEMP_INPUT**, but the Memory Reference is tagged as output. The Implementation may update the size field to reflect the required output size in some use cases.

TEEC.MEMREF_TEMP_INOUT A Temporary Memory Reference tagged as both input and output, i.e., for which both the behaviors of **TEEC.MEMREF_TEMP_INPUT** and **TEEC.MEMREF_TEMP_OUTPUT** apply.

TEEC.MEMREF_WHOLE The Parameter is a Registered Memory Reference that refers to the entirety of its parent Shared Memory block. The parameter structure is a `TEEC_MemoryReference`. In this structure, the Implementation **MUST** read only the parent field and **MAY** update the size field when the operation completes.

TEEC.MEMREF_PARTIAL_INPUT A Registered Memory Reference structure that refers to a partial region of its parent Shared Memory block and is tagged as input.

TEEC.MEMREF_PARTIAL_OUTPUT Registered Memory Reference structure that refers to a partial region of its parent Shared Memory block and is tagged as output.

TEEC.MEMREF_PARTIAL_INOUT The Registered Memory Reference structure that refers to a partial region of its parent Shared Memory block and is tagged as both input and output, i.e., for which both the behaviors of **TEEC.MEMREF_PARTIAL_INPUT** and **TEEC.MEMREF_PARTIAL_OUTPUT** apply.

10.15.1.38 TEEC_ORIGIN_API `#define TEEC_ORIGIN_API 0x00000001`

Function error origins, of type `TEEC.ErrorOrigin`. These indicate where in the software stack a particular return value originates from.

`TEEC_ORIGIN_API` The error originated within the TEE Client API implementation. `TEEC_ORIGIN_COMMS` The error originated within the underlying communications stack linking the rich OS with the TEE. `TEEC_ORIGIN_TEE` The error originated within the common TEE code. `TEEC_ORIGIN_TRUSTED_APP` The error originated within the Trusted Application code.

10.15.1.39 TEEC_ORIGIN_COMMS `#define TEEC_ORIGIN_COMMS 0x00000002`**10.15.1.40 TEEC_ORIGIN_TEE** `#define TEEC_ORIGIN_TEE 0x00000003`**10.15.1.41 TEEC_ORIGIN_TRUSTED_APP** `#define TEEC_ORIGIN_TRUSTED_APP 0x00000004`**10.15.1.42 TEEC_PARAM_TYPE_GET** `#define TEEC_PARAM_TYPE_GET (`

```

    p,
    i ) (((p) >> (i * 4)) & 0xF)
```

Get the *i*-th param type from the paramType.

Parameters

<i>p</i>	The paramType.
<i>i</i>	The <i>i</i> -th parameter to get the type for.

10.15.1.43 TEEC_PARAM_TYPES `#define TEEC_PARAM_TYPES (`

```

    p0,
    p1,
    p2,
    p3 ) ((p0) | ((p1) << 4) | ((p2) << 8) | ((p3) << 12))
```

Encode the paramTypes according to the supplied types.

Parameters

<i>p0</i>	The first param type.
<i>p1</i>	The second param type.
<i>p2</i>	The third param type.
<i>p3</i>	The fourth param type.

10.15.1.44 TEEC_SUCCESS `#define TEEC_SUCCESS 0x00000000`

Return values. Type is `TEEC_Result`

`TEEC_SUCCESS` The operation was successful. `TEEC_ERROR_GENERIC` Non-specific cause. `TEEC_ERROR_ACCESS_DENIED` Access privileges are not sufficient. `TEEC_ERROR_CANCEL` The operation was canceled. `TEEC_ERROR_ACCESS_CONFLICT` Concurrent accesses caused conflict. `TEEC_ERROR_EXCESS_DATA` Too much data for the requested operation was passed. `TEEC_ERROR_BAD_FORMAT` Input data was of invalid format. `TEEC_ERROR_BAD_PARAMETERS` Input parameters were invalid. `TEEC_ERROR_BAD_STATE` Operation is not valid in the current state. `TEEC_ERROR_ITEM_NOT_FOUND` The requested data item is not found. `TEEC_ERROR_NOT_IMPLEMENTED` The requested operation should exist but is not yet implemented. `TEEC_ERROR_NOT_SUPPORTED` The requested operation is valid but is not supported in this implementation. `TEEC_ERROR_NO_DATA` Expected data was missing. `TEEC_ERROR_OUT_OF_MEMORY` System ran out of resources. `TEEC_ERROR_BUSY` The system is busy working on something else. `TEEC_ERROR_COMMUNICATION` Communication with a remote party failed. `TEEC_ERROR_SECURITY` A security fault was detected. `TEEC_ERROR_SHORT_BUFFER` The supplied buffer is too short for the generated output. `TEEC_ERROR_TARGET_DEAD` Trusted Application has panicked during the operation. Standard defined error codes.

10.15.1.45 TEEC_VALUE_INOUT `#define TEEC_VALUE_INOUT 0x00000003`**10.15.1.46 TEEC_VALUE_INPUT** `#define TEEC_VALUE_INPUT 0x00000001`**10.15.1.47 TEEC_VALUE_OUTPUT** `#define TEEC_VALUE_OUTPUT 0x00000002`**10.15.2 Typedef Documentation****10.15.2.1 TEEC_Result** `typedef uint32_t TEEC_Result`**10.15.3 Function Documentation****10.15.3.1 TEEC_AllocateSharedMemory()** `TEEC_Result TEEC_AllocateSharedMemory (`
`TEEC_Context * context,`
`TEEC_SharedMemory * sharedMem)`

`TEEC_AllocateSharedMemory()` - Allocate shared memory for TEE.

Parameters

<i>context</i>	The initialized TEE context structure in which scope to open the session.
<i>sharedMem</i>	Pointer to the allocated shared memory.

Returns

TEEC_SUCCESS The registration was successful.
TEEC_ERROR_OUT_OF_MEMORY Memory exhaustion.
TEEC_Result Something failed.

10.15.3.2 TEEC.CloseSession()

```
void TEEC.CloseSession (
    TEEC_Session * session )
```

[TEEC.CloseSession\(\)](#) - Closes the session which has been opened with the specific trusted application.

Parameters

<i>session</i>	The opened session to close.
----------------	------------------------------

10.15.3.3 TEEC.FinalizeContext()

```
void TEEC.FinalizeContext (
    TEEC_Context * context )
```

[TEEC.FinalizeContext\(\)](#) - Destroys a context holding connection information on the specific TEE.

This function destroys an initialized TEE context, closing the connection between the client application and the TEE. This function must only be called when all sessions related to this TEE context have been closed and all shared memory blocks have been released.

Parameters

<i>context</i>	The context to be destroyed.
----------------	------------------------------

[TEEC.FinalizeContext\(\)](#) - Destroys a context holding connection information on the specific TEE.

This function finalizes an initialized TEE context, closing the connection between the client application and the TEE. This function must only be called when all sessions related to this TEE context have been closed and all shared memory blocks have been released.

Parameters

<i>context</i>	The context to be finalized.
----------------	------------------------------

10.15.3.4 TEEC.InitializeContext() `TEEC_Result TEEC.InitializeContext (`
 `const char * name,`
 `TEEC_Context * context)`

[TEEC.InitializeContext\(\)](#) - Initializes a context holding connection information on the specific TEE, designated by the name string.

Parameters

<i>name</i>	A zero-terminated string identifying the TEE to connect to. If name is set to NULL, the default TEE is connected to. NULL is the only supported value in this version of the API implementation.
<i>context</i>	The context structure which is to be initialized.

Returns

TEEC.SUCCESS The initialization was successful.

TEEC.Result Something failed.

10.15.3.5 TEEC.InvokeCommand() `TEEC_Result TEEC.InvokeCommand (`
 `TEEC_Session * session,`
 `uint32_t commandID,`
 `TEEC_Operation * operation,`
 `uint32_t * returnOrigin)`

[TEEC.InvokeCommand\(\)](#) - Executes a command in the specified trusted application.

Parameters

<i>session</i>	A handle to an open connection to the trusted application.
<i>commandID</i>	Identifier of the command in the trusted application to invoke.
<i>operation</i>	An operation structure to use in the invoke command. May be set to NULL to signify no operation structure needed.
<i>returnOrigin</i>	A parameter which will hold the error origin if this function returns any value other than TEEC_SUCCESS.

Returns

TEEC_SUCCESS OpenSession successfully opened a new session.

TEEC_Result Something failed.

10.15.3.6 TEEC_OpenSession() `TEEC_Result TEEC_OpenSession (`

```
TEEC_Context * context,
TEEC_Session * session,
const TEEC_UUID * destination,
uint32_t connectionMethod,
const void * connectionData,
TEEC_Operation * operation,
uint32_t * returnOrigin )
```

`TEEC_OpenSession()` - Opens a new session with the specified trusted application.

Parameters

<i>context</i>	The initialized TEE context structure in which scope to open the session.
<i>session</i>	The session to initialize.
<i>destination</i>	A structure identifying the trusted application with which to open a session.
<i>connectionMethod</i>	The connection method to use.
<i>connectionData</i>	Any data necessary to connect with the chosen connection method. Not supported, should be set to NULL.
<i>operation</i>	An operation structure to use in the session. May be set to NULL to signify no operation structure needed.
<i>returnOrigin</i>	A parameter which will hold the error origin if this function returns any value other than TEEC_SUCCESS.

Returns

TEEC_SUCCESS OpenSession successfully opened a new session.

TEEC_Result Something failed.

10.15.3.7 TEEC_RegisterSharedMemory() `TEEC_Result TEEC_RegisterSharedMemory (`

```
TEEC_Context * context,
TEEC_SharedMemory * sharedMem )
```

`TEEC_RegisterSharedMemory()` - Register a block of existing memory as a shared block within the scope of the specified context.

Parameters

<i>context</i>	The initialized TEE context structure in which scope to open the session.
<i>sharedMem</i>	pointer to the shared memory structure to register.

Returns

TEEC_SUCCESS The registration was successful.

TEEC_ERROR_OUT_OF_MEMORY Memory exhaustion.

TEEC_Result Something failed.

10.15.3.8 TEEC.ReleaseSharedMemory() `void TEEC_ReleaseSharedMemory (`
`TEEC_SharedMemory * sharedMemory)`

[TEEC.ReleaseSharedMemory\(\)](#) - Free or deregister the shared memory.

Parameters

<i>sharedMem</i>	Pointer to the shared memory to be freed.
------------------	---

10.15.3.9 TEEC.RequestCancellation() `void TEEC_RequestCancellation (`
`TEEC_Operation * operation)`

[TEEC.RequestCancellation\(\)](#) - Request the cancellation of a pending open session or command invocation.

Parameters

<i>operation</i>	Pointer to an operation previously passed to open session or invoke.
------------------	--

10.16 tee_client_api.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  * Copyright (c) 2015, Linaro Limited
5  * All rights reserved.
6  *
7  * Redistribution and use in source and binary forms, with or without
8  * modification, are permitted provided that the following conditions are met:
9  *
10 * 1. Redistributions of source code must retain the above copyright notice,
11 * this list of conditions and the following disclaimer.
12 *
13 * 2. Redistributions in binary form must reproduce the above copyright notice,
14 * this list of conditions and the following disclaimer in the documentation
15 * and/or other materials provided with the distribution.
16 *
17 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
18 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
19 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
20 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
21 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
22 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
23 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
24 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
25 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

```

```

26  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
27  * POSSIBILITY OF SUCH DAMAGE.
28  */
29 #ifndef TEE_CLIENT_API_H
30 #define TEE_CLIENT_API_H
31
32 #ifdef __cplusplus
33 extern "C" {
34 #endif
35
36 #include <stdint.h>
37 #include <stddef.h>
38 #include <stdbool.h>
39 #include <limits.h>
40
41 /*
42  * Defines the number of available memory references in an open session or
43  * invoke command operation payload.
44  */
45 #define TEEC_CONFIG_PAYLOAD_REF_COUNT 4
46
47 #define TEEC_CONFIG_SHARED_MEM_MAX_SIZE ULONG_MAX
48
49 #define TEEC_NONE 0x00000000
50 #define TEEC_VALUE_INPUT 0x00000001
51 #define TEEC_VALUE_OUTPUT 0x00000002
52 #define TEEC_VALUE_INOUT 0x00000003
53 #define TEEC_MEMREF_TEMP_INPUT 0x00000005
54 #define TEEC_MEMREF_TEMP_OUTPUT 0x00000006
55 #define TEEC_MEMREF_TEMP_INOUT 0x00000007
56 #define TEEC_MEMREF_WHOLE 0x0000000C
57 #define TEEC_MEMREF_PARTIAL_INPUT 0x0000000D
58 #define TEEC_MEMREF_PARTIAL_OUTPUT 0x0000000E
59 #define TEEC_MEMREF_PARTIAL_INOUT 0x0000000F
60
61 #define TEEC_MEM_INPUT 0x00000001
62 #define TEEC_MEM_OUTPUT 0x00000002
63
64 #define TEEC_SUCCESS 0x00000000
65 #define TEEC_ERROR_GENERIC 0xFFFF0000
66 #define TEEC_ERROR_ACCESS_DENIED 0xFFFF0001
67 #define TEEC_ERROR_CANCEL 0xFFFF0002
68 #define TEEC_ERROR_ACCESS_CONFLICT 0xFFFF0003
69 #define TEEC_ERROR_EXCESS_DATA 0xFFFF0004
70 #define TEEC_ERROR_BAD_FORMAT 0xFFFF0005
71 #define TEEC_ERROR_BAD_PARAMETERS 0xFFFF0006
72 #define TEEC_ERROR_BAD_STATE 0xFFFF0007
73 #define TEEC_ERROR_ITEM_NOT_FOUND 0xFFFF0008
74 #define TEEC_ERROR_NOT_IMPLEMENTED 0xFFFF0009
75 #define TEEC_ERROR_NOT_SUPPORTED 0xFFFF000A
76 #define TEEC_ERROR_NO_DATA 0xFFFF000B
77 #define TEEC_ERROR_OUT_OF_MEMORY 0xFFFF000C
78 #define TEEC_ERROR_BUSY 0xFFFF000D
79 #define TEEC_ERROR_COMMUNICATION 0xFFFF000E
80 #define TEEC_ERROR_SECURITY 0xFFFF000F
81 #define TEEC_ERROR_SHORT_BUFFER 0xFFFF0010
82 #define TEEC_ERROR_EXTERNAL_CANCEL 0xFFFF0011
83 #define TEEC_ERROR_TARGET_DEAD 0xFFFF3024
84
85 #define TEEC_ORIGIN_API 0x00000001
86 #define TEEC_ORIGIN_COMMS 0x00000002
87 #define TEEC_ORIGIN_TEE 0x00000003
88 #define TEEC_ORIGIN_TRUSTED_APP 0x00000004
89
90 #define TEEC_LOGIN_PUBLIC 0x00000000
91 #define TEEC_LOGIN_USER 0x00000001
92 #define TEEC_LOGIN_GROUP 0x00000002
93 #define TEEC_LOGIN_APPLICATION 0x00000004
94 #define TEEC_LOGIN_USER_APPLICATION 0x00000005
95 #define TEEC_LOGIN_GROUP_APPLICATION 0x00000006
96
97 #define TEEC_PARAM_TYPES(p0, p1, p2, p3) \
98     ((p0) | ((p1) << 4) | ((p2) << 8) | ((p3) << 12))
99
100 #define TEEC_PARAM_TYPE_GET(p, i) (((p) >> (i * 4)) & 0xF)
101
102 typedef uint32_t TEEC_Result;
103
104 typedef struct {
105     /* Implementation defined */
106     int fd;
107     bool reg_mem;
108 } TEEC_Context;
109
110 typedef struct {
111     uint32_t timeLow;
112     uint16_t timeMid;

```

```

268     uint16_t timeHiAndVersion;
269     uint8_t clockSeqAndNode[8];
270 } TEEC.UUID;
271
272 typedef struct {
273     void *buffer;
274     size_t size;
275     uint32_t flags;
276     /*
277      * Implementation-Defined
278      */
279     int id;
280     size_t allocated_size;
281     void *shadow_buffer;
282     int registered_fd;
283     bool buffer_allocated;
284 } TEEC.SharedMemory;
285
286 typedef struct {
287     void *buffer;
288     size_t size;
289 } TEEC.TempMemoryReference;
290
291 typedef struct {
292     TEEC.SharedMemory *parent;
293     size_t size;
294     size_t offset;
295 } TEEC.RegisteredMemoryReference;
296
297 typedef struct {
298     uint32_t a;
299     uint32_t b;
300 } TEEC.Value;
301
302 typedef union {
303     TEEC.TempMemoryReference tmpref;
304     TEEC.RegisteredMemoryReference memref;
305     TEEC.Value value;
306 } TEEC.Parameter;
307
308 typedef struct {
309     /* Implementation defined */
310     TEEC.Context *ctx;
311     uint32_t session_id;
312 } TEEC.Session;
313
314 typedef struct {
315     uint32_t started;
316     uint32_t paramTypes;
317     TEEC.Parameter params[TEEC_CONFIG_PAYLOAD_REF_COUNT];
318     /* Implementation-Defined */
319     TEEC.Session *session;
320 } TEEC.Operation;
321
322 TEEC_Result TEEC_InitializeContext(const char *name, TEEC.Context *context);
323
324 void TEEC_FinalizeContext(TEEC.Context *context);
325
326 TEEC_Result TEEC_OpenSession(TEEC.Context *context,
327                             TEEC.Session *session,
328                             const TEEC.UUID *destination,
329                             uint32_t connectionMethod,
330                             const void *connectionData,
331                             TEEC.Operation *operation,
332                             uint32_t *returnOrigin);
333
334 void TEEC_CloseSession(TEEC.Session *session);
335
336 TEEC_Result TEEC_InvokeCommand(TEEC.Session *session,
337                                uint32_t commandID,
338                                TEEC.Operation *operation,
339                                uint32_t *returnOrigin);
340
341 TEEC_Result TEEC_RegisterSharedMemory(TEEC.Context *context,
342                                       TEEC.SharedMemory *sharedMem);
343
344 TEEC_Result TEEC_AllocateSharedMemory(TEEC.Context *context,
345                                       TEEC.SharedMemory *sharedMem);
346
347 void TEEC_ReleaseSharedMemory(TEEC.SharedMemory *sharedMemory);
348
349 void TEEC_RequestCancellation(TEEC.Operation *operation);
350
351 #ifdef __cplusplus
352 }
353 #endif

```

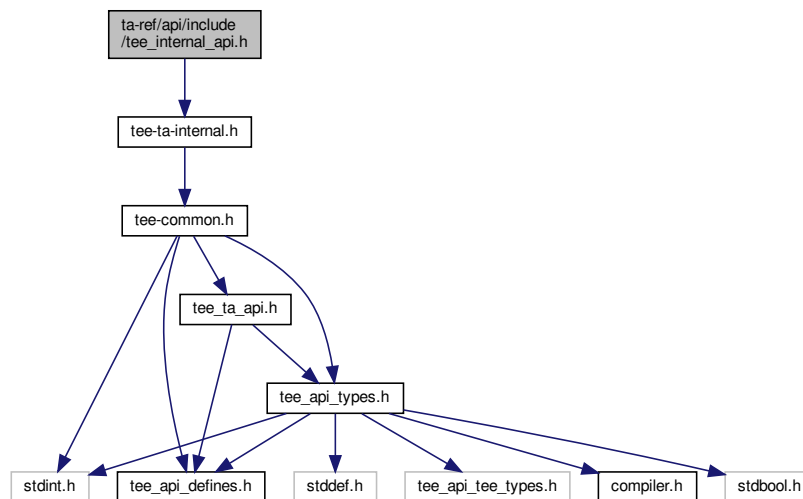


```
549 #endif
```

10.17 ta-ref/api/include/tee_internal_api.h File Reference

```
#include "tee-ta-internal.h"
```

Include dependency graph for tee_internal_api.h:



10.18 tee_internal_api.h

[Go to the documentation of this file.](#)

```
1 #include "tee-ta-internal.h"
```

10.19 ta-ref/api/include/tee_internal_api_extensions.h File Reference

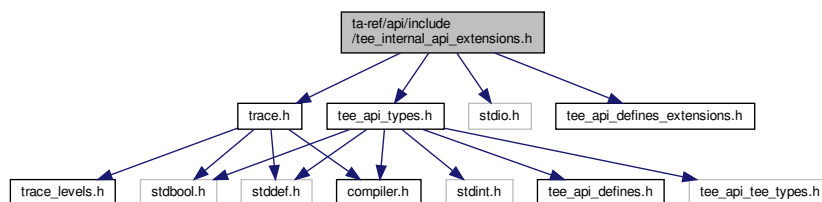
```
#include <trace.h>
```

```
#include <stdio.h>
```

```
#include <tee_api_defines_extensions.h>
```

```
#include <tee_api_types.h>
```

Include dependency graph for tee_internal_api_extensions.h:



Macros

- `#define TEE_USER_MEM_HINT_NO_FILL_ZERO 0x80000000`

Functions

- void `tee_user_mem_mark_heap` (void)
- `size_t` `tee_user_mem_check_heap` (void)
- `TEE_Result` `TEE_CacheClean` (char *buf, `size_t` len)
- `TEE_Result` `TEE_CacheFlush` (char *buf, `size_t` len)
- `TEE_Result` `TEE_CacheInvalidate` (char *buf, `size_t` len)
- void * `tee_map_zi` (`size_t` len, `uint32_t` flags)
- `TEE_Result` `tee_unmap` (void *buf, `size_t` len)
- `TEE_Result` `tee_uuid_from_str` (`TEE_UUID` *uuid, const char *s)

10.19.1 Macro Definition Documentation

10.19.1.1 TEE_USER_MEM_HINT_NO_FILL_ZERO `#define TEE_USER_MEM_HINT_NO_FILL_ZERO 0x80000000`

10.19.2 Function Documentation

10.19.2.1 TEE_CacheClean() `TEE_Result` `TEE_CacheClean` (
 char * *buf*,
 size_t *len*)

10.19.2.2 TEE_CacheFlush() `TEE_Result` `TEE_CacheFlush` (
 char * *buf*,
 size_t *len*)

10.19.2.3 TEE_CacheInvalidate() `TEE_Result` `TEE_CacheInvalidate` (
 char * *buf*,
 size_t *len*)

10.19.2.4 tee_map_zi() void * `tee_map_zi` (
 size_t *len*,
 uint32_t *flags*)

```

10.19.2.5 tee_unmap() TEE_Result tee_unmap (
    void * buf,
    size_t len )

```

```

10.19.2.6 tee_user_mem_check_heap() size_t tee_user_mem_check_heap (
    void )

```

```

10.19.2.7 tee_user_mem_mark_heap() void tee_user_mem_mark_heap (
    void )

```

```

10.19.2.8 tee_uuid_from_str() TEE_Result tee_uuid_from_str (
    TEE_UUID * uuid,
    const char * s )

```

10.20 tee_internal_api_extensions.h

[Go to the documentation of this file.](#)

```

1 /* SPDX-License-Identifier: BSD-2-Clause */
2 /*
3  * Copyright (c) 2014, STMicroelectronics International N.V.
4  */
5
6 #ifndef TEE_INTERNAL_API_EXTENSIONS_H
7 #define TEE_INTERNAL_API_EXTENSIONS_H
8
9 /* trace support */
10 #include <trace.h>
11 #include <stdio.h>
12 #include <tee_api_defines_extensions.h>
13 #include <tee_api_types.h>
14
15 void tee_user_mem_mark_heap(void);
16 size_t tee_user_mem_check_heap(void);
17 /* Hint implementation defines */
18 #define TEE_USER_MEM_HINT_NO_FILL_ZERO 0x80000000
19
20 /*
21  * Cache maintenance support (TA requires the CACHE_MAINTENANCE property)
22  *
23  * TEE_CacheClean() Write back to memory any dirty data cache lines. The line
24  * is marked as not dirty. The valid bit is unchanged.
25  *
26  * TEE_CacheFlush() Purges any valid data cache lines. Any dirty cache lines
27  * are first written back to memory, then the cache line is
28  * invalidated.
29  *
30  * TEE_CacheInvalidate() Invalidate any valid data cache lines. Any dirty line
31  * are not written back to memory.
32  */
33 TEE_Result TEE_CacheClean(char *buf, size_t len);
34 TEE_Result TEE_CacheFlush(char *buf, size_t len);
35 TEE_Result TEE_CacheInvalidate(char *buf, size_t len);
36
37 /*
38  * tee_map_zi() - Map zero initialized memory
39  * @len: Number of bytes
40  * @flags: 0 or TEE_MEMORY_ACCESS_ANY_OWNER to allow sharing with other TAs
41  *
42  * Returns valid pointer on success or NULL on error.
43  */
44 void *tee_map_zi(size_t len, uint32_t flags);
45

```

```

46 /*
47 * tee_unmap() - Unmap previously mapped memory
48 * @buf:      Buffer
49 * @len:      Number of bytes
50 *
51 * Note that supplied @buf and @len has to match exactly what has
52 * previously been returned by tee_map_zi().
53 *
54 * Return TEE_SUCCESS on success or TEE_ERROR_* on failure.
55 */
56 TEE_Result tee_unmap(void *buf, size_t len);
57
58 /*
59 * Convert a UUID string @s into a TEE_UUID @uuid
60 * Expected format for @s is: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
61 * 'x' being any hexadecimal digit (0-9a-fA-F)
62 */
63 TEE_Result tee_uuid_from_str(TEE_UUID *uuid, const char *s);
64
65 #endif

```

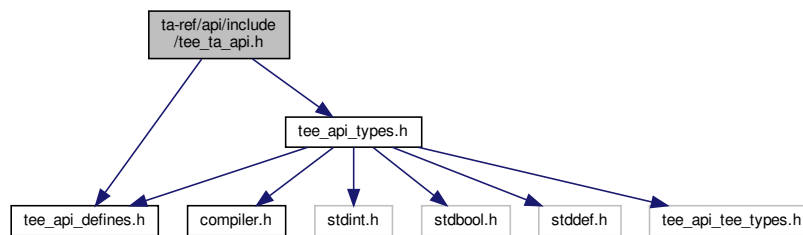
10.21 ta-ref/api/include/tee_ta_api.h File Reference

```

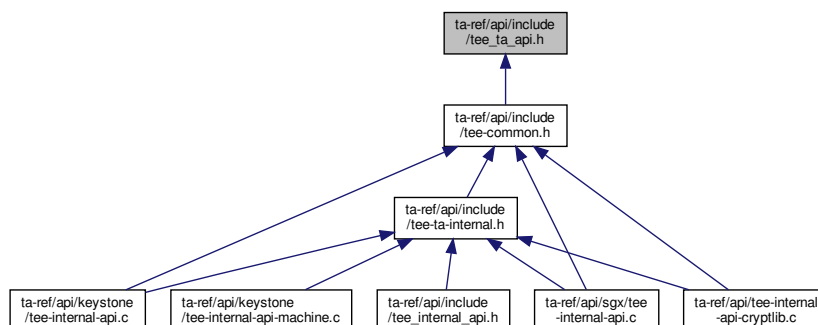
#include <tee_api_defines.h>
#include <tee_api_types.h>

```

Include dependency graph for tee_ta_api.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define TA_EXPORT`

Functions

- [TEE_Result](#) [TA_EXPORT](#) [TA_CreateEntryPoint](#) (void)
- void [TA_EXPORT](#) [TA_DestroyEntryPoint](#) (void)
- [TEE_Result](#) [TA_EXPORT](#) [TA_OpenSessionEntryPoint](#) (uint32_t paramTypes, [TEE_Param](#) params[[TEE_NUM_PARAMS](#)], void **sessionContext)
- void [TA_EXPORT](#) [TA_CloseSessionEntryPoint](#) (void *sessionContext)
- [TEE_Result](#) [TA_EXPORT](#) [TA_InvokeCommandEntryPoint](#) (void *sessionContext, uint32_t commandID, uint32_t paramTypes, [TEE_Param](#) params[[TEE_NUM_PARAMS](#)])

10.21.1 Macro Definition Documentation

10.21.1.1 TA_EXPORT `#define TA_EXPORT`

10.21.2 Function Documentation

10.21.2.1 TA_CloseSessionEntryPoint() void [TA_EXPORT](#) [TA_CloseSessionEntryPoint](#) (void * sessionContext)

10.21.2.2 TA_CreateEntryPoint() [TEE_Result](#) [TA_EXPORT](#) [TA_CreateEntryPoint](#) (void)

10.21.2.3 TA_DestroyEntryPoint() void [TA_EXPORT](#) [TA_DestroyEntryPoint](#) (void)

10.21.2.4 TA_InvokeCommandEntryPoint() [TEE_Result](#) [TA_EXPORT](#) [TA_InvokeCommandEntryPoint](#) (void * sessionContext, uint32_t commandID, uint32_t paramTypes, [TEE_Param](#) params[[TEE_NUM_PARAMS](#)])

10.21.2.5 TA_OpenSessionEntryPoint() [TEE_Result](#) [TA_EXPORT](#) [TA_OpenSessionEntryPoint](#) (uint32_t paramTypes, [TEE_Param](#) params[[TEE_NUM_PARAMS](#)], void ** sessionContext)

10.22 tee_ta_api.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27
28 /* Based on GP TEE Internal API Specification Version 0.22 */
29 #ifndef TEE_TA_API_H
30 #define TEE_TA_API_H
31
32 #include <tee_api_defines.h>
33 #include <tee_api_types.h>
34
35 /* This is a null define in STE TEE environment */
36 #define TA_EXPORT
37
38 /*
39  * TA Interface
40  *
41  * Each Trusted Application must provide the Implementation with a number
42  * of functions, collectively called the \TA interface". These functions
43  * are the entry points called by the Trusted Core Framework to create the
44  * instance, notify the instance that a new client is connecting, notify
45  * the instance when the client invokes a command, etc.
46  *
47  * Trusted Application Entry Points:
48  */
49
50 /*
51  * The function TA_CreateEntryPoint is the Trusted Application's
52  * constructor, which the Framework calls when it creates a new instance of
53  * the Trusted Application. To register instance data, the implementation
54  * of this constructor can use either global variables or the function
55  * TEE_InstanceSetData.
56  *
57  * Return Value:
58  * - TEE_SUCCESS: if the instance is successfully created, the function
59  * must return TEE_SUCCESS.
60  * - Any other value: if any other code is returned the instance is not
61  * created, and no other entry points of this instance will be called.
62  * The Framework MUST reclaim all resources and dereference all objects
63  * related to the creation of the instance.
64  *
65  * If this entry point was called as a result of a client opening a
66  * session, the error code is returned to the client and the session is
67  * not opened.
68  */
69 TEE_Result TA_EXPORT TA_CreateEntryPoint(void);
70
71 /*
72  * The function TA_DestroyEntryPoint is the Trusted Applications
73  * destructor, which the Framework calls when the instance is being
74  * destroyed.
75  *
76  * When the function TA_DestroyEntryPoint is called, the Framework
77  * guarantees that no client session is currently open. Once the call to
78  * TA_DestroyEntryPoint has been completed, no other entry point of this
79  * instance will ever be called.
80  *
81  * Note that when this function is called, all resources opened by the
82  * instance are still available. It is only after the function returns that
83  * the Implementation MUST start automatically reclaiming resources left

