

Trusted Application Programming Reference on Portable TEE

National Institute of Advanced Industrial Science and Technology

2021-02-19

1 Preparation	1
1.1 Keystone(RISC-V Unleashed)	1
1.1.1 Required Packages	1
1.1.2 Build Keystone	1
1.1.3 Run Keystone examples	2
1.2 OPTEE (ARM64 RPI3)	2
1.2.1 Required Packages	2
1.2.2 Build OPTEE v3.9.0	2
1.2.3 Run Optee examples	3
1.3 SGX (Intel NUC)	4
1.3.1 List of machines which are confirmed to work	4
1.3.2 BIOS Versions which are failed or succeeded in IAS Test	4
1.3.3 BIOS Settings	5
1.3.4 Required Packages	5
1.3.5 Build SGX	5
1.3.6 Run sgx-ra-sample	6
2 Building	8
2.1 ta-ref with Keystone	8
2.1.1 Cloning source and building	8
2.1.2 Check ta-ref by running test_gp, test_hello, on qemu	9
2.2 ta-ref with OPTEE	10
2.2.1 Cloning source and building	11
2.2.2 check ta-ref by running test_gp, test_hello, on qemu	11
2.3 ta-ref with SGX	12
2.3.1 Cloning source and building	12
2.3.2 Check ta-ref by running test_gp, test_hello, simulation mode on any pc	12
3 Running on Dev Boards	14
3.1 Keystone, Unleashed	14
3.1.1 Preparation of rootfs on SD Card	14
3.1.2 Copying binaries of test_hello and test_gp	15
3.1.3 Check test_hello and test_gp on Unleashed	15
3.2 OPTEE, RPI3	17
3.2.1 Preparation of rootfs on SD Card	17
3.2.2 Copying binaries of test_hello and test_gp to rootfs partition	18
3.2.3 Check test_hello and test_gp	18
3.3 SGX, NUC	19
3.3.1 Copying binaries of test_hello and test_gp to NUC machine	19
3.3.2 Check test_hello and test_gp	19
4 Overview of ta-ref	20
4.1 Features	20

4.1.1 What we did on RISC-V	20
4.1.2 Separate GP TEE Internal API	21
4.2 Diagram	21
4.2.1 Dependency of category	21
5 How to program on ta-ref	21
5.1 Time Functions	21
5.2 Random Functions	22
5.3 Hash Functions	22
5.4 Symmetric crypto Functions	23
5.5 Asymmetric crypto Functions	23
5.6 Asymmetric crypto gcm Functions	25
5.7 open, read, write, close on secure storage	26
6 API compare with full-set GP API	27
6.1 GP API	27
7 Class Index	29
7.1 Class List	29
8 File Index	30
8.1 File List	30
9 Class Documentation	34
9.1 __profiler_data Struct Reference	34
9.1.1 Member Data Documentation	34
9.2 __profiler_header Struct Reference	35
9.2.1 Member Data Documentation	35
9.3 __TEE_ObjectHandle Struct Reference	35
9.3.1 Member Data Documentation	35
9.4 __TEE_OperationHandle Struct Reference	36
9.4.1 Member Data Documentation	36
9.5 _sgx_errlist_t Struct Reference	38
9.5.1 Member Data Documentation	38
9.6 addrinfo Struct Reference	38
9.6.1 Member Data Documentation	39
9.7 enclave_report Struct Reference	40
9.7.1 Member Data Documentation	40
9.8 invoke_command_param.t Struct Reference	40
9.8.1 Member Data Documentation	41
9.9 invoke_command.t Struct Reference	41
9.9.1 Member Data Documentation	42
9.10 list Struct Reference	42
9.10.1 Member Data Documentation	42

9.11 nm.info Struct Reference	43
9.11.1 Member Data Documentation	43
9.12 ob16_t Struct Reference	43
9.12.1 Member Data Documentation	44
9.13 ob196_t Struct Reference	44
9.13.1 Member Data Documentation	44
9.14 ob256_t Struct Reference	44
9.14.1 Member Data Documentation	45
9.15 out_fct_wrap_type Struct Reference	45
9.15.1 Member Data Documentation	45
9.16 param_buffer_t Struct Reference	45
9.16.1 Member Data Documentation	46
9.17 pollfd Struct Reference	46
9.17.1 Member Data Documentation	46
9.18 ree_time_t Struct Reference	47
9.18.1 Member Data Documentation	47
9.19 report Struct Reference	47
9.19.1 Member Data Documentation	48
9.20 result Struct Reference	48
9.20.1 Member Data Documentation	48
9.21 sm_report Struct Reference	49
9.21.1 Member Data Documentation	49
9.22 TEE_Attribute Struct Reference	50
9.22.1 Member Data Documentation	50
9.23 TEE_Identity Struct Reference	51
9.23.1 Member Data Documentation	52
9.24 TEE_ObjectInfo Struct Reference	52
9.24.1 Member Data Documentation	52
9.25 TEE_OperationInfo Struct Reference	53
9.25.1 Member Data Documentation	54
9.26 TEE_OperationInfoKey Struct Reference	55
9.26.1 Member Data Documentation	55
9.27 TEE_OperationInfoMultiple Struct Reference	55
9.27.1 Member Data Documentation	56
9.28 TEE_Param Union Reference	57
9.28.1 Member Data Documentation	57
9.29 TEE_SEAID Struct Reference	58
9.29.1 Member Data Documentation	58
9.30 TEE_SEReaderProperties Struct Reference	58
9.30.1 Member Data Documentation	58
9.31 TEE_Time Struct Reference	59
9.31.1 Member Data Documentation	59

9.32 TEE.UUID Struct Reference	59
9.32.1 Member Data Documentation	60
9.33 TEEC.Context Struct Reference	60
9.33.1 Detailed Description	60
9.33.2 Member Data Documentation	60
9.34 TEEC.Operation Struct Reference	61
9.34.1 Detailed Description	61
9.34.2 Member Data Documentation	62
9.35 TEEC.Parameter Union Reference	62
9.35.1 Detailed Description	62
9.35.2 Member Data Documentation	63
9.36 TEEC.RegisteredMemoryReference Struct Reference	63
9.36.1 Detailed Description	64
9.36.2 Member Data Documentation	64
9.37 TEEC.Session Struct Reference	65
9.37.1 Detailed Description	65
9.37.2 Member Data Documentation	65
9.38 TEEC.SharedMemory Struct Reference	65
9.38.1 Detailed Description	66
9.38.2 Member Data Documentation	66
9.39 TEEC.TempMemoryReference Struct Reference	67
9.39.1 Detailed Description	67
9.39.2 Member Data Documentation	67
9.40 TEEC.UUID Struct Reference	68
9.40.1 Detailed Description	68
9.40.2 Member Data Documentation	68
9.41 TEEC.Value Struct Reference	69
9.41.1 Detailed Description	69
9.41.2 Member Data Documentation	69
10 File Documentation	70
10.1 ta-ref/api/include/compiler.h File Reference	70
10.1.1 Macro Definition Documentation	71
10.2 ta-ref/api/include/report.h File Reference	77
10.2.1 Macro Definition Documentation	77
10.3 ta-ref/api/include/tee-common.h File Reference	78
10.3.1 Detailed Description	79
10.3.2 Macro Definition Documentation	79
10.4 ta-ref/api/include/tee-ta-internal.h File Reference	79
10.4.1 Detailed Description	81
10.4.2 Function Documentation	81
10.5 ta-ref/api/include/tee_api.h File Reference	103

10.5.1 Function Documentation	107
10.6 ta-ref/api/include/tee_api_defines.h File Reference	139
10.6.1 Macro Definition Documentation	145
10.7 ta-ref/api/include/tee_api_defines_extensions.h File Reference	177
10.7.1 Macro Definition Documentation	178
10.8 ta-ref/api/include/tee_api_types.h File Reference	181
10.8.1 Macro Definition Documentation	182
10.8.2 Typedef Documentation	183
10.8.3 Enumeration Type Documentation	185
10.9 ta-ref/api/include/tee_client_api.h File Reference	185
10.9.1 Macro Definition Documentation	188
10.9.2 Typedef Documentation	194
10.9.3 Function Documentation	194
10.10 ta-ref/api/include/tee_internal_api.h File Reference	197
10.11 ta-ref/api/include/tee_internal_api_extensions.h File Reference	198
10.11.1 Macro Definition Documentation	199
10.11.2 Function Documentation	199
10.12 ta-ref/api/include/tee_ta_api.h File Reference	201
10.12.1 Macro Definition Documentation	201
10.12.2 Function Documentation	202
10.13 ta-ref/api/include/test_dev_key.h File Reference	203
10.13.1 Variable Documentation	203
10.14 ta-ref/api/include/trace.h File Reference	204
10.14.1 Macro Definition Documentation	205
10.14.2 Function Documentation	208
10.14.3 Variable Documentation	209
10.15 ta-ref/api/include/trace_levels.h File Reference	209
10.15.1 Macro Definition Documentation	209
10.16 ta-ref/api/keystone/tee-internal-api-machine.c File Reference	210
10.16.1 Function Documentation	211
10.17 ta-ref/api/keystone/tee-internal-api.c File Reference	211
10.17.1 Macro Definition Documentation	212
10.17.2 Function Documentation	213
10.18 ta-ref/api/sgx/tee-internal-api.c File Reference	222
10.18.1 Macro Definition Documentation	223
10.18.2 Function Documentation	224
10.19 ta-ref/api/keystone/tee_api_tee_types.h File Reference	231
10.19.1 Macro Definition Documentation	232
10.20 ta-ref/api/optee/tee_api_tee_types.h File Reference	233
10.21 ta-ref/api/sgx/tee_api_tee_types.h File Reference	233
10.21.1 Macro Definition Documentation	234
10.22 ta-ref/api/keystone/teec_stub.c File Reference	235

10.22.1 Function Documentation	235
10.23 ta-ref/api/keystone/trace.c File Reference	238
10.23.1 Function Documentation	239
10.24 ta-ref/test_gp/keystone/Enclave/trace.c File Reference	240
10.24.1 Function Documentation	241
10.25 ta-ref/test_gp/optee/Enclave/trace.c File Reference	241
10.25.1 Function Documentation	242
10.26 ta-ref/test_gp/sgx/Enclave/trace.c File Reference	243
10.26.1 Function Documentation	243
10.27 ta-ref/api/keystone/vsnprintf.c File Reference	244
10.27.1 Macro Definition Documentation	246
10.27.2 Typedef Documentation	247
10.27.3 Function Documentation	247
10.28 ta-ref/test_gp/vsnprintf.c File Reference	250
10.28.1 Macro Definition Documentation	251
10.28.2 Typedef Documentation	253
10.28.3 Function Documentation	253
10.29 ta-ref/api/tee-internal-api-cryptlib.c File Reference	263
10.29.1 Macro Definition Documentation	265
10.29.2 Function Documentation	266
10.30 ta-ref/benchmark/bench.c File Reference	277
10.30.1 Function Documentation	278
10.30.2 Variable Documentation	280
10.31 ta-ref/benchmark/bench.h File Reference	281
10.31.1 Macro Definition Documentation	282
10.31.2 Function Documentation	282
10.32 ta-ref/benchmark/cpu_bench.c File Reference	284
10.32.1 Function Documentation	285
10.33 ta-ref/benchmark/include/config_bench.h File Reference	285
10.33.1 Macro Definition Documentation	286
10.33.2 Enumeration Type Documentation	286
10.33.3 Function Documentation	287
10.34 ta-ref/benchmark/io_bench.c File Reference	287
10.34.1 Macro Definition Documentation	288
10.34.2 Function Documentation	288
10.35 ta-ref/benchmark/keystone/tee_def.h File Reference	289
10.35.1 Function Documentation	290
10.35.2 Variable Documentation	290
10.36 ta-ref/benchmark/optee/tee_def.h File Reference	290
10.36.1 Macro Definition Documentation	291
10.36.2 Function Documentation	291
10.36.3 Variable Documentation	291