```

```

84 * opened.
85 *
86 * Return Value:
87 * This function can return no success or error code. After this function
88 * returns the Implementation MUST consider the instance destroyed and
89 * reclaims all resources left open by the instance.
90 */
91 void TA_EXPORT TA_DestroyEntryPoint(void);
92
93 /*
94 * The Framework calls the function TA.OpenSessionEntryPoint when a client
95 * requests to open a session with the Trusted Application. The open
96 * session request may result in a new Trusted Application instance being
97 * created as defined in section 4.5.
98 *
99 * The client can specify parameters in an open operation which are passed
100 * to the Trusted Application instance in the arguments paramTypes and
101 * params. These arguments can also be used by the Trusted Application
102 * instance to transfer response data back to the client. See section 4.3.6
103 * for a specification of how to handle the operation parameters.
104 *
105 * If this function returns TEE_SUCCESS, the client is connected to a
106 * Trusted Application instance and can invoke Trusted Application
107 * commands. When the client disconnects, the Framework will eventually
108 * call the TA.CloseSessionEntryPoint entry point.
109 *
110 * If the function returns any error, the Framework rejects the connection
111 * and returns the error code and the current TEE content of the parameters the
112 * client. The return origin is then set to TEE_ORIGIN_TRUSTED_APP.
113 *
114 * The Trusted Application instance can register a session data pointer by
115 * setting *pSessionContext. The value of this pointer is not interpreted
116 * by the Framework, and is simply passed back to other TA functions
117 * within this session. Note that *pSessionContext may be set with a pointer
118 * to a memory allocated by the Trusted Application instance or with
119 * anything else, like an integer, a handle etc. The Framework will not
120 * automatically free *pSessionContext when the session is closed; the
121 * Trusted Application instance is responsible for freeing memory if
122 * required.
123 *
124 * During the call to TA.OpenSessionEntryPoint the client may request to
125 * cancel the operation. See section 4.10 for more details on
126 * cancellations. If the call to TA.OpenSessionEntryPoint returns
127 * TEE_SUCCESS, the client must consider the session as successfully opened
128 * and explicitly close it if necessary.
129 *
130 * Parameters:
131 * - paramTypes: the types of the four parameters.
132 * - params: a pointer to an array of four parameters.
133 * - sessionContext: A pointer to a variable that can be filled by the
134 *   Trusted Application instance with an opaque void* data pointer
135 *
136 * Return Value:
137 * - TEE_SUCCESS if the session is successfully opened.
138 * - Any other value if the session could not be open.
139 *   o The error code may be one of the pre-defined codes, or may be a new
140 *     error code defined by the Trusted Application implementation itself.
141 */
142 TEE_Result TA_EXPORT TA_OpenSessionEntryPoint(uint32_t paramTypes,
143       TEE_Param params[TEE_NUM_PARAMS],
144       void **sessionContext);
145
146 /*
147 * The Framework calls this function to close a client session. During the
148 * call to this function the implementation can use any session functions.
149 *
150 * The Trusted Application implementation is responsible for freeing any
151 * resources consumed by the session being closed. Note that the Trusted
152 * Application cannot refuse to close a session, but can hold the closing
153 * until it returns from TA.CloseSessionEntryPoint. This is why this
154 * function cannot return an error code.
155 *
156 * Parameters:
157 * - sessionContext: The value of the void* opaque data pointer set by the
158 *   Trusted Application in the function TA.OpenSessionEntryPoint for this
159 *   session.
160 */
161 void TA_EXPORT TA_CloseSessionEntryPoint(void *sessionContext);
162
163 /*
164 * The Framework calls this function when the client invokes a command
165 * within the given session.
166 *
167 * The Trusted Application can access the parameters sent by the client
168 * through the paramTypes and params arguments. It can also use these
169 * arguments to transfer response data back to the client.
170 */

```

```

171 * During the call to TA_InvokeCommandEntryPoint the client may request to
172 * cancel the operation.
173 *
174 * A command is always invoked within the context of a client session.
175 * Thus, any session function can be called by the command implementation.
176 *
177 * Parameter:
178 * - sessionContext: The value of the void* opaque data pointer set by the
179 *   Trusted Application in the function TA_OpenSessionEntryPoint.
180 * - commandID: A Trusted Application-specific code that identifies the
181 *   command to be invoked.
182 * - paramTypes: the types of the four parameters.
183 * - params: a pointer to an array of four parameters.
184 *
185 * Return Value:
186 * - TEE_SUCCESS: if the command is successfully executed, the function
187 *   must return this value.
188 * - Any other value: if the invocation of the command fails for any
189 *   reason.
190 *   o The error code may be one of the pre-defined codes, or may be a new
191 *     error code defined by the Trusted Application implementation itself.
192 */
193
194 TEE_Result TA_EXPORT TA_InvokeCommandEntryPoint(void *sessionContext,
195         uint32_t commandID,
196         uint32_t paramTypes,
197         TEE_Param params[TEE_NUM_PARAMS]);
198
199 /*
200 * Correspondance Client Functions <--> TA Functions
201 *
202 * TEE_OpenSession or TEE_OpenTASession:
203 * If a new Trusted Application instance is needed to handle the session,
204 * TA_CreateEntryPoint is called.
205 * Then, TA_OpenSessionEntryPoint is called.
206 *
207 *
208 * TEE_InvokeCommand or TEE_InvokeTACommand:
209 * TA_InvokeCommandEntryPoint is called.
210 *
211 *
212 * TEE_CloseSession or TEE_CloseTASession:
213 * TA_CloseSessionEntryPoint is called.
214 * For a multi-instance TA or for a single-instance, non keep-alive TA, if
215 * the session closed was the last session on the instance, then
216 * TA_DestroyEntryPoint is called. Otherwise, the instance is kept until
217 * the TEE shuts down.
218 *
219 */
220
221 #endif

```

10.23 ta-ref/api/include/test_dev_key.h File Reference

Variables

- static const unsigned char [_sanctum_dev_secret_key](#) []
- static const size_t [_sanctum_dev_secret_key_len](#) = 64
- static const unsigned char [_sanctum_dev_public_key](#) []
- static const size_t [_sanctum_dev_public_key_len](#) = 32

10.23.1 Variable Documentation

10.23.1.1 [_sanctum_dev_public_key](#) const unsigned char [_sanctum_dev_public_key](#)[] [static]

Initial value:

```

= {
    0x0f, 0xaa, 0xd4, 0xff, 0x01, 0x17, 0x85, 0x83, 0xba, 0xa5, 0x88, 0x96,
    0x6f, 0x7c, 0x1f, 0xf3, 0x25, 0x64, 0xdd, 0x17, 0xd7, 0xdc, 0x2b, 0x46,
    0xcb, 0x50, 0xa8, 0x4a, 0x69, 0x27, 0x0b, 0x4c
}

```


10.23.1.2 `_sanctum_dev_public_key_len` `const size_t _sanctum_dev_public_key_len = 32 [static]`

10.23.1.3 `_sanctum_dev_secret_key` `const unsigned char _sanctum_dev_secret_key[] [static]`

Initial value:

```
= {
    0x40, 0xa0, 0x99, 0x47, 0x8c, 0xce, 0xfa, 0x3a, 0x06, 0x63, 0xab, 0xc9,
    0x5e, 0x7a, 0x1e, 0xc9, 0x54, 0xb4, 0xf5, 0xf6, 0x45, 0xba, 0xd8, 0x04,
    0xdb, 0x13, 0xe7, 0xd7, 0x82, 0x6c, 0x70, 0x73, 0x57, 0x6a, 0x9a, 0xb6,
    0x21, 0x60, 0xd9, 0xd1, 0xc6, 0xae, 0xdc, 0x29, 0x85, 0x2f, 0xb9, 0x60,
    0xee, 0x51, 0x32, 0x83, 0x5a, 0x16, 0x89, 0xec, 0x06, 0xa8, 0x72, 0x34,
    0x51, 0xaa, 0x0e, 0x4a
}
```

10.23.1.4 `_sanctum_dev_secret_key_len` `const size_t _sanctum_dev_secret_key_len = 64 [static]`

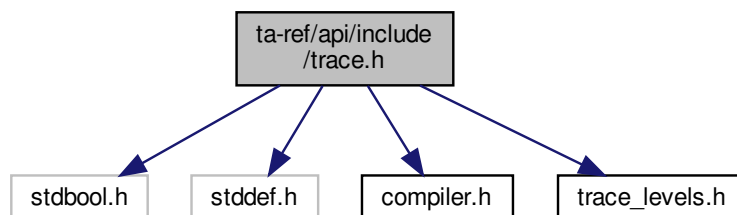
10.24 test_dev_key.h

[Go to the documentation of this file.](#)

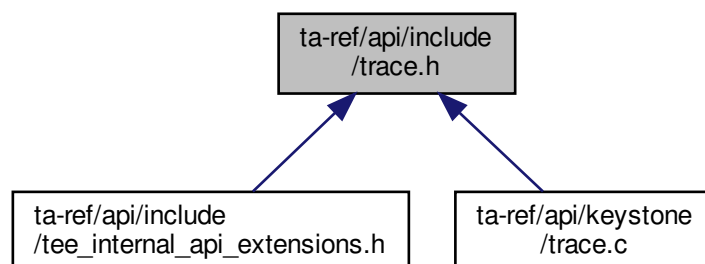
```
1 /* These are known device TESTING keys, use them for testing on platforms/qemu */
2
3 #warning Using TEST device root key. No integrity guarantee.
4 static const unsigned char _sanctum_dev_secret_key[] = {
5     0x40, 0xa0, 0x99, 0x47, 0x8c, 0xce, 0xfa, 0x3a, 0x06, 0x63, 0xab, 0xc9,
6     0x5e, 0x7a, 0x1e, 0xc9, 0x54, 0xb4, 0xf5, 0xf6, 0x45, 0xba, 0xd8, 0x04,
7     0xdb, 0x13, 0xe7, 0xd7, 0x82, 0x6c, 0x70, 0x73, 0x57, 0x6a, 0x9a, 0xb6,
8     0x21, 0x60, 0xd9, 0xd1, 0xc6, 0xae, 0xdc, 0x29, 0x85, 0x2f, 0xb9, 0x60,
9     0xee, 0x51, 0x32, 0x83, 0x5a, 0x16, 0x89, 0xec, 0x06, 0xa8, 0x72, 0x34,
10    0x51, 0xaa, 0x0e, 0x4a
11 };
12 static const size_t _sanctum_dev_secret_key_len = 64;
13
14 static const unsigned char _sanctum_dev_public_key[] = {
15     0x0f, 0xaa, 0xd4, 0xff, 0x01, 0x17, 0x85, 0x83, 0xba, 0xa5, 0x88, 0x96,
16     0x6f, 0x7c, 0x1f, 0xf3, 0x25, 0x64, 0xdd, 0x17, 0xd7, 0xdc, 0x2b, 0x46,
17     0xcb, 0x50, 0xa8, 0x4a, 0x69, 0x27, 0x0b, 0x4c
18 };
19 static const size_t _sanctum_dev_public_key_len = 32;
```

10.25 ta-ref/api/include/trace.h File Reference

```
#include <stdbool.h>
#include <stddef.h>
#include <compiler.h>
#include <trace_levels.h>
Include dependency graph for trace.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- #define `MAX_PRINT_SIZE` 256
- #define `MAX_FUNC_PRINT_SIZE` 32
- #define `TRACE_LEVEL` `TRACE_MAX`
- #define `trace_printf_helper`(level, level.ok, ...)
- #define `MSG`(...) (void)0
- #define `EMSG`(...) `trace_printf_helper`(`TRACE_ERROR`, true, __VA_ARGS__)
- #define `IMSG`(...) `trace_printf_helper`(`TRACE_INFO`, true, __VA_ARGS__)
- #define `DMSG`(...) `trace_printf_helper`(`TRACE_DEBUG`, true, __VA_ARGS__)
- #define `FMSG`(...) `trace_printf_helper`(`TRACE_FLOW`, true, __VA_ARGS__)
- #define `INMSG`(...) `FMSG`("> " __VA_ARGS__)
- #define `OUTMSG`(...) `FMSG`("< " __VA_ARGS__)
- #define `OUTRMSG`(r)
- #define `DHEXDUMP`(buf, len)
- #define `trace_printf_helper_raw`(level, level.ok, ...) `trace_printf`(NULL, 0, (level), (level.ok), __VA_ARGS__)
- #define `MSG_RAW`(...) (void)0
- #define `EMSG_RAW`(...) `trace_printf_helper_raw`(`TRACE_ERROR`, true, __VA_ARGS__)
- #define `IMSG_RAW`(...) `trace_printf_helper_raw`(`TRACE_INFO`, true, __VA_ARGS__)
- #define `DMSG_RAW`(...) `trace_printf_helper_raw`(`TRACE_DEBUG`, true, __VA_ARGS__)
- #define `FMSG_RAW`(...) `trace_printf_helper_raw`(`TRACE_FLOW`, true, __VA_ARGS__)
- #define `SMSG`(...) (void)0
- #define `EPRINT_STACK`() (void)0
- #define `IPRINT_STACK`() (void)0
- #define `DPRINT_STACK`() (void)0
- #define `FPRINT_STACK`() (void)0

Functions

- void `trace_ext_puts` (const char *str)
- int `trace_ext_get_thread_id` (void)
- void `trace_set_level` (int level)
- int `trace_get_level` (void)
- void `trace_printf` (const char *func, int line, int level, bool level.ok, const char *fmt,...) `_printf`(5)
- void `dhex_dump` (const char *function, int line, int level, const void *buf, int len)

Variables

- int `trace_level`
- const char `trace_ext_prefix` []

10.25.1 Macro Definition Documentation

10.25.1.1 DHEXDUMP `#define DHEXDUMP(
 buf,
 len)`

Value:

```
dhex_dump(__func__, __LINE__, TRACE_DEBUG, \  
buf, len)
```

10.25.1.2 DMSG `#define DMSG(
 ...) trace_printf_helper(TRACE_DEBUG, true, __VA_ARGS__)`

10.25.1.3 DMSG_RAW `#define DMSG_RAW(
 ...) trace_printf_helper_raw(TRACE_DEBUG, true, __VA_ARGS__)`

10.25.1.4 DPRINT_STACK `#define DPRINT_STACK() (void)0`

10.25.1.5 EMSG `#define EMSG(
 ...) trace_printf_helper(TRACE_ERROR, true, __VA_ARGS__)`

10.25.1.6 EMSG_RAW `#define EMSG_RAW(
 ...) trace_printf_helper_raw(TRACE_ERROR, true, __VA_ARGS__)`

10.25.1.7 EPRINT_STACK `#define EPRINT_STACK() (void)0`

10.25.1.8 FMSG `#define FMSG(
...) trace.printf.helper(TRACE_FLOW, true, __VA_ARGS__)`

10.25.1.9 FMSG_RAW `#define FMSG_RAW(
...) trace.printf.helper_raw(TRACE_FLOW, true, __VA_ARGS__)`

10.25.1.10 FPRINT_STACK `#define FPRINT_STACK() (void)0`

10.25.1.11 IMSG `#define IMSG(
...) trace.printf.helper(TRACE_INFO, true, __VA_ARGS__)`

10.25.1.12 IMSG_RAW `#define IMSG_RAW(
...) trace.printf.helper_raw(TRACE_INFO, true, __VA_ARGS__)`

10.25.1.13 INMSG `#define INMSG(
...) FMSG("> " __VA_ARGS__)`

10.25.1.14 IPRINT_STACK `#define IPRINT_STACK() (void)0`

10.25.1.15 MAX_FUNC_PRINT_SIZE `#define MAX_FUNC_PRINT_SIZE 32`

10.25.1.16 MAX_PRINT_SIZE `#define MAX_PRINT_SIZE 256`

10.25.1.17 MSG `#define MSG(
...) (void)0`

10.25.1.18 MSG_RAW `#define MSG_RAW(
...) (void)0`

10.25.1.19 OUTMSG `#define OUTMSG(
...) FMSG("< " __VA_ARGS__)`

10.25.1.20 OUTRMSG `#define OUTRMSG(
r)`

Value:

```
do {  
    OUTMSG("r=[%x]", r);  
    return r;  
} while (0)
```

10.25.1.21 SMSG `#define SMSG(
...) (void)0`

10.25.1.22 TRACE_LEVEL `#define TRACE_LEVEL TRACE_MAX`

10.25.1.23 trace_printf_helper `#define trace_printf_helper(
level,
level_ok,
...)`

Value:

```
trace_printf(__func__, __LINE__, (level), (level_ok), \  
__VA_ARGS__)
```

10.25.1.24 trace_printf_helper_raw `#define trace_printf_helper_raw(
level,
level_ok,
...) trace_printf(NULL, 0, (level), (level_ok), __VA_ARGS__)`

10.25.2 Function Documentation

10.25.2.1 dhex_dump() `void dhex_dump (`
 `const char * function,`
 `int line,`
 `int level,`
 `const void * buf,`
 `int len)`

10.25.2.2 trace_ext_get_thread_id() `int trace_ext_get_thread_id (`
 `void)`

10.25.2.3 trace_ext_puts() `void trace_ext_puts (`
 `const char * str)`

10.25.2.4 trace_get_level() `int trace_get_level (`
 `void)`

10.25.2.5 trace_printf() `void trace_printf (`
 `const char * func,`
 `int line,`
 `int level,`
 `bool level_ok,`
 `const char * fmt,`
 `...)`

10.25.2.6 trace_set_level() `void trace_set_level (`
 `int level)`

10.25.3 Variable Documentation

10.25.3.1 trace_ext_prefix `const char trace_ext_prefix[]` [extern]

10.25.3.2 trace_level `int trace_level` [extern]

10.26 trace.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27 #ifndef TRACE_H
28 #define TRACE_H
29
30 #include <stdbool.h>
31 #include <stddef.h>
32 #include <compiler.h>
33 #include <trace_levels.h>
34
35 #define MAX_PRINT_SIZE 256
36 #define MAX_FUNC_PRINT_SIZE 32
37
38 #ifndef TRACE_LEVEL
39 #define TRACE_LEVEL TRACE_MAX
40 #endif
41
42 /*
43  * Symbols provided by the entity that uses this API.
44  */
45 extern int trace_level;
46 extern const char trace_ext_prefix[];
47 void trace_ext_puts(const char *str);
48 int trace_ext_get_thread_id(void);
49 void trace_set_level(int level);
50 int trace_get_level(void);
51
52 /* Internal functions used by the macros below */
53 void trace_printf(const char *func, int line, int level, bool level_ok,
54                  const char *fmt, ...) _printf(5, 6);
55
56 #define trace_printf_helper(level, level_ok, ...) \
57     trace_printf(__func__, __LINE__, (level), (level_ok), \
58                 __VA_ARGS__)
59
60 /* Formatted trace tagged with level independent */
61 #if (TRACE_LEVEL <= 0)
62 #define MSG(...) (void)0
63 #else
64 #define MSG(...) trace_printf_helper(0, false, __VA_ARGS__)
65 #endif
66
67 /* Formatted trace tagged with TRACE_ERROR level */
68 #if (TRACE_LEVEL < TRACE_ERROR)
69 #define EMSG(...) (void)0
70 #else
71 #define EMSG(...) trace_printf_helper(TRACE_ERROR, true, __VA_ARGS__)
72 #endif
73
74 /* Formatted trace tagged with TRACE_INFO level */
75 #if (TRACE_LEVEL < TRACE_INFO)
76 #define MSG(...) (void)0
77 #else
78 #define MSG(...) trace_printf_helper(TRACE_INFO, true, __VA_ARGS__)
79 #endif
80
81 /* Formatted trace tagged with TRACE_DEBUG level */
82 #if (TRACE_LEVEL < TRACE_DEBUG)
83 #define DMSG(...) (void)0

```

```

84 #else
85 #define DMSG(...)    trace_printf_helper(TRACE_DEBUG, true, __VA_ARGS__)
86 #endif
87
88 /* Formatted trace tagged with TRACE_FLOW level */
89 #if (TRACE_LEVEL < TRACE_FLOW)
90 #define FMSG(...)    (void)0
91 #else
92 #define FMSG(...)    trace_printf_helper(TRACE_FLOW, true, __VA_ARGS__)
93 #endif
94
95 /* Formatted trace tagged with TRACE_FLOW level and prefix with '>' */
96 #define INMSG(...)    FMSG("> " __VA_ARGS__)
97 /* Formatted trace tagged with TRACE_FLOW level and prefix with '<' */
98 #define OUTMSG(...)    FMSG("< " __VA_ARGS__)
99 /* Formatted trace tagged with TRACE_FLOW level and prefix with '<' and print
100  * an error message if r != 0 */
101 #define OUTRMSG(r)    \
102     do {              \
103         OUTMSG("r=[%x]", r); \
104         return r;        \
105     } while (0)
106
107 void dhex_dump(const char *function, int line, int level,
108               const void *buf, int len);
109 #if (TRACE_LEVEL < TRACE_DEBUG)
110 #define DHEXDUMP(buf, len) (void)0
111 #else
112 #define DHEXDUMP(buf, len) dhex_dump(__func__, __LINE__, TRACE_DEBUG, \
113                                     buf, len)
114 #endif
115
116
117 /* Trace api without trace formatting */
118
119 #define trace_printf_helper_raw(level, level_ok, ...) \
120     trace_printf(NULL, 0, (level), (level_ok), __VA_ARGS__)
121
122 /* No formatted trace tagged with level independent */
123 #if (TRACE_LEVEL <= 0)
124 #define MSG_RAW(...)    (void)0
125 #else
126 #define MSG_RAW(...)    trace_printf_helper_raw(0, false, __VA_ARGS__)
127 #endif
128
129 /* No formatted trace tagged with TRACE_ERROR level */
130 #if (TRACE_LEVEL < TRACE_ERROR)
131 #define EMSG_RAW(...)    (void)0
132 #else
133 #define EMSG_RAW(...)    trace_printf_helper_raw(TRACE_ERROR, true, __VA_ARGS__)
134 #endif
135
136 /* No formatted trace tagged with TRACE_INFO level */
137 #if (TRACE_LEVEL < TRACE_INFO)
138 #define IMSG_RAW(...)    (void)0
139 #else
140 #define IMSG_RAW(...)    trace_printf_helper_raw(TRACE_INFO, true, __VA_ARGS__)
141 #endif
142
143 /* No formatted trace tagged with TRACE_DEBUG level */
144 #if (TRACE_LEVEL < TRACE_DEBUG)
145 #define DMSG_RAW(...)    (void)0
146 #else
147 #define DMSG_RAW(...)    trace_printf_helper_raw(TRACE_DEBUG, true, __VA_ARGS__)
148 #endif
149
150 /* No formatted trace tagged with TRACE_FLOW level */
151 #if (TRACE_LEVEL < TRACE_FLOW)
152 #define FMSG_RAW(...)    (void)0
153 #else
154 #define FMSG_RAW(...)    trace_printf_helper_raw(TRACE_FLOW, true, __VA_ARGS__)
155 #endif
156
157 #if (TRACE_LEVEL <= 0)
158 #define SMSG(...)    (void)0
159 #else
160 /*
161  * Synchronised flushed trace, an Always message straight to HW trace IP.
162  * Current only supported inside OP-TEE kernel, will be just like an EMSG()
163  * in another context.
164  */
165 #define SMSG(...)    \
166     trace_printf(__func__, __LINE__, TRACE_ERROR, true, __VA_ARGS__)
167 #endif /* TRACE_LEVEL */
168
169
170 #if defined(__KERNEL__) && defined(CFG_UNWIND)

```



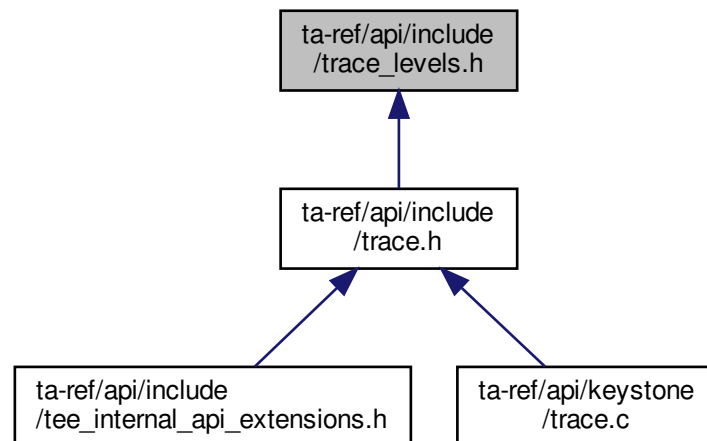
```

171 #include <kernel/unwind.h>
172 #define _PRINT_STACK
173 #endif
174
175 #if defined(_PRINT_STACK) && (TRACE_LEVEL >= TRACE_ERROR)
176 #define EPRINT_STACK() print_kernel_stack(TRACE_ERROR)
177 #else
178 #define EPRINT_STACK() (void)0
179 #endif
180
181 #if defined(_PRINT_STACK) && (TRACE_LEVEL >= TRACE_INFO)
182 #define IPRINT_STACK() print_kernel_stack(TRACE_INFO)
183 #else
184 #define IPRINT_STACK() (void)0
185 #endif
186
187 #if defined(_PRINT_STACK) && (TRACE_LEVEL >= TRACE_DEBUG)
188 #define DPRINT_STACK() print_kernel_stack(TRACE_DEBUG)
189 #else
190 #define DPRINT_STACK() (void)0
191 #endif
192
193 #if defined(_PRINT_STACK) && (TRACE_LEVEL >= TRACE_FLOW)
194 #define FPRINT_STACK() print_kernel_stack(TRACE_FLOW)
195 #else
196 #define FPRINT_STACK() (void)0
197 #endif
198
199 #if defined(__KERNEL__) && defined(CFG_UNWIND)
200 #undef _PRINT_STACK
201 #endif
202
203 #endif /* TRACE_H */

```

10.27 ta-ref/api/include/trace_levels.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define [TRACE_MIN](#) 1
- #define [TRACE_ERROR](#) [TRACE_MIN](#)
- #define [TRACE_INFO](#) 2

- `#define TRACE_DEBUG 3`
- `#define TRACE_FLOW 4`
- `#define TRACE_MAX TRACE_FLOW`
- `#define TRACE_PRINTF_LEVEL TRACE_ERROR`

10.27.1 Macro Definition Documentation

10.27.1.1 TRACE_DEBUG `#define TRACE_DEBUG 3`

10.27.1.2 TRACE_ERROR `#define TRACE_ERROR TRACE_MIN`

10.27.1.3 TRACE_FLOW `#define TRACE_FLOW 4`

10.27.1.4 TRACE_INFO `#define TRACE_INFO 2`

10.27.1.5 TRACE_MAX `#define TRACE_MAX TRACE_FLOW`

10.27.1.6 TRACE_MIN `#define TRACE_MIN 1`

10.27.1.7 TRACE_PRINTF_LEVEL `#define TRACE_PRINTF_LEVEL TRACE_ERROR`

10.28 trace.levels.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27 #ifndef TRACE_LEVELS_H
28 #define TRACE_LEVELS_H
29
30 /*
31  * Trace levels.
32  *
33  * ALWAYS is used when you always want a print to be seen, but it is not always
34  * an error.
35  *
36  * ERROR is used when some kind of error has happened, this is most likely the
37  * print you will use most of the time when you report some kind of error.
38  *
39  * INFO is used when you want to print some 'normal' text to the user.
40  * This is the default level.
41  *
42  * DEBUG is used to print extra information to enter deeply in the module.
43  *
44  * FLOW is used to print the execution flow, typically the in/out of functions.
45  *
46  */
47
48 #define TRACE_MIN          1
49 #define TRACE_ERROR        TRACE_MIN
50 #define TRACE_INFO         2
51 #define TRACE_DEBUG        3
52 #define TRACE_FLOW         4
53 #define TRACE_MAX          TRACE_FLOW
54
55 /* Trace level of the casual printf */
56 #define TRACE_PRINTF_LEVEL TRACE_ERROR
57
58 #endif /*TRACE_LEVELS_H*/

```

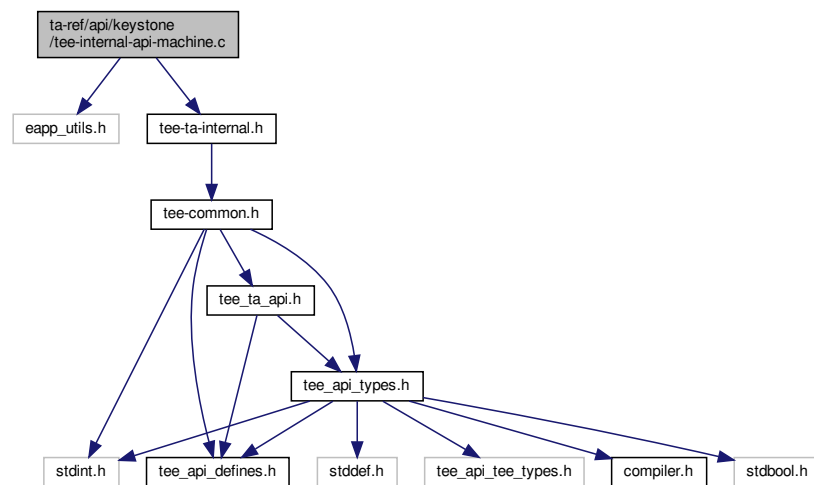
10.29 ta-ref/api/keystone/tee-internal-api-machine.c File Reference