10.37 ta-ref/benchmark/sgx/tee_def.h File Reference	291
10.37.1 Function Documentation	292
10.37.2 Variable Documentation	292
10.38 ta-ref/benchmark/memory_bench.c File Reference	293
10.38.1 Macro Definition Documentation	293
10.38.2 Function Documentation	293
10.39 ta-ref/benchmark/time_test.c File Reference	294
10.39.1 Function Documentation	295
10.40 ta-ref/docs/building.md File Reference	295
10.41 ta-ref/docs/gp_api.md File Reference	295
10.42 ta-ref/docs/how_to_program_on_ta-ref.md File Reference	295
10.43 ta-ref/docs/overview_of_ta-ref.md File Reference	295
10.44 ta-ref/docs/preparation.md File Reference	295
10.45 ta-ref/docs/running_on_dev_boards.md File Reference	295
10.46 ta-ref/edger/edger8r/user_types.h File Reference	295
10.46.1 Macro Definition Documentation	296
10.46.2 Typedef Documentation	296
10.47 ta-ref/edger/keyedge/Enclave.t.c File Reference	296
10.48 ta-ref/edger/keyedge/Enclave.t.h File Reference	297
10.48.1 Function Documentation	297
10.49 ta-ref/edger/optee/Enclave.t.h File Reference	298
10.50 ta-ref/edger/keyedge/Enclave.u.c File Reference	298
10.51 ta-ref/edger/keyedge/Enclave.u.h File Reference	299
10.51.1 Macro Definition Documentation	299
10.51.2 Function Documentation	299
10.52 ta-ref/edger/keyedge/ocalls.h File Reference	300
10.52.1 Macro Definition Documentation	301
10.52.2 Typedef Documentation	302
10.52.3 Function Documentation	302
10.53 ta-ref/edger/optee/Enclave.h File Reference	305
10.53.1 Macro Definition Documentation	305
10.54 ta-ref/test_gp/sgx/Enclave/Enclave.h File Reference	306
10.54.1 Function Documentation	306
10.55 ta-ref/gp/asymmetric_key.c File Reference	308
10.55.1 Macro Definition Documentation	309
10.55.2 Function Documentation	309
10.56 ta-ref/gp/include/config_ref_ta.h File Reference	309
10.56.1 Macro Definition Documentation	310
10.56.2 Function Documentation	310
10.57 ta-ref/gp/include/gp_test.h File Reference	311
10.57.1 Function Documentation	312
10.58 ta-ref/gp/invoke_command.c File Reference	314

10.58.1 Macro Definition Documentation	314
10.59 ta-ref/gp/message_digest.c File Reference	315
10.59.1 Macro Definition Documentation	315
10.59.2 Function Documentation	316
10.60 ta-ref/gp/random.c File Reference	316
10.60.1 Function Documentation	317
10.61 ta-ref/gp/secure_storage.c File Reference	317
10.61.1 Macro Definition Documentation	317
10.61.2 Function Documentation	318
10.62 ta-ref/gp/symmetric_key.c File Reference	318
10.62.1 Macro Definition Documentation	319
10.62.2 Function Documentation	319
10.63 ta-ref/gp/symmetric_key_gcm.c File Reference	319
10.63.1 Macro Definition Documentation	320
10.63.2 Function Documentation	320
10.64 ta-ref/gp/time.c File Reference	320
10.64.1 Function Documentation	321
10.65 ta-ref/profiler/analyzer/analyzer.c File Reference	321
10.65.1 Macro Definition Documentation	322
10.65.2 Function Documentation	322
10.66 ta-ref/profiler/analyzer/analyzer.h File Reference	323
10.67 ta-ref/profiler/analyzer/nm_parse.c File Reference	324
10.67.1 Macro Definition Documentation	325
10.67.2 Function Documentation	325
10.67.3 Variable Documentation	327
10.68 ta-ref/profiler/analyzer/nm_parse.h File Reference	327
10.68.1 Macro Definition Documentation	328
10.68.2 Function Documentation	328
10.69 ta-ref/profiler/analyzer/stack.h File Reference	329
10.69.1 Macro Definition Documentation	330
10.69.2 Function Documentation	330
10.69.3 Variable Documentation	331
10.70 ta-ref/profiler/app/tools.c File Reference	331
10.70.1 Function Documentation	331
10.71 ta-ref/profiler/keystone/Enclave/tools.c File Reference	332
10.71.1 Function Documentation	332
10.72 ta-ref/profiler/optee/Enclave/tools.c File Reference	333
10.72.1 Function Documentation	333
10.73 ta-ref/profiler/sgx/Enclave/tools.c File Reference	334
10.73.1 Function Documentation	334
10.74 ta-ref/test_gp/tools.c File Reference	335
10.74.1 Function Documentation	335

10.75 ta-ref/profiler/keystone/tee_config.h File Reference	337
10.75.1 Function Documentation	337
10.75.2 Variable Documentation	337
10.76 ta-ref/profiler/optee/tee_config.h File Reference	338
10.76.1 Macro Definition Documentation	338
10.76.2 Function Documentation	338
10.76.3 Variable Documentation	339
10.77 ta-ref/profiler/sgx/tee_config.h File Reference	339
10.77.1 Function Documentation	339
10.77.2 Variable Documentation	340
10.78 ta-ref/profiler/keystone/tee_profiler.c File Reference	340
10.78.1 Function Documentation	340
10.78.2 Variable Documentation	342
10.79 ta-ref/profiler/optee/tee_profiler.c File Reference	342
10.79.1 Function Documentation	342
10.79.2 Variable Documentation	343
10.80 ta-ref/profiler/sgx/tee_profiler.c File Reference	343
10.80.1 Function Documentation	344
10.80.2 Variable Documentation	345
10.81 ta-ref/profiler/keystone/tee_profiler.h File Reference	346
10.81.1 Function Documentation	346
10.82 ta-ref/profiler/optee/tee_profiler.h File Reference	347
10.82.1 Function Documentation	347
10.83 ta-ref/profiler/sgx/tee_profiler.h File Reference	347
10.83.1 Function Documentation	348
10.84 ta-ref/profiler/profiler.c File Reference	348
10.84.1 Function Documentation	349
10.84.2 Variable Documentation	351
10.85 ta-ref/profiler/profiler.h File Reference	351
10.85.1 Function Documentation	351
10.86 ta-ref/profiler/profiler_attrs.h File Reference	352
10.86.1 Macro Definition Documentation	352
10.87 ta-ref/profiler/profiler_data.h File Reference	353
10.87.1 Macro Definition Documentation	354
10.87.2 Typedef Documentation	354
10.87.3 Enumeration Type Documentation	354
10.87.4 Function Documentation	355
10.87.5 Variable Documentation	355
10.88 ta-ref/test_gp/crt.c File Reference	355
10.88.1 Function Documentation	356
10.88.2 Variable Documentation	356
10.89 ta-ref/test_gp/optee/Enclave/crt.c File Reference	357

10.89.1 Macro Definition Documentation	358
10.89.2 Function Documentation	358
10.89.3 Variable Documentation	362
10.90 ta-ref/test_gp/include/crt.h File Reference	362
10.90.1 Function Documentation	363
10.91 ta-ref/test_gp/include/ocall_wrapper.h File Reference	364
10.91.1 Function Documentation	364
10.92 ta-ref/test_gp/include/random.h File Reference	365
10.93 ta-ref/test_gp/include/tools.h File Reference	366
10.93.1 Function Documentation	366
10.94 ta-ref/test_gp/keystone/Enclave/ocall_wrapper.c File Reference	367
10.94.1 Function Documentation	368
10.95 ta-ref/test_gp/sgx/Enclave/ocall_wrapper.c File Reference	368
10.95.1 Function Documentation	369
10.96 ta-ref/test_gp/keystone/Enclave/startup.c File Reference	369
10.96.1 Function Documentation	369
10.97 ta-ref/test_gp/sgx/Enclave/startup.c File Reference	370
10.97.1 Function Documentation	370
10.98 ta-ref/test_hello/keystone/App/App.cpp File Reference	371
10.98.1 Function Documentation	371
10.98.2 Variable Documentation	372
10.99 ta-ref/test_hello/sgx/App/App.cpp File Reference	372
10.99.1 Macro Definition Documentation	373
10.99.2 Function Documentation	373
10.100 ta-ref/test_gp/keystone/App/App.cpp File Reference	374
10.100.1 Function Documentation	374
10.100.2 Variable Documentation	375
10.101 ta-ref/test_gp/sgx/App/App.cpp File Reference	375
10.101.1 Macro Definition Documentation	376
10.101.2 Function Documentation	376
10.102 ta-ref/test_hello/keystone/App/App_ocalls.cpp File Reference	377
10.102.1 Function Documentation	378
10.103 ta-ref/test_hello/sgx/App/App_ocalls.cpp File Reference	381
10.103.1 Function Documentation	382
10.104 ta-ref/test_gp/keystone/App/App_ocalls.cpp File Reference	384
10.104.1 Macro Definition Documentation	385
10.104.2 Function Documentation	385
10.105 ta-ref/test_gp/sgx/App/App_ocalls.cpp File Reference	389
10.105.1 Macro Definition Documentation	389
10.105.2 Function Documentation	390
10.106 ta-ref/test_hello/keystone/Enclave/Enclave.c File Reference	392
10.106.1 Macro Definition Documentation	393