```

#include "eapp_utils.h"
#include "tee-ta-internal.h"

```

Include dependency graph for tee-internal-api-machine.c:



Functions

- void `__attribute__((noreturn))`

10.29.1 Function Documentation

10.29.1.1 `__attribute__()` void `__attribute__` (
(noreturn))

TEE.Panic() - Raises a panic in the Trusted Application instance.

When a Trusted Application calls the TEE.Panic function, the current instance shall be destroyed and all the resources opened by the instance shall be reclaimed. All sessions opened from the panicking instance on another TA shall be gracefully closed and all cryptographic objects and operations shall be closed properly.

Parameters

<i>code</i>	An informative panic code defined by the TA.
-------------	--

Returns

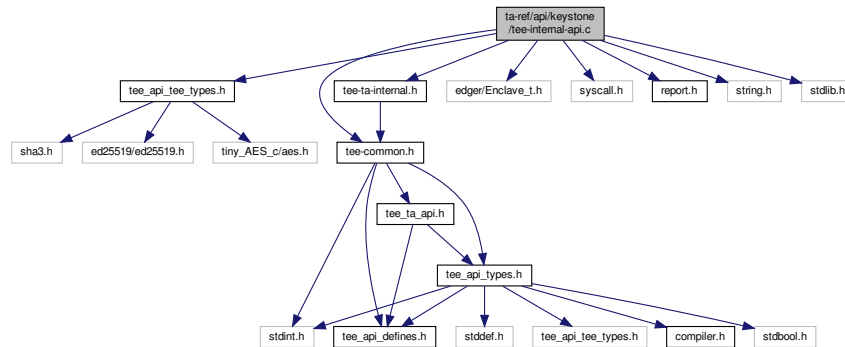
panic code will be returned.

10.30 ta-ref/api/keystone/tee-internal-api.c File Reference

```
#include "tee_api_tee_types.h"
#include "tee-common.h"
```

```
#include "tee-ta-internal.h"
#include "edger/Enclave_t.h"
#include "syscall.h"
#include "report.h"
#include <string.h>
#include <stdlib.h>
```

Include dependency graph for tee-internal-api.c:



Macros

- `#define O_RDONLY 0`
- `#define O_WRONLY 00001`
- `#define O_RDWR 00002`
- `#define O_CREAT 00100`
- `#define O_EXCL 00200`
- `#define O_TRUNC 01000`
- `#define FPERMS 0600`

Functions

- `void * TEE_Malloc (uint32_t size, uint32_t hint)`
- `void * TEE_Realloc (void *buffer, uint32_t newSize)`
- `void TEE_Free (void *buffer)`
- `void TEE_GetREETime (TEE_Time *time)`
Core Functions, Time Functions.
- `void TEE_GetSystemTime (TEE_Time *time)`
Core Functions, Time Functions.
- `TEE_Result GetRelTimeStart (uint64_t start)`
Core Functions, Time Functions.
- `TEE_Result GetRelTimeEnd (uint64_t end)`
Core Functions, Time Functions.
- `static int flags2flags (int flags)`
- `static int set_object_key (void *id, unsigned int idlen, TEE_ObjectHandle object)`
- `static TEE_Result OpenPersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle *object, int ocreat)`
- `TEE_Result TEE_CreatePersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle attributes, const void *initialData, uint32_t initialDataLen, TEE_ObjectHandle *object)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

- **TEE_Result TEE_OpenPersistentObject** (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, **TEE_ObjectHandle** *object)

Core Functions, Secure Storage Functions (data is isolated for each TA)

- **TEE_Result TEE_GetObjectInfo1** (**TEE_ObjectHandle** object, **TEE_ObjectInfo** *objectInfo)

Core Functions, Secure Storage Functions (data is isolated for each TA)

- **TEE_Result TEE_WriteObjectData** (**TEE_ObjectHandle** object, const void *buffer, uint32_t size)

Core Functions, Secure Storage Functions (data is isolated for each TA)

- **TEE_Result TEE_ReadObjectData** (**TEE_ObjectHandle** object, void *buffer, uint32_t size, uint32_t *count)

Core Functions, Secure Storage Functions (data is isolated for each TA)

- void **TEE_CloseObject** (**TEE_ObjectHandle** object)

Core Functions, Secure Storage Functions (data is isolated for each TA)

- void **TEE_GenerateRandom** (void *randomBuffer, uint32_t randomBufferLen)

Crypto, common.

10.30.1 Macro Definition Documentation

10.30.1.1 FPERMS #define FPERMS 0600

10.30.1.2 O_CREAT #define O_CREAT 00100

10.30.1.3 O_EXCL #define O_EXCL 00200

10.30.1.4 O_RDONLY #define O_RDONLY 0

10.30.1.5 O_RDWR #define O_RDWR 00002

10.30.1.6 O_TRUNC #define O_TRUNC 01000

10.30.1.7 O_WRONLY #define O_WRONLY 00001

10.30.2 Function Documentation

10.30.2.1 flags2flags() static int flags2flags (
int flags) [inline], [static]

flags2flags() - Checks the status for reading or writing of the file operational.

This function is used to check the status for reading or writing of the file operational.

Parameters

<i>flags</i>	Flags of the referencing node.
--------------	--------------------------------

Returns

ret if success.

10.30.2.2 GetRelTimeEnd() `TEE_Result GetRelTimeEnd (uint64_t end)`

Core Functions, Time Functions.

[GetRelTimeEnd\(\)](#) - finds the real time of the end timing.

This function prints the ending time.

Parameters

<i>end</i>	End timing
------------	------------

Returns

0 If success

10.30.2.3 GetRelTimeStart() `TEE_Result GetRelTimeStart (uint64_t start)`

Core Functions, Time Functions.

[GetRelTimeStart\(\)](#) - Gets the real time of the start timing.

This function prints the starting time.

Parameters

<i>start</i>	Start timing
--------------	--------------

Returns

0 on success

10.30.2.4 OpenPersistentObject() static `TEE_Result` OpenPersistentObject (
 uint32_t storageID,
 const void * objectID,
 uint32_t objectIDLen,
 uint32_t flags,
 TEE_ObjectHandle * object,
 int ocreat) [static]

`OpenPersistentObject()` - Opens a handle on an existing persistent object.

The flags parameter is a set of flags that controls the access rights and sharing permissions with which the object handle is opened. The value of the flags parameter is constructed by a bitwise-inclusive OR of flags TEE_DATA_FLAG_ACCESS_READ, the object is opened with the read access right. This allows the Trusted Application to call the function TEE_ReadObjectData. TEE_DATA_FLAG_ACCESS_WRITE, the object is opened with the write access right. TEE_DATA_FLAG_ACCESS_WRITE_META, the object is opened with the write-meta access right.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	length of the identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion.

Returns

0 if success else error occurred.

10.30.2.5 set_object_key() static int set_object_key (
 void * id,
 unsigned int idlen,
 TEE_ObjectHandle object) [static]

`set_object_key()` - Initialize report and then attest enclave with file.

This function describes the initialization of report, attest the enclave with file id and its length then assigned to ret. Based on "mbedtls" key encryption and decryption position of the object will be copied. Finally ret value returns on success else signature too short error will appear on failure.

Parameters

<i>id</i>	id of the object.
<i>idlen</i>	length of the id.
<i>object</i>	TEE_ObjectHandle type handle.

Returns

ret if success.

10.30.2.6 TEE_CloseObject() `void TEE_CloseObject (`
`TEE_ObjectHandle object)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_CloseObject\(\)](#) - Closes an opened object handle.

The object can be persistent or transient. For transient objects, TEE_CloseObject is equivalent to TEE_Free↔TransientObject.

Parameters

<i>object</i>	Handle of the object.
---------------	-----------------------

Returns

TEE_SUCCESS if success else error occurred.

10.30.2.7 TEE_CreatePersistentObject() `TEE_Result TEE_CreatePersistentObject (`
`uint32_t storageID,`
`const void * objectID,`
`uint32_t objectIDLen,`
`uint32_t flags,`
`TEE_ObjectHandle attributes,`
`const void * initialData,`
`uint32_t initialDataLen,`
`TEE_ObjectHandle * object)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_CreatePersistentObject\(\)](#) - Creates a persistent object with initial attributes.

In this function an initial data stream content returns either a handle on the created object or TEE_HANDLE_NULL upon failure.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>attributes</i>	A handle on a persistent object or an initialized transient object from which to take the persistent object attributes
<i>initialData</i>	The initial data content of the persistent object
<i>initialDataLen</i>	The initial data content of the persistent object
<i>object</i>	A pointer to the handle which contains the opened handle upon successful completion

Returns

0 if success else error occurred.

10.30.2.8 TEE.Free() `void TEEFree (`
`void * buffer)`

[TEE.Free\(\)](#) - causes the space pointed to by *buffer* to be deallocated; that is made available for further allocation.

This function describes if *buffer* is a NULL pointer, `TEE.Free` does nothing. Otherwise, it is a Programmer Error if the argument does not match a pointer previously returned by the `TEE.Malloc` or `TEE.Realloc` if the space has been deallocated by a call to `TEE.Free` or `TEE.Realloc`.

Parameters

<i>buffer</i>	The pointer to the memory block to be freed.
---------------	--

10.30.2.9 TEE.GenerateRandom() `void TEEGenerateRandom (`
`void * randomBuffer,`
`uint32_t randomBufferLen)`

Crypto, common.

`ocall_getrandom()` - For getting random data.

This function describes that the *retval* is returned based on the size of *buffer* by calling the functions `ocall_getrandom196` and `ocall_getrandom16`

Parameters

<i>buf</i>	character type buffer
<i>len</i>	size of the buffer
<i>flags</i>	unassigned integer flag

Returns

retval value will be returned based on length of buffer. [TEE.GenerateRandom\(\)](#) - Function generates random data.

This function generates random data of random bufferlength and is stored in to randomBuffer by calling ocall_↔getrandom(). If ret is not equal to randomBufferLen then TEE.Panic function is called.

Parameters

<i>randomBuffer</i>	Reference to generated random data
<i>randomBufferLen</i>	Byte length of requested random data

Returns

ocall version random data

10.30.2.10 TEE_GetObjectInfo1() `TEE_Result TEE_GetObjectInfo1 (`
`TEE_ObjectHandle object,`
`TEE_ObjectInfo * objectInfo)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_GetObjectInfo1\(\)](#) - Returns the characteristics of an object.

This function returns a handle which can be used to access the object's attributes and data stream.

Parameters

<i>objectInfo</i>	Pointer to a structure filled with the object information
<i>object</i>	Handle of the object

Returns

0 if success else error occurred.

10.30.2.11 TEE_GetREETime() `void TEE_GetREETime (`
`TEE_Time * time)`

Core Functions, Time Functions.

[TEE_GetREETime\(\)](#) - Retrieves the current REE system time.

This function retrieves the current time as seen from the point of view of the REE.

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds
-------------	--

10.30.2.12 TEE_GetSystemTime() `void TEE_GetSystemTime (`
`TEE_Time * time)`

Core Functions, Time Functions.

[TEE_GetSystemTime\(\)](#) - Retrieves the current system time.

This function describes the system time has an arbitrary implementation defined origin that can vary across TA instances. The minimum guarantee is that the system time shall be monotonic for a given TA instance.

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds
-------------	--

10.30.2.13 TEE_Malloc() `void * TEE_Malloc (`
`uint32_t size,`
`uint32_t hint)`

[TEE_Malloc\(\)](#) - Allocates space for an object whose size in bytes is specified in the parameter size.

This function describes the pointer returned is guaranteed to be aligned such that it may be assigned as a pointer to any basic C type. The valid hint values are a bitmask and can be independently set. This parameter allows Trusted Applications to refer to various pools of memory or to request special characteristics for the allocated memory by using an implementation-defined hint. Future versions of this specification may introduce additional standard hints.

Parameters

<i>size</i>	The size of the buffer to be allocated.
<i>hint</i>	A hint to the allocator.

Returns

Upon successful completion, with size not equal to zero, the function returns a pointer to the allocated space.

10.30.2.14 TEE_OpenPersistentObject() `TEE_Result TEE_OpenPersistentObject (`
 uint32_t *storageID*,
 const void * *objectID*,
 uint32_t *objectIDLen*,
 uint32_t *flags*,
 TEE_ObjectHandle * *object*)

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_OpenPersistentObject\(\)](#) - Opens a handle on an existing persistent object.

This function returns a handle which can be used to access the object's attributes and data stream.

Parameters

<i>storageID</i>	The storage to use
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion

Returns

0 if success else error occurred.

10.30.2.15 TEE_ReadObjectData() `TEE_Result TEE_ReadObjectData (`
 TEE_ObjectHandle *object*,
 void * *buffer*,
 uint32_t *size*,
 uint32_t * *count*)

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_ReadObjectData\(\)](#) - Attempts to read size bytes from the data stream associated with the object into the buffer pointed to by buffer.

The bytes are read starting at the position in the data stream currently stored in the object handle. The handle's position is incremented by the number of bytes actually read. On completion of TEE_ReadObjectData sets the number of bytes actually read in the "uint32_t" pointed to by count. The value written to *count may be less than size if the number of bytes until the end-of-stream is less than size. It is set to 0 if the position at the start of the read operation is at or beyond the end-of-stream. These are the only cases where *count may be less than size.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write
<i>count</i>	size of the buffer.

Returns

TEE_SUCCESS if success else error occurred.

10.30.2.16 TEE_Realloc() `void * TEE_Realloc (`
 `void * buffer,`
 `uint32_t newSize)`

[TEE_Realloc\(\)](#) - Changes the size of the memory object pointed to by *buffer* to the size specified by *new size*.

This function describes the content of the object remains unchanged up to the lesser of the new and old sizes. Space in excess of the old size contains unspecified content. If the new size of the memory object requires movement of the object, the space for the previous instantiation of the object is deallocated. If the space cannot be allocated, the original object remains allocated, and this function returns a NULL pointer.

Parameters

<i>buffer</i>	The pointer to the object to be reallocated.
<i>newSize</i>	The new size required for the object

Returns

Upon successful completion, TEE_Realloc returns a pointer to the (possibly moved) allocated space. If there is not enough available memory, TEE_Realloc returns a NULL pointer and the original buffer is still allocated and unchanged.

10.30.2.17 TEE_WriteObjectData() `TEE_Result TEE_WriteObjectData (`
 `TEE_ObjectHandle object,`
 `const void * buffer,`
 `uint32_t size)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_WriteObjectData\(\)](#) - Writes the buffer data in to persistent objects.

In this function it checks if object is present or not, the encryption/ decryption buffer is taken by calling `MBEDTLS_AES_CRYPT_CBC()` then that buffer data is encrypted and mapped to object. On the base of object creation TEE_SUCCESS appears else TEE_ERROR_GENERIC appears.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write

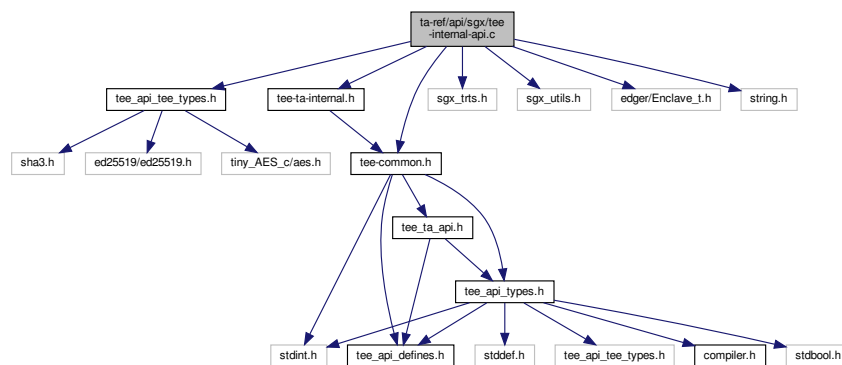
Returns

TEE_SUCCESS if success else error occurred.

10.31 ta-ref/api/sgx/tee-internal-api.c File Reference

```
#include "tee_api_tee_types.h"
#include "tee-common.h"
#include "tee-ta-internal.h"
#include "sgx_trts.h"
#include "sgx_utils.h"
#include "edger/Enclave_t.h"
#include <string.h>
```

Include dependency graph for tee-internal-api.c:



Macros

- #define [O_RDONLY](#) 0
- #define [O_WRONLY](#) 00001
- #define [O_RDWR](#) 00002
- #define [O_CREAT](#) 00100
- #define [O_EXCL](#) 00200
- #define [O_TRUNC](#) 01000
- #define [FPERMS](#) 0600

Functions

- void [__attribute__\(\(noreturn\)\)](#)
- void [TEE_GetREETime](#) (TEE_Time *time)
Core Functions, Time Functions.
- void [TEE_GetSystemTime](#) (TEE_Time *time)
Core Functions, Time Functions.
- [TEE_Result GetRelTimeStart](#) (uint64_t start)
Core Functions, Time Functions.
- [TEE_Result GetRelTimeEnd](#) (uint64_t end)
Core Functions, Time Functions.
- static int [flags2flags](#) (int flags)

- static int **set_object_key** (const void *id, unsigned int idlen, **TEE_ObjectHandle** object)
- static **TEE_Result** **OpenPersistentObject** (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, **TEE_ObjectHandle** *object, int ocreat)
- **TEE_Result** **TEE_CreatePersistentObject** (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, **TEE_ObjectHandle** attributes, const void *initialData, uint32_t initialDataLen, **TEE_ObjectHandle** *object)

Core Functions, Secure Storage Functions (data is isolated for each TA)

- **TEE_Result** **TEE_OpenPersistentObject** (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, **TEE_ObjectHandle** *object)

Core Functions, Secure Storage Functions (data is isolated for each TA)

- **TEE_Result** **TEE_GetObjectInfo1** (**TEE_ObjectHandle** object, **TEE_ObjectInfo** *objectInfo)

Core Functions, Secure Storage Functions (data is isolated for each TA)

- **TEE_Result** **TEE_WriteObjectData** (**TEE_ObjectHandle** object, const void *buffer, uint32_t size)

Core Functions, Secure Storage Functions (data is isolated for each TA)

- **TEE_Result** **TEE_ReadObjectData** (**TEE_ObjectHandle** object, void *buffer, uint32_t size, uint32_t *count)

Core Functions, Secure Storage Functions (data is isolated for each TA)

- void **TEE_CloseObject** (**TEE_ObjectHandle** object)

Core Functions, Secure Storage Functions (data is isolated for each TA)

- void **TEE_GenerateRandom** (void *randomBuffer, uint32_t randomBufferLen)

Crypto, common.

10.31.1 Macro Definition Documentation

10.31.1.1 FPERMS `#define FPERMS 0600`

10.31.1.2 O_CREAT `#define O_CREAT 00100`

10.31.1.3 O_EXCL `#define O_EXCL 00200`

10.31.1.4 O_RDONLY `#define O_RDONLY 0`

10.31.1.5 O_RDWR `#define O_RDWR 00002`

10.31.1.6 O_TRUNC `#define O_TRUNC 01000`

10.31.1.7 O_WRONLY `#define O_WRONLY 00001`

10.31.2 Function Documentation

10.31.2.1 __attribute__() `void __attribute__ (`
`(noreturn))`

TEE.Panic() - Raises a Panic in the Trusted Application instance

When a Trusted Application calls the TEE_Panic function, the current instance shall be destroyed and all the resources opened by the instance shall be reclaimed.

Parameters

<i>ec</i>	An informative panic code defined by the TA. May be displayed in traces if traces are available.
-----------	--

10.31.2.2 flags2flags() `static int flags2flags (`
`int flags) [inline], [static]`

[flags2flags\(\)](#) - Checks the status for reading or writing of the file operational.

This function is to check the status for reading or writing of the file operational.

Parameters

<i>flags</i>	Flags of the referencing node.
--------------	--------------------------------

Returns

0 if success else error occurred.

10.31.2.3 GetRelTimeEnd() `TEE_Result GetRelTimeEnd (`
`uint64_t end)`

Core Functions, Time Functions.

[GetRelTimeStart\(\)](#) - find the real time of the end timing.

This function prints the End timing.

Parameters

<i>end</i>	End timing
------------	------------

Returns

0 if success else error occurred

10.31.2.4 GetRelTimeStart() `TEE_Result GetRelTimeStart (`
`uint64_t start)`

Core Functions, Time Functions.

[GetRelTimeStart\(\)](#) - Gets the real time of the start timing.

This function prints the start timing.

Parameters

<i>start</i>	start timing
--------------	--------------

Returns

0 if success else error occurred.

10.31.2.5 OpenPersistentObject() `static TEE_Result OpenPersistentObject (`
`uint32_t storageID,`
`const void * objectID,`
`uint32_t objectIDLen,`
`uint32_t flags,`
`TEE_ObjectHandle * object,`
`int ocreat) [static]`

[OpenPersistentObject\(\)](#) - Opens a handle on an existing persistent object.

The flags parameter is a set of flags that controls the access rights and sharing permissions with which the object handle is opened. The value of the flags parameter is constructed by a bitwise-inclusive OR of flags TEE_DATA_FLAG_ACCESS_READ, the object is opened with the read access right. This allows the Trusted Application to call the function TEE_ReadObjectData. TEE_DATA_FLAG_ACCESS_WRITE, the object is opened with the write access right. TEE_DATA_FLAG_ACCESS_WRITE_META, the object is opened with the write-meta access right.

Parameters

<i>storageID</i>	The storage to use.
Paramter list continued on next page	

<i>objectID</i>	The object identifier
<i>objectIDLen</i>	length of the identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion.

Returns

0 if success else error occurred.

10.31.2.6 set_object_key() static int set_object_key (
const void * id,
unsigned int idlen,
TEE_ObjectHandle object) [static]

set_object_key - To initialize report and then attest enclave with file.

This function describes objectID as key_id to make the key dependent on it sgx report key is 128-bit. Fill another 128-bit with seal key. seal key doesn't change with enclave. Better than nothing, though. random nonce can not use for AES here because of persistency. the digest of attestation report and objectID as the last resort has been used.

Parameters

<i>id</i>	id of the object.
<i>idlen</i>	length of the id.
<i>object</i>	TEE_ObjectHandle type handle.

Returns

0 if success else error occurred.

10.31.2.7 TEE_CloseObject() void TEE_CloseObject (
TEE_ObjectHandle object)

Core Functions, Secure Storage Functions (data is isolated for each TA)

TEE_CloseObject() - Function closes an opened object handle.

The object can be persistent or transient. For transient objects, TEE_CloseObject is equivalent to TEE_Free↔TransientObject.

Parameters

<i>object</i>	Handle of the object
---------------	----------------------

Returns

TEE_SUCCESS if success else error occurred.

10.31.2.8 TEE_CreatePersistentObject() `TEE_Result TEE_CreatePersistentObject (`
`uint32_t storageID,`
`const void * objectID,`
`uint32_t objectIDLen,`
`uint32_t flags,`
`TEE_ObjectHandle attributes,`
`const void * initialData,`
`uint32_t initialDataLen,`
`TEE_ObjectHandle * object)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_CreatePersistentObject\(\)](#) - Creates a persistent object with initial attributes.

An initial data stream content, and optionally returns either a handle on the created object, or TEE_HANDLE_NULL upon failure.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>attributes</i>	A handle on a persistent object or an initialized transient object from which to take the persistent object attributes
<i>initialData</i>	The initial data content of the persistent object
<i>initialDataLen</i>	The initial data content of the persistent object
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion

Returns

0 if success, else error occurred.

10.31.2.9 TEE_GenerateRandom() `void TEE_GenerateRandom (`
`void * randomBuffer,`
`uint32_t randomBufferLen)`

Crypto, common.

[TEE.GenerateRandom\(\)](#) - Generates random data.

This function generates random data of random bufferlength and is stored in to randomBuffer by calling `sgx_read_rand()`.

Parameters

<i>randomBuffer</i>	Reference to generated random data
<i>randomBufferLen</i>	Byte length of requested random data

10.31.2.10 TEE.GetObjectInfo1() `TEE_Result TEE.GetObjectInfo1 (`
`TEE_ObjectHandle object,`
`TEE_ObjectInfo * objectInfo)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE.GetObjectInfo1\(\)](#) - Function returns the characteristics of an object.

It returns a handle that can be used to access the object's attributes and data stream.

Parameters

<i>objectInfo</i>	Pointer to a structure filled with the object information
<i>object</i>	Handle of the object

Returns

0 if success else error occurred.

10.31.2.11 TEE.GetREETime() `void TEE.GetREETime (`
`TEE_Time * time)`

Core Functions, Time Functions.

[TEE.GetREETime\(\)](#) - Function retrieves the current REE system time.

This function retrieves the current time as seen from the point of view of the REE.

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds.
-------------	---

10.31.2.12 TEE_GetSystemTime() `void TEE_GetSystemTime (`
`TEE_Time * time)`

Core Functions, Time Functions.

[TEE_GetSystemTime\(\)](#) - Retrieves the current system time.

The system time has an arbitrary implementation-defined origin that can vary across TA instances

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds.
-------------	---

10.31.2.13 TEE_OpenPersistentObject() `TEE_Result TEE_OpenPersistentObject (`
`uint32_t storageID,`
`const void * objectID,`
`uint32_t objectIDLen,`
`uint32_t flags,`
`TEE_ObjectHandle * object)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_OpenPersistentObject\(\)](#) - Opens a handle on an existing persistent object.

This function returns a handle that can be used to access the object's attributes and data stream.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion

Returns

0 if success, else error occurred.

10.31.2.14 TEE_ReadObjectData() `TEE_Result TEE_ReadObjectData (`
`TEE_ObjectHandle object,`
`void * buffer,`

```
uint32_t size,
uint32_t * count )
```

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_ReadObjectData\(\)](#) - Attempts to read size bytes from the data stream associated with the object object into the buffer pointed to by buffer.

The bytes are read starting at the position in the data stream currently stored in the object handle. The handle's position is incremented by the number of bytes actually read. On completion TEE_ReadObjectData sets the number of bytes actually read in the uint32_t pointed to by count. The value written to *count may be less than size if the number of bytes until the end-of-stream is less than size. It is set to 0 if the position at the start of the read operation is at or beyond the end-of-stream. These are the only cases where *count may be less than size.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write
<i>count</i>	size of the buffer.

Returns

TEE_SUCCESS if success, else error occurred.

10.31.2.15 TEE_WriteObjectData() [TEE_Result](#) TEE_WriteObjectData (
[TEE_ObjectHandle](#) object,
const void * buffer,
uint32_t size)

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_WriteObjectData\(\)](#) - writes size bytes from the buffer pointed to by buffer to the data stream associated with the open object handle object.

If the current data position points before the end-of-stream, then size bytes are written to the data stream, overwriting bytes starting at the current data position. If the current data position points beyond the stream's end, then the data stream is first extended with zero bytes until the length indicated by the data position indicator is reached, and then size bytes are written to the stream.

Parameters

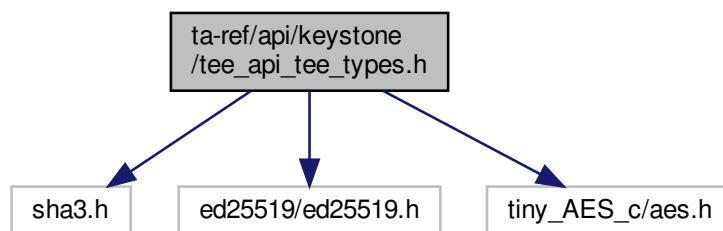
<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write

Returns

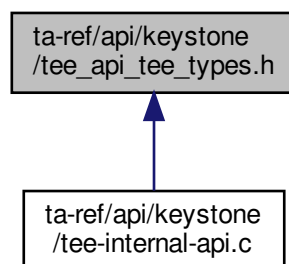
TEE_SUCCESS if success else error occurred.

10.32 ta-ref/api/keystone/tee_api_tee_types.h File Reference

```
#include "sha3.h"
#include "ed25519/ed25519.h"
#include "tiny_AES_c/aes.h"
Include dependency graph for tee_api_tee_types.h:
```



This graph shows which files directly or indirectly include this file:

**Classes**

- [struct __TEE.OperationHandle](#)
- [struct __TEE.ObjectHandle](#)

Macros

- #define MBEDCRYPT 1
- #define WOLFCRYPT 2
- #define AES256 1
- #define SHA_LENGTH (256/8)
- #define TEE_OBJECT_NONCE_SIZE 16
- #define TEE_OBJECT_KEY_SIZE 32
- #define TEE_OBJECT_SKEY_SIZE 64
- #define TEE_OBJECT_AAD_SIZE 16
- #define TEE_OBJECT_TAG_SIZE 16

10.32.1 Macro Definition Documentation

10.32.1.1 AES256 #define AES256 1

10.32.1.2 MBEDCRYPT #define MBEDCRYPT 1

10.32.1.3 SHA_LENGTH #define SHA_LENGTH (256/8)

10.32.1.4 TEE_OBJECT_AAD_SIZE #define TEE_OBJECT_AAD_SIZE 16

10.32.1.5 TEE_OBJECT_KEY_SIZE #define TEE_OBJECT_KEY_SIZE 32

10.32.1.6 TEE_OBJECT_NONCE_SIZE #define TEE_OBJECT_NONCE_SIZE 16

10.32.1.7 TEE_OBJECT_SKEY_SIZE #define TEE_OBJECT_SKEY_SIZE 64

10.32.1.8 TEE_OBJECT_TAG_SIZE #define TEE_OBJECT_TAG_SIZE 16

10.32.1.9 WOLFCRYPT #define WOLFCRYPT 2

10.33 tee_api_tee_types.h

[Go to the documentation of this file.](#)