10.106.2 Function Documentation	393
10.107 ta-ref/test_hello/optee/Enclave/Enclave.c File Reference	393
10.107.1 Macro Definition Documentation	394
10.107.2 Function Documentation	394
10.107.3 Variable Documentation	397
10.108 ta-ref/test_hello/sgx/Enclave/Enclave.c File Reference	398
10.108.1 Macro Definition Documentation	398
10.108.2 Function Documentation	398
10.109 ta-ref/test_gp/keystone/Enclave/Enclave.c File Reference	399
10.109.1 Function Documentation	399
10.110 ta-ref/test_gp/optee/Enclave/Enclave.c File Reference	400
10.110.1 Function Documentation	400
10.111 ta-ref/test_gp/sgx/Enclave/Enclave.c File Reference	401
10.111.1 Function Documentation	401
10.112 ta-ref/test_hello/optee/App/main.c File Reference	401
10.112.1 Macro Definition Documentation	402
10.112.2 Function Documentation	402
10.112.3 Variable Documentation	403
10.113 ta-ref/test_gp/optee/App/main.c File Reference	403
10.113.1 Macro Definition Documentation	404
10.113.2 Function Documentation	404
10.114 ta-ref/test_hello/optee/Enclave/user_ta_header.c File Reference	405
10.114.1 Macro Definition Documentation	405
10.114.2 Function Documentation	406
10.114.3 Variable Documentation	407
10.115 ta-ref/test_gp/optee/Enclave/user_ta_header.c File Reference	408
10.115.1 Macro Definition Documentation	409
10.115.2 Function Documentation	409
10.115.3 Variable Documentation	410
10.116 ta-ref/test_hello/optee/Enclave/user_ta_header_defines.h File Reference	411
10.116.1 Macro Definition Documentation	412
10.117 ta-ref/test_gp/optee/Enclave/user_ta_header_defines.h File Reference	413
10.117.1 Macro Definition Documentation	414
10.118 ta-ref/test_hello/sgx/App/App.h File Reference	415
10.118.1 Macro Definition Documentation	415
10.118.2 Variable Documentation	416
10.119 ta-ref/test_gp/sgx/App/App.h File Reference	416
10.119.1 Macro Definition Documentation	417
10.119.2 Variable Documentation	417
10.120 ta-ref/test_hello/sgx/App/App_ocalls.h File Reference	417
10.120.1 Typedef Documentation	418
10.120.2 Function Documentation	418

10.121 ta-ref/test_gp/sgx/App/App_ocalls.h File Reference	424
10.121.1 Typedef Documentation	426
10.121.2 Function Documentation	426
10.122 ta-ref/test_hello/sgx/App/types.h File Reference	433
10.122.1 Typedef Documentation	434
10.122.2 Variable Documentation	434
10.123 ta-ref/test_gp/sgx/App/types.h File Reference	435
10.123.1 Typedef Documentation	436
10.123.2 Variable Documentation	436
Index	437

1 Preparation

1.1 Keystone(RISC-V Unleashed)

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

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

1.1.1 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 zlib1g-dev wget
bzip2 patch vim-common lbzip2 python pkg-config libglib2.0-dev libpixman-1-dev libssl-dev screen
device-tree-compiler expect makeelf unzip cpio rsync cmake
```

1.1.2 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 `$KEYSTONE_DIR/riscv/bin`
- Add to your PATH i.e, `export PATH=$PATH:$KEYSTONE_DIR/riscv/bin`

1.1.3 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
```

1.2 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 see here:

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

1.2.1 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
```

1.2.2 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
```

1.2.2.1 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}
```

1.2.2.2 Clone and Build optee 3.9.0 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/knknkn162/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
```

1.2.2.3 Clone and Build optee 3.9.0 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/knknkn162/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
```

1.2.3 Run Optee examples

1.2.3.1 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 keep.bootcon 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

```

1.2.3.2 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
```

1.3 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. See here:

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

1.3.1 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)

1.3.2 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->

1.3.3 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>

1.3.4 Required Packages

Install 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
```

1.3.5 Build SGX

There are 3 components which need to be build for SGX

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

1.3.5.1 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
```

1.3.5.2 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`

1.3.6 Run sgx-ra-sample

1.3.6.1 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
```

1.3.6.2 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 form 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
```

1.3.6.3 Run IAS Test

Run "run-server"

[illegible]

Open another terminal and run "run-client"

```
./run-client
---- Copy/Paste Msg0||Msg1 Below to SP -----
00000000a7fa6ed63bec97891885abc2e2e80bd4bb2bd5bb32a7e142337f486bb9f6e76a9db59aa9aac50cd24c3625451a79bce7c51e24447981444cf516f
-----
Waiting for msg2
---- Copy/Paste Msg3 Below to SP -----
787d992031b5ed7d57f149aec7f04912a7fa6ed63bec97891885abc2e2e80bd4bb2bd5bb32a7e142337f486bb9f6e76a9db59aa9aac50cd24c3625451a79bce7c51e24447981444cf516f
-----
---- Enclave Trust Status from Service Provider -----
Enclave TRUSTED
```

1.3.6.4 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.

1.3.6.5 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->

1.3.6.6 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...
```

2 Building

2.1 ta-ref with Keystone

Make sure Keystone and other dependant sources have been built

2.1.1 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=<keystone directory path>
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
```

2.1.2 Check ta-ref by running test_gp, test.hello, on qemu

Copy the test.hello and test_gp programs to QEMU.

2.1.2.1 Launch QEMU console

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

2.1.2.2 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!
```

2.1.2.3 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:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
hash: 9b04c091da96b997afb8f2585d608aebe9c4a904f7d52c8f28c7e4d2dd9fba5f
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:
e94431cd22a6029185d0dbb1a17b5d62843bfeef25591583d2d668ec6fed1c692f88ce4754d690c346c8d9f2726630e0386abf4e45699a2ca2b34b
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
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:
c23e9ce04589e80a66debe23a788ae5393bdcd8e875e87e1bcf2b2d998f6418ccc6ee4ab112fdbfc5175868691efb40781a318ff439d30b49cc9f7
@tag: a551f999317b3fbd1eea7b622ce2caee
TEE.AllocateOperation(): start
TEE.AEInit(): start
TEE.AEDecryptFinal(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aebe9c4a904f7d52c8f28c7e4d2dd9fba5f
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:
d6e6b6e54db8b6a62fc1927886938bead27f4813f19ce77182e3016b5426bcad067ca98cd75f9dfddafe9eb0655c48df992d3ad674db69d831f26a
TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation:
TEE.FreeTransientObject(): start
verify ok
main end

```

2.2 ta-ref with OPTEE

Make sure optee_3.9.0.rpi3 has been built already.

2.2.1 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
```

2.2.2 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/
```

2.2.2.1 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

2.2.2.2 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: fe0c7d3eefb9bd5e63b8a0cce29af7eb
@GP REE time 1612156259 sec 390 millis
@GP System time 249187 sec 954 millis
256 bytes read:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
verify ok
hash: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@cipher:
30a558176172c53be4a2ac320776de105da79c29726879fe67d06b629f065731285f8a90f8a521ce34ecee51e15e928d157ea10d149bb687dd78b
decrypted to:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
verify ok
```

```
@cipher:
    ff409d8fe203bf0d81de36832b86c702f07edd343f408d3a2fb5ab347b4f72b10031efff0c17b7e0bc56c3f2f95f53c0d731ed87eb3e1187b6714a
@tag: 9b357baf76d2632fa7d16231640d6324
decrypted to:
    000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
verify ok
@digest: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@signature:
    719fa9898f3423b754675b835268f9b2368b77a429eeabf7369d60d135dee08158c3902fd2ed3c1bf17cb34e76f2ba25da915fa3970c757962f753
@@TEEFreeOperation:
verify ok
ecall.ta.main() end
--- enclave log end---
res = TEEC_SUCCESS; TEEC_InvokeCommand succeeded!
```

If executed successfully, you see above messages

2.3 ta-ref with SGX

Build ta-ref for Intel SGX platforms

2.3.1 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
```

2.3.2 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

2.3.2.1 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.
```


2.3.2.2 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: f35c1dle4bbf6641c5511c9dc5aaf638
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:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
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:
7427bff21e729a824a239e25332ebd455d18fa6aec1ec6618b77c252f768e0a9345608b0135727568867ce5b0fac872f6647787861b88220840281
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
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:
e33f34122c80b9a10002725e4e21542256da7c7cd3f6dd1b62b71cf8308f9e4a0daa50b29880a8f76707c4ed432549c4da9e68e7930189d2127fdd
@tag: 4c920ce2aef079e468ab24e25730d9d2
TEE.AllocateOperation(): start
TEE.AEInit(): start
TEE.AEDecryptFinal(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
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:
100b392ce043e9b8dc703088f505dd3083ec47bfcb8d59d968a66b54e80464d684d56dc9c44336f08fd9630979863a2d8fb7cd672a819ef609357e

```

```

TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation:
TEE.FreeTransientObject(): start
verify ok
main end
Info: Enclave successfully returned.

```

3 Running on Dev Boards

3.1 Keystone, Unleashed

Make sure Keystone and other dependant sources have been built

3.1.1 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

```

3.1.1.1 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.

```

3.1.1.2 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

3.1.2 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.

3.1.3 Check test_hello and test_gp on Unleashed

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

```

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(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:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
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:
e94431cd22a6029185d0dbb1a17b5d62843bfeef25591583d2d668ec6fed1c692f88ce4754d690c346c8d9f2726630e0386abf4e45699a2ca2b34b
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
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:
c23e9ce04589e80a66debe23a788ae5393bdcd8e875e87e1bcf2b2d998f6418ccc6ee4ab112fdbfc5175868691efb40781a318ff439d30b49cc9f7
@tag: a551f999317b3fbd1eea7b622ce2caee
TEE.AllocateOperation(): start
TEE.AEInit(): start
TEE.AEDecryptFinal(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
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(0x000000003FFFE28, 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:
    d6e6b6e54db8b6a62fc1927886938bead27f4813f19ce77182e3016b5426bcad067ca98cd75f9dfddafe9eb0655c48df992d3ad674db69d831f26a
TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation:
TEE.FreeTransientObject(): start
verify ok
main end

```

Test is successful.

3.2 OPTEE, RPI3

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

3.2.1 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.

3.2.2 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/
```

3.2.3 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:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
verify ok
hash: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@cipher:
30a558176172c53be4a2ac320776de105da79c29726879fe67d06b629f065731285f8a90f8a521ce34ecee51e15e928d157ea10d149bb687dd78b
decrypted to:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
verify ok
@cipher:
ff409d8fe203bf0d81de36832b86c702f07edd343f408d3a2fb5ab347b4f72b10031efff0c17b7e0bc56c3f2f95f53c0d731ed87eb3e1187b6714a
@tag: 9b357baf76d2632fa7d16231640d6324
decrypted to:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
verify ok
@digest: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@signature:
719fa9898f3423b754675b835268f9b2368b77a429eeabf7369d60d135dee08158c3902fd2ed3c1bf17cb34e76f2ba25da915fa3970c757962f753
@@TEEFreeOperation:
verify ok
ecall.ta_main() end
--- enclave log end---
res = TEEC.SUCCESS; TEEC.InvokeCommand succeeded!
```

If executed successfully, you see above messages

3.3 SGX, NUC

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

3.3.1 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.

3.3.2 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:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
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:
7427bff21e729a824a239e25332ebd455d18fa6aec1ec6618b77c252f768e0a9345608b0135727568867ce5b0fac872f6647787861b88220840281
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
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:
    e33f34122c80b9a10002725e4e21542256da7c7cd3f6dd1b62b71cf8308f9e4a0daa50b29880a8f76707c4ed432549c4da9e68e7930189d2127fdd
@tag: 4c920ce2aef079e468ab24e25730d9d2
TEE.AllocateOperation(): start
TEE.AEInit(): start
TEE.AEDecryptFinal(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to:
    000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
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:
    100b392ce043e9b8dc703088f505dd3083ec47bfc8d59d968a66b54e80464d684d56dc9c44336f08fd9630979863a2d8fb7cd672a819ef609357e
TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation:
TEE.FreeTransientObject(): start
verify ok
main end
Info: Enclave successfully returned.