```

1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                               and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10 *
11 * 1. Redistributions of source code must retain the above copyright notice,
12 * this list of conditions and the following disclaimer.
13 *
14 * 2. Redistributions in binary form must reproduce the above copyright notice,
15 * this list of conditions and the following disclaimer in the documentation
16 * and/or other materials provided with the distribution.
17 *
18 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28 * POSSIBILITY OF SUCH DAMAGE.
29 */
30
31 #ifndef TEE_API_TYPES_KEYSTONE_H
32 #define TEE_API_TYPES_KEYSTONE_H
33
34 #define MBEDCRYPT 1
35 #define WOLFCRYPT 2
36
37 #if CRYPTLIB==MBEDCRYPT
38 # define MBEDTLS_CONFIG_FILE "mbed-crypto-config.h"
39 # include "mbedtls/gcm.h"
40 # include "mbedtls/aes.h"
41 # include "sha3.h"
42 # include "ed25519/ed25519.h"
43 #define AES256 1
44 #elif CRYPTLIB==WOLFCRYPT
45 # define HAVE_AESGCM 1
46 # define HAVE_AES_CBC 1
47 # define HAVE_AES_DECRYPT 1
48 # define HAVE_FIPS 1
49 # define HAVE_FIPS_VERSION 2
50 # define HAVE_ED25519 1
51 # define HAVE_ED25519_SIGN 1
52 # define HAVE_ED25519_VERIFY 1
53 # define WOLFSSL_SHA512 1
54 # define WOLFSSL_SHA3 1
55 # define WOLFSSL_SHA3_SMALL 1
56 # define WOLFCRYPT_ONLY 1
57 # define WOLF_CRYPT_PORT_H
58 # include "wolfssl/wolfcrypt/sha3.h"
59 # include "wolfssl/wolfcrypt/aes.h"
60 # include "wolfssl/wolfcrypt/sha512.h"
61 # include "wolfssl/wolfcrypt/ed25519.h"
62 #else
63 #include "sha3.h"
64 #include "ed25519/ed25519.h"
65 #define AES256 1
66 # include "tiny_aes.c/aes.h"
67 #endif
68
69 #define SHA_LENGTH (256/8)
70 #define TEE_OBJECT_NONCE_SIZE 16
71 #define TEE_OBJECT_KEY_SIZE 32
72 #define TEE_OBJECT_SKEY_SIZE 64
73 #define TEE_OBJECT_AAD_SIZE 16
74 #define TEE_OBJECT_TAG_SIZE 16
75
76 struct __TEE_OperationHandle
77 {

```

```

78  int mode;
79  int flags;
80  int alg;
81  #if CRYPTLIB==MBEDCRYPT
82  sha3_ctx_t ctx;
83  mbedtls_aes_context aectx;
84  mbedtls_gcm_context aegcmctx;
85  #elif CRYPTLIB==WOLFCRYPT
86  wc_Sha3 ctx;
87  Aes aectx;
88  Aes aegcmctx;
89  unsigned int aegcm_aadsz;
90  unsigned char aegcm_aad[TEE_OBJECT_AAD_SIZE];
91  ed25519_key key;
92  #else
93  sha3_ctx_t ctx;
94  struct AES_ctx aectx;
95  #endif
96  int aegcm_state;
97  unsigned char aeiv[TEE_OBJECT_NONCE_SIZE];
98  unsigned char aekey[32];
99  unsigned char pubkey[TEE_OBJECT_KEY_SIZE];
100 unsigned char prikey[TEE_OBJECT_SKEY_SIZE];
101 };
102
103 struct __TEE_ObjectHandle
104 {
105     unsigned int type;
106     int flags;
107     int desc;
108     #if CRYPTLIB==MBEDCRYPT
109     mbedtls_aes_context persist_ctx;
110     unsigned char persist_iv[TEE_OBJECT_NONCE_SIZE];
111     #elif CRYPTLIB==WOLFCRYPT
112     Aes persist_ctx;
113     unsigned char persist_iv[TEE_OBJECT_NONCE_SIZE];
114     ed25519_key key;
115     #else
116     struct AES_ctx persist_ctx;
117     #endif
118     unsigned char public_key[TEE_OBJECT_KEY_SIZE];
119     unsigned char private_key[TEE_OBJECT_SKEY_SIZE];
120 };
121
122 // defined in tee_api_defines.h
123 // enum Data_Flag_Constants {
124 //     TEE_DATA_FLAG_ACCESS_READ = 0x00000001,
125 //     TEE_DATA_FLAG_ACCESS_WRITE = 0x00000002,
126 //     //TEE_DATA_FLAG_ACCESS_WRITE_META = 0x00000004,
127 //     //TEE_DATA_FLAG_SHARE_READ = 0x00000010,
128 //     //TEE_DATA_FLAG_SHARE_WRITE = 0x00000020,
129 //     TEE_DATA_FLAG_OVERWRITE = 0x00000400
130 // };
131 // enum Data_Flag_Constants {
132 //     TEE_DATA_FLAG_ACCESS_READ = 0x00000001,
133 //     TEE_DATA_FLAG_ACCESS_WRITE = 0x00000002,
134 //     //TEE_DATA_FLAG_ACCESS_WRITE_META = 0x00000004,
135 //     //TEE_DATA_FLAG_SHARE_READ = 0x00000010,
136 //     //TEE_DATA_FLAG_SHARE_WRITE = 0x00000020,
137 //     TEE_DATA_FLAG_OVERWRITE = 0x00000400
138 // };
139 #endif

```

10.34 ta-ref/api/optee/tee_api_tee_types.h File Reference

10.35 tee_api_tee_types.h

[Go to the documentation of this file.](#)

```
1 // empty
```

10.36 ta-ref/api/sgx/tee_api_tee_types.h File Reference

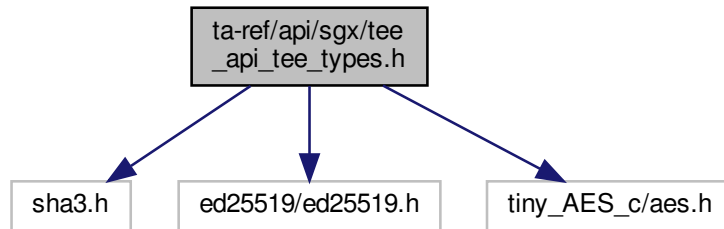
```

#include "sha3.h"
#include "ed25519/ed25519.h"

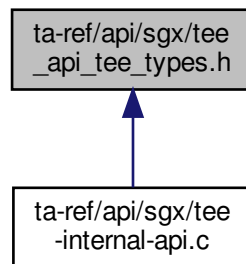
```

```
#include "tiny_AES_c/aes.h"
```

Include dependency graph for tee_api_tee_types.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct [__TEE_OperationHandle](#)
- struct [__TEE_ObjectHandle](#)

Macros

- #define [MBEDCRYPT](#) 1
- #define [WOLFCRYPT](#) 2
- #define [SHA_LENGTH](#) (256/8)
- #define [AES256](#) 1
- #define [TEE_OBJECT_NONCE_SIZE](#) 16
- #define [TEE_OBJECT_KEY_SIZE](#) 32
- #define [TEE_OBJECT_SKEY_SIZE](#) 64
- #define [TEE_OBJECT_AAD_SIZE](#) 16
- #define [TEE_OBJECT_TAG_SIZE](#) 16
- #define [TEE_HANDLE_NULL](#) 0

10.36.1 Macro Definition Documentation

10.36.1.1 AES256 `#define AES256 1`

10.36.1.2 MBEDCRYPT `#define MBEDCRYPT 1`

10.36.1.3 SHA_LENGTH `#define SHA_LENGTH (256/8)`

10.36.1.4 TEE_HANDLE_NULL `#define TEE_HANDLE_NULL 0`

10.36.1.5 TEE_OBJECT_AAD_SIZE `#define TEE_OBJECT_AAD_SIZE 16`

10.36.1.6 TEE_OBJECT_KEY_SIZE `#define TEE_OBJECT_KEY_SIZE 32`

10.36.1.7 TEE_OBJECT_NONCE_SIZE `#define TEE_OBJECT_NONCE_SIZE 16`

10.36.1.8 TEE_OBJECT_SKEY_SIZE `#define TEE_OBJECT_SKEY_SIZE 64`

10.36.1.9 TEE_OBJECT_TAG_SIZE `#define TEE_OBJECT_TAG_SIZE 16`

10.36.1.10 WOLFCRYPT `#define WOLFCRYPT 2`

10.37 tee_api_tee_types.h

[Go to the documentation of this file.](#)

```

1  /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                               and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10 *
11 * 1. Redistributions of source code must retain the above copyright notice,
12 * this list of conditions and the following disclaimer.
13 *
14 * 2. Redistributions in binary form must reproduce the above copyright notice,
15 * this list of conditions and the following disclaimer in the documentation
16 * and/or other materials provided with the distribution.
17 *
18 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28 * POSSIBILITY OF SUCH DAMAGE.
29 */
30
31 #ifndef TEE_API_TYPES_KEYSTONE_H
32 #define TEE_API_TYPES_KEYSTONE_H
33
34 #define MBEDCRYPT 1
35 #define WOLFCRYPT 2
36 #define SHA_LENGTH (256/8)
37
38 #include "sha3.h"
39 #include "ed25519/ed25519.h"
40 #define AES256 1
41 #if CRYPTLIB==MBEDCRYPT
42 # define MBEDTLS_CONFIG_FILE "mbed-crypto-config.h"
43 # include "mbedtls/gcm.h"
44 # include "mbedtls/aes.h"
45 #elif CRYPTLIB==WOLFCRYPT
46 # define HAVE_AESGCM 1
47 # define HAVE_AES_CBC 1
48 # define HAVE_AES_DECRYPT 1
49 # define HAVE_FIPS 1
50 # define HAVE_FIPS_VERSION 2
51 # define HAVE_ED25519 1
52 # define HAVE_ED25519_SIGN 1
53 # define HAVE_ED25519_VERIFY 1
54 # define WOLFSSL_SHA3 1
55 # define WOLF_CRYPT_PORT_H
56 # include "wolfssl/wolfcrypt/sha3.h"
57 # include "wolfssl/wolfcrypt/aes.h"
58 # include "wolfssl/wolfcrypt/sha512.h"
59 # include "wolfssl/wolfcrypt/ed25519.h"
60 #else
61 # include "tiny_AES.c/aes.h"
62 #endif
63
64 #define TEE_OBJECT_NONCE_SIZE 16
65 #define TEE_OBJECT_KEY_SIZE 32
66 #define TEE_OBJECT_SKEY_SIZE 64
67 #define TEE_OBJECT_AAD_SIZE 16
68 #define TEE_OBJECT_TAG_SIZE 16
69
70 struct __TEE_OperationHandle
71 {
72     int mode;
73     int flags;
74     int alg;
75     #if CRYPTLIB==MBEDCRYPT
76     sha3_ctx_t ctx;
77     mbedtls_aes_context aectx;
78     mbedtls_gcm_context aegmctx;
79     #elif CRYPTLIB==WOLFCRYPT
80     wc_Sha3 ctx;
81     Aes aectx;
82     Aes aegmctx;
83     unsigned int aegm_aadsz;

```

```

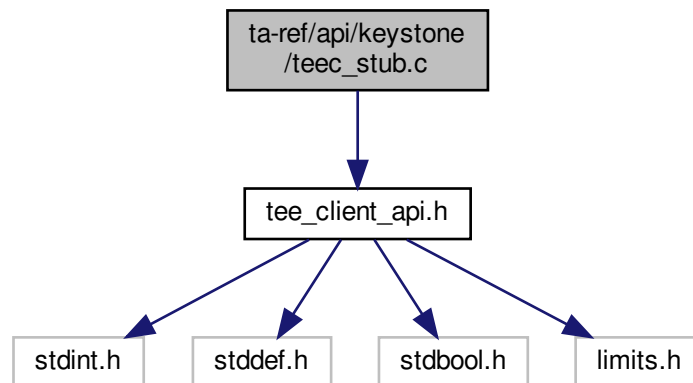
84 unsigned char aegcm_aad[TEE_OBJECT_AAD_SIZE];
85 ed25519_key key;
86 #else
87 sha3_ctx_t ctx;
88 struct AES_ctx aectx;
89 #endif
90 int aegcm_state;
91 unsigned char aeiv[TEE_OBJECT_NONCE_SIZE];
92 unsigned char aekey[32];
93 unsigned char pubkey[TEE_OBJECT_KEY_SIZE];
94 unsigned char prikey[TEE_OBJECT_SKEY_SIZE];
95 };
96
97 struct __TEE_ObjectHandle
98 {
99     unsigned int type;
100     int flags;
101     int desc;
102 #if CRYPTLIB==MBEDCRYPT
103     mbedtls_aes_context persist_ctx;
104     unsigned char persist_iv[TEE_OBJECT_NONCE_SIZE];
105 #elif CRYPTLIB==WOLFCRYPT
106     Aes persist_ctx;
107     unsigned char persist_iv[TEE_OBJECT_NONCE_SIZE];
108     ed25519_key key;
109 #else
110     struct AES_ctx persist_ctx;
111 #endif
112     unsigned char public_key[TEE_OBJECT_KEY_SIZE];
113     unsigned char private_key[TEE_OBJECT_SKEY_SIZE];
114 };
115
116 // Minimal constant definitions
117
118 #define TEE_HANDLE_NULL 0
119 #endif

```

10.38 ta-ref/api/keystone/teec_stub.c File Reference

#include <tee_client_api.h>

Include dependency graph for teec_stub.c:



Functions

- `TEEC_Result TEEC_InitializeContext` (const char *name, `TEEC_Context` *context)
- void `TEEC_FinalizeContext` (`TEEC_Context` *context)

- `TEEC_Result TEEC_OpenSession (TEEC_Context *context, TEEC_Session *session, const TEEC_UUID *destination, uint32_t connectionMethod, const void *connectionData, TEEC_Operation *operation, uint32_t *returnOrigin)`
- `void TEEC_CloseSession (TEEC_Session *session)`
- `TEEC_Result TEEC_RegisterSharedMemory (TEEC_Context *context, TEEC_SharedMemory *sharedMem)`
- `TEEC_Result TEEC_AllocateSharedMemory (TEEC_Context *context, TEEC_SharedMemory *sharedMem)`
- `void TEEC_ReleaseSharedMemory (TEEC_SharedMemory *sharedMemory)`
- `void TEEC_RequestCancellation (TEEC_Operation *operation)`

10.38.1 Function Documentation

10.38.1.1 TEEC_AllocateSharedMemory() `TEEC_Result TEEC_AllocateSharedMemory (TEEC_Context * context, TEEC_SharedMemory * sharedMem)`

`TEEC_AllocateSharedMemory()` - Allocate shared memory for TEE.

Parameters

<i>context</i>	The initialized TEE context structure in which scope to open the session.
<i>sharedMem</i>	Pointer to the allocated shared memory.

Returns

TEEC_SUCCESS The registration was successful.
 TEEC_ERROR_OUT_OF_MEMORY Memory exhaustion.
 TEEC_Result Something failed.

10.38.1.2 TEEC_CloseSession() `void TEEC_CloseSession (TEEC_Session * session)`

`TEEC_CloseSession()` - Closes the session which has been opened with the specific trusted application.

Parameters

<i>session</i>	The opened session to close.
----------------	------------------------------

10.38.1.3 TEEC_FinalizeContext() `void TEEC_FinalizeContext (TEEC_Context * context)`

[TEEC_FinalizeContext\(\)](#) - Destroys a context holding connection information on the specific TEE.

This function finalizes an initialized TEE context, closing the connection between the client application and the TEE. This function must only be called when all sessions related to this TEE context have been closed and all shared memory blocks have been released.

Parameters

<i>context</i>	The context to be finalized.
----------------	------------------------------

10.38.1.4 TEEC.InitializeContext() `TEEC_Result TEEC.InitializeContext (`
 `const char * name,`
 `TEEC_Context * context)`

[TEEC.InitializeContext\(\)](#) - Initializes a context holding connection information on the specific TEE, designated by the name string.

Parameters

<i>name</i>	A zero-terminated string identifying the TEE to connect to. If name is set to NULL, the default TEE is connected to. NULL is the only supported value in this version of the API implementation.
<i>context</i>	The context structure which is to be initialized.

Returns

TEEC.SUCCESS The initialization was successful.

TEEC.Result Something failed.

10.38.1.5 TEEC.OpenSession() `TEEC_Result TEEC.OpenSession (`
 `TEEC_Context * context,`
 `TEEC_Session * session,`
 `const TEEC_UUID * destination,`
 `uint32_t connectionMethod,`
 `const void * connectionData,`
 `TEEC_Operation * operation,`
 `uint32_t * returnOrigin)`

[TEEC.OpenSession\(\)](#) - Opens a new session with the specified trusted application.

Parameters

<i>context</i>	The initialized TEE context structure in which scope to open the session.
<i>session</i>	The session to initialize.
<i>destination</i>	A structure identifying the trusted application with which to open a session.
Paramter list continued on next page	

<i>connectionMethod</i>	The connection method to use.
<i>connectionData</i>	Any data necessary to connect with the chosen connection method. Not supported, should be set to NULL.
<i>operation</i>	An operation structure to use in the session. May be set to NULL to signify no operation structure needed.
<i>returnOrigin</i>	A parameter which will hold the error origin if this function returns any value other than TEEC_SUCCESS.

Returns

TEEC_SUCCESS OpenSession successfully opened a new session.

TEEC_Result Something failed.

10.38.1.6 TEEC_RegisterSharedMemory() `TEEC_Result TEEC_RegisterSharedMemory (`
`TEEC_Context * context,`
`TEEC_SharedMemory * sharedMem)`

[TEEC_RegisterSharedMemory\(\)](#) - Register a block of existing memory as a shared block within the scope of the specified context.

Parameters

<i>context</i>	The initialized TEE context structure in which scope to open the session.
<i>sharedMem</i>	pointer to the shared memory structure to register.

Returns

TEEC_SUCCESS The registration was successful.

TEEC_ERROR_OUT_OF_MEMORY Memory exhaustion.

TEEC_Result Something failed.

10.38.1.7 TEEC_ReleaseSharedMemory() `void TEEC_ReleaseSharedMemory (`
`TEEC_SharedMemory * sharedMemory)`

[TEEC_ReleaseSharedMemory\(\)](#) - Free or deregister the shared memory.

Parameters

<i>sharedMem</i>	Pointer to the shared memory to be freed.
------------------	---

10.38.1.8 TEEC.RequestCancellation() void TEEC.RequestCancellation (
 TEEC_Operation * operation)

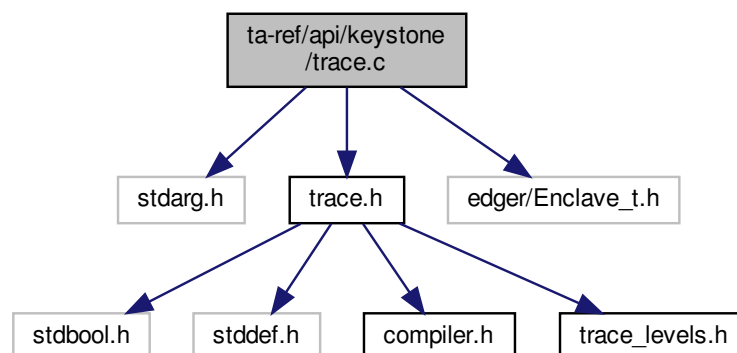
[TEEC.RequestCancellation\(\)](#) - Request the cancellation of a pending open session or command invocation.

Parameters

<i>operation</i>	Pointer to an operation previously passed to open session or invoke.
------------------	--

10.39 ta-ref/api/keystone/trace.c File Reference

```
#include <stdarg.h>
#include "trace.h"
#include "edger/Enclave_t.h"
Include dependency graph for trace.c:
```



Functions

- void [trace_vprintf](#) (const char *func, int line, int level, bool level_ok, const char *fmt, va_list ap)
- void [trace_printf](#) (const char *func, int line, int level, bool level_ok, const char *fmt,...)

10.39.1 Function Documentation

10.39.1.1 trace_printf() `void trace_printf (`
 `const char * func,`
 `int line,`
 `int level,`
 `bool level_ok,`
 `const char * fmt,`
 `...)`

[trace_printf\(\)](#) - Prints the formatted data to stdout.

This function returns the value of `ap` by calling `va_end()`.

Parameters

<i>func</i>	Pointer to a buffer where the resulting C-string is stored.
<i>line</i>	integer type of line
<i>level_ok</i>	boolean value
<i>fmt</i>	C string that contains a format string
<i>ap</i>	A value identifying a variable arguments list

Returns

Total number of characters is returned.

10.39.1.2 trace_vprintf() `void trace_vprintf (`
 `const char * func,`
 `int line,`
 `int level,`
 `bool level_ok,`
 `const char * fmt,`
 `va_list ap)`

[trace_vprintf\(\)](#) - Writes the formatted data from variable argument list to sized buffer.

This function returns the buffer character by calling `ocall_print_string()`

Parameters

<i>func</i>	Pointer to a buffer where the resulting C-string is stored.
<i>line</i>	integer type of line
<i>level_ok</i>	boolean value
<i>fmt</i>	C string that contains a format string
<i>ap</i>	A value identifying a variable arguments list

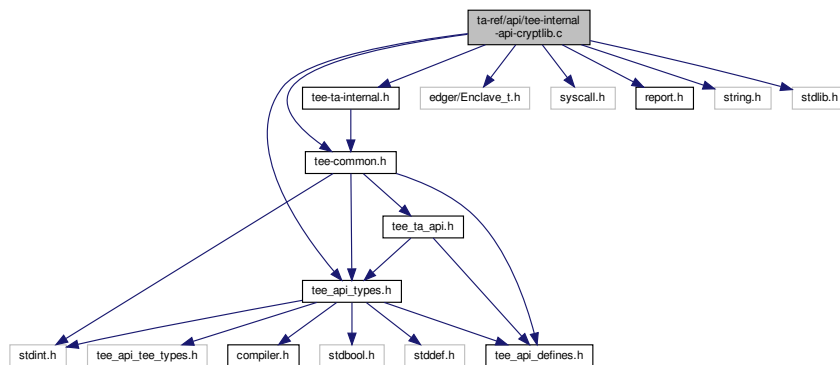
Returns

buf The total number of characters written is returned.

10.40 ta-ref/api/tee-internal-api-cryptlib.c File Reference

```
#include "tee_api_types.h"
#include "tee-common.h"
#include "tee-ta-internal.h"
#include "edger/Enclave_t.h"
#include "syscall.h"
#include "report.h"
#include <string.h>
#include <stdlib.h>
```

Include dependency graph for tee-internal-api-cryptlib.c:



Macros

- #define [GCM_ST_INIT](#) 1
- #define [GCM_ST_AAD](#) 2
- #define [GCM_ST_ACTIVE](#) 3
- #define [GCM_ST_FINAL](#) 4
- #define [SIG_LENGTH](#) 64

Functions

- void [wolfSSL_Free](#) (void *p)
- void * [wolfSSL_Malloc](#) (size_t n)
- [TEE_Result TEE_AllocateOperation](#) ([TEE_OperationHandle](#) *operation, uint32_t algorithm, uint32_t mode, uint32_t maxKeySize)
Crypto, for all Crypto Functions.
- void [TEE_FreeOperation](#) ([TEE_OperationHandle](#) operation)
Crypto, for all Crypto Functions.
- void [TEE_DigestUpdate](#) ([TEE_OperationHandle](#) operation, const void *chunk, uint32_t chunkSize)
Crypto, Message Digest Functions.
- [TEE_Result TEE_DigestDoFinal](#) ([TEE_OperationHandle](#) operation, const void *chunk, uint32_t chunkLen, void *hash, uint32_t *hashLen)
- [TEE_Result TEE_SetOperationKey](#) ([TEE_OperationHandle](#) operation, [TEE_ObjectHandle](#) key)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- [TEE_Result TEE_AEInit](#) ([TEE_OperationHandle](#) operation, const void *nonce, uint32_t nonceLen, uint32_t tagLen, uint32_t AADLen, uint32_t payloadLen)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

- void **TEE_AEUpdateAAD** (**TEE.OperationHandle** operation, const void *AADdata, uint32_t AADdataLen)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

- **TEE.Result** **TEE_AEUpdate** (**TEE.OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

- **TEE.Result** **TEE_AEEncryptFinal** (**TEE.OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen, void *tag, uint32_t *tagLen)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

- **TEE.Result** **TEE_AEDecryptFinal** (**TEE.OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen, void *tag, uint32_t tagLen)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

- void **TEE.CipherInit** (**TEE.OperationHandle** operation, const void *nonce, uint32_t nonceLen)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

- **TEE.Result** **TEE.CipherUpdate** (**TEE.OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

- **TEE.Result** **TEE.CipherDoFinal** (**TEE.OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)

- **TEE.Result** **TEE.GenerateKey** (**TEE.ObjectHandle** object, uint32_t keySize, const **TEE.Attribute** *params, uint32_t paramCount)

Crypto, Asymmetric key Verification Functions.

- **TEE.Result** **TEE.AllocateTransientObject** (**TEE.ObjectType** objectType, uint32_t maxKeySize, **TEE.ObjectHandle** *object)

Crypto, Asymmetric key Verification Functions.

- void **TEE.InitRefAttribute** (**TEE.Attribute** *attr, uint32_t attributeID, const void *buffer, uint32_t length)

Crypto, Asymmetric key Verification Functions.

- void **TEE.InitValueAttribute** (**TEE.Attribute** *attr, uint32_t attributeID, uint32_t a, uint32_t b)

Crypto, Asymmetric key Verification Functions.

- void **TEE.FreeTransientObject** (**TEE.ObjectHandle** object)

Crypto, Asymmetric key Verification Functions.

- **TEE.Result** **TEE.AsymmetricSignDigest** (**TEE.OperationHandle** operation, const **TEE.Attribute** *params, uint32_t paramCount, const void *digest, uint32_t digestLen, void *signature, uint32_t *signatureLen)

Crypto, Asymmetric key Verification Functions.

- **TEE.Result** **TEE.AsymmetricVerifyDigest** (**TEE.OperationHandle** operation, const **TEE.Attribute** *params, uint32_t paramCount, const void *digest, uint32_t digestLen, const void *signature, uint32_t signatureLen)

Crypto, Asymmetric key Verification Functions.

10.40.1 Macro Definition Documentation

10.40.1.1 GCM_ST_AAD #define GCM_ST_AAD 2

10.40.1.2 GCM_ST_ACTIVE #define GCM_ST_ACTIVE 3

10.40.1.3 GCM_ST_FINAL `#define GCM_ST_FINAL 4`

10.40.1.4 GCM_ST_INIT `#define GCM_ST_INIT 1`

10.40.1.5 SIG_LENGTH `#define SIG_LENGTH 64`

10.40.2 Function Documentation

10.40.2.1 TEE_AEDecryptFinal() `TEE_Result TEE_AEDecryptFinal (`
`TEE_OperationHandle operation,`
`const void * srcData,`
`uint32_t srcLen,`
`void * destData,`
`uint32_t * destLen,`
`void * tag,`
`uint32_t tagLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

TEE_AEDecryptFinal() - Processes data that has not been processed by previous calls to TEE_AEUpdate as well as data supplied in srcData.

This function completes the AE operation and compares the computed tag with the tag supplied in the parameter tag. The operation handle can be reused or newly initialized. The buffers srcData and destData shall be either completely disjoint or equal in their starting positions. The operation may be in either initial or active state and enters initial state afterwards.

Parameters

<i>operation</i>	Handle of a running AE operation
<i>srcData</i>	Reference to final chunk of input data to be encrypted
<i>srcLen</i>	length of the input data
<i>destData</i>	Output buffer. Can be omitted if the output is to be discarded.
<i>destLen</i>	length of the buffer.
<i>tag</i>	Output buffer filled with the computed tag
<i>tagLen</i>	length of the tag.

Returns

0 on success.

TEE_ERROR_SHORT_BUFFER If the output buffer is not large enough to contain the output

TEE_ERROR_MAC_INVALID If the computed tag does not match the supplied tag

10.40.2.2 TEE_AEEncryptFinal() `TEE_Result TEE_AEEncryptFinal (`
`TEE_OperationHandle operation,`
`const void * srcData,`
`uint32_t srcLen,`
`void * destData,`
`uint32_t * destLen,`
`void * tag,`
`uint32_t * tagLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

TEE_AEEncryptFinal() - processes data that has not been processed by previous calls to TEE_AEUpdate as well as data supplied in srcData .

TEE_AEEncryptFinal completes the AE operation and computes the tag. The operation handle can be reused or newly initialized. The buffers srcData and destData SHALL be either completely disjoint or equal in their starting positions. The operation may be in either initial or active state and enters initial state afterwards.

Parameters

<i>operation</i>	Handle of a running AE operation
<i>srcData</i>	Reference to final chunk of input data to be encrypted
<i>srcLen</i>	length of the input data
<i>destData</i>	Output buffer. Can be omitted if the output is to be discarded.
<i>destLen</i>	length of the buffer.
<i>tag</i>	Output buffer filled with the computed tag
<i>tagLen</i>	length of the tag.

Returns

0 on success.

TEE_ERROR_SHORT_BUFFER If the output or tag buffer is not large enough to contain the output.

10.40.2.3 TEE_AEInit() `TEE_Result TEE_AEInit (`
`TEE_OperationHandle operation,`
`const void * nonce,`
`uint32_t nonceLen,`
`uint32_t tagLen,`
`uint32_t AADLen,`
`uint32_t payloadLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

TEE_AEInit() - Initializes an Authentication Encryption operation.

The operation must be in initial state and remains in the initial state afterwards.

Parameters

<i>operation</i>	A handle on the operation.
<i>nonce</i>	The operation nonce or IV
<i>nonceLen</i>	length of nonce
<i>tagLen</i>	Size in bits of the tag
<i>AADLen</i>	Length in bytes of the AAD
<i>payloadLen</i>	Length in bytes of the payload.

Returns

0 on success.

TEE_ERROR_NOT_SUPPORTED If the tag length is not supported by the algorithm.

10.40.2.4 TEE_AEUpdate() `TEE_Result TEE_AEUpdate (`
`TEE_OperationHandle operation,`
`const void * srcData,`
`uint32_t srcLen,`
`void * destData,`
`uint32_t * destLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

TEE_AEUpdate() - Accumulates data for an Authentication Encryption operation

This function describes Input data does not have to be a multiple of block size. Subsequent calls to this function are possible. Unless one or more calls of this function have supplied sufficient input data, no output is generated. when using this routine to decrypt the returned data may be corrupt since the integrity check is not performed until all the data has been processed. If this is a concern then only use the TEE_AEDecryptFinal routine.

Parameters

<i>operation</i>	Handle of a running AE operation.
<i>srcData</i>	Input data buffer to be encrypted or decrypted
<i>srcLen</i>	length of the input buffer.
<i>destData</i>	Output buffer
<i>destLen</i>	length of the out put buffer.

Returns

0 on success.

TEE_ERROR_SHORT_BUFFER if the output buffer is not large enough to contain the output.

10.40.2.5 TEE_AEUpdateAAD() `void TEE_AEUpdateAAD (`
`TEE_OperationHandle operation,`
`const void * AADdata,`
`uint32_t AADdataLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

[TEE_AEUpdateAAD\(\)](#) - Feeds a new chunk of Additional Authentication Data (AAD) to the AE operation. Subsequent calls to this function are possible.

The TEE_AEUpdateAAD function feeds a new chunk of Additional Authentication Data (AAD) to the AE operation. Subsequent calls to this function are possible. The buffers srcData and destData shall be either completely disjoint or equal in their starting positions. The operation SHALL be in initial state and remains in initial state afterwards.

Parameters

<i>operation</i>	Handle on the AE operation
<i>AADdata</i>	Input buffer containing the chunk of AAD
<i>AADdataLen</i>	length of the chunk of AAD.

10.40.2.6 TEE_AllocateOperation() `TEE_Result TEE_AllocateOperation (`
`TEE_OperationHandle * operation,`
`uint32_t algorithm,`
`uint32_t mode,`
`uint32_t maxKeySize)`

Crypto, for all Crypto Functions.

[TEE_AllocateOperation\(\)](#) - Allocates a handle for a new cryptographic operation and sets the mode and algorithm type.

If this function does not return with TEE_SUCCESS then there is no valid handle value. Once a cryptographic operation has been created, the implementation shall guarantee that all resources necessary for the operation are allocated and that any operation with a key of at most maxKeySize bits can be performed. For algorithms that take multiple keys, for example the AES XTS algorithm, the maxKeySize parameter specifies the size of the largest key. It is up to the implementation to properly allocate space for multiple keys if the algorithm so requires.

Parameters

<i>operation</i>	reference to generated operation handle.
<i>algorithm</i>	One of the cipher algorithms.
<i>mode</i>	The operation mode.
<i>maxKeySize</i>	Maximum key size in bits for the operation.

Returns

0 in case of success

TEE_ERROR_OUT_OF_MEMORY If there are not enough resources to allocate the operation.

TEE_ERROR_NOT_SUPPORTED If the mode is not compatible with the algorithm or key size or if the algorithm is not one of the listed algorithms or if maxKeySize is not appropriate for the algorithm.

10.40.2.7 TEE_AllocateTransientObject() `TEE_Result TEE_AllocateTransientObject (`
`TEE_ObjectType objectType,`
`uint32_t maxKeySize,`
`TEE_ObjectHandle * object)`

Crypto, Asymmetric key Verification Functions.

[TEE_AllocateTransientObject\(\)](#) - Allocates an uninitialized transient object. Transient objects are used to hold a cryptographic object (key or key-pair).

The value TEE_KEYSIZE_NO_KEY should be used for maxObjectSize for object types that do not require a key so that all the container resources can be pre-allocated. As allocated, the container is uninitialized. It can be initialized by subsequently importing the object material, generating an object, deriving an object, or loading an object from the Trusted Storage.

Parameters

<i>objectType</i>	Type of uninitialized object container to be created
<i>maxKeySize</i>	Key Size of the object.
<i>object</i>	Filled with a handle on the newly created key container.

Returns

0 on success

TEE_ERROR_OUT_OF_MEMORY If not enough resources are available to allocate the object handle.

TEE_ERROR_NOT_SUPPORTED If the key size is not supported or the object type is not supported.

10.40.2.8 TEE_AsymmetricSignDigest() `TEE_Result TEE_AsymmetricSignDigest (`
`TEE_OperationHandle operation,`
`const TEE_Attribute * params,`
`uint32_t paramCount,`
`const void * digest,`
`uint32_t digestLen,`
`void * signature,`
`uint32_t * signatureLen)`

Crypto, Asymmetric key Verification Functions.

[TEE_AsymmetricSignDigest\(\)](#) - Signs a message digest within an asymmetric operation.

Parameters

<i>operation</i>	Handle on the operation, which SHALL have been suitably set up with an operation key.
<i>params</i>	Optional operation parameters
<i>paramCount</i>	size of param
<i>digest</i>	Input buffer containing the input message digest
<i>digestLen</i>	length of input buffer.
<i>signature</i>	Output buffer written with the signature of the digest
<i>signatureLen</i>	length of output buffer.

Returns

0 on success

TEE_ERROR_SHORT_BUFFER If the signature buffer is not large enough to hold the result

10.40.2.9 TEE_AsymmetricVerifyDigest() `TEE_Result` TEE_AsymmetricVerifyDigest (
 `TEE_OperationHandle` operation,
 const `TEE_Attribute` * params,
 uint32_t paramCount,
 const void * digest,
 uint32_t digestLen,
 const void * signature,
 uint32_t signatureLen)

Crypto, Asymmetric key Verification Functions.

[TEE_AsymmetricVerifyDigest\(\)](#) - verifies a message digest signature within an asymmetric operation.

This function describes the message digest signature verify by calling ed25519_verify().

Parameters

<i>operation</i>	Handle on the operation, which SHALL have been suitably set up with an operation key.
<i>params</i>	Optional operation parameters
<i>paramCount</i>	size of param.
<i>digest</i>	Input buffer containing the input message digest
<i>digestLen</i>	length of input buffer.
<i>signature</i>	Output buffer written with the signature of the digest
<i>signatureLen</i>	length of output buffer.

Returns

TEE_SUCCESS on success

TEE_ERROR_SIGNATURE_INVALID if the signature is invalid.

10.40.2.10 TEE_CipherDoFinal() `TEE_Result TEE_CipherDoFinal (`
`TEE_OperationHandle operation,`
`const void * srcData,`
`uint32_t srcLen,`
`void * destData,`
`uint32_t * destLen)`

[TEE_CipherDoFinal\(\)](#) - Finalizes the cipher operation, processing data that has not been processed by previous calls to [TEE_CipherUpdate](#) as well as data supplied in `srcData` .

This function describes The operation handle can be reused or re-initialized. The buffers `srcData` and `destData` shall be either completely disjoint or equal in their starting positions. The operation SHALL be in active state and is set to initial state afterwards.

Parameters

<i>operation</i>	Handle of a running Cipher operation
<i>srcData</i>	Input data buffer to be encrypted or decrypted
<i>srcLen</i>	length of input buffer
<i>destData</i>	output buffer
<i>destLen</i>	ouput buffer length.

Returns

0 on success

TEE_ERROR_SHORT_BUFFER If the output buffer is not large enough to contain the output

10.40.2.11 TEE_CipherInit() `void TEE_CipherInit (`
`TEE_OperationHandle operation,`
`const void * nonce,`
`uint32_t nonceLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

[TEE_CipherInit\(\)](#) - starts the symmetric cipher operation.

The operation shall have been associated with a key. If the operation is in active state, it is reset and then initialized. If the operation is in initial state, it is moved to active state.

Parameters

<i>operation</i>	A handle on an opened cipher operation setup with a key
<i>nonce</i>	Buffer containing the operation Initialization Vector as appropriate.
<i>nonceLen</i>	length of the buffer

10.40.2.12 TEE_CipherUpdate() `TEE_Result TEE_CipherUpdate (`
`TEE.OperationHandle operation,`
`const void * srcData,`
`uint32_t srcLen,`
`void * destData,`
`uint32_t * destLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

[TEE.CipherUpdate\(\)](#) - encrypts or decrypts input data.

Input data does not have to be a multiple of block size. Subsequent calls to this function are possible. Unless one or more calls of this function have supplied sufficient input data, no output is generated. The cipher operation is finalized with a call to `TEE.CipherDoFinal`. The buffers `srcData` and `destData` SHALL be either completely disjoint or equal in their starting positions. The operation SHALL be in active state.

Parameters

<i>operation</i>	Handle of a running Cipher operation
<i>srcData</i>	Input data buffer to be encrypted or decrypted
<i>srcLen</i>	length of input buffer
<i>destData</i>	output buffer
<i>destLen</i>	output buffer length.

Returns

0 on success else

`TEE_ERROR_SHORT_BUFFER` If the output buffer is not large enough to contain the output. In this case, the input is not fed into the algorithm.

10.40.2.13 TEE_DigestDoFinal() `TEE_Result TEE_DigestDoFinal (`
`TEE.OperationHandle operation,`
`const void * chunk,`
`uint32_t chunkLen,`
`void * hash,`
`uint32_t * hashLen)`

[TEE.DigestDoFinal\(\)](#) - Finalizes the message digest operation and produces the message hash.

This function finalizes the message digest operation and produces the message hash. Afterwards the Message Digest operation is reset to initial state and can be reused.

Parameters

<i>operation</i>	Handle of a running Message Digest operation.
<i>chunk</i>	Chunk of data to be hashed.
<i>chunkLen</i>	size of the chunk.
<i>hash</i>	Output buffer filled with the message hash.
<i>hashLen</i>	length of the message hash.

Returns

0 on success

TEE_ERROR_SHORT_BUFFER If the output buffer is too small. In this case, the operation is not finalized.

10.40.2.14 TEE_DigestUpdate() void TEE_DigestUpdate (
 TEE_OperationHandle operation,
 const void * chunk,
 uint32_t chunkSize)

Crypto, Message Digest Functions.

[TEE_DigestUpdate\(\)](#) - Accumulates message data for hashing.

This function describes the message does not have to be block aligned. Subsequent calls to this function are possible. The operation may be in either initial or active state and becomes active.

Parameters

<i>operation</i>	Handle of a running Message Digest operation.
<i>chunk</i>	Chunk of data to be hashed
<i>chunkSize</i>	size of the chunk.

10.40.2.15 TEE_FreeOperation() void TEE_FreeOperation (
 TEE_OperationHandle operation)

Crypto, for all Crypto Functions.

[TEE_FreeOperation\(\)](#) - Deallocates all resources associated with an operation handle.

This function deallocates all resources associated with an operation handle. After this function is called, the operation handle is no longer valid. All cryptographic material in the operation is destroyed. The function does nothing if operation is TEE_HANDLE_NULL.

Parameters

<i>operation</i>	Reference to operation handle.
------------------	--------------------------------

Returns

nothing after the operation free.

10.40.2.16 TEE_FreeTransientObject() `void TEE_FreeTransientObject (`
`TEE_ObjectHandle object)`

Crypto, Asymmetric key Verification Functions.

[TEE_FreeTransientObject\(\)](#) - Deallocates a transient object previously allocated with [TEE_AllocateTransientObject](#) .

this function describes the object handle is no longer valid and all resources associated with the transient object shall have been reclaimed after the [TEE_AllocateTransientObject\(\)](#) call.

Parameters

<i>object</i>	Handle on the object to free.
---------------	-------------------------------

10.40.2.17 TEE_GenerateKey() `TEE_Result TEE_GenerateKey (`
`TEE_ObjectHandle object,`
`uint32_t keySize,`
`const TEE_Attribute * params,`
`uint32_t paramCount)`

Crypto, Asymmetric key Verification Functions.

[TEE_GenerateKey \(\)](#) - Generates a random key or a key-pair and populates a transient key object with the generated key material.

The size of the desired key is passed in the `keySize` parameter and shall be less than or equal to the maximum key size specified when the transient object was created.

Parameters

<i>object</i>	Handle on an uninitialized transient key to populate with the generated key.
<i>keySize</i>	Requested key size shall be less than or equal to the maximum key size specified when the object container was created
<i>params</i>	Parameters for the key generation.
<i>paramCount</i>	The values of all parameters are copied into the object so that the params array and all the memory buffers it points to may be freed after this routine returns without affecting the object.

Returns

0 on succes

`TEE_ERROR_BAD_PARAMETERS` If an incorrect or inconsistent attribute is detected. The checks that are performed depend on the implementation.

10.40.2.18 TEE_InitRefAttribute() `void TEE_InitRefAttribute (`
`TEE_Attribute * attr,`