```

4 Overview of ta-ref

4.1 Features

4.1.1 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.

4.1.2 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

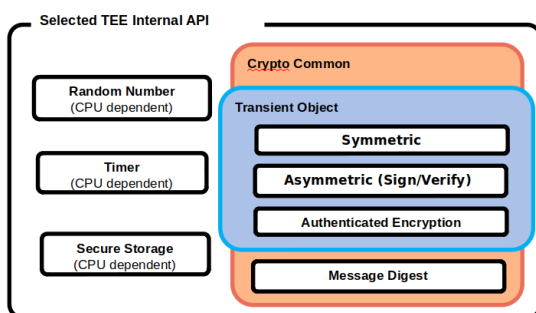
Category	CPU (In)Dependent	Functions
Random Number	Dependent	TEE_GenerateRandom
Time	Dependent	TEE_GetREETime, TEE_GetSystemTime
Secure Storage	Dependent	TEE_CreatePersistentObject, TEE_OpenPersistentObject, TEE_ReadObjectData, TEE_WriteObjectData, TEE_CloseObject
Transient Object	Dependent Independent	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

4.2 Diagram

4.2.1 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

5 How to program on ta-ref

5.1 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);
}
--- -- --

```

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 ---

```

5.2 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");
}
--- -----

```

5.3 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 ---

```

5.4 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 ---

```

5.5 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 ---

```

5.6 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 ---

```

5.7 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 data 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 data 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 ---

```

6 API compare with full-set 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:

__profiler_data	34
__profiler_header	35
__TEE_ObjectHandle	35
__TEE_OperationHandle	36
_sgx_errlist_t	38
addrinfo	38
enclave_report	40
invoke_command_param_t	40
invoke_command_t	41
list	42
nm_info	43
ob16_t	43
ob196_t	44
ob256_t	44
out_fct_wrap_type	45
param_buffer_t	45
pollfd	46
ree_time_t	47
report	47
result	48
sm_report	49
TEE_Attribute	50
TEE_Identity	51
TEE_ObjectInfo	52
TEE_OperationInfo	53
TEE_OperationInfoKey	55
TEE_OperationInfoMultiple	55
TEE_Param	57

TEE_SEAID	58
TEE_SEReaderProperties	58
TEE_Time	59
TEE_UUID	59
TEEC_Context	60
TEEC_Operation	61
TEEC_Parameter	62
TEEC_RegisteredMemoryReference	63
TEEC_Session	65
TEEC_SharedMemory	65
TEEC_TempMemoryReference	67
TEEC_UUID	68
TEEC_Value	69

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	263
ta-ref/api/include/compiler.h	70
ta-ref/api/include/report.h	77
ta-ref/api/include/tee-common.h Common type and definitions of RISC-V TEE	78
ta-ref/api/include/tee-ta-internal.h Candidate API list for Global Platform like RISC-V TEE	79
ta-ref/api/include/tee_api.h	103
ta-ref/api/include/tee_api_defines.h	139
ta-ref/api/include/tee_api_defines_extensions.h	177
ta-ref/api/include/tee_api_types.h	181
ta-ref/api/include/tee_client_api.h	185
ta-ref/api/include/tee_internal_api.h	197
ta-ref/api/include/tee_internal_api_extensions.h	198
ta-ref/api/include/tee_ta_api.h	201

ta-ref/api/include/test_dev_key.h	203
ta-ref/api/include/trace.h	204
ta-ref/api/include/trace_levels.h	209
ta-ref/api/keystone/tee-internal-api-machine.c	210
ta-ref/api/keystone/tee-internal-api.c	211
ta-ref/api/keystone/tee_api_tee_types.h	231
ta-ref/api/keystone/teec_stub.c	235
ta-ref/api/keystone/trace.c	238
ta-ref/api/keystone/vsnprintf.c	244
ta-ref/api/optee/tee_api_tee_types.h	233
ta-ref/api/sgx/tee-internal-api.c	222
ta-ref/api/sgx/tee_api_tee_types.h	233
ta-ref/benchmark/bench.c	277
ta-ref/benchmark/bench.h	281
ta-ref/benchmark/cpu_bench.c	284
ta-ref/benchmark/io_bench.c	287
ta-ref/benchmark/memory_bench.c	293
ta-ref/benchmark/time_test.c	294
ta-ref/benchmark/include/config_bench.h	285
ta-ref/benchmark/keystone/tee_def.h	289
ta-ref/benchmark/optee/tee_def.h	290
ta-ref/benchmark/sgx/tee_def.h	291
ta-ref/edger/edger8r/user_types.h	295
ta-ref/edger/keyedge/Enclave.t.c	296
ta-ref/edger/keyedge/Enclave.t.h	297
ta-ref/edger/keyedge/Enclave.u.c	298
ta-ref/edger/keyedge/Enclave.u.h	299
ta-ref/edger/keyedge/ocalls.h	300
ta-ref/edger/optee/Enclave.h	305
ta-ref/edger/optee/Enclave.t.h	298
ta-ref/gp/asymmetric_key.c	308
ta-ref/gp/invoke_command.c	314

ta-ref/gp/message_digest.c	315
ta-ref/gp/random.c	316
ta-ref/gp/secure_stoage.c	317
ta-ref/gp/symmetric_key.c	318
ta-ref/gp/symmetric_key_gcm.c	319
ta-ref/gp/time.c	320
ta-ref/gp/include/config_ref_ta.h	309
ta-ref/gp/include/gp_test.h	311
ta-ref/profiler/profiler.c	348
ta-ref/profiler/profiler.h	351
ta-ref/profiler/profiler_attrs.h	352
ta-ref/profiler/profiler_data.h	353
ta-ref/profiler/analyzer/analyzer.c	321
ta-ref/profiler/analyzer/analyzer.h	323
ta-ref/profiler/analyzer/nm_parse.c	324
ta-ref/profiler/analyzer/nm_parse.h	327
ta-ref/profiler/analyzer/stack.h	329
ta-ref/profiler/app/tools.c	331
ta-ref/profiler/keystone/tee_config.h	337
ta-ref/profiler/keystone/tee_profiler.c	340
ta-ref/profiler/keystone/tee_profiler.h	346
ta-ref/profiler/keystone/Enclave/tools.c	332
ta-ref/profiler/optee/tee_config.h	338
ta-ref/profiler/optee/tee_profiler.c	342
ta-ref/profiler/optee/tee_profiler.h	347
ta-ref/profiler/optee/Enclave/tools.c	333
ta-ref/profiler/sgx/tee_config.h	339
ta-ref/profiler/sgx/tee_profiler.c	343
ta-ref/profiler/sgx/tee_profiler.h	347
ta-ref/profiler/sgx/Enclave/tools.c	334
ta-ref/test_gp/crt.c	355
ta-ref/test_gp/tools.c	335

ta-ref/test_gp/vsnprintf.c	250
ta-ref/test_gp/include/crt.h	362
ta-ref/test_gp/include/ocall_wrapper.h	364
ta-ref/test_gp/include/random.h	365
ta-ref/test_gp/include/tools.h	366
ta-ref/test_gp/keystone/App/App.cpp	374
ta-ref/test_gp/keystone/App/App_ocalls.cpp	384
ta-ref/test_gp/keystone/Enclave/Enclave.c	399
ta-ref/test_gp/keystone/Enclave/ocall_wrapper.c	367
ta-ref/test_gp/keystone/Enclave/startup.c	369
ta-ref/test_gp/keystone/Enclave/trace.c	240
ta-ref/test_gp/optee/App/main.c	403
ta-ref/test_gp/optee/Enclave/crt.c	357
ta-ref/test_gp/optee/Enclave/Enclave.c	400
ta-ref/test_gp/optee/Enclave/trace.c	241
ta-ref/test_gp/optee/Enclave/user_ta_header.c	408
ta-ref/test_gp/optee/Enclave/user_ta_header_defines.h	413
ta-ref/test_gp/sgx/App/App.cpp	375
ta-ref/test_gp/sgx/App/App.h	416
ta-ref/test_gp/sgx/App/App_ocalls.cpp	389
ta-ref/test_gp/sgx/App/App_ocalls.h	424
ta-ref/test_gp/sgx/App/types.h	435
ta-ref/test_gp/sgx/Enclave/Enclave.c	401
ta-ref/test_gp/sgx/Enclave/Enclave.h	306
ta-ref/test_gp/sgx/Enclave/ocall_wrapper.c	368
ta-ref/test_gp/sgx/Enclave/startup.c	370
ta-ref/test_gp/sgx/Enclave/trace.c	243
ta-ref/test_hello/keystone/App/App.cpp	371
ta-ref/test_hello/keystone/App/App_ocalls.cpp	377
ta-ref/test_hello/keystone/Enclave/Enclave.c	392
ta-ref/test_hello/optee/App/main.c	401
ta-ref/test_hello/optee/Enclave/Enclave.c	393

ta-ref/test_hello/optee/Enclave/user_ta_header.c	405
ta-ref/test_hello/optee/Enclave/user_ta_header_defines.h	411
ta-ref/test_hello/sgx/App/App.cpp	372
ta-ref/test_hello/sgx/App/App.h	415
ta-ref/test_hello/sgx/App/App_ocalls.cpp	381
ta-ref/test_hello/sgx/App/App_ocalls.h	417
ta-ref/test_hello/sgx/App/types.h	433
ta-ref/test_hello/sgx/Enclave/Enclave.c	398

9 Class Documentation

9.1 __profiler_data Struct Reference

```
#include <profiler_data.h>
```

Public Attributes

- [uint8_t direction](#)
- [uint8_t hartid](#)
- [__profiler_nsec_t nsec](#)
- [uintptr_t callee](#)

9.1.1 Member Data Documentation

9.1.1.1 callee `uintptr_t __profiler_data::callee`

9.1.1.2 direction `uint8_t __profiler_data::direction`

9.1.1.3 hartid `uint8_t __profiler_data::hartid`

9.1.1.4 nsec `__profiler_nsec_t __profiler_data::nsec`

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

- [ta-ref/profiler/profiler_data.h](#)

9.2 __profiler_header Struct Reference

```
#include <profiler_data.h>
```

Public Attributes

- uint64_t [size](#)
- uint64_t [idx](#)
- uintptr_t [start](#)

9.2.1 Member Data Documentation

9.2.1.1 idx uint64_t __profiler_header::idx

9.2.1.2 size uint64_t __profiler_header::size

9.2.1.3 start uintptr_t __profiler_header::start

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

- [ta-ref/profiler/profiler_data.h](#)

9.3 __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]

9.3.1 Member Data Documentation

9.3.1.1 desc `int __TEE_ObjectHandle::desc`

9.3.1.2 flags `int __TEE_ObjectHandle::flags`

9.3.1.3 persist_ctx `struct AES_ctx __TEE_ObjectHandle::persist_ctx`

9.3.1.4 private_key `unsigned char __TEE_ObjectHandle::private_key`

9.3.1.5 public_key `unsigned char __TEE_ObjectHandle::public_key`

9.3.1.6 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](#)

9.4 __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]

9.4.1 Member Data Documentation

9.4.1.1 aectx struct AES_ctx __TEE_OperationHandle::aectx

9.4.1.2 aegcm.state int __TEE_OperationHandle::aegcm.state

9.4.1.3 aeiv unsigned char __TEE_OperationHandle::aeiv

9.4.1.4 aekey unsigned char __TEE_OperationHandle::aekey

9.4.1.5 alg int __TEE_OperationHandle::alg

9.4.1.6 ctx sha3_ctx_t __TEE_OperationHandle::ctx

9.4.1.7 flags int __TEE_OperationHandle::flags

9.4.1.8 mode int __TEE_OperationHandle::mode

9.4.1.9 prikey unsigned char __TEE_OperationHandle::prikey

9.4.1.10 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](#)

9.5 `_sgx_errlist_t` Struct Reference

```
#include <types.h>
```

Public Attributes

- `sgx_status_t` [err](#)
- `const char *` [msg](#)
- `const char *` [sug](#)

9.5.1 Member Data Documentation

9.5.1.1 `err` `sgx_status_t _sgx_errlist_t::err`

9.5.1.2 `msg` `const char * _sgx_errlist_t::msg`

9.5.1.3 `sug` `const char * _sgx_errlist_t::sug`

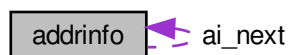
The documentation for this struct was generated from the following files:

- [ta-ref/test_hello/sgx/App/types.h](#)
- [ta-ref/test_gp/sgx/App/types.h](#)

9.6 `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](#)

9.6.1 Member Data Documentation

9.6.1.1 ai_addr struct sockaddr* addrinfo::ai_addr

9.6.1.2 ai_addrlen [socklen_t](#) addrinfo::ai_addrlen

9.6.1.3 ai_canonname char* addrinfo::ai_canonname

9.6.1.4 ai_family int addrinfo::ai_family

9.6.1.5 ai_flags int addrinfo::ai_flags

9.6.1.6 ai_next struct [addrinfo](#)* addrinfo::ai_next

9.6.1.7 ai_protocol int addrinfo::ai_protocol

9.6.1.8 **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](#)

9.7 **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](#)]

9.7.1 Member Data Documentation

9.7.1.1 data `uint8_t enclave_report::data` [[ATTEST_DATA_MAXLEN](#)]

9.7.1.2 data_len `uint64_t enclave_report::data_len`

9.7.1.3 hash `uint8_t enclave_report::hash` [[MDSIZE](#)]

9.7.1.4 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](#)

9.8 **invoke_command_param_t** Struct Reference

```
#include <ocalls.h>
```

Public Attributes

- unsigned int [a](#)
- unsigned int [b](#)
- unsigned int [size](#)

9.8.1 Member Data Documentation

9.8.1.1 a unsigned int invoke_command_param_t::a

9.8.1.2 b unsigned int invoke_command_param_t::b

9.8.1.3 size unsigned int invoke_command_param_t::size

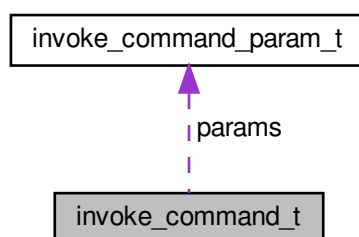
The documentation for this struct was generated from the following file:

- [ta-ref/edger/keyedge/ocalls.h](#)

9.9 invoke_command_t Struct Reference

```
#include <ocalls.h>
```

Collaboration diagram for invoke_command_t:



Public Attributes

- unsigned int [commandID](#)
- unsigned int [paramTypes](#)
- [invoke_command_param_t params](#) [4]

9.9.1 Member Data Documentation

9.9.1.1 commandID `unsigned int invoke_command_t::commandID`

9.9.1.2 params `invoke_command_param_t invoke_command_t::params[4]`

9.9.1.3 paramTypes `unsigned int invoke_command_t::paramTypes`

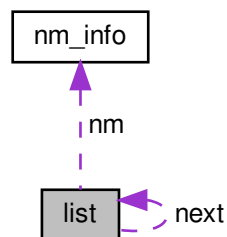
The documentation for this struct was generated from the following file:

- [ta-ref/edger/keyedge/ocalls.h](#)

9.10 list Struct Reference

```
#include <nm_parse.h>
```

Collaboration diagram for list:



Public Attributes

- `struct list * next`
- `uintptr_t addr`
- `struct nm_info * nm`

9.10.1 Member Data Documentation

9.10.1.1 addr `uintptr_t list::addr`

9.10.1.2 next `struct list* list::next`

9.10.1.3 nm `struct nm_info* list::nm`

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

- [ta-ref/profiler/analyzer/nm_parse.h](#)

9.11 nm_info Struct Reference

```
#include <nm_parse.h>
```

Public Attributes

- char [type](#)
- char [func_name](#) [256]

9.11.1 Member Data Documentation

9.11.1.1 func_name `char nm_info::func_name[256]`

9.11.1.2 type `char nm_info::type`

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

- [ta-ref/profiler/analyzer/nm_parse.h](#)

9.12 ob16_t Struct Reference

```
#include <ocalls.h>
```

Public Attributes

- int [ret](#)
- unsigned char [b](#) [16]

9.12.1 Member Data Documentation

9.12.1.1 **b** `unsigned char ob16_t::b[16]`

9.12.1.2 **ret** `int ob16_t::ret`

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

- [ta-ref/edger/keyedge/ocalls.h](#)

9.13 ob196_t Struct Reference

```
#include <ocalls.h>
```

Public Attributes

- `int` [ret](#)
- `unsigned char` [b](#) [196]

9.13.1 Member Data Documentation

9.13.1.1 **b** `unsigned char ob196_t::b[196]`

9.13.1.2 **ret** `int ob196_t::ret`

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

- [ta-ref/edger/keyedge/ocalls.h](#)

9.14 ob256_t Struct Reference

```
#include <ocalls.h>
```


Public Attributes

- int [ret](#)
- unsigned char [b](#) [256]

9.14.1 Member Data Documentation

9.14.1.1 [b](#) unsigned char ob256_t::b[256]

9.14.1.2 [ret](#) int ob256_t::ret

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

- [ta-ref/edger/keyedge/ocalls.h](#)

9.15 out_fct_wrap_type Struct Reference

Public Attributes

- void(* [fct](#))(char character, void *[arg](#))
- void * [arg](#)

9.15.1 Member Data Documentation

9.15.1.1 [arg](#) void * out_fct_wrap_type::arg

9.15.1.2 [fct](#) void(* out_fct_wrap_type::fct)(char character, void *[arg](#))

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

- [ta-ref/api/keystone/vsnprintf.c](#)
- [ta-ref/test_gp/vsnprintf.c](#)

9.16 param_buffer_t Struct Reference

```
#include <ocalls.h>
```

Public Attributes

- `size_t` [size](#)
- `char` [buf](#) [256]

9.16.1 Member Data Documentation

9.16.1.1 `buf` `char param_buffer_t::buf[256]`

9.16.1.2 `size` `size_t param_buffer_t::size`

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

- [ta-ref/edger/keyedge/ocalls.h](#)

9.17 pollfd Struct Reference

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

Public Attributes

- `int` [fd](#)
- `short int` [events](#)
- `short int` [revents](#)

9.17.1 Member Data Documentation

9.17.1.1 `events` `short int pollfd::events`

9.17.1.2 `fd` `int pollfd::fd`

9.17.1.3 `revents` `short int pollfd::revents`

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

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

9.18 ree_time_t Struct Reference

```
#include <ocalls.h>
```

Public Attributes

- unsigned int [seconds](#)
- unsigned int [millis](#)

9.18.1 Member Data Documentation

9.18.1.1 millis unsigned int ree_time_t::millis

9.18.1.2 seconds unsigned int ree_time_t::seconds

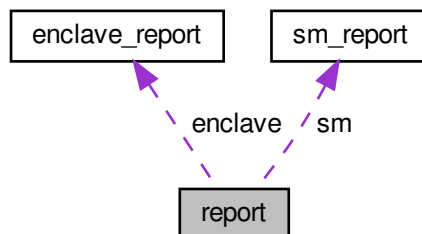
The documentation for this struct was generated from the following files:

- ta-ref/edger/keyedge/[ocalls.h](#)
- ta-ref/test_hello/sgx/App/[App_ocalls.h](#)
- ta-ref/test_gp/sgx/App/[App_ocalls.h](#)

9.19 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](#)]

9.19.1 Member Data Documentation

9.19.1.1 dev_public_key uint8_t report::dev_public_key [[PUBLIC_KEY_SIZE](#)]

9.19.1.2 enclave struct [enclave_report](#) report::enclave

9.19.1.3 sm struct [sm_report](#) report::sm

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

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

9.20 result Struct Reference

```
#include <analyzer.h>
```

Public Attributes

- size_t [idx](#)
- uintptr_t [callee](#)
- uint8_t [start_hartid](#)
- uint8_t [end_hartid](#)
- __profiler_nsec_t [start](#)
- __profiler_nsec_t [end](#)
- size_t [depth](#)

9.20.1 Member Data Documentation

9.20.1.1 callee uintptr_t result::callee

9.20.1.2 depth `size_t result::depth`

9.20.1.3 end `_profiler_nsec_t result::end`

9.20.1.4 end_hartid `uint8_t result::end_hartid`

9.20.1.5 idx `size_t result::idx`

9.20.1.6 start `_profiler_nsec_t result::start`

9.20.1.7 start_hartid `uint8_t result::start_hartid`

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

- ta-ref/profiler/analyzer/[analyzer.h](#)

9.21 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](#)]

9.21.1 Member Data Documentation

9.21.1.1 hash `uint8_t sm_report::hash` [[MDSIZE](#)]

9.21.1.2 public_key uint8_t sm_report::public_key[PUBLIC_KEY_SIZE]