```
uint32_t attributeID,
const void * buffer,
uint32_t length )
```

Crypto, Asymmetric key Verification Functions.

[TEE_InitRefAttribute\(\)](#) - The helper function can be used to populate a single attribute either with a reference to a buffer or with integer values.

In `TEE_InitRefAttribute()` only the buffer pointer is copied, not the content of the buffer. This means that the attribute structure maintains a pointer back to the supplied buffer. It is the responsibility of the TA author to ensure that the contents of the buffer maintain their value until the attributes array is no longer in use.

Parameters

<i>attr</i>	attribute structure to initialize.
<i>attributeID</i>	Identifier of the attribute to populate.
<i>buffer</i>	input buffer that holds the content of the attribute.
<i>length</i>	buffer length.

10.40.2.19 TEE_InitValueAttribute() `void TEE_InitValueAttribute (`
`TEE_Attribute * attr,`
`uint32_t attributeID,`
`uint32_t a,`
`uint32_t b)`

Crypto, Asymmetric key Verification Functions.

[TEE_InitValueAttribute\(\)](#) - The helper function can be used to populate a single attribute either with a reference to a buffer or with integer values.

Parameters

<i>attr</i>	attribute structure to initialize.
<i>attributeID</i>	Identifier of the attribute to populate.
<i>a</i>	unsigned integer value to assign to the a member of the attribute structure.
<i>b</i>	unsigned integer value to assign to the b member of the attribute structure

10.40.2.20 TEE_SetOperationKey() `TEE_Result TEE_SetOperationKey (`
`TEE_OperationHandle operation,`
`TEE_ObjectHandle key)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

[TEE_SetOperationKey\(\)](#) - Programs the key of an operation; that is, it associates an operation with a key.

The key material is copied from the key object handle into the operation. After the key has been set, there is no longer any link between the operation and the key object. The object handle can be closed or reset and this will not affect the operation. This copied material exists until the operation is freed using `TEE_FreeOperation` or another key is set into the operation.

Parameters

<i>operation</i>	Operation handle.
<i>key</i>	A handle on a key object.

Returns

0 on success return

`TEE_ERROR_CORRUPT_OBJECT` If the object is corrupt. The object handle is closed.

`TEE_ERROR_STORAGE_NOT_AVAILABLE` If the persistent object is stored in a storage area which is currently inaccessible.

10.40.2.21 `wolfSSL_Free()` `void wolfSSLFree (`
`void * p)`

[`wolfSSL_Free\(\)`](#) - Deallocates the memory which allocated previously.

Parameters

<i>p</i>	This is the pointer to a memory block.
----------	--

10.40.2.22 `wolfSSL_Malloc()` `void * wolfSSLMalloc (`
`size_t n)`

[`wolfSSL_Malloc\(\)`](#) - Allocates the requested memory and returns a pointer to it.

Parameters

<i>n</i>	size of the memory block.
----------	---------------------------

10.41 ta-ref/docs/building.md File Reference

10.42 ta-ref/docs/gp_api.md File Reference

10.43 ta-ref/docs/how_to_program_on_ta-ref.md File Reference

10.44 ta-ref/docs/overview_of_ta-ref.md File Reference

10.45 ta-ref/docs/preparation.md File Reference

10.46 ta-ref/docs/running_on_dev_boards.md File Reference

Index

- `__GCC_VERSION`
 - compiler.h, 62
- `__INTOF_ADD`
 - compiler.h, 62
- `__INTOF_ASSIGN`
 - compiler.h, 62
- `__INTOF_HALF_MAX_SIGNED`
 - compiler.h, 62
- `__INTOF_MAX`
 - compiler.h, 63
- `__INTOF_MAX_SIGNED`
 - compiler.h, 63
- `__INTOF_MIN`
 - compiler.h, 63
- `__INTOF_MIN_SIGNED`
 - compiler.h, 63
- `__INTOF_MUL`
 - compiler.h, 63
- `__INTOF_SUB`
 - compiler.h, 64
- `__TEE_ObjectHandle`, 32
 - desc, 33
 - flags, 33
 - persist_ctx, 33
 - private_key, 33
 - public_key, 33
 - type, 33
- `__TEE_OperationHandle`, 33
 - aectx, 34
 - aegcm.state, 34
 - aev, 34
 - aekey, 34
 - alg, 34
 - ctx, 34
 - flags, 34
 - mode, 34
 - prikey, 35
 - pubkey, 35
- `__aligned`
 - compiler.h, 60
 - tee_api.types.h, 151
- `__attr_const`
 - compiler.h, 60
- `__attribute__`
 - tee-internal-api-machine.c, 190
 - tee-internal-api.c, 203
 - tee-ta-internal.h, 74
- `__bss`
 - compiler.h, 60
- `__cold`
 - compiler.h, 60
- `__compiler_add_overflow`
 - compiler.h, 60
- `__compiler_atomic_load`
 - compiler.h, 61
- `__compiler_atomic_store`
 - compiler.h, 61
- `__compiler_bswap16`
 - compiler.h, 61
- `__compiler_bswap32`
 - compiler.h, 61
- `__compiler_bswap64`
 - compiler.h, 61
- `__compiler_compare_and_swap`
 - compiler.h, 61
- `__compiler_mul_overflow`
 - compiler.h, 61
- `__compiler_sub_overflow`
 - compiler.h, 61
- `__data`
 - compiler.h, 62
- `__deprecated`
 - compiler.h, 62
- `__early_ta`
 - compiler.h, 62
- `__intof_mul_a0`
 - compiler.h, 63
- `__intof_mul_a1`
 - compiler.h, 63
- `__intof_mul_b0`
 - compiler.h, 64
- `__intof_mul_b1`
 - compiler.h, 64
- `__intof_mul_hmask`
 - compiler.h, 64
- `__intof_mul_hshift`
 - compiler.h, 64
- `__intof_mul_negate`
 - compiler.h, 64
- `__intof_mul_t`
 - compiler.h, 64
- `__maybe_unused`
 - compiler.h, 64
- `__must_check`
 - compiler.h, 65
- `__noinline`
 - compiler.h, 65
- `__noprof`
 - compiler.h, 65
- `__noreturn`
 - compiler.h, 65
- `__packed`
 - compiler.h, 65
- `__printf`
 - compiler.h, 65
- `__pure`
 - compiler.h, 65
- `__rodata`
 - compiler.h, 65
- `__rodata_unpaged`

- compiler.h, 65
 - __section
 - compiler.h, 65
 - __unused
 - compiler.h, 65
 - __used
 - compiler.h, 66
 - __weak
 - compiler.h, 66
 - _sanctum_dev_public_key
 - test_dev_key.h, 178
 - _sanctum_dev_public_key_len
 - test_dev_key.h, 178
 - _sanctum_dev_secret_key
 - test_dev_key.h, 179
 - _sanctum_dev_secret_key_len
 - test_dev_key.h, 179
- a
 - TEE_Attribute, 39
 - TEE_Param, 46
 - TEEC_Value, 58
- addrinfo, 35
 - ai_addr, 35
 - ai_addrlen, 35
 - ai_canonname, 36
 - ai_family, 36
 - ai_flags, 36
 - ai_next, 36
 - ai_protocol, 36
 - ai_socktype, 36
- aectx
 - __TEE_OperationHandle, 34
- aegcm_state
 - __TEE_OperationHandle, 34
- aeiv
 - __TEE_OperationHandle, 34
- aekey
 - __TEE_OperationHandle, 34
- AES256
 - tee_api_tee_types.h, 211, 215
- ai_addr
 - addrinfo, 35
- ai_addrlen
 - addrinfo, 35
- ai_canonname
 - addrinfo, 36
- ai_family
 - addrinfo, 36
- ai_flags
 - addrinfo, 36
- ai_next
 - addrinfo, 36
- ai_protocol
 - addrinfo, 36
- ai_socktype
 - addrinfo, 36
- alg
 - __TEE_OperationHandle, 34
- algorithm
 - TEE_OperationInfo, 43
 - TEE_OperationInfoMultiple, 45
- allocated_size
 - TEEC_SharedMemory, 55
- ATTEST_DATA_MAXLEN
 - report.h, 69
- attributeID
 - TEE_Attribute, 40
- b
 - TEE_Attribute, 40
 - TEE_Param, 46
 - TEEC_Value, 58
- buffer
 - TEE_Attribute, 40
 - TEE_Param, 46
 - TEE_SEAID, 47
 - TEEC_SharedMemory, 55
 - TEEC_TempMemoryReference, 56
- buffer_allocated
 - TEEC_SharedMemory, 55
- bufferLen
 - TEE_SEAID, 47
- clockSeqAndNode
 - TEE_UUID, 49
 - TEEC_UUID, 57
- compiler.h
 - __GCC_VERSION, 62
 - __INTOF_ADD, 62
 - __INTOF_ASSIGN, 62
 - __INTOF_HALF_MAX_SIGNED, 62
 - __INTOF_MAX, 63
 - __INTOF_MAX_SIGNED, 63
 - __INTOF_MIN, 63
 - __INTOF_MIN_SIGNED, 63
 - __INTOF_MUL, 63
 - __INTOF_SUB, 64
 - __aligned, 60
 - __attr_const, 60
 - __bss, 60
 - __cold, 60
 - __compiler_add_overflow, 60
 - __compiler_atomic_load, 61
 - __compiler_atomic_store, 61
 - __compiler_bswap16, 61
 - __compiler_bswap32, 61
 - __compiler_bswap64, 61
 - __compiler_compare_and_swap, 61
 - __compiler_mul_overflow, 61
 - __compiler_sub_overflow, 61
 - __data, 62
 - __deprecated, 62
 - __early_ta, 62
 - __intof_mul.a0, 63
 - __intof_mul.a1, 63
 - __intof_mul.b0, 64
 - __intof_mul.b1, 64

- __intof_mul_hmask, 64
- __intof_mul_hshift, 64
- __intof_mul_negate, 64
- __intof_mul_t, 64
- __maybe_unused, 64
- __must_check, 65
- __noinline, 65
- __noprof, 65
- __noreturn, 65
- __packed, 65
- __printf, 65
- __pure, 65
- __rodata, 65
- __rodata_unpaged, 65
- __section, 65
- __unused, 65
- __used, 66
- __weak, 66
- content
 - TEE_Attribute, 40
- ctx
 - __TEE_OperationHandle, 34
 - TEEC_Session, 54
- data
 - enclave_report, 36
- data_len
 - enclave_report, 37
- dataPosition
 - TEE_ObjectInfo, 42
- dataSize
 - TEE_ObjectInfo, 42
- desc
 - __TEE_ObjectHandle, 33
- dev_public_key
 - report, 38
- dhex_dump
 - trace.h, 183
- DHEXDUMP
 - trace.h, 181
- digestLength
 - TEE_OperationInfo, 43
 - TEE_OperationInfoMultiple, 45
- DMREQ_FINISH
 - tee_api_types.h, 150
- DMREQ_WRITE
 - tee_api_types.h, 150
- DMSG
 - trace.h, 181
- DMSG_RAW
 - trace.h, 181
- DPRINT_STACK
 - trace.h, 181
- EMSG
 - trace.h, 181
- EMSG_RAW
 - trace.h, 181
- enclave
 - report, 38
- enclave_report, 36
 - data, 36
 - data_len, 37
 - hash, 37
 - signature, 37
- EPRINT_STACK
 - trace.h, 181
- events
 - pollfd, 37
- fd
 - pollfd, 37
 - TEEC_Context, 49
- flags
 - __TEE_ObjectHandle, 33
 - __TEE_OperationHandle, 34
 - TEEC_SharedMemory, 55
- flags2flags
 - tee-internal-api.c, 192, 203
- FMSG
 - trace.h, 181
- FMSG_RAW
 - trace.h, 182
- FPERMS
 - tee-internal-api.c, 192, 202
- FPRINT_STACK
 - trace.h, 182
- GCM_ST_AAD
 - tee-internal-api-cryptlib.c, 224
- GCM_ST_ACTIVE
 - tee-internal-api-cryptlib.c, 224
- GCM_ST_FINAL
 - tee-internal-api-cryptlib.c, 224
- GCM_ST_INIT
 - tee-internal-api-cryptlib.c, 225
- GetRelTimeEnd
 - tee-internal-api.c, 193, 203
 - tee-ta-internal.h, 75
- GetRelTimeStart
 - tee-internal-api.c, 193, 204
 - tee-ta-internal.h, 75
- handleFlags
 - TEE_ObjectInfo, 42
- handleState
 - TEE_OperationInfo, 43
 - TEE_OperationInfoMultiple, 45
- hash
 - enclave_report, 37
 - sm_report, 39
- id
 - TEEC_SharedMemory, 55
- IMSG
 - trace.h, 182
- IMSG_RAW
 - trace.h, 182

- INMSG
 - trace.h, 182
- IPRINT_STACK
 - trace.h, 182
- keyInformation
 - TEE_OperationInfoMultiple, 45
- keySize
 - TEE_ObjectInfo, 42
 - TEE_OperationInfo, 43
 - TEE_OperationInfoKey, 44
- length
 - TEE_Attribute, 40
- login
 - TEE_Identity, 41
- MAX_FUNC_PRINT_SIZE
 - trace.h, 182
- MAX_PRINT_SIZE
 - trace.h, 182
- maxKeySize
 - TEE_ObjectInfo, 42
 - TEE_OperationInfo, 43
 - TEE_OperationInfoMultiple, 45
- maxObjectSize
 - TEE_ObjectInfo, 42
- MBEDCRYPT
 - tee_api_tee_types.h, 211, 215
- MDSIZE
 - report.h, 69
- memref
 - TEE_Param, 46
 - TEEC_Parameter, 52
- millis
 - TEE_Time, 48
- mode
 - __TEE_OperationHandle, 34
 - TEE_OperationInfo, 43
 - TEE_OperationInfoMultiple, 45
- MSG
 - trace.h, 182
- MSG_RAW
 - trace.h, 182
- nfds_t
 - tee_api_types.h, 151
- numberOfKeys
 - TEE_OperationInfoMultiple, 45
- O_CREAT
 - tee-internal-api.c, 192, 202
- O_EXCL
 - tee-internal-api.c, 192, 202
- O_RDONLY
 - tee-internal-api.c, 192, 202
- O_RDWR
 - tee-internal-api.c, 192, 202
- O_TRUNC
 - tee-internal-api.c, 192, 202
- O_WRONLY
 - tee-internal-api.c, 192, 202
- objectSize
 - TEE_ObjectInfo, 42
- objectType
 - TEE_ObjectInfo, 42
- objectUsage
 - TEE_ObjectInfo, 42
- offset
 - TEEC_RegisteredMemoryReference, 53
- OpenPersistentObject
 - tee-internal-api.c, 194, 204
- operationClass
 - TEE_OperationInfo, 43
 - TEE_OperationInfoMultiple, 45
- operationState
 - TEE_OperationInfoMultiple, 45
- OUTMSG
 - trace.h, 183
- OUTRMSG
 - trace.h, 183
- params
 - TEEC_Operation, 51
- paramTypes
 - TEEC_Operation, 51
- parent
 - TEEC_RegisteredMemoryReference, 53
- persist_ctx
 - __TEE_ObjectHandle, 33
- pollfd, 37
 - events, 37
 - fd, 37
 - revents, 37
- pr_deb
 - tee-common.h, 71
- prikey
 - __TEE_OperationHandle, 35
- private_key
 - __TEE_ObjectHandle, 33
- pubkey
 - __TEE_OperationHandle, 35
- public_key
 - __TEE_ObjectHandle, 33
 - sm_report, 39
- PUBLIC_KEY_SIZE
 - report.h, 69
- ref
 - TEE_Attribute, 40
- reg_mem
 - TEEC_Context, 50
- registered_fd
 - TEEC_SharedMemory, 56
- report, 38
 - dev_public_key, 38
 - enclave, 38
 - sm, 38

- report.h
 - ATTEST_DATA_MAXLEN, 69
 - MDSIZE, 69
 - PUBLIC_KEY_SIZE, 69
 - SIGNATURE_SIZE, 69
- requiredKeyUsage
 - TEE_OperationInfo, 43
 - TEE_OperationInfoKey, 44
- revents
 - pollfd, 37
- seconds
 - TEE_Time, 48
- selectResponseEnable
 - TEE_SEReaderProperties, 47
- sePresent
 - TEE_SEReaderProperties, 48
- session
 - TEEC_Operation, 51
- session_id
 - TEEC_Session, 54
- set_object_key
 - tee-internal-api.c, 194, 205
- SHA_LENGTH
 - tee_api_tee_types.h, 211, 215
- shadow_buffer
 - TEEC_SharedMemory, 56
- SIG_LENGTH
 - tee-internal-api-cryptlib.c, 225
- signature
 - enclave_report, 37
 - sm_report, 39
- SIGNATURE_SIZE
 - report.h, 69
- size
 - TEE_Param, 46
 - TEEC_RegisteredMemoryReference, 53
 - TEEC_SharedMemory, 56
 - TEEC_TempMemoryReference, 57
- sm
 - report, 38
- sm_report, 38
 - hash, 39
 - public_key, 39
 - signature, 39
- SMSG
 - trace.h, 183
- socklen_t
 - tee_api_types.h, 151
- started
 - TEEC_Operation, 51
- ta-ref/api/include/compiler.h, 59, 66
- ta-ref/api/include/report.h, 68, 69
- ta-ref/api/include/tee-common.h, 70, 71
- ta-ref/api/include/tee-ta-internal.h, 71, 96
- ta-ref/api/include/tee_api_defines.h, 99, 138
- ta-ref/api/include/tee_api_defines_extensions.h, 143, 147
- ta-ref/api/include/tee_api_types.h, 148, 153
- ta-ref/api/include/tee_client_api.h, 156, 168
- ta-ref/api/include/tee_internal_api.h, 171
- ta-ref/api/include/tee_internal_api_extensions.h, 171, 173
- ta-ref/api/include/tee_ta_api.h, 174, 176
- ta-ref/api/include/test_dev_key.h, 178, 179
- ta-ref/api/include/trace.h, 179, 185
- ta-ref/api/include/trace_levels.h, 187, 189
- ta-ref/api/keystone/tee-internal-api-machine.c, 189
- ta-ref/api/keystone/tee-internal-api.c, 190
- ta-ref/api/keystone/tee_api_tee_types.h, 210, 212
- ta-ref/api/keystone/teec_stub.c, 217
- ta-ref/api/keystone/trace.c, 221
- ta-ref/api/optee/tee_api_tee_types.h, 213
- ta-ref/api/sgx/tee-internal-api.c, 201
- ta-ref/api/sgx/tee_api_tee_types.h, 213, 216
- ta-ref/api/tee-internal-api-cryptlib.c, 223
- ta-ref/docs/building.md, 237
- ta-ref/docs/gp_api.md, 237
- ta-ref/docs/how_to_program_on_ta-ref.md, 237
- ta-ref/docs/overview_of_ta-ref.md, 237
- ta-ref/docs/preparation.md, 237
- ta-ref/docs/running_on_dev_boards.md, 237
- TA_CloseSessionEntryPoint
 - tee_ta_api.h, 175
- TA_CreateEntryPoint
 - tee_ta_api.h, 175
- TA_DestroyEntryPoint
 - tee_ta_api.h, 175
- TA_EXPORT
 - tee_ta_api.h, 175
- TA_InvokeCommandEntryPoint
 - tee_ta_api.h, 175
- TA_OpenSessionEntryPoint
 - tee_ta_api.h, 175
- tee-common.h
 - pr_deb, 71
- tee-internal-api-cryptlib.c
 - GCM_ST_AAD, 224
 - GCM_ST_ACTIVE, 224
 - GCM_ST_FINAL, 224
 - GCM_ST_INIT, 225
 - SIG_LENGTH, 225
 - TEE_AEDecryptFinal, 225
 - TEE_AEEncryptFinal, 226
 - TEE_AEInit, 226
 - TEE_AEUpdate, 227
 - TEE_AEUpdateAAD, 227
 - TEE_AllocateOperation, 228
 - TEE_AllocateTransientObject, 229
 - TEE_AsymmetricSignDigest, 229
 - TEE_AsymmetricVerifyDigest, 230
 - TEE_CipherDoFinal, 230
 - TEE_CipherInit, 231
 - TEE_CipherUpdate, 231
 - TEE_DigestDoFinal, 232
 - TEE_DigestUpdate, 233
 - TEE_FreeOperation, 233
 - TEE_FreeTransientObject, 233

- TEE_GenerateKey, 234
- TEE_InitRefAttribute, 234
- TEE_InitValueAttribute, 235
- TEE_SetOperationKey, 235
- wolfSSL_Free, 236
- wolfSSL_Malloc, 236
- tee-internal-api-machine.c
 - __attribute__, 190
- tee-internal-api.c
 - __attribute__, 203
 - flags2flags, 192, 203
 - FPERMS, 192, 202
 - GetRelTimeEnd, 193, 203
 - GetRelTimeStart, 193, 204
 - O_CREAT, 192, 202
 - O_EXCL, 192, 202
 - O_RDONLY, 192, 202
 - O_RDWR, 192, 202
 - O_TRUNC, 192, 202
 - O_WRONLY, 192, 202
 - OpenPersistentObject, 194, 204
 - set_object_key, 194, 205
 - TEE_CloseObject, 195, 205
 - TEE_CreatePersistentObject, 195, 206
 - TEE_Free, 196
 - TEE_GenerateRandom, 196, 206
 - TEE_GetObjectInfo1, 197, 207
 - TEE_GetREETime, 197, 207
 - TEE_GetSystemTime, 198, 208
 - TEE_Malloc, 198
 - TEE_OpenPersistentObject, 198, 208
 - TEE_ReadObjectData, 199, 208
 - TEE_Realloc, 200
 - TEE_WriteObjectData, 200, 209
- tee-ta-internal.h
 - __attribute__, 74
 - GetRelTimeEnd, 75
 - GetRelTimeStart, 75
 - TEE_AEDecryptFinal, 76
 - TEE_AEEncryptFinal, 77
 - TEE_AEInit, 78
 - TEE_AEUpdate, 78
 - TEE_AEUpdateAAD, 80
 - TEE_AllocateOperation, 80
 - TEE_AllocateTransientObject, 81
 - TEE_AsymmetricSignDigest, 82
 - TEE_AsymmetricVerifyDigest, 82
 - TEE_CipherInit, 83
 - TEE_CipherUpdate, 84
 - TEE_CloseObject, 84
 - TEE_CreatePersistentObject, 85
 - TEE_DigestDoFinal, 86
 - TEE_DigestUpdate, 87
 - TEE_FreeOperation, 87
 - TEE_FreeTransientObject, 88
 - TEE_GenerateKey, 88
 - TEE_GenerateRandom, 89
 - TEE_GetObjectInfo1, 90
 - TEE_GetREETime, 91
 - TEE_GetSystemTime, 91
 - TEE_InitRefAttribute, 92
 - TEE_InitValueAttribute, 92
 - TEE_OpenPersistentObject, 93
 - TEE_ReadObjectData, 94
 - TEE_SetOperationKey, 95
 - TEE_WriteObjectData, 95
 - TEE_AEDecryptFinal
 - tee-internal-api-cryptlib.c, 225
 - tee-ta-internal.h, 76
 - TEE_AEEncryptFinal
 - tee-internal-api-cryptlib.c, 226
 - tee-ta-internal.h, 77
 - TEE_AEInit
 - tee-internal-api-cryptlib.c, 226
 - tee-ta-internal.h, 78
 - TEE_AEUpdate
 - tee-internal-api-cryptlib.c, 227
 - tee-ta-internal.h, 78
 - TEE_AEUpdateAAD
 - tee-internal-api-cryptlib.c, 227
 - tee-ta-internal.h, 80
 - TEE_ALG_AES_CBC_MAC_NOPAD
 - tee_api_defines.h, 105
 - TEE_ALG_AES_CBC_MAC_PKCS5
 - tee_api_defines.h, 105
 - TEE_ALG_AES_CBC_NOPAD
 - tee_api_defines.h, 105
 - TEE_ALG_AES_CCM
 - tee_api_defines.h, 105
 - TEE_ALG_AES_CMAC
 - tee_api_defines.h, 106
 - TEE_ALG_AES_CTR
 - tee_api_defines.h, 106
 - TEE_ALG_AES_CTS
 - tee_api_defines.h, 106
 - TEE_ALG_AES_ECB_NOPAD
 - tee_api_defines.h, 106
 - TEE_ALG_AES_GCM
 - tee_api_defines.h, 106
 - TEE_ALG_AES_XTS
 - tee_api_defines.h, 106
 - TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY
 - tee_api_defines_extensions.h, 144
 - TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY
 - tee_api_defines_extensions.h, 144
 - TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY
 - tee_api_defines_extensions.h, 144
 - TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY
 - tee_api_defines_extensions.h, 144
 - TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY
 - tee_api_defines_extensions.h, 145
 - TEE_ALG_DES3_CBC_MAC_NOPAD
 - tee_api_defines.h, 106
 - TEE_ALG_DES3_CBC_MAC_PKCS5
 - tee_api_defines.h, 106
 - TEE_ALG_DES3_CBC_NOPAD

- [tee_api.defines.h](#), [106](#)
- TEE_ALG_DES3_ECB_NOPAD
 - [tee_api.defines.h](#), [106](#)
- TEE_ALG_DES_CBC_MAC_NOPAD
 - [tee_api.defines.h](#), [106](#)
- TEE_ALG_DES_CBC_MAC_PKCS5
 - [tee_api.defines.h](#), [107](#)
- TEE_ALG_DES_CBC_NOPAD
 - [tee_api.defines.h](#), [107](#)
- TEE_ALG_DES_ECB_NOPAD
 - [tee_api.defines.h](#), [107](#)
- TEE_ALG_DH_DERIVE_SHARED_SECRET
 - [tee_api.defines.h](#), [107](#)
- TEE_ALG_DSA_SHA1
 - [tee_api.defines.h](#), [107](#)
- TEE_ALG_DSA_SHA224
 - [tee_api.defines.h](#), [107](#)
- TEE_ALG_DSA_SHA256
 - [tee_api.defines.h](#), [107](#)
- TEE_ALG_ECDH_P192
 - [tee_api.defines.h](#), [107](#)
- TEE_ALG_ECDH_P224
 - [tee_api.defines.h](#), [107](#)
- TEE_ALG_ECDH_P256
 - [tee_api.defines.h](#), [107](#)
- TEE_ALG_ECDH_P384
 - [tee_api.defines.h](#), [107](#)
- TEE_ALG_ECDH_P521
 - [tee_api.defines.h](#), [108](#)
- TEE_ALG_ECDSA_P192
 - [tee_api.defines.h](#), [108](#)
- TEE_ALG_ECDSA_P224
 - [tee_api.defines.h](#), [108](#)
- TEE_ALG_ECDSA_P256
 - [tee_api.defines.h](#), [108](#)
- TEE_ALG_ECDSA_P384
 - [tee_api.defines.h](#), [108](#)
- TEE_ALG_ECDSA_P521
 - [tee_api.defines.h](#), [108](#)
- TEE_ALG_HKDF_MD5_DERIVE_KEY
 - [tee_api.defines_extensions.h](#), [145](#)
- TEE_ALG_HKDF_SHA1_DERIVE_KEY
 - [tee_api.defines_extensions.h](#), [145](#)
- TEE_ALG_HKDF_SHA224_DERIVE_KEY
 - [tee_api.defines_extensions.h](#), [145](#)
- TEE_ALG_HKDF_SHA256_DERIVE_KEY
 - [tee_api.defines_extensions.h](#), [145](#)
- TEE_ALG_HKDF_SHA384_DERIVE_KEY
 - [tee_api.defines_extensions.h](#), [145](#)
- TEE_ALG_HKDF_SHA512_DERIVE_KEY
 - [tee_api.defines_extensions.h](#), [145](#)
- TEE_ALG_HMAC_MD5
 - [tee_api.defines.h](#), [108](#)
- TEE_ALG_HMAC_SHA1
 - [tee_api.defines.h](#), [108](#)
- TEE_ALG_HMAC_SHA224
 - [tee_api.defines.h](#), [108](#)
- TEE_ALG_HMAC_SHA256

tee_api.defines.h, [108](#)
TEE_ALG_HMAC_SHA384
tee_api.defines.h, [108](#)
TEE_ALG_HMAC_SHA512
tee_api.defines.h, [109](#)
TEE_ALG_MD5
tee_api.defines.h, [109](#)
TEE_ALG_MD5SHA1
tee_api.defines.h, [109](#)
TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY
tee_api.defines_extensions.h, [145](#)
TEE_ALG_RSA_NOPAD
tee_api.defines.h, [109](#)
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1
tee_api.defines.h, [109](#)
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224
tee_api.defines.h, [109](#)
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256
tee_api.defines.h, [109](#)
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384
tee_api.defines.h, [109](#)
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512
tee_api.defines.h, [109](#)
TEE_ALG_RSAES_PKCS1_V1_5
tee_api.defines.h, [109](#)
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1
tee_api.defines.h, [110](#)
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224
tee_api.defines.h, [110](#)
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256
tee_api.defines.h, [110](#)
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384
tee_api.defines.h, [110](#)
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512
tee_api.defines.h, [110](#)
TEE_ALG_RSASSA_PKCS1_V1_5_MD5
tee_api.defines.h, [110](#)
TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1
tee_api.defines.h, [110](#)
TEE_ALG_RSASSA_PKCS1_V1_5_SHA1
tee_api.defines.h, [110](#)
TEE_ALG_RSASSA_PKCS1_V1_5_SHA224
tee_api.defines.h, [110](#)
TEE_ALG_RSASSA_PKCS1_V1_5_SHA256
tee_api.defines.h, [110](#)
TEE_ALG_RSASSA_PKCS1_V1_5_SHA384
tee_api.defines.h, [111](#)
TEE_ALG_RSASSA_PKCS1_V1_5_SHA512
tee_api.defines.h, [111](#)
TEE_ALG_SHA1
tee_api.defines.h, [111](#)
TEE_ALG_SHA224
tee_api.defines.h, [111](#)
TEE_ALG_SHA256
tee_api.defines.h, [111](#)
TEE_ALG_SHA384
tee_api.defines.h, [111](#)
TEE_ALG_SHA512

- tee_api_defines.h, 111
- TEE_AllocateOperation
 - tee-internal-api-cryptlib.c, 228
 - tee-ta-internal.h, 80
- TEE_AllocateTransientObject
 - tee-internal-api-cryptlib.c, 229
 - tee-ta-internal.h, 81
- tee_api_defines.h
 - TEE_ALG_AES_CBC_MAC_NOPAD, 105
 - TEE_ALG_AES_CBC_MAC_PKCS5, 105
 - TEE_ALG_AES_CBC_NOPAD, 105
 - TEE_ALG_AES_CCM, 105
 - TEE_ALG_AES_CMAC, 106
 - TEE_ALG_AES_CTR, 106
 - TEE_ALG_AES_CTS, 106
 - TEE_ALG_AES_ECB_NOPAD, 106
 - TEE_ALG_AES_GCM, 106
 - TEE_ALG_AES_XTS, 106
 - TEE_ALG_DES3_CBC_MAC_NOPAD, 106
 - TEE_ALG_DES3_CBC_MAC_PKCS5, 106
 - TEE_ALG_DES3_CBC_NOPAD, 106
 - TEE_ALG_DES3_ECB_NOPAD, 106
 - TEE_ALG_DES_CBC_MAC_NOPAD, 106
 - TEE_ALG_DES_CBC_MAC_PKCS5, 107
 - TEE_ALG_DES_CBC_NOPAD, 107
 - TEE_ALG_DES_ECB_NOPAD, 107
 - TEE_ALG_DH_DERIVE_SHARED_SECRET, 107
 - TEE_ALG_DSA_SHA1, 107
 - TEE_ALG_DSA_SHA224, 107
 - TEE_ALG_DSA_SHA256, 107
 - TEE_ALG_ECDH_P192, 107
 - TEE_ALG_ECDH_P224, 107
 - TEE_ALG_ECDH_P256, 107
 - TEE_ALG_ECDH_P384, 107
 - TEE_ALG_ECDH_P521, 108
 - TEE_ALG_ECDSA_P192, 108
 - TEE_ALG_ECDSA_P224, 108
 - TEE_ALG_ECDSA_P256, 108
 - TEE_ALG_ECDSA_P384, 108
 - TEE_ALG_ECDSA_P521, 108
 - TEE_ALG_HMAC_MD5, 108
 - TEE_ALG_HMAC_SHA1, 108
 - TEE_ALG_HMAC_SHA224, 108
 - TEE_ALG_HMAC_SHA256, 108
 - TEE_ALG_HMAC_SHA384, 108
 - TEE_ALG_HMAC_SHA512, 109
 - TEE_ALG_MD5, 109
 - TEE_ALG_MD5SHA1, 109
 - TEE_ALG_RSA_NOPAD, 109
 - TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1, 109
 - TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224, 109
 - TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256, 109
 - TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384, 109
 - TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512, 109
 - TEE_ALG_RSAES_PKCS1_V1_5, 109
 - TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1, 110
 - TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224, 110
 - TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256, 110
 - TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384, 110
 - TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512, 110
 - TEE_ALG_RSASSA_PKCS1_V1_5_MD5, 110
 - TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1, 110
 - TEE_ALG_RSASSA_PKCS1_V1_5_SHA1, 110
 - TEE_ALG_RSASSA_PKCS1_V1_5_SHA224, 110
 - TEE_ALG_RSASSA_PKCS1_V1_5_SHA256, 110
 - TEE_ALG_RSASSA_PKCS1_V1_5_SHA384, 111
 - TEE_ALG_RSASSA_PKCS1_V1_5_SHA512, 111
 - TEE_ALG_SHA1, 111
 - TEE_ALG_SHA224, 111
 - TEE_ALG_SHA256, 111
 - TEE_ALG_SHA384, 111
 - TEE_ALG_SHA512, 111
 - TEE_ATTR_BIT_PROTECTED, 111
 - TEE_ATTR_BIT_VALUE, 111
 - TEE_ATTR_DH_BASE, 111
 - TEE_ATTR_DH_PRIME, 111
 - TEE_ATTR_DH_PRIVATE_VALUE, 112
 - TEE_ATTR_DH_PUBLIC_VALUE, 112
 - TEE_ATTR_DH_SUBPRIME, 112
 - TEE_ATTR_DH_X_BITS, 112
 - TEE_ATTR_DSA_BASE, 112
 - TEE_ATTR_DSA_PRIME, 112
 - TEE_ATTR_DSA_PRIVATE_VALUE, 112
 - TEE_ATTR_DSA_PUBLIC_VALUE, 112
 - TEE_ATTR_DSA_SUBPRIME, 112
 - TEE_ATTR_ECC_CURVE, 112
 - TEE_ATTR_ECC_PRIVATE_VALUE, 112
 - TEE_ATTR_ECC_PUBLIC_VALUE_X, 113
 - TEE_ATTR_ECC_PUBLIC_VALUE_Y, 113
 - TEE_ATTR_RSA_COEFFICIENT, 113
 - TEE_ATTR_RSA_EXPONENT1, 113
 - TEE_ATTR_RSA_EXPONENT2, 113
 - TEE_ATTR_RSA_MODULUS, 113
 - TEE_ATTR_RSA_OAEP_LABEL, 113
 - TEE_ATTR_RSA_PRIME1, 113
 - TEE_ATTR_RSA_PRIME2, 113
 - TEE_ATTR_RSA_PRIVATE_EXPONENT, 113
 - TEE_ATTR_RSA_PSS_SALT_LENGTH, 113
 - TEE_ATTR_RSA_PUBLIC_EXPONENT, 114
 - TEE_ATTR_SECRET_VALUE, 114
 - TEE_BigIntSizeInU32, 114
 - TEE_DATA_FLAG_ACCESS_READ, 114
 - TEE_DATA_FLAG_ACCESS_WRITE, 114
 - TEE_DATA_FLAG_ACCESS_WRITE_META, 114
 - TEE_DATA_FLAG_OVERWRITE, 114

- TEE_DATA_FLAG.SHARE_READ, 114
- TEE_DATA_FLAG.SHARE_WRITE, 114
- TEE_DATA_MAX_POSITION, 114
- TEE_ECC_CURVE_NIST_P192, 114
- TEE_ECC_CURVE_NIST_P224, 115
- TEE_ECC_CURVE_NIST_P256, 115
- TEE_ECC_CURVE_NIST_P384, 115
- TEE_ECC_CURVE_NIST_P521, 115
- TEE_ERROR.ACCESS.CONFLICT, 115
- TEE_ERROR.ACCESS.DENIED, 115
- TEE_ERROR.BAD.FORMAT, 115
- TEE_ERROR.BAD.PARAMETERS, 115
- TEE_ERROR.BAD.STATE, 115
- TEE_ERROR.BUSY, 115
- TEE_ERROR.CANCEL, 115
- TEE_ERROR.COMMUNICATION, 116
- TEE_ERROR.CORRUPT.OBJECT, 116
- TEE_ERROR.CORRUPT.OBJECT.2, 116
- TEE_ERROR.EXCESS.DATA, 116
- TEE_ERROR.EXTERNAL.CANCEL, 116
- TEE_ERROR.GENERIC, 116
- TEE_ERROR.ITEM.NOT.FOUND, 116
- TEE_ERROR.MAC.INVALID, 116
- TEE_ERROR.NO.DATA, 116
- TEE_ERROR.NOT.IMPLEMENTED, 116
- TEE_ERROR.NOT.SUPPORTED, 116
- TEE_ERROR.OUT.OF.MEMORY, 117
- TEE_ERROR.OVERFLOW, 117
- TEE_ERROR.SECURITY, 117
- TEE_ERROR.SHORT.BUFFER, 117
- TEE_ERROR.SIGNATURE.INVALID, 117
- TEE_ERROR.STORAGE.NO.SPACE, 117
- TEE_ERROR.STORAGE.NOT.AVAILABLE, 117
- TEE_ERROR.STORAGE.NOT.AVAILABLE.2, 117
- TEE_ERROR.TARGET.DEAD, 117
- TEE_ERROR.TIME.NEEDS.RESET, 117
- TEE_ERROR.TIME.NOT.SET, 117
- TEE_HANDLE.FLAG.EXPECT.TWO.KEYS, 118
- TEE_HANDLE.FLAG.INITIALIZED, 118
- TEE_HANDLE.FLAG.KEY.SET, 118
- TEE_HANDLE.FLAG.PERSISTENT, 118
- TEE_HANDLE.NULL, 118
- TEE.INT.CORE.API.SPEC.VERSION, 118
- TEE.LOGIN.APPLICATION, 118
- TEE.LOGIN.APPLICATION.GROUP, 118
- TEE.LOGIN.APPLICATION.USER, 118
- TEE.LOGIN.GROUP, 118
- TEE.LOGIN.PUBLIC, 118
- TEE.LOGIN.TRUSTED.APP, 119
- TEE.LOGIN.USER, 119
- TEE.MALLOC.FILL.ZERO, 119
- TEE.MEMORY.ACCESS.ANY.OWNER, 119
- TEE.MEMORY.ACCESS.READ, 119
- TEE.MEMORY.ACCESS.WRITE, 119
- TEE.NUM.PARAMS, 119
- TEE.OBJECT.ID.MAX.LEN, 119
- TEE.OPERATION.AE, 119
- TEE.OPERATION.ASYMMETRIC.CIPHER, 119
- TEE.OPERATION.ASYMMETRIC.SIGNATURE, 119
- TEE.OPERATION.CIPHER, 120
- TEE.OPERATION.DIGEST, 120
- TEE.OPERATION.KEY.DERIVATION, 120
- TEE.OPERATION.MAC, 120
- TEE.OPERATION.STATE.ACTIVE, 120
- TEE.OPERATION.STATE.INITIAL, 120
- TEE.ORIGIN.API, 120
- TEE.ORIGIN.COMMS, 120
- TEE.ORIGIN.TEE, 120
- TEE.ORIGIN.TRUSTED.APP, 120
- TEE.PANIC.ID.TA.CLOSESESSIONENTRYPOINT, 120
- TEE.PANIC.ID.TA.CREATEENTRYPOINT, 121
- TEE.PANIC.ID.TA.DESTROYENTRYPOINT, 121
- TEE.PANIC.ID.TA.INVOKECOMMANDENTRYPOINT, 121
- TEE.PANIC.ID.TA.OPENSESSIONENTRYPOINT, 121
- TEE.PANIC.ID.TEE.AEDECRYPTFINAL, 121
- TEE.PANIC.ID.TEE.AEENCRYPTFINAL, 121
- TEE.PANIC.ID.TEE.AEINIT, 121
- TEE.PANIC.ID.TEE.AEUPDATE, 121
- TEE.PANIC.ID.TEE.AEUPDATEAAD, 121
- TEE.PANIC.ID.TEE.ALLOCATEOPERATION, 121
- TEE.PANIC.ID.TEE.ALLOCATEPERSISTENTOBJECTENUMERATOR, 122
- TEE.PANIC.ID.TEE.ALLOCATEPROPERTYENUMERATOR, 122
- TEE.PANIC.ID.TEE.ALLOCATETRANSIENTOBJECT, 122
- TEE.PANIC.ID.TEE.ASYMMETRICDECRYPT, 122
- TEE.PANIC.ID.TEE.ASYMMETRICENCRYPT, 122
- TEE.PANIC.ID.TEE.ASYMMETRICSIGNDIGEST, 122
- TEE.PANIC.ID.TEE.ASYMMETRICVERIFYDIGEST, 122
- TEE.PANIC.ID.TEE.BIGINTADD, 122
- TEE.PANIC.ID.TEE.BIGINTADDMOD, 122
- TEE.PANIC.ID.TEE.BIGINTCMP, 122
- TEE.PANIC.ID.TEE.BIGINTCMPS32, 123
- TEE.PANIC.ID.TEE.BIGINTCOMPUTEEXTENDEDGCD, 123
- TEE.PANIC.ID.TEE.BIGINTCOMPUTEFGMM, 123
- TEE.PANIC.ID.TEE.BIGINTCONVERTFROMFGMM, 123
- TEE.PANIC.ID.TEE.BIGINTCONVERTFROMOCTETSTRING, 123
- TEE.PANIC.ID.TEE.BIGINTCONVERTFROMS32, 123
- TEE.PANIC.ID.TEE.BIGINTCONVERTTOFGMM, 123
- TEE.PANIC.ID.TEE.BIGINTCONVERTTOOCTETSTRING, 123
- TEE.PANIC.ID.TEE.BIGINTCONVERTTOS32, 123

- 123
- TEE.PANIC.ID.TEE.BIGINTDIV, 123
- TEE.PANIC.ID.TEE.BIGINTFMMCONTEXTSIZEINU32, 124
- TEE.PANIC.ID.TEE.BIGINTFMMSIZEINU32, 124
- TEE.PANIC.ID.TEE.BIGINTGETBIT, 124
- TEE.PANIC.ID.TEE.BIGINTGETBITCOUNT, 124
- TEE.PANIC.ID.TEE.BIGINTINIT, 124
- TEE.PANIC.ID.TEE.BIGINTINITFMM, 124
- TEE.PANIC.ID.TEE.BIGINTINITFMMCONTEXT, 124
- TEE.PANIC.ID.TEE.BIGINTINVMOD, 124
- TEE.PANIC.ID.TEE.BIGINTISPROBABLEPRIME, 124
- TEE.PANIC.ID.TEE.BIGINTMOD, 124
- TEE.PANIC.ID.TEE.BIGINTMUL, 125
- TEE.PANIC.ID.TEE.BIGINTMULMOD, 125
- TEE.PANIC.ID.TEE.BIGINTNEG, 125
- TEE.PANIC.ID.TEE.BIGINTRELATIVEPRIME, 125
- TEE.PANIC.ID.TEE.BIGINTSHIFTRIGHT, 125
- TEE.PANIC.ID.TEE.BIGINTSQUARE, 125
- TEE.PANIC.ID.TEE.BIGINTSQUAREMOD, 125
- TEE.PANIC.ID.TEE.BIGINTSUB, 125
- TEE.PANIC.ID.TEE.BIGINTSUBMOD, 125
- TEE.PANIC.ID.TEE.CHECKMEMORYACCESSRIGHTS, 125
- TEE.PANIC.ID.TEE.CIPHERDOFINAL, 125
- TEE.PANIC.ID.TEE.CIPHERINIT, 126
- TEE.PANIC.ID.TEE.CIPHERUPDATE, 126
- TEE.PANIC.ID.TEE.CLOSEANDDELETEPERSISTENTOBJECT, 126
- TEE.PANIC.ID.TEE.CLOSEANDDELETEPERSISTENTOBJECT, 126
- TEE.PANIC.ID.TEE.CLOSEOBJECT, 126
- TEE.PANIC.ID.TEE.CLOSETASESSION, 126
- TEE.PANIC.ID.TEE.COPYOBJECTATTRIBUTES, 126
- TEE.PANIC.ID.TEE.COPYOBJECTATTRIBUTES1, 126
- TEE.PANIC.ID.TEE.COPYOPERATION, 126
- TEE.PANIC.ID.TEE.CREATEPERSISTENTOBJECT, 126
- TEE.PANIC.ID.TEE.DERIVEKEY, 127
- TEE.PANIC.ID.TEE.DIGESTDOFINAL, 127
- TEE.PANIC.ID.TEE.DIGESTUPDATE, 127
- TEE.PANIC.ID.TEE.FREE, 127
- TEE.PANIC.ID.TEE.FREEOPERATION, 127
- TEE.PANIC.ID.TEE.FREEPERSISTENTOBJECTENUMERATOR, 127
- TEE.PANIC.ID.TEE.FREEPROPERTYENUMERATOR, 127
- TEE.PANIC.ID.TEE.FREETRANSIENTOBJECT, 127
- TEE.PANIC.ID.TEE.GENERATEKEY, 127
- TEE.PANIC.ID.TEE.GENERATERANDOM, 127
- TEE.PANIC.ID.TEE.GETCANCELLATIONFLAG, 128
- TEE.PANIC.ID.TEE.GETINSTANCEDATA, 128
- TEE.PANIC.ID.TEE.GETNEXTPERSISTENTOBJECT, 128
- TEE.PANIC.ID.TEE.GETNEXTPROPERTY, 128
- TEE.PANIC.ID.TEE.GETOBJECTBUFFERATTRIBUTE, 128
- TEE.PANIC.ID.TEE.GETOBJECTINFO, 128
- TEE.PANIC.ID.TEE.GETOBJECTINFO1, 128
- TEE.PANIC.ID.TEE.GETOBJECTVALUEATTRIBUTE, 128
- TEE.PANIC.ID.TEE.GETOPERATIONINFO, 128
- TEE.PANIC.ID.TEE.GETOPERATIONINFOMULTIPLE, 128
- TEE.PANIC.ID.TEE.GETPROPERTYASBINARYBLOCK, 129
- TEE.PANIC.ID.TEE.GETPROPERTYASBOOL, 129
- TEE.PANIC.ID.TEE.GETPROPERTYASIDENTITY, 129
- TEE.PANIC.ID.TEE.GETPROPERTYASSTRING, 129
- TEE.PANIC.ID.TEE.GETPROPERTYASU32, 129
- TEE.PANIC.ID.TEE.GETPROPERTYASUUID, 129
- TEE.PANIC.ID.TEE.GETPROPERTYNAME, 129
- TEE.PANIC.ID.TEE.GETREETIME, 129
- TEE.PANIC.ID.TEE.GETSYSTEMTIME, 129
- TEE.PANIC.ID.TEE.GETTAPERSISTENTTIME, 129
- TEE.PANIC.ID.TEE.INITREFATTRIBUTE, 130
- TEE.PANIC.ID.TEE.INITVALUEATTRIBUTE, 130
- TEE.PANIC.ID.TEE.INVOKETACOMMAND, 130
- TEE.PANIC.ID.TEE.MACCOMPAREFINAL, 130
- TEE.PANIC.ID.TEE.MACCOMPUTEFINAL, 130
- TEE.PANIC.ID.TEE.MACINIT, 130
- TEE.PANIC.ID.TEE.MACUPDATE, 130
- TEE.PANIC.ID.TEE.MALLOC, 130
- TEE.PANIC.ID.TEE.MASKCANCELLATION, 130
- TEE.PANIC.ID.TEE.MEMCOMPARE, 130
- TEE.PANIC.ID.TEE.MEMFILL, 131
- TEE.PANIC.ID.TEE.MEMMOVE, 131
- TEE.PANIC.ID.TEE.OPENPERSISTENTOBJECT, 131
- TEE.PANIC.ID.TEE.OPENTASESSION, 131
- TEE.PANIC.ID.TEE.PANIC, 131
- TEE.PANIC.ID.TEE.POPULATETRANSIENTOBJECT, 131
- TEE.PANIC.ID.TEE.READOBJECTDATA, 131
- TEE.PANIC.ID.TEE.REALLOC, 131
- TEE.PANIC.ID.TEE.RENAMEPERSISTENTOBJECT, 131
- TEE.PANIC.ID.TEE.RESETOPERATION, 131
- TEE.PANIC.ID.TEE.RESETPERSISTENTOBJECTENUMERATOR, 132
- TEE.PANIC.ID.TEE.RESETPROPERTYENUMERATOR, 132
- TEE.PANIC.ID.TEE.RESETTRANSIENTOBJECT, 132

- TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE, 132
- TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE1, 132
- TEE_PANIC_ID_TEE_SEEKOBJECTDATA, 132
- TEE_PANIC_ID_TEE_SETINSTANCEDATA, 132
- TEE_PANIC_ID_TEE_SETOPERATIONKEY, 132
- TEE_PANIC_ID_TEE_SETOPERATIONKEY2, 132
- TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME, 132
- TEE_PANIC_ID_TEE_STARTPERSISTENTOBJECTENUMERATOR, 133
- TEE_PANIC_ID_TEE_STARTPROPERTYENUMERATOR, 133
- TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA, 133
- TEE_PANIC_ID_TEE_UNMASKCANCELLATION, 133
- TEE_PANIC_ID_TEE_WAIT, 133
- TEE_PANIC_ID_TEE_WRITEOBJECTDATA, 133
- TEE_PARAM_TYPE_GET, 133
- TEE_PARAM_TYPE_MEMREF_INOUT, 133
- TEE_PARAM_TYPE_MEMREF_INPUT, 133
- TEE_PARAM_TYPE_MEMREF_OUTPUT, 133
- TEE_PARAM_TYPE_NONE, 134
- TEE_PARAM_TYPE_SET, 134
- TEE_PARAM_TYPE_VALUE_INOUT, 134
- TEE_PARAM_TYPE_VALUE_INPUT, 134
- TEE_PARAM_TYPE_VALUE_OUTPUT, 134
- TEE_PARAM_TYPES, 134
- TEE_PROPSET_CURRENT_CLIENT, 134
- TEE_PROPSET_CURRENT_TA, 134
- TEE_PROPSET_TEE_IMPLEMENTATION, 134
- TEE_STORAGE_PRIVATE, 134
- TEE_SUCCESS, 135
- TEE_TIMEOUT_INFINITE, 135
- TEE_TYPE_AES, 135
- TEE_TYPE_CORRUPTED_OBJECT, 135
- TEE_TYPE_DATA, 135
- TEE_TYPE_DES, 135
- TEE_TYPE_DES3, 135
- TEE_TYPE_DH_KEYPAIR, 135
- TEE_TYPE_DSA_KEYPAIR, 135
- TEE_TYPE_DSA_PUBLIC_KEY, 135
- TEE_TYPE_ECDH_KEYPAIR, 135
- TEE_TYPE_ECDH_PUBLIC_KEY, 136
- TEE_TYPE_ECDSA_KEYPAIR, 136
- TEE_TYPE_ECDSA_PUBLIC_KEY, 136
- TEE_TYPE_GENERIC_SECRET, 136
- TEE_TYPE_HMAC_MD5, 136
- TEE_TYPE_HMAC_SHA1, 136
- TEE_TYPE_HMAC_SHA224, 136
- TEE_TYPE_HMAC_SHA256, 136
- TEE_TYPE_HMAC_SHA384, 136
- TEE_TYPE_HMAC_SHA512, 136
- TEE_TYPE_RSA_KEYPAIR, 136
- TEE_TYPE_RSA_PUBLIC_KEY, 137
- TEE_USAGE_DECRYPT, 137
- TEE_USAGE_DERIVE, 137
- TEE_USAGE_ENCRYPT, 137
- TEE_USAGE_EXTRACTABLE, 137
- TEE_USAGE_MAC, 137
- TEE_USAGE_SIGN, 137
- TEE_USAGE_VERIFY, 137
- tee_api_defines_extensions.h
 - TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY, 144
 - TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY, 144
 - TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY, 144
 - TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY, 144
 - TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY, 145
 - TEE_ALG_HKDF_MD5_DERIVE_KEY, 145
 - TEE_ALG_HKDF_SHA1_DERIVE_KEY, 145
 - TEE_ALG_HKDF_SHA224_DERIVE_KEY, 145
 - TEE_ALG_HKDF_SHA256_DERIVE_KEY, 145
 - TEE_ALG_HKDF_SHA384_DERIVE_KEY, 145
 - TEE_ALG_HKDF_SHA512_DERIVE_KEY, 145
 - TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY, 145
 - TEE_ATTR_CONCAT_KDF_DKM_LENGTH, 145
 - TEE_ATTR_CONCAT_KDF_OTHER_INFO, 145
 - TEE_ATTR_CONCAT_KDF_Z, 145
 - TEE_ATTR_HKDF_IKM, 146
 - TEE_ATTR_HKDF_INFO, 146
 - TEE_ATTR_HKDF_OKM_LENGTH, 146
 - TEE_ATTR_HKDF_SALT, 146
 - TEE_ATTR_PBKDF2_DKM_LENGTH, 146
 - TEE_ATTR_PBKDF2_ITERATION_COUNT, 146
 - TEE_ATTR_PBKDF2_PASSWORD, 146
 - TEE_ATTR_PBKDF2_SALT, 146
 - TEE_MEMORY_ACCESS_NONSECURE, 146
 - TEE_MEMORY_ACCESS_SECURE, 146
 - TEE_STORAGE_PRIVATE_REE, 146
 - TEE_STORAGE_PRIVATE_RPMB, 147
 - TEE_STORAGE_PRIVATE_SQL_RESERVED, 147
 - TEE_TYPE_CONCAT_KDF_Z, 147
 - TEE_TYPE_HKDF_IKM, 147
 - TEE_TYPE_PBKDF2_PASSWORD, 147
- tee_api_tee_types.h
 - AES256, 211, 215
 - MBEDCRYPT, 211, 215
 - SHA_LENGTH, 211, 215
 - TEE_HANDLE_NULL, 215
 - TEE_OBJECT_AAD_SIZE, 211, 215
 - TEE_OBJECT_KEY_SIZE, 211, 215
 - TEE_OBJECT_NONCE_SIZE, 211, 215
 - TEE_OBJECT_SKEY_SIZE, 211, 215
 - TEE_OBJECT_TAG_SIZE, 211, 215
 - WOLFCRYPT, 211, 215
- tee_api_types.h
 - ...aligned, 151
 - DMREQ_FINISH, 150

- DMREQ.WRITE, 150
- nfds_t, 151
- socklen_t, 151
- TEE_BigInt, 151
- TEE_BigIntFMM, 151
- TEE_DATA_SEEK_CUR, 153
- TEE_DATA_SEEK_END, 153
- TEE_DATA_SEEK_SET, 153
- TEE_ErrorOrigin, 151
- TEE_MEM_INPUT, 150
- TEE_MEM_OUTPUT, 150
- TEE_MEMREF_0_USED, 150
- TEE_MEMREF_1_USED, 151
- TEE_MEMREF_2_USED, 151
- TEE_MEMREF_3_USED, 151
- TEE_MODE_DECRYPT, 153
- TEE_MODE_DERIVE, 153
- TEE_MODE_DIGEST, 153
- TEE_MODE_ENCRYPT, 153
- TEE_MODE_MAC, 153
- TEE_MODE_SIGN, 153
- TEE_MODE_VERIFY, 153
- TEE_ObjectEnumHandle, 151
- TEE_ObjectHandle, 152
- TEE_ObjectType, 152
- TEE_OperationHandle, 152
- TEE_OperationMode, 152
- TEE_PropSetHandle, 152
- TEE_Result, 152
- TEE_SE_READER_NAME_MAX, 151
- TEE_SEChannelHandle, 152
- TEE_SEReaderHandle, 152
- TEE_SEServiceHandle, 152
- TEE_SESessionHandle, 152
- TEE_Session, 152
- TEE_TASessionHandle, 152
- TEE_Whence, 153
- TEE_AsymmetricSignDigest
 - tee-internal-api-cryptlib.c, 229
 - tee-ta-internal.h, 82
- TEE_AsymmetricVerifyDigest
 - tee-internal-api-cryptlib.c, 230
 - tee-ta-internal.h, 82
- TEE_ATTR_BIT_PROTECTED
 - tee_api_defines.h, 111
- TEE_ATTR_BIT_VALUE
 - tee_api_defines.h, 111
- TEE_ATTR_CONCAT_KDF_DKM_LENGTH
 - tee_api_defines_extensions.h, 145
- TEE_ATTR_CONCAT_KDF_OTHER_INFO
 - tee_api_defines_extensions.h, 145
- TEE_ATTR_CONCAT_KDF_Z
 - tee_api_defines_extensions.h, 145
- TEE_ATTR_DH_BASE
 - tee_api_defines.h, 111
- TEE_ATTR_DH_PRIME
 - tee_api_defines.h, 111
- TEE_ATTR_DH_PRIVATE_VALUE
 - tee_api_defines.h, 112
- TEE_ATTR_DH_PUBLIC_VALUE
 - tee_api_defines.h, 112
- TEE_ATTR_DH_SUBPRIME
 - tee_api_defines.h, 112
- TEE_ATTR_DH_X_BITS
 - tee_api_defines.h, 112
- TEE_ATTR_DSA_BASE
 - tee_api_defines.h, 112
- TEE_ATTR_DSA_PRIME
 - tee_api_defines.h, 112
- TEE_ATTR_DSA_PRIVATE_VALUE
 - tee_api_defines.h, 112
- TEE_ATTR_DSA_PUBLIC_VALUE
 - tee_api_defines.h, 112
- TEE_ATTR_DSA_SUBPRIME
 - tee_api_defines.h, 112
- TEE_ATTR_ECC_CURVE
 - tee_api_defines.h, 112
- TEE_ATTR_ECC_PRIVATE_VALUE
 - tee_api_defines.h, 112
- TEE_ATTR_ECC_PUBLIC_VALUE_X
 - tee_api_defines.h, 113
- TEE_ATTR_ECC_PUBLIC_VALUE_Y
 - tee_api_defines.h, 113
- TEE_ATTR_HKDF_IKM
 - tee_api_defines_extensions.h, 146
- TEE_ATTR_HKDF_INFO
 - tee_api_defines_extensions.h, 146
- TEE_ATTR_HKDF_OKM_LENGTH
 - tee_api_defines_extensions.h, 146
- TEE_ATTR_HKDF_SALT
 - tee_api_defines_extensions.h, 146
- TEE_ATTR_PBKDF2_DKM_LENGTH
 - tee_api_defines_extensions.h, 146
- TEE_ATTR_PBKDF2_ITERATION_COUNT
 - tee_api_defines_extensions.h, 146
- TEE_ATTR_PBKDF2_PASSWORD
 - tee_api_defines_extensions.h, 146
- TEE_ATTR_PBKDF2_SALT
 - tee_api_defines_extensions.h, 146
- TEE_ATTR_RSA_COEFFICIENT
 - tee_api_defines.h, 113
- TEE_ATTR_RSA_EXPONENT1
 - tee_api_defines.h, 113
- TEE_ATTR_RSA_EXPONENT2
 - tee_api_defines.h, 113
- TEE_ATTR_RSA_MODULUS
 - tee_api_defines.h, 113
- TEE_ATTR_RSA_OAEP_LABEL
 - tee_api_defines.h, 113
- TEE_ATTR_RSA_PRIME1
 - tee_api_defines.h, 113
- TEE_ATTR_RSA_PRIME2
 - tee_api_defines.h, 113
- TEE_ATTR_RSA_PRIVATE_EXPONENT
 - tee_api_defines.h, 113
- TEE_ATTR_RSA_PSS_SALT_LENGTH

- tee_api_defines.h, 113
- TEE_ATTR_RSA_PUBLIC_EXPONENT
 - tee_api_defines.h, 114
- TEE_ATTR_SECRET_VALUE
 - tee_api_defines.h, 114
- TEE_Attribute, 39
 - a, 39
 - attributeID, 40
 - b, 40
 - buffer, 40
 - content, 40
 - length, 40
 - ref, 40
 - value, 40
- TEE_BigInt
 - tee_api_types.h, 151
- TEE_BigIntFMM
 - tee_api_types.h, 151
- TEE_BigIntSizeInU32
 - tee_api_defines.h, 114
- TEE_CacheClean
 - tee_internal_api_extensions.h, 172
- TEE_CacheFlush
 - tee_internal_api_extensions.h, 172
- TEE_CacheInvalidate
 - tee_internal_api_extensions.h, 172
- TEE_CipherDoFinal
 - tee-internal-api-cryptlib.c, 230
- TEE_CipherInit
 - tee-internal-api-cryptlib.c, 231
 - tee-ta-internal.h, 83
- TEE_CipherUpdate
 - tee-internal-api-cryptlib.c, 231
 - tee-ta-internal.h, 84
- tee_client_api.h
 - TEEC_AllocateSharedMemory, 164
 - TEEC_CloseSession, 165
 - TEEC_CONFIG_PAYLOAD_REF_COUNT, 158
 - TEEC_CONFIG_SHARED_MEM_MAX_SIZE, 159
 - TEEC_ERROR_ACCESS_CONFLICT, 159
 - TEEC_ERROR_ACCESS_DENIED, 159
 - TEEC_ERROR_BAD_FORMAT, 159
 - TEEC_ERROR_BAD_PARAMETERS, 159
 - TEEC_ERROR_BAD_STATE, 159
 - TEEC_ERROR_BUSY, 159
 - TEEC_ERROR_CANCEL, 159
 - TEEC_ERROR_COMMUNICATION, 159
 - TEEC_ERROR_EXCESS_DATA, 159
 - TEEC_ERROR_EXTERNAL_CANCEL, 159
 - TEEC_ERROR_GENERIC, 160
 - TEEC_ERROR_ITEM_NOT_FOUND, 160
 - TEEC_ERROR_NO_DATA, 160
 - TEEC_ERROR_NOT_IMPLEMENTED, 160
 - TEEC_ERROR_NOT_SUPPORTED, 160
 - TEEC_ERROR_OUT_OF_MEMORY, 160
 - TEEC_ERROR_SECURITY, 160
 - TEEC_ERROR_SHORT_BUFFER, 160
 - TEEC_ERROR_TARGET_DEAD, 160
 - TEEC_FinalizeContext, 165
 - TEEC_InitializeContext, 166
 - TEEC_InvokeCommand, 166
 - TEEC_LOGIN_APPLICATION, 160
 - TEEC_LOGIN_GROUP, 160
 - TEEC_LOGIN_GROUP_APPLICATION, 161
 - TEEC_LOGIN_PUBLIC, 161
 - TEEC_LOGIN_USER, 161
 - TEEC_LOGIN_USER_APPLICATION, 161
 - TEEC_MEM_INPUT, 161
 - TEEC_MEM_OUTPUT, 161
 - TEEC_MEMREF_PARTIAL_INOUT, 161
 - TEEC_MEMREF_PARTIAL_INPUT, 161
 - TEEC_MEMREF_PARTIAL_OUTPUT, 161
 - TEEC_MEMREF_TEMP_INOUT, 162
 - TEEC_MEMREF_TEMP_INPUT, 162
 - TEEC_MEMREF_TEMP_OUTPUT, 162
 - TEEC_MEMREF_WHOLE, 162
 - TEEC_NONE, 162
 - TEEC_OpenSession, 167
 - TEEC_ORIGIN_API, 162
 - TEEC_ORIGIN_COMMS, 163
 - TEEC_ORIGIN_TEE, 163
 - TEEC_ORIGIN_TRUSTED_APP, 163
 - TEEC_PARAM_TYPE_GET, 163
 - TEEC_PARAM_TYPES, 163
 - TEEC_RegisterSharedMemory, 167
 - TEEC_ReleaseSharedMemory, 168
 - TEEC_RequestCancellation, 168
 - TEEC_Result, 164
 - TEEC_SUCCESS, 164
 - TEEC_VALUE_INOUT, 164
 - TEEC_VALUE_INPUT, 164
 - TEEC_VALUE_OUTPUT, 164
- TEE_CloseObject
 - tee-internal-api.c, 195, 205
 - tee-ta-internal.h, 84
- TEE_CreatePersistentObject
 - tee-internal-api.c, 195, 206
 - tee-ta-internal.h, 85
- TEE_DATA_FLAG_ACCESS_READ
 - tee_api_defines.h, 114
- TEE_DATA_FLAG_ACCESS_WRITE
 - tee_api_defines.h, 114
- TEE_DATA_FLAG_ACCESS_WRITE_META
 - tee_api_defines.h, 114
- TEE_DATA_FLAG_OVERWRITE
 - tee_api_defines.h, 114
- TEE_DATA_FLAG_SHARE_READ
 - tee_api_defines.h, 114
- TEE_DATA_FLAG_SHARE_WRITE
 - tee_api_defines.h, 114
- TEE_DATA_MAX_POSITION
 - tee_api_defines.h, 114
- TEE_DATA_SEEK_CUR
 - tee_api_types.h, 153
- TEE_DATA_SEEK_END
 - tee_api_types.h, 153

- TEE_DATA_SEEK_SET
 - tee_api.types.h, 153
- TEE_DigestDoFinal
 - tee-internal-api-cryptlib.c, 232
 - tee-ta-internal.h, 86
- TEE_DigestUpdate
 - tee-internal-api-cryptlib.c, 233
 - tee-ta-internal.h, 87
- TEE_ECC_CURVE_NIST_P192
 - tee_api.defines.h, 114
- TEE_ECC_CURVE_NIST_P224
 - tee_api.defines.h, 115
- TEE_ECC_CURVE_NIST_P256
 - tee_api.defines.h, 115
- TEE_ECC_CURVE_NIST_P384
 - tee_api.defines.h, 115
- TEE_ECC_CURVE_NIST_P521
 - tee_api.defines.h, 115
- TEE_ERROR_ACCESS_CONFLICT
 - tee_api.defines.h, 115
- TEE_ERROR_ACCESS_DENIED
 - tee_api.defines.h, 115
- TEE_ERROR_BAD_FORMAT
 - tee_api.defines.h, 115
- TEE_ERROR_BAD_PARAMETERS
 - tee_api.defines.h, 115
- TEE_ERROR_BAD_STATE
 - tee_api.defines.h, 115
- TEE_ERROR_BUSY
 - tee_api.defines.h, 115
- TEE_ERROR_CANCEL
 - tee_api.defines.h, 115
- TEE_ERROR_COMMUNICATION
 - tee_api.defines.h, 116
- TEE_ERROR_CORRUPT_OBJECT
 - tee_api.defines.h, 116
- TEE_ERROR_CORRUPT_OBJECT_2
 - tee_api.defines.h, 116
- TEE_ERROR_EXCESS_DATA
 - tee_api.defines.h, 116
- TEE_ERROR_EXTERNAL_CANCEL
 - tee_api.defines.h, 116
- TEE_ERROR_GENERIC
 - tee_api.defines.h, 116
- TEE_ERROR_ITEM_NOT_FOUND
 - tee_api.defines.h, 116
- TEE_ERROR_MAC_INVALID
 - tee_api.defines.h, 116
- TEE_ERROR_NO_DATA
 - tee_api.defines.h, 116
- TEE_ERROR_NOT_IMPLEMENTED
 - tee_api.defines.h, 116
- TEE_ERROR_NOT_SUPPORTED
 - tee_api.defines.h, 116
- TEE_ERROR_OUT_OF_MEMORY
 - tee_api.defines.h, 117
- TEE_ERROR_OVERFLOW
 - tee_api.defines.h, 117
- TEE_ERROR_SECURITY
 - tee_api.defines.h, 117
- TEE_ERROR_SHORT_BUFFER
 - tee_api.defines.h, 117
- TEE_ERROR_SIGNATURE_INVALID
 - tee_api.defines.h, 117
- TEE_ERROR_STORAGE_NO_SPACE
 - tee_api.defines.h, 117
- TEE_ERROR_STORAGE_NOT_AVAILABLE
 - tee_api.defines.h, 117
- TEE_ERROR_STORAGE_NOT_AVAILABLE_2
 - tee_api.defines.h, 117
- TEE_ERROR_TARGET_DEAD
 - tee_api.defines.h, 117
- TEE_ERROR_TIME_NEEDS_RESET
 - tee_api.defines.h, 117
- TEE_ERROR_TIME_NOT_SET
 - tee_api.defines.h, 117
- TEE_ErrorOrigin
 - tee_api.types.h, 151
- TEE_Free
 - tee-internal-api.c, 196
- TEE_FreeOperation
 - tee-internal-api-cryptlib.c, 233
 - tee-ta-internal.h, 87
- TEE_FreeTransientObject
 - tee-internal-api-cryptlib.c, 233
 - tee-ta-internal.h, 88
- TEE_GenerateKey
 - tee-internal-api-cryptlib.c, 234
 - tee-ta-internal.h, 88
- TEE_GenerateRandom
 - tee-internal-api.c, 196, 206
 - tee-ta-internal.h, 89
- TEE_GetObjectInfo1
 - tee-internal-api.c, 197, 207
 - tee-ta-internal.h, 90
- TEE_GetREETime
 - tee-internal-api.c, 197, 207
 - tee-ta-internal.h, 91
- TEE_GetSystemTime
 - tee-internal-api.c, 198, 208
 - tee-ta-internal.h, 91
- TEE_HANDLE_FLAG_EXPECT_TWO_KEYS
 - tee_api.defines.h, 118
- TEE_HANDLE_FLAG_INITIALIZED
 - tee_api.defines.h, 118
- TEE_HANDLE_FLAG_KEY_SET
 - tee_api.defines.h, 118