9.21.1.3 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

9.22 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

9.22.1 Member Data Documentation

9.22.1.1 a uint32_t TEE_Attribute::a

9.22.1.2 attributeID uint32_t TEE_Attribute::attributeID

9.22.1.3 b uint32_t TEE_Attribute::b

9.22.1.4 buffer `void* TEEAttribute::buffer`

9.22.1.5 `union { ... } TEEAttribute::content`

9.22.1.6 length `uint32_t TEEAttribute::length`

9.22.1.7 `struct { ... } TEEAttribute::ref`

9.22.1.8 `struct { ... } TEEAttribute::value`

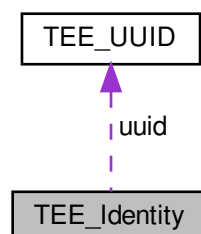
The documentation for this struct was generated from the following file:

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

9.23 TEE_Identity Struct Reference

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

Collaboration diagram for TEE_Identity:



Public Attributes

- `uint32_t login`
- `TEE_UUID uuid`

9.23.1 Member Data Documentation

9.23.1.1 login `uint32_t TEE_Identity::login`

9.23.1.2 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](#)

9.24 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`

9.24.1 Member Data Documentation

9.24.1.1 `__extension__ { ... }`

9.24.1.2 `__extension__ { ... }`

9.24.1.3 dataPosition uint32_t TEE_ObjectInfo::dataPosition

9.24.1.4 dataSize uint32_t TEE_ObjectInfo::dataSize

9.24.1.5 handleFlags uint32_t TEE_ObjectInfo::handleFlags

9.24.1.6 keySize uint32_t TEE_ObjectInfo::keySize

9.24.1.7 maxKeySize uint32_t TEE_ObjectInfo::maxKeySize

9.24.1.8 maxObjectSize uint32_t TEE_ObjectInfo::maxObjectSize

9.24.1.9 objectSize uint32_t TEE_ObjectInfo::objectSize

9.24.1.10 objectType uint32_t TEE_ObjectInfo::objectType

9.24.1.11 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](#)

9.25 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](#)

9.25.1 Member Data Documentation

9.25.1.1 algorithm uint32_t TEE_OperationInfo::algorithm

9.25.1.2 digestLength uint32_t TEE_OperationInfo::digestLength

9.25.1.3 handleState uint32_t TEE_OperationInfo::handleState

9.25.1.4 keySize uint32_t TEE_OperationInfo::keySize

9.25.1.5 maxKeySize uint32_t TEE_OperationInfo::maxKeySize

9.25.1.6 mode uint32_t TEE_OperationInfo::mode

9.25.1.7 operationClass uint32_t TEE_OperationInfo::operationClass

9.25.1.8 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](#)

9.26 TEE_OperationInfoKey Struct Reference

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

Public Attributes

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

9.26.1 Member Data Documentation

9.26.1.1 keySize `uint32_t TEE_OperationInfoKey::keySize`

9.26.1.2 requiredKeyUsage `uint32_t TEE_OperationInfoKey::requiredKeyUsage`

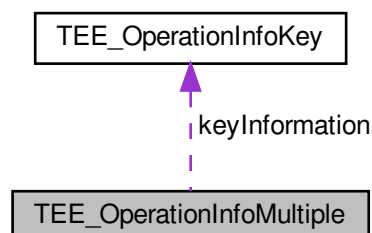
The documentation for this struct was generated from the following file:

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

9.27 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](#) []

9.27.1 Member Data Documentation

9.27.1.1 algorithm uint32_t TEE_OperationInfoMultiple::algorithm

9.27.1.2 digestLength uint32_t TEE_OperationInfoMultiple::digestLength

9.27.1.3 handleState uint32_t TEE_OperationInfoMultiple::handleState

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

9.27.1.5 maxKeySize uint32_t TEE_OperationInfoMultiple::maxKeySize

9.27.1.6 mode uint32_t TEE_OperationInfoMultiple::mode

9.27.1.7 numberOfKeys uint32_t TEE_OperationInfoMultiple::numberOfKeys

9.27.1.8 operationClass uint32_t TEE_OperationInfoMultiple::operationClass

9.27.1.9 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

9.28 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](#)

9.28.1 Member Data Documentation**9.28.1.1 a** uint32_t TEE_Param::a**9.28.1.2 b** uint32_t TEE_Param::b**9.28.1.3 buffer** void* TEE_Param::buffer**9.28.1.4** struct { ... } TEE_Param::memref**9.28.1.5 size** uint32_t TEE_Param::size

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

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

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

9.29 TEE_SEAID Struct Reference

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

Public Attributes

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

9.29.1 Member Data Documentation

9.29.1.1 **buffer** `uint8_t* TEE_SEAID::buffer`

9.29.1.2 **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](#)

9.30 TEE_SEReaderProperties Struct Reference

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

Public Attributes

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

9.30.1 Member Data Documentation

9.30.1.1 selectResponseEnable `bool TEE_SEReadProperties::selectResponseEnable`

9.30.1.2 sePresent `bool TEE_SEReadProperties::sePresent`

9.30.1.3 teeOnly `bool TEE_SEReadProperties::teeOnly`

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

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

9.31 TEE_Time Struct Reference

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

Public Attributes

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

9.31.1 Member Data Documentation

9.31.1.1 millis `uint32_t TEE_Time::millis`

9.31.1.2 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](#)

9.32 TEE_UUID Struct Reference

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

Public Attributes

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

9.32.1 Member Data Documentation

9.32.1.1 clockSeqAndNode `uint8_t TEE_UUID::clockSeqAndNode[8]`

9.32.1.2 timeHiAndVersion `uint16_t TEE_UUID::timeHiAndVersion`

9.32.1.3 timeLow `uint32_t TEE_UUID::timeLow`

9.32.1.4 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](#)

9.33 TEEC_Context Struct Reference

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

Public Attributes

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

9.33.1 Detailed Description

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

9.33.2 Member Data Documentation

9.33.2.1 `fd` `int TEEC_Context::fd`

9.33.2.2 `reg_mem` `bool TEEC_Context::reg_mem`

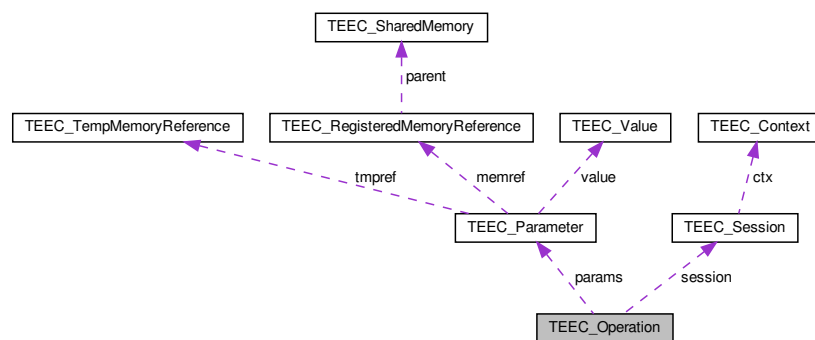
The documentation for this struct was generated from the following file:

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

9.34 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`

9.34.1 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.

9.34.2 Member Data Documentation

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

9.34.2.2 paramTypes `uint32_t` `TEEC_Operation::paramTypes`

9.34.2.3 session `TEEC_Session*` `TEEC_Operation::session`

9.34.2.4 started `uint32_t` `TEEC_Operation::started`

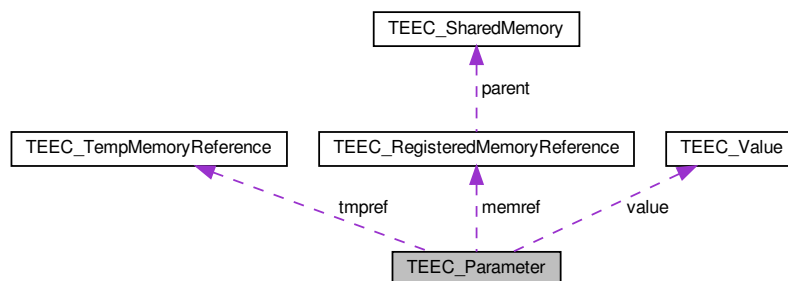
The documentation for this struct was generated from the following file:

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

9.35 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`

9.35.1 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

9.35.2 Member Data Documentation

9.35.2.1 memref [TEEC_RegisteredMemoryReference](#) `TEEC_Parameter::memref`

9.35.2.2 tmpref [TEEC_TempMemoryReference](#) `TEEC_Parameter::tmpref`

9.35.2.3 value [TEEC_Value](#) `TEEC_Parameter::value`

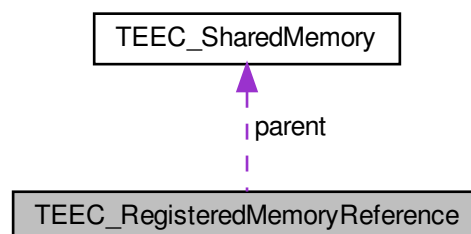
The documentation for this union was generated from the following file:

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

9.36 TEEC_RegisteredMemoryReference Struct Reference

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

Collaboration diagram for TEEC_RegisteredMemoryReference:



Public Attributes

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

9.36.1 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.

9.36.2 Member Data Documentation

9.36.2.1 offset `size_t TEEC_RegisteredMemoryReference::offset`

9.36.2.2 parent `TEEC_SharedMemory* TEEC_RegisteredMemoryReference::parent`

9.36.2.3 size `size_t TEEC_RegisteredMemoryReference::size`

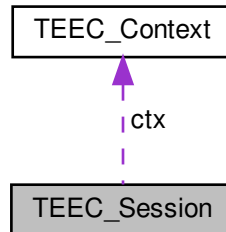
The documentation for this struct was generated from the following file:

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

9.37 TEEC_Session Struct Reference

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

Collaboration diagram for TEEC_Session:



Public Attributes

- [TEEC_Context](#) * `ctx`
- `uint32_t` [session_id](#)

9.37.1 Detailed Description

struct [TEEC_Session](#) - Represents a connection between a client application and a trusted application.

9.37.2 Member Data Documentation

9.37.2.1 `ctx` [TEEC_Context](#)* `TEEC_Session::ctx`

9.37.2.2 `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`

9.38 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`

9.38.1 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 <code>TEEC_MEM_INPUT</code> and <code>TEEC_MEM_OUTPUT</code> 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.

9.38.2 Member Data Documentation

9.38.2.1 `allocated_size` `size_t TEEC_SharedMemory::allocated_size`

9.38.2.2 `buffer` `void* TEEC_SharedMemory::buffer`

9.38.2.3 `buffer_allocated` `bool TEEC_SharedMemory::buffer_allocated`

9.38.2.4 `flags` `uint32_t TEEC_SharedMemory::flags`

9.38.2.5 id `int TEEC_SharedMemory::id`

9.38.2.6 registered_fd `int TEEC_SharedMemory::registered_fd`

9.38.2.7 shadow_buffer `void* TEEC_SharedMemory::shadow_buffer`

9.38.2.8 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](#)

9.39 TEEC_TempMemoryReference Struct Reference

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

Public Attributes

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

9.39.1 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.