- TEE_HANDLE_FLAG_PERSISTENT
 - tee_api.defines.h, 118
- TEE_HANDLE_NULL
 - tee_api.defines.h, 118
 - tee_api.tee.types.h, 215
- TEE_Identity, 40
 - login, 41
 - uuid, 41
- TEE_InitRefAttribute

- tee-internal-api-cryptlib.c, 234
- tee-ta-internal.h, 92
- TEE_InitValueAttribute
 - tee-internal-api-cryptlib.c, 235
 - tee-ta-internal.h, 92
- TEE_INT_CORE_API_SPEC_VERSION
 - tee_api_defines.h, 118
- tee_internal_api_extensions.h
 - TEE_CacheClean, 172
 - TEE_CacheFlush, 172
 - TEE_CacheInvalidate, 172
 - tee_map_zi, 172
 - tee_unmap, 172
 - tee_user_mem_check_heap, 173
 - TEE_USER_MEM_HINT_NO_FILL_ZERO, 172
 - tee_user_mem_mark_heap, 173
 - tee_uuid_from_str, 173
- TEE_LOGIN_APPLICATION
 - tee_api_defines.h, 118
- TEE_LOGIN_APPLICATION_GROUP
 - tee_api_defines.h, 118
- TEE_LOGIN_APPLICATION_USER
 - tee_api_defines.h, 118
- TEE_LOGIN_GROUP
 - tee_api_defines.h, 118
- TEE_LOGIN_PUBLIC
 - tee_api_defines.h, 118
- TEE_LOGIN_TRUSTED_APP
 - tee_api_defines.h, 119
- TEE_LOGIN_USER
 - tee_api_defines.h, 119
- TEE_Malloc
 - tee-internal-api.c, 198
- TEE_MALLOC_FILL_ZERO
 - tee_api_defines.h, 119
- tee_map_zi
 - tee_internal_api_extensions.h, 172
- TEE_MEM_INPUT
 - tee_api_types.h, 150
- TEE_MEM_OUTPUT
 - tee_api_types.h, 150
- TEE_MEMORY_ACCESS_ANY_OWNER
 - tee_api_defines.h, 119
- TEE_MEMORY_ACCESS_NONSECURE
 - tee_api_defines_extensions.h, 146
- TEE_MEMORY_ACCESS_READ
 - tee_api_defines.h, 119
- TEE_MEMORY_ACCESS_SECURE
 - tee_api_defines_extensions.h, 146
- TEE_MEMORY_ACCESS_WRITE
 - tee_api_defines.h, 119
- TEE_MEMREF_0_USED
 - tee_api_types.h, 150
- TEE_MEMREF_1_USED
 - tee_api_types.h, 151
- TEE_MEMREF_2_USED
 - tee_api_types.h, 151
- TEE_MEMREF_3_USED
 - tee_api_types.h, 151
- TEE_MODE_DECRYPT
 - tee_api_types.h, 153
- TEE_MODE_DERIVE
 - tee_api_types.h, 153
- TEE_MODE_DIGEST
 - tee_api_types.h, 153
- TEE_MODE_ENCRYPT
 - tee_api_types.h, 153
- TEE_MODE_MAC
 - tee_api_types.h, 153
- TEE_MODE_SIGN
 - tee_api_types.h, 153
- TEE_MODE_VERIFY
 - tee_api_types.h, 153
- TEE_NUM_PARAMS
 - tee_api_defines.h, 119
- TEE_OBJECT_AAD_SIZE
 - tee_api_tee_types.h, 211, 215
- TEE_OBJECT_ID_MAX_LEN
 - tee_api_defines.h, 119
- TEE_OBJECT_KEY_SIZE
 - tee_api_tee_types.h, 211, 215
- TEE_OBJECT_NONCE_SIZE
 - tee_api_tee_types.h, 211, 215
- TEE_OBJECT_SKEY_SIZE
 - tee_api_tee_types.h, 211, 215
- TEE_OBJECT_TAG_SIZE
 - tee_api_tee_types.h, 211, 215
- TEE_ObjectEnumHandle
 - tee_api_types.h, 151
- TEE_ObjectHandle
 - tee_api_types.h, 152
- TEE_ObjectInfo, 41
 - dataPosition, 42
 - dataSize, 42
 - handleFlags, 42
 - keySize, 42
 - maxKeySize, 42
 - maxObjectSize, 42
 - objectSize, 42
 - objectType, 42
 - objectUsage, 42
- TEE_ObjectType
 - tee_api_types.h, 152
- TEE_OpenPersistentObject
 - tee-internal-api.c, 198, 208
 - tee-ta-internal.h, 93
- TEE_OPERATION_AE
 - tee_api_defines.h, 119
- TEE_OPERATION_ASYMMETRIC_CIPHER
 - tee_api_defines.h, 119
- TEE_OPERATION_ASYMMETRIC_SIGNATURE
 - tee_api_defines.h, 119
- TEE_OPERATION_CIPHER
 - tee_api_defines.h, 120
- TEE_OPERATION_DIGEST
 - tee_api_defines.h, 120

- TEE_OPERATION_KEY_DERIVATION
 - tee_api_defines.h, 120
- TEE_OPERATION_MAC
 - tee_api_defines.h, 120
- TEE_OPERATION_STATE_ACTIVE
 - tee_api_defines.h, 120
- TEE_OPERATION_STATE_INITIAL
 - tee_api_defines.h, 120
- TEE_OperationHandle
 - tee_api_types.h, 152
- TEE_OperationInfo, 42
 - algorithm, 43
 - digestLength, 43
 - handleState, 43
 - keySize, 43
 - maxKeySize, 43
 - mode, 43
 - operationClass, 43
 - requiredKeyUsage, 43
- TEE_OperationInfoKey, 44
 - keySize, 44
 - requiredKeyUsage, 44
- TEE_OperationInfoMultiple, 44
 - algorithm, 45
 - digestLength, 45
 - handleState, 45
 - keyInformation, 45
 - maxKeySize, 45
 - mode, 45
 - numberOfKeys, 45
 - operationClass, 45
 - operationState, 45
- TEE_OperationMode
 - tee_api_types.h, 152
- TEE_ORIGIN_API
 - tee_api_defines.h, 120
- TEE_ORIGIN_COMMS
 - tee_api_defines.h, 120
- TEE_ORIGIN_TEE
 - tee_api_defines.h, 120
- TEE_ORIGIN_TRUSTED_APP
 - tee_api_defines.h, 120
- TEE_PANIC_ID_TA_CLOSESESSIONENTRYPOINT
 - tee_api_defines.h, 120
- TEE_PANIC_ID_TA_CREATEENTRYPOINT
 - tee_api_defines.h, 121
- TEE_PANIC_ID_TA_DESTROYENTRYPOINT
 - tee_api_defines.h, 121
- TEE_PANIC_ID_TA_INVOKECOMMANDENTRYPOINT
 - tee_api_defines.h, 121
- TEE_PANIC_ID_TA_OPENSESSIONENTRYPOINT
 - tee_api_defines.h, 121
- TEE_PANIC_ID_TEE_AEDECRIPTFINAL
 - tee_api_defines.h, 121
- TEE_PANIC_ID_TEE_AEENCRYPTFINAL
 - tee_api_defines.h, 121
- TEE_PANIC_ID_TEE_AEINIT
 - tee_api_defines.h, 121
- TEE_PANIC_ID_TEE_AEUPDATE
 - tee_api_defines.h, 121
- TEE_PANIC_ID_TEE_AEUPDATEAAD
 - tee_api_defines.h, 121
- TEE_PANIC_ID_TEE_ALLOCATEOPERATION
 - tee_api_defines.h, 121
- TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR
 - tee_api_defines.h, 122
- TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERATOR
 - tee_api_defines.h, 122
- TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT
 - tee_api_defines.h, 122
- TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT
 - tee_api_defines.h, 122
- TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT
 - tee_api_defines.h, 122
- TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST
 - tee_api_defines.h, 122
- TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST
 - tee_api_defines.h, 122
- TEE_PANIC_ID_TEE_BIGINTADD
 - tee_api_defines.h, 122
- TEE_PANIC_ID_TEE_BIGINTADDMOD
 - tee_api_defines.h, 122
- TEE_PANIC_ID_TEE_BIGINTCMP
 - tee_api_defines.h, 122
- TEE_PANIC_ID_TEE_BIGINTCMPS32
 - tee_api_defines.h, 123
- TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD
 - tee_api_defines.h, 123
- TEE_PANIC_ID_TEE_BIGINTCOMPUTEFMM
 - tee_api_defines.h, 123
- TEE_PANIC_ID_TEE_BIGINTCONVERTFROMFMM
 - tee_api_defines.h, 123
- TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING
 - tee_api_defines.h, 123
- TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32
 - tee_api_defines.h, 123
- TEE_PANIC_ID_TEE_BIGINTCONVERTTOFMM
 - tee_api_defines.h, 123
- TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING
 - tee_api_defines.h, 123
- TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32
 - tee_api_defines.h, 123
- TEE_PANIC_ID_TEE_BIGINTDIV
 - tee_api_defines.h, 123
- TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32
 - tee_api_defines.h, 124
- TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32
 - tee_api_defines.h, 124
- TEE_PANIC_ID_TEE_BIGINTGETBIT
 - tee_api_defines.h, 124
- TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT
 - tee_api_defines.h, 124
- TEE_PANIC_ID_TEE_BIGINTINIT
 - tee_api_defines.h, 124
- TEE_PANIC_ID_TEE_BIGINTINITFMM
 - tee_api_defines.h, 124

- TEE.PANIC.ID.TEE.BIGINTINITFMMCONTEXT
tee_api_defines.h, 124
- TEE.PANIC.ID.TEE.BIGINTINVMOD
tee_api_defines.h, 124
- TEE.PANIC.ID.TEE.BIGINTISPROBABLEPRIME
tee_api_defines.h, 124
- TEE.PANIC.ID.TEE.BIGINTMOD
tee_api_defines.h, 124
- TEE.PANIC.ID.TEE.BIGINTMUL
tee_api_defines.h, 125
- TEE.PANIC.ID.TEE.BIGINTMULMOD
tee_api_defines.h, 125
- TEE.PANIC.ID.TEE.BIGINTNEG
tee_api_defines.h, 125
- TEE.PANIC.ID.TEE.BIGINTRELATIVEPRIME
tee_api_defines.h, 125
- TEE.PANIC.ID.TEE.BIGINTSHIFTRIGHT
tee_api_defines.h, 125
- TEE.PANIC.ID.TEE.BIGINTSQUARE
tee_api_defines.h, 125
- TEE.PANIC.ID.TEE.BIGINTSQUAREMOD
tee_api_defines.h, 125
- TEE.PANIC.ID.TEE.BIGINTSUB
tee_api_defines.h, 125
- TEE.PANIC.ID.TEE.BIGINTSUBMOD
tee_api_defines.h, 125
- TEE.PANIC.ID.TEE.CHECKMEMORYACCESSRIGHTS
tee_api_defines.h, 125
- TEE.PANIC.ID.TEE.CIPHERDOFINAL
tee_api_defines.h, 125
- TEE.PANIC.ID.TEE.CIPHERINIT
tee_api_defines.h, 126
- TEE.PANIC.ID.TEE.CIPHERUPDATE
tee_api_defines.h, 126
- TEE.PANIC.ID.TEE.CLOSEANDDELETEPERSISTENTOBJECT
tee_api_defines.h, 126
- TEE.PANIC.ID.TEE.CLOSEANDDELETEPERSISTENTOBJECT
tee_api_defines.h, 126
- TEE.PANIC.ID.TEE.CLOSEOBJECT
tee_api_defines.h, 126
- TEE.PANIC.ID.TEE.CLOSETASESSION
tee_api_defines.h, 126
- TEE.PANIC.ID.TEE.COPYOBJECTATTRIBUTES
tee_api_defines.h, 126
- TEE.PANIC.ID.TEE.COPYOBJECTATTRIBUTES1
tee_api_defines.h, 126
- TEE.PANIC.ID.TEE.COPYOPERATION
tee_api_defines.h, 126
- TEE.PANIC.ID.TEE.CREATEPERSISTENTOBJECT
tee_api_defines.h, 126
- TEE.PANIC.ID.TEE.DERIVEKEY
tee_api_defines.h, 127
- TEE.PANIC.ID.TEE.DIGESTDOFINAL
tee_api_defines.h, 127
- TEE.PANIC.ID.TEE.DIGESTUPDATE
tee_api_defines.h, 127
- TEE.PANIC.ID.TEE.FREE
tee_api_defines.h, 127
- TEE.PANIC.ID.TEE.FREEOPERATION
tee_api_defines.h, 127
- TEE.PANIC.ID.TEE.FREEPERSISTENTOBJECTENUMERATOR
tee_api_defines.h, 127
- TEE.PANIC.ID.TEE.FREEPROPERTYENUMERATOR
tee_api_defines.h, 127
- TEE.PANIC.ID.TEE.FREETRANSIENTOBJECT
tee_api_defines.h, 127
- TEE.PANIC.ID.TEE.GENERATEKEY
tee_api_defines.h, 127
- TEE.PANIC.ID.TEE.GENERATERANDOM
tee_api_defines.h, 127
- TEE.PANIC.ID.TEE.GETCANCELLATIONFLAG
tee_api_defines.h, 128
- TEE.PANIC.ID.TEE.GETINSTANCEDATA
tee_api_defines.h, 128
- TEE.PANIC.ID.TEE.GETNEXTPERSISTENTOBJECT
tee_api_defines.h, 128
- TEE.PANIC.ID.TEE.GETNEXTPROPERTY
tee_api_defines.h, 128
- TEE.PANIC.ID.TEE.GETOBJECTBUFFERATTRIBUTE
tee_api_defines.h, 128
- TEE.PANIC.ID.TEE.GETOBJECTINFO
tee_api_defines.h, 128
- TEE.PANIC.ID.TEE.GETOBJECTINFO1
tee_api_defines.h, 128
- TEE.PANIC.ID.TEE.GETOBJECTVALUEATTRIBUTE
tee_api_defines.h, 128
- TEE.PANIC.ID.TEE.GETOPERATIONINFO
tee_api_defines.h, 128
- TEE.PANIC.ID.TEE.GETOPERATIONINFOMULTIPLE
tee_api_defines.h, 128
- TEE.PANIC.ID.TEE.GETPROPERTYASBINARYBLOCK
tee_api_defines.h, 129
- TEE.PANIC.ID.TEE.GETPROPERTYASBOOL
tee_api_defines.h, 129
- TEE.PANIC.ID.TEE.GETPROPERTYASIDENTITY
tee_api_defines.h, 129
- TEE.PANIC.ID.TEE.GETPROPERTYASSTRING
tee_api_defines.h, 129
- TEE.PANIC.ID.TEE.GETPROPERTYASU32
tee_api_defines.h, 129
- TEE.PANIC.ID.TEE.GETPROPERTYASUUID
tee_api_defines.h, 129
- TEE.PANIC.ID.TEE.GETPROPERTYNAME
tee_api_defines.h, 129
- TEE.PANIC.ID.TEE.GETREETIME
tee_api_defines.h, 129
- TEE.PANIC.ID.TEE.GETSYSTEMTIME
tee_api_defines.h, 129
- TEE.PANIC.ID.TEE.GETTAPERSISTENTTIME
tee_api_defines.h, 129
- TEE.PANIC.ID.TEE.INITREFATTRIBUTE
tee_api_defines.h, 130
- TEE.PANIC.ID.TEE.INITVALUEATTRIBUTE
tee_api_defines.h, 130
- TEE.PANIC.ID.TEE.INVOKETACOMMAND
tee_api_defines.h, 130

- TEE.PANIC.ID.TEE.MACCOMPAREFINAL
tee_api_defines.h, 130
- TEE.PANIC.ID.TEE.MACCOMPUTEFINAL
tee_api_defines.h, 130
- TEE.PANIC.ID.TEE.MACINIT
tee_api_defines.h, 130
- TEE.PANIC.ID.TEE.MACUPDATE
tee_api_defines.h, 130
- TEE.PANIC.ID.TEE.MALLOC
tee_api_defines.h, 130
- TEE.PANIC.ID.TEE.MASKCANCELLATION
tee_api_defines.h, 130
- TEE.PANIC.ID.TEE.MEMCOMPARE
tee_api_defines.h, 130
- TEE.PANIC.ID.TEE.MEMFILL
tee_api_defines.h, 131
- TEE.PANIC.ID.TEE.MEMMOVE
tee_api_defines.h, 131
- TEE.PANIC.ID.TEE.OPENPERSISTENTOBJECT
tee_api_defines.h, 131
- TEE.PANIC.ID.TEE.OPENTASESSION
tee_api_defines.h, 131
- TEE.PANIC.ID.TEE.PANIC
tee_api_defines.h, 131
- TEE.PANIC.ID.TEE.POPULATETRANSIENTOBJECT
tee_api_defines.h, 131
- TEE.PANIC.ID.TEE.READOBJECTDATA
tee_api_defines.h, 131
- TEE.PANIC.ID.TEE.REALLOC
tee_api_defines.h, 131
- TEE.PANIC.ID.TEE.RENAMEPERSISTENTOBJECT
tee_api_defines.h, 131
- TEE.PANIC.ID.TEE.RESETOPERATION
tee_api_defines.h, 131
- TEE.PANIC.ID.TEE.RESETPERSISTENTOBJECTENUMERATOR
tee_api_defines.h, 132
- TEE.PANIC.ID.TEE.RESETPROPERTYENUMERATOR
tee_api_defines.h, 132
- TEE.PANIC.ID.TEE.RESETTRANSIENTOBJECT
tee_api_defines.h, 132
- TEE.PANIC.ID.TEE.RESTRICTOBJECTUSAGE
tee_api_defines.h, 132
- TEE.PANIC.ID.TEE.RESTRICTOBJECTUSAGE1
tee_api_defines.h, 132
- TEE.PANIC.ID.TEE.SEEKOBJECTDATA
tee_api_defines.h, 132
- TEE.PANIC.ID.TEE.SETINSTANCEDATA
tee_api_defines.h, 132
- TEE.PANIC.ID.TEE.SETOPERATIONKEY
tee_api_defines.h, 132
- TEE.PANIC.ID.TEE.SETOPERATIONKEY2
tee_api_defines.h, 132
- TEE.PANIC.ID.TEE.SETTAPERSISTENTTIME
tee_api_defines.h, 132
- TEE.PANIC.ID.TEE.STARTPERSISTENTOBJECTENUMERATOR
tee_api_defines.h, 133
- TEE.PANIC.ID.TEE.STARTPROPERTYENUMERATOR
tee_api_defines.h, 133
- TEE.PANIC.ID.TEE.TRUNCATEOBJECTDATA
tee_api_defines.h, 133
- TEE.PANIC.ID.TEE.UNMASKCANCELLATION
tee_api_defines.h, 133
- TEE.PANIC.ID.TEE.WAIT
tee_api_defines.h, 133
- TEE.PANIC.ID.TEE.WRITEOBJECTDATA
tee_api_defines.h, 133
- TEE.Param, 46
 - a, 46
 - b, 46
 - buffer, 46
 - memref, 46
 - size, 46
 - value, 46
- TEE.PARAM.TYPE.GET
tee_api_defines.h, 133
- TEE.PARAM.TYPE.MEMREF.INOUT
tee_api_defines.h, 133
- TEE.PARAM.TYPE.MEMREF.INPUT
tee_api_defines.h, 133
- TEE.PARAM.TYPE.MEMREF.OUTPUT
tee_api_defines.h, 133
- TEE.PARAM.TYPE.NONE
tee_api_defines.h, 134
- TEE.PARAM.TYPE.SET
tee_api_defines.h, 134
- TEE.PARAM.TYPE.VALUE.INOUT
tee_api_defines.h, 134
- TEE.PARAM.TYPE.VALUE.INPUT
tee_api_defines.h, 134
- TEE.PARAM.TYPE.VALUE.OUTPUT
tee_api_defines.h, 134
- TEE.PARAM.TYPES
tee_api_defines.h, 134
- TEE.PROPSET.CURRENT.CLIENT
tee_api_defines.h, 134
- TEE.PROPSET.CURRENT.TA
tee_api_defines.h, 134
- TEE.PROPSET.TEE.IMPLEMENTATION
tee_api_defines.h, 134
- TEE.PropSetHandle
tee_api_types.h, 152
- TEE.ReadObjectData
tee-internal-api.c, 199, 208
tee-ta-internal.h, 94
- TEE.Realloc
tee-internal-api.c, 200
- TEE.Result
tee_api_types.h, 152
- TEE.SE.READER.NAME.MAX
tee_api_types.h, 151
- TEE.SEAID, 47
 - buffer, 47
 - OfferLen, 47
- TEE.SEChannelHandle
tee_api_types.h, 152
- TEE.SEReaderHandle

- tee_api.types.h, 152
- TEE_SEReaderProperties, 47
 - selectResponseEnable, 47
 - sePresent, 48
 - teeOnly, 48
- TEE_SEServiceHandle
 - tee_api.types.h, 152
- TEE_SESessionHandle
 - tee_api.types.h, 152
- TEE_Session
 - tee_api.types.h, 152
- TEE_SetOperationKey
 - tee-internal-api-cryptlib.c, 235
 - tee-ta-internal.h, 95
- TEE_STORAGE_PRIVATE
 - tee_api.defines.h, 134
- TEE_STORAGE_PRIVATE_REE
 - tee_api.defines_extensions.h, 146
- TEE_STORAGE_PRIVATE_RPMB
 - tee_api.defines_extensions.h, 147
- TEE_STORAGE_PRIVATE_SQL_RESERVED
 - tee_api.defines_extensions.h, 147
- TEE_SUCCESS
 - tee_api.defines.h, 135
- tee.ta.api.h
 - TA_CloseSessionEntryPoint, 175
 - TA_CreateEntryPoint, 175
 - TA_DestroyEntryPoint, 175
 - TA_EXPORT, 175
 - TA_InvokeCommandEntryPoint, 175
 - TA_OpenSessionEntryPoint, 175
- TEE_TASessionHandle
 - tee_api.types.h, 152
- TEE_Time, 48
 - millis, 48
 - seconds, 48
- TEE_TIMEOUT_INFINITE
 - tee_api.defines.h, 135
- TEE_TYPE_AES
 - tee_api.defines.h, 135
- TEE_TYPE_CONCAT_KDF_Z
 - tee_api.defines_extensions.h, 147
- TEE_TYPE_CORRUPTED_OBJECT
 - tee_api.defines.h, 135
- TEE_TYPE_DATA
 - tee_api.defines.h, 135
- TEE_TYPE_DES
 - tee_api.defines.h, 135
- TEE_TYPE_DES3
 - tee_api.defines.h, 135
- TEE_TYPE_DH_KEYPAIR
 - tee_api.defines.h, 135
- TEE_TYPE_DSA_KEYPAIR
 - tee_api.defines.h, 135
- TEE_TYPE_DSA_PUBLIC_KEY
 - tee_api.defines.h, 135
- TEE_TYPE_ECDH_KEYPAIR
 - tee_api.defines.h, 135
- TEE_TYPE_ECDH_PUBLIC_KEY
 - tee_api.defines.h, 136
- TEE_TYPE_ECDSA_KEYPAIR
 - tee_api.defines.h, 136
- TEE_TYPE_ECDSA_PUBLIC_KEY
 - tee_api.defines.h, 136
- TEE_TYPE_GENERIC_SECRET
 - tee_api.defines.h, 136
- TEE_TYPE_HKDF_IKM
 - tee_api.defines_extensions.h, 147
- TEE_TYPE_HMAC_MD5
 - tee_api.defines.h, 136
- TEE_TYPE_HMAC_SHA1
 - tee_api.defines.h, 136
- TEE_TYPE_HMAC_SHA224
 - tee_api.defines.h, 136
- TEE_TYPE_HMAC_SHA256
 - tee_api.defines.h, 136
- TEE_TYPE_HMAC_SHA384
 - tee_api.defines.h, 136
- TEE_TYPE_HMAC_SHA512
 - tee_api.defines.h, 136
- TEE_TYPE_PBKDF2_PASSWORD
 - tee_api.defines_extensions.h, 147
- TEE_TYPE_RSA_KEYPAIR
 - tee_api.defines.h, 136
- TEE_TYPE_RSA_PUBLIC_KEY
 - tee_api.defines.h, 137
- tee.unmap
 - tee_internal.api.extensions.h, 172
- TEE_USAGE_DECRYPT
 - tee_api.defines.h, 137
- TEE_USAGE_DERIVE
 - tee_api.defines.h, 137
- TEE_USAGE_ENCRYPT
 - tee_api.defines.h, 137
- TEE_USAGE_EXTRACTABLE
 - tee_api.defines.h, 137
- TEE_USAGE_MAC
 - tee_api.defines.h, 137
- TEE_USAGE_SIGN
 - tee_api.defines.h, 137
- TEE_USAGE_VERIFY
 - tee_api.defines.h, 137
- tee_user_mem_check_heap
 - tee_internal.api.extensions.h, 173
- TEE_USER_MEM_HINT_NO_FILL_ZERO
 - tee_internal.api.extensions.h, 172
- tee_user_mem_mark_heap
 - tee_internal.api.extensions.h, 173
- TEE_UUID, 48
 - clockSeqAndNode, 49
 - timeHiAndVersion, 49
 - timeLow, 49
 - timeMid, 49
- tee_uuid_from_str
 - tee_internal.api.extensions.h, 173
- TEE_Whence

- tee_api_types.h, 153
- TEE.WriteObjectData
 - tee-internal-api.c, 200, 209
 - tee-ta-internal.h, 95
- TEEC.AllocateSharedMemory
 - tee_client_api.h, 164
 - teec_stub.c, 218
- TEEC.CloseSession
 - tee_client_api.h, 165
 - teec_stub.c, 218
- TEEC.CONFIG_PAYLOAD_REF_COUNT
 - tee_client_api.h, 158
- TEEC.CONFIG_SHAREDMEM_MAX_SIZE
 - tee_client_api.h, 159
- TEEC.Context, 49
 - fd, 49
 - reg_mem, 50
- TEEC.ERROR_ACCESS_CONFLICT
 - tee_client_api.h, 159
- TEEC.ERROR_ACCESS_DENIED
 - tee_client_api.h, 159
- TEEC.ERROR_BAD_FORMAT
 - tee_client_api.h, 159
- TEEC.ERROR_BAD_PARAMETERS
 - tee_client_api.h, 159
- TEEC.ERROR_BAD_STATE
 - tee_client_api.h, 159
- TEEC.ERROR_BUSY
 - tee_client_api.h, 159
- TEEC.ERROR_CANCEL
 - tee_client_api.h, 159
- TEEC.ERROR_COMMUNICATION
 - tee_client_api.h, 159
- TEEC.ERROR_EXCESS_DATA
 - tee_client_api.h, 159
- TEEC.ERROR_EXTERNAL_CANCEL
 - tee_client_api.h, 159
- TEEC.ERROR_GENERIC
 - tee_client_api.h, 160
- TEEC.ERROR_ITEM_NOT_FOUND
 - tee_client_api.h, 160
- TEEC.ERROR_NO_DATA
 - tee_client_api.h, 160
- TEEC.ERROR_NOT_IMPLEMENTED
 - tee_client_api.h, 160
- TEEC.ERROR_NOT_SUPPORTED
 - tee_client_api.h, 160
- TEEC.ERROR_OUT_OF_MEMORY
 - tee_client_api.h, 160
- TEEC.ERROR_SECURITY
 - tee_client_api.h, 160
- TEEC.ERROR_SHORT_BUFFER
 - tee_client_api.h, 160
- TEEC.ERROR_TARGET_DEAD
 - tee_client_api.h, 160
- TEEC.FinalizeContext
 - tee_client_api.h, 165
 - teec_stub.c, 218
- TEEC.InitializeContext
 - tee_client_api.h, 166
 - teec_stub.c, 219
- TEEC.InvokeCommand
 - tee_client_api.h, 166
- TEEC.LOGIN_APPLICATION
 - tee_client_api.h, 160
- TEEC.LOGIN_GROUP
 - tee_client_api.h, 160
- TEEC.LOGIN_GROUP_APPLICATION
 - tee_client_api.h, 161
- TEEC.LOGIN_PUBLIC
 - tee_client_api.h, 161
- TEEC.LOGIN_USER
 - tee_client_api.h, 161
- TEEC.LOGIN_USER_APPLICATION
 - tee_client_api.h, 161
- TEEC.MEM.INPUT
 - tee_client_api.h, 161
- TEEC.MEM.OUTPUT
 - tee_client_api.h, 161
- TEEC.MEMREF_PARTIAL_INOUT
 - tee_client_api.h, 161
- TEEC.MEMREF_PARTIAL_INPUT
 - tee_client_api.h, 161
- TEEC.MEMREF_PARTIAL_OUTPUT
 - tee_client_api.h, 161
- TEEC.MEMREF_TEMP_INOUT
 - tee_client_api.h, 162
- TEEC.MEMREF_TEMP_INPUT
 - tee_client_api.h, 162
- TEEC.MEMREF_TEMP_OUTPUT
 - tee_client_api.h, 162
- TEEC.MEMREF_WHOLE
 - tee_client_api.h, 162
- TEEC.NONE
 - tee_client_api.h, 162
- TEEC.OpenSession
 - tee_client_api.h, 167
 - teec_stub.c, 219
- TEEC.Operation, 50
 - params, 51
 - paramTypes, 51
 - session, 51
 - started, 51
- TEEC.ORIGIN_API
 - tee_client_api.h, 162
- TEEC.ORIGIN_COMMS
 - tee_client_api.h, 163
- TEEC.ORIGIN_TEE
 - tee_client_api.h, 163
- TEEC.ORIGIN_TRUSTED_APP
 - tee_client_api.h, 163
- TEEC.PARAM_TYPE_GET
 - tee_client_api.h, 163
- TEEC.PARAM_TYPES
 - tee_client_api.h, 163
- TEEC.Parameter, 51

- memref, 52
- tmpref, 52
- value, 52
- TEEC.RegisteredMemoryReference, 52
 - offset, 53
 - parent, 53
 - size, 53
- TEEC.RegisterSharedMemory
 - tee_client_api.h, 167
 - teec_stub.c, 220
- TEEC.ReleaseSharedMemory
 - tee_client_api.h, 168
 - teec_stub.c, 220
- TEEC.RequestCancellation
 - tee_client_api.h, 168
 - teec_stub.c, 221
- TEEC.Result
 - tee_client_api.h, 164
- TEEC.Session, 54
 - ctx, 54
 - session_id, 54
- TEEC.SharedMemory, 54
 - allocated_size, 55
 - buffer, 55
 - buffer_allocated, 55
 - flags, 55
 - id, 55
 - registered_fd, 56
 - shadow_buffer, 56
 - size, 56
- teec_stub.c
 - TEEC.AllocateSharedMemory, 218
 - TEEC.CloseSession, 218
 - TEEC.FinalizeContext, 218
 - TEEC.InitializeContext, 219
 - TEEC.OpenSession, 219
 - TEEC.RegisterSharedMemory, 220
 - TEEC.ReleaseSharedMemory, 220
 - TEEC.RequestCancellation, 221
- TEEC.SUCCESS
 - tee_client_api.h, 164
- TEEC.TempMemoryReference, 56
 - buffer, 56
 - size, 57
- TEEC.UUID, 57
 - clockSeqAndNode, 57
 - timeHiAndVersion, 57
 - timeLow, 57
 - timeMid, 57
- TEEC.Value, 58
 - a, 58
 - b, 58
- TEEC.VALUE_INOUT
 - tee_client_api.h, 164
- TEEC.VALUE_INPUT
 - tee_client_api.h, 164
- TEEC.VALUE_OUTPUT
 - tee_client_api.h, 164
- teeOnly
 - TEE_SEReaderProperties, 48
- test_dev_key.h
 - _sanctum_dev_public_key, 178
 - _sanctum_dev_public_key_len, 178
 - _sanctum_dev_secret_key, 179
 - _sanctum_dev_secret_key_len, 179
- timeHiAndVersion
 - TEE.UUID, 49
 - TEEC.UUID, 57
- timeLow
 - TEE.UUID, 49
 - TEEC.UUID, 57
- timeMid
 - TEE.UUID, 49
 - TEEC.UUID, 57
- tmpref
 - TEEC.Parameter, 52
- trace.c
 - trace_printf, 221
 - trace_vprintf, 222
- trace.h
 - dhex_dump, 183
 - DHEXDUMP, 181
 - DMSG, 181
 - DMSG_RAW, 181
 - DPRINT_STACK, 181
 - EMSG, 181
 - EMSG_RAW, 181
 - EPRINT_STACK, 181
 - FMSG, 181
 - FMSG_RAW, 182
 - FPRINT_STACK, 182
 - IMSG, 182
 - IMSG_RAW, 182
 - INMSG, 182
 - IPRINT_STACK, 182
 - MAX_FUNC_PRINT_SIZE, 182
 - MAX_PRINT_SIZE, 182
 - MSG, 182
 - MSG_RAW, 182
 - OUTMSG, 183
 - OUTRMSG, 183
 - SMSG, 183
 - trace_ext_get_thread_id, 184
 - trace_ext_prefix, 184
 - trace_ext_puts, 184
 - trace_get_level, 184
 - TRACE_LEVEL, 183
 - trace_level, 184
 - trace_printf, 184
 - trace_printf_helper, 183
 - trace_printf_helper_raw, 183
 - trace_set_level, 184
- TRACE_DEBUG
 - trace_levels.h, 188
- TRACE_ERROR
 - trace_levels.h, 188

- trace_ext_get_thread_id
 - trace.h, [184](#)
- trace_ext_prefix
 - trace.h, [184](#)
- trace_ext_puts
 - trace.h, [184](#)
- TRACE_FLOW
 - trace_levels.h, [188](#)
- trace_get_level
 - trace.h, [184](#)
- TRACE_INFO
 - trace_levels.h, [188](#)
- TRACE_LEVEL
 - trace.h, [183](#)
- trace_level
 - trace.h, [184](#)
- trace_levels.h
 - TRACE_DEBUG, [188](#)
 - TRACE_ERROR, [188](#)
 - TRACE_FLOW, [188](#)
 - TRACE_INFO, [188](#)
 - TRACE_MAX, [188](#)
 - TRACE_MIN, [188](#)
 - TRACE_PRINTF_LEVEL, [188](#)
- TRACE_MAX
 - trace_levels.h, [188](#)
- TRACE_MIN
 - trace_levels.h, [188](#)
- trace_printf
 - trace.c, [221](#)
 - trace.h, [184](#)
- trace_printf_helper
 - trace.h, [183](#)
- trace_printf_helper_raw
 - trace.h, [183](#)
- TRACE_PRINTF_LEVEL
 - trace_levels.h, [188](#)
- trace_set_level
 - trace.h, [184](#)
- trace_vprintf
 - trace.c, [222](#)
- type
 - __TEE_ObjectHandle, [33](#)
- uuid
 - TEE_Identity, [41](#)
- value
 - TEE_Attribute, [40](#)
 - TEE_Param, [46](#)
 - TEEC_Parameter, [52](#)
- WOLFCRYPT
 - tee_api_tee_types.h, [211](#), [215](#)
- wolfSSL_Free
 - tee-internal-api-cryptlib.c, [236](#)
- wolfSSL_Malloc
 - tee-internal-api-cryptlib.c, [236](#)