9.39.2 Member Data Documentation

9.39.2.1 buffer `void* TEEC_TempMemoryReference::buffer`

9.39.2.2 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](#)

9.40 TEEC_UUID Struct Reference

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

Public Attributes

- `uint32_t timeLow`
- `uint16_t timeMid`
- `uint16_t timeHiAndVersion`
- `uint8_t clockSeqAndNode [8]`

9.40.1 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.

9.40.2 Member Data Documentation

9.40.2.1 clockSeqAndNode `uint8_t TEEC_UUID::clockSeqAndNode[8]`

9.40.2.2 timeHiAndVersion `uint16_t TEEC_UUID::timeHiAndVersion`

9.40.2.3 timeLow `uint32_t TEEC_UUID::timeLow`

9.40.2.4 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](#)

9.41 TEEC_Value Struct Reference

```
#include <tee-client_api.h>
```

Public Attributes

- `uint32_t` [a](#)
- `uint32_t` [b](#)

9.41.1 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.

9.41.2 Member Data Documentation

9.41.2.1 **a** `uint32_t TEEC_Value::a`

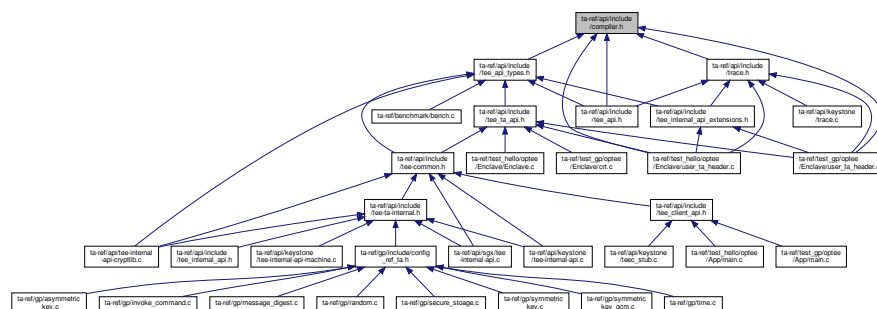
9.41.2.2 **b** `uint32_t TEEC_Value::b`

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

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

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

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



- #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)))`

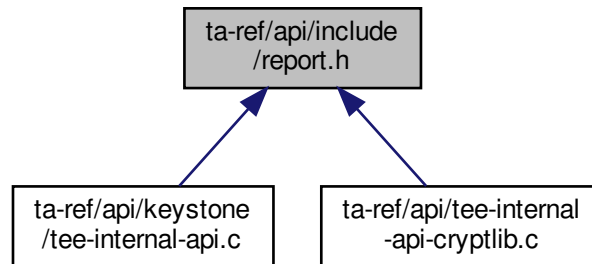
10.1.1.46 `__unused` `#define __unused __attribute__((unused))`

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

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

10.2 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

10.2.1 Macro Definition Documentation

10.2.1.1 ATTEST_DATA_MAXLEN `#define ATTEST_DATA_MAXLEN 1024`

10.2.1.2 MDSIZE `#define MDSIZE 64`

10.3.1 Detailed Description

Common type and definitions of RISC-V TEE.

draft RISC-V Internal TEE API

Author

Akira Tsukamoto, AIST

Date

2019/09/25

10.3.2 Macro Definition Documentation

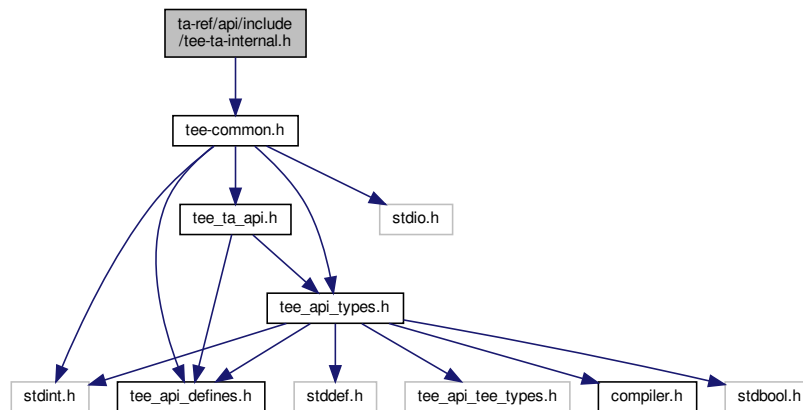
10.3.2.1 pr_deb `#define pr_deb(
...) do { } while (0)`

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

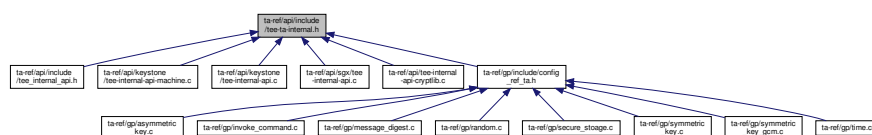
Candidate API list for Global Platform like RISC-V TEE.

`#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)
- 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.*

10.4.1 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

10.4.2 Function Documentation

10.4.2.1 [__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. 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.
-----------	--

10.4.2.2 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

10.4.2.3 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.

10.4.2.4 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

10.4.2.5 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.

10.4.2.6 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.

10.4.2.7 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 output buffer.

Returns

0 on success.

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

10.4.2.8 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.

10.4.2.9 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.

10.4.2.10 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.

10.4.2.11 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
Paramter list continued on next page	

<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 sccess

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

10.4.2.12 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.

10.4.2.13 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

10.4.2.14 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.

10.4.2.15 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.Free↔TransientObject.

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.

10.4.2.16 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
<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.

10.4.2.17 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.

10.4.2.18 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.

10.4.2.19 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.

10.4.2.20 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.
---------------	-------------------------------

10.4.2.21 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.
<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.

10.4.2.22 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 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

[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

10.4.2.23 TEE_GetObjectInfo1() `TEE_Result TEE_GetObjectInfo1 (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.

10.4.2.24 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.
-------------	---

10.4.2.25 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.
-------------	---

10.4.2.26 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.

10.4.2.27 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

10.4.2.28 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.
<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.

```
10.4.2.29 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-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.

10.4.2.30 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.

10.4.2.31 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

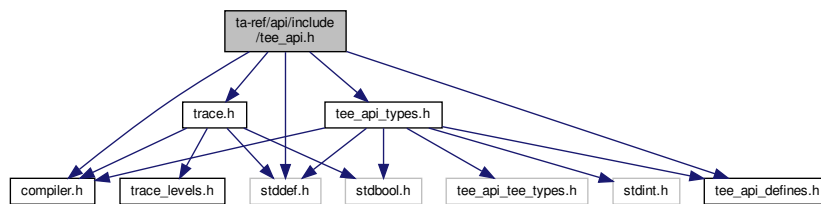
Returns

TEE_SUCCESS if success else error occurred.

10.5 ta-ref/api/include/tee_api.h File Reference

```
#include <stddef.h>
#include <compiler.h>
#include <tee_api_defines.h>
#include <tee_api_types.h>
#include <trace.h>
```

Include dependency graph for tee_api.h:



Functions

- [TEE_Result TEE_GetPropertyAsString](#) ([TEE_PropSetHandle](#) propsetOrEnumerator, const char *name, char *valueBuffer, uint32_t *valueBufferLen)
- [TEE_Result TEE_GetPropertyAsBool](#) ([TEE_PropSetHandle](#) propsetOrEnumerator, const char *name, bool *value)
- [TEE_Result TEE_GetPropertyAsU32](#) ([TEE_PropSetHandle](#) propsetOrEnumerator, const char *name, uint32_t *value)
- [TEE_Result TEE_GetPropertyAsBinaryBlock](#) ([TEE_PropSetHandle](#) propsetOrEnumerator, const char *name, void *valueBuffer, uint32_t *valueBufferLen)
- [TEE_Result TEE_GetPropertyAsUUID](#) ([TEE_PropSetHandle](#) propsetOrEnumerator, const char *name, [TEE_UUID](#) *value)
- [TEE_Result TEE_GetPropertyAsIdentity](#) ([TEE_PropSetHandle](#) propsetOrEnumerator, const char *name, [TEE_Identity](#) *value)
- [TEE_Result TEE_AllocatePropertyEnumerator](#) ([TEE_PropSetHandle](#) *enumerator)
- void [TEE_FreePropertyEnumerator](#) ([TEE_PropSetHandle](#) enumerator)
- void [TEE_StartPropertyEnumerator](#) ([TEE_PropSetHandle](#) enumerator, [TEE_PropSetHandle](#) propSet)
- void [TEE_ResetPropertyEnumerator](#) ([TEE_PropSetHandle](#) enumerator)
- [TEE_Result TEE_GetPropertyName](#) ([TEE_PropSetHandle](#) enumerator, void *nameBuffer, uint32_t *nameBufferLen)
- [TEE_Result TEE_GetNextProperty](#) ([TEE_PropSetHandle](#) enumerator)

- void `TEE_Panic` (`TEE_Result` panicCode)
- `TEE_Result` `TEE_OpenTASession` (const `TEE_UUID` *destination, uint32_t cancellationRequestTimeout, uint32_t paramTypes, `TEE_Param` params[`TEE_NUM_PARAMS`], `TEE_TASessionHandle` *session, uint32_t *returnOrigin)
- void `TEE_CloseTASession` (`TEE_TASessionHandle` session)
- `TEE_Result` `TEE_InvokeTACommand` (`TEE_TASessionHandle` session, uint32_t cancellationRequestTimeout, uint32_t commandID, uint32_t paramTypes, `TEE_Param` params[`TEE_NUM_PARAMS`], uint32_t *returnOrigin)
- bool `TEE_GetCancellationFlag` (void)
- bool `TEE_UnmaskCancellation` (void)
- bool `TEE_MaskCancellation` (void)
- `TEE_Result` `TEE_CheckMemoryAccessRights` (uint32_t accessFlags, void *buffer, uint32_t size)
- void `TEE_SetInstanceData` (const void *instanceData)
- const void * `TEE_GetInstanceData` (void)
- void * `TEE_Malloc` (uint32_t size, uint32_t hint)
- void * `TEE_Realloc` (void *buffer, uint32_t newSize)
- void `TEE_Free` (void *buffer)
- void * `TEE_MemMove` (void *dest, const void *src, uint32_t size)
- int32_t `TEE_MemCompare` (const void *buffer1, const void *buffer2, uint32_t size)
- void * `TEE_MemFill` (void *buff, uint32_t x, uint32_t size)
- void `TEE_GetObjectInfo` (`TEE_ObjectHandle` object, `TEE_ObjectInfo` *objectInfo)
- `TEE_Result` `TEE_GetObjectInfo1` (`TEE_ObjectHandle` object, `TEE_ObjectInfo` *objectInfo)

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

- void `TEE_RestrictObjectUsage` (`TEE_ObjectHandle` object, uint32_t objectUsage)
- `TEE_Result` `TEE_RestrictObjectUsage1` (`TEE_ObjectHandle` object, uint32_t objectUsage)
- `TEE_Result` `TEE_GetObjectBufferAttribute` (`TEE_ObjectHandle` object, uint32_t attributeID, void *buffer, uint32_t *size)
- `TEE_Result` `TEE_GetObjectValueAttribute` (`TEE_ObjectHandle` object, uint32_t attributeID, uint32_t *a, uint32_t *b)
- void `TEE_CloseObject` (`TEE_ObjectHandle` object)

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

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

Crypto, Asymmetric key Verification Functions.

- void `TEE_FreeTransientObject` (`TEE_ObjectHandle` object)

Crypto, Asymmetric key Verification Functions.

- void `TEE_ResetTransientObject` (`TEE_ObjectHandle` object)
- `TEE_Result` `TEE_PopulateTransientObject` (`TEE_ObjectHandle` object, const `TEE_Attribute` *attrs, uint32_t attrCount)
- 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_CopyObjectAttributes` (`TEE_ObjectHandle` destObject, `TEE_ObjectHandle` srcObject)
- `TEE_Result` `TEE_CopyObjectAttributes1` (`TEE_ObjectHandle` destObject, `TEE_ObjectHandle` srcObject)
- `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_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_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)

- void [TEE_CloseAndDeletePersistentObject](#) ([TEE_ObjectHandle](#) object)
- [TEE_Result](#) [TEE_CloseAndDeletePersistentObject1](#) ([TEE_ObjectHandle](#) object)
- [TEE_Result](#) [TEE_RenamePersistentObject](#) ([TEE_ObjectHandle](#) object, const void *newObjectID, uint32_t newObjectIDLen)
- [TEE_Result](#) [TEE_AllocatePersistentObjectEnumerator](#) ([TEE_ObjectEnumHandle](#) *objectEnumerator)
- void [TEE_FreePersistentObjectEnumerator](#) ([TEE_ObjectEnumHandle](#) objectEnumerator)
- void [TEE_ResetPersistentObjectEnumerator](#) ([TEE_ObjectEnumHandle](#) objectEnumerator)
- [TEE_Result](#) [TEE_StartPersistentObjectEnumerator](#) ([TEE_ObjectEnumHandle](#) objectEnumerator, uint32_t storageID)
- [TEE_Result](#) [TEE_GetNextPersistentObject](#) ([TEE_ObjectEnumHandle](#) objectEnumerator, [TEE_ObjectInfo](#) *objectInfo, void *objectID, uint32_t *objectIDLen)
- [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_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_TruncateObjectData](#) ([TEE_ObjectHandle](#) object, uint32_t size)
- [TEE_Result](#) [TEE_SeekObjectData](#) ([TEE_ObjectHandle](#) object, int32_t offset, [TEE_Whence](#) whence)
- [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_GetOperationInfo](#) ([TEE_OperationHandle](#) operation, [TEE_OperationInfo](#) *operationInfo)
- [TEE_Result](#) [TEE_GetOperationInfoMultiple](#) ([TEE_OperationHandle](#) operation, [TEE_OperationInfoMultiple](#) *operationInfoMultiple, uint32_t *operationSize)
- void [TEE_ResetOperation](#) ([TEE_OperationHandle](#) operation)
- [TEE_Result](#) [TEE_SetOperationKey](#) ([TEE_OperationHandle](#) operation, [TEE_ObjectHandle](#) key)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- [TEE_Result](#) [TEE_SetOperationKey2](#) ([TEE_OperationHandle](#) operation, [TEE_ObjectHandle](#) key1, [TEE_ObjectHandle](#) key2)
- void [TEE_CopyOperation](#) ([TEE_OperationHandle](#) dstOperation, [TEE_OperationHandle](#) srcOperation)
- [TEE_Result](#) [TEE_IsAlgorithmSupported](#) (uint32_t algId, uint32_t element)
- 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)
- void [TEE_CipherInit](#) ([TEE_OperationHandle](#) operation, const void *IV, uint32_t IVLen)
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)
- void [TEE_MACInit](#) ([TEE_OperationHandle](#) operation, const void *IV, uint32_t IVLen)
- void [TEE_MACUpdate](#) ([TEE_OperationHandle](#) operation, const void *chunk, uint32_t chunkSize)
- [TEE_Result](#) [TEE_MACComputeFinal](#) ([TEE_OperationHandle](#) operation, const void *message, uint32_t messageLen, void *mac, uint32_t *macLen)
- [TEE_Result](#) [TEE_MACCompareFinal](#) ([TEE_OperationHandle](#) operation, const void *message, uint32_t messageLen, const void *mac, uint32_t macLen)
- [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.
- **TEE_Result TEE_AsymmetricEncrypt** (**TEE.OperationHandle** operation, const **TEE.Attribute** *params, uint32_t paramCount, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)
- **TEE_Result TEE_AsymmetricDecrypt** (**TEE.OperationHandle** operation, const **TEE.Attribute** *params, uint32_t paramCount, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)
- **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.
- void **TEE_DeriveKey** (**TEE.OperationHandle** operation, const **TEE.Attribute** *params, uint32_t paramCount, **TEE.ObjectHandle** derivedKey)
- void **TEE.GenerateRandom** (void *randomBuffer, uint32_t randomBufferLen)
Crypto, common.
- void **TEE.GetSystemTime** (**TEE.Time** *time)
Core Functions, Time Functions.
- **TEE_Result TEE.Wait** (uint32_t timeout)
- **TEE_Result TEE_GetTAPersistentTime** (**TEE.Time** *time)
- **TEE_Result TEE_SetTAPersistentTime** (const **TEE.Time** *time)
- void **TEE_GetREETime** (**TEE.Time** *time)
Core Functions, Time Functions.
- uint32_t **TEE_BigIntFMMSizeInU32** (uint32_t modulusSizeInBits)
- uint32_t **TEE_BigIntFMMContextSizeInU32** (uint32_t modulusSizeInBits)
- void **TEE_BigIntInit** (**TEE_BigInt** *bigInt, uint32_t len)
- void **TEE_BigIntInitFMMContext** (**TEE_BigIntFMMContext** *context, uint32_t len, const **TEE_BigInt** *modulus)
- void **TEE_BigIntInitFMM** (**TEE_BigIntFMM** *bigIntFMM, uint32_t len)
- **TEE_Result TEE_BigIntConvertFromOctetString** (**TEE_BigInt** *dest, const uint8_t *buffer, uint32_t bufferLen, int32_t sign)
- **TEE_Result TEE_BigIntConvertToOctetString** (uint8_t *buffer, uint32_t *bufferLen, const **TEE_BigInt** *bigInt)
- void **TEE_BigIntConvertFromS32** (**TEE_BigInt** *dest, int32_t shortVal)
- **TEE_Result TEE_BigIntConvertToS32** (int32_t *dest, const **TEE_BigInt** *src)
- int32_t **TEE_BigIntCmp** (const **TEE_BigInt** *op1, const **TEE_BigInt** *op2)
- int32_t **TEE_BigIntCmpS32** (const **TEE_BigInt** *op, int32_t shortVal)
- void **TEE_BigIntShiftRight** (**TEE_BigInt** *dest, const **TEE_BigInt** *op, size_t bits)
- bool **TEE_BigIntGetBit** (const **TEE_BigInt** *src, uint32_t bitIndex)
- uint32_t **TEE_BigIntGetBitCount** (const **TEE_BigInt** *src)
- void **TEE_BigIntAdd** (**TEE_BigInt** *dest, const **TEE_BigInt** *op1, const **TEE_BigInt** *op2)
- void **TEE_BigIntSub** (**TEE_BigInt** *dest, const **TEE_BigInt** *op1, const **TEE_BigInt** *op2)
- void **TEE_BigIntNeg** (**TEE_BigInt** *dest, const **TEE_BigInt** *op)
- void **TEE_BigIntMul** (**TEE_BigInt** *dest, const **TEE_BigInt** *op1, const **TEE_BigInt** *op2)
- void **TEE_BigIntSquare** (**TEE_BigInt** *dest, const **TEE_BigInt** *op)
- void **TEE_BigIntDiv** (**TEE_BigInt** *dest_q, **TEE_BigInt** *dest_r, const **TEE_BigInt** *op1, const **TEE_BigInt** *op2)
- void **TEE_BigIntMod** (**TEE_BigInt** *dest, const **TEE_BigInt** *op, const **TEE_BigInt** *n)
- void **TEE_BigIntAddMod** (**TEE_BigInt** *dest, const **TEE_BigInt** *op1, const **TEE_BigInt** *op2, const **TEE_BigInt** *n)

- void `TEE_BigIntSubMod` (`TEE_BigInt` *dest, const `TEE_BigInt` *op1, const `TEE_BigInt` *op2, const `TEE_BigInt` *n)
- void `TEE_BigIntMulMod` (`TEE_BigInt` *dest, const `TEE_BigInt` *op1, const `TEE_BigInt` *op2, const `TEE_BigInt` *n)
- void `TEE_BigIntSquareMod` (`TEE_BigInt` *dest, const `TEE_BigInt` *op, const `TEE_BigInt` *n)
- void `TEE_BigIntInvMod` (`TEE_BigInt` *dest, const `TEE_BigInt` *op, const `TEE_BigInt` *n)
- bool `TEE_BigIntRelativePrime` (const `TEE_BigInt` *op1, const `TEE_BigInt` *op2)
- void `TEE_BigIntComputeExtendedGcd` (`TEE_BigInt` *gcd, `TEE_BigInt` *u, `TEE_BigInt` *v, const `TEE_BigInt` *op1, const `TEE_BigInt` *op2)
- int32_t `TEE_BigIntIsProbablePrime` (const `TEE_BigInt` *op, uint32_t confidenceLevel)
- void `TEE_BigIntConvertToFMM` (`TEE_BigIntFMM` *dest, const `TEE_BigInt` *src, const `TEE_BigInt` *n, const `TEE_BigIntFMMContext` *context)
- void `TEE_BigIntConvertFromFMM` (`TEE_BigInt` *dest, const `TEE_BigIntFMM` *src, const `TEE_BigInt` *n, const `TEE_BigIntFMMContext` *context)
- void `TEE_BigIntFMMConvertToBigInt` (`TEE_BigInt` *dest, const `TEE_BigIntFMM` *src, const `TEE_BigInt` *n, const `TEE_BigIntFMMContext` *context)
- void `TEE_BigIntComputeFMM` (`TEE_BigIntFMM` *dest, const `TEE_BigIntFMM` *op1, const `TEE_BigIntFMM` *op2, const `TEE_BigInt` *n, const `TEE_BigIntFMMContext` *context)

10.5.1 Function Documentation

10.5.1.1 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

10.5.1.2 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.

10.5.1.3 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.

10.5.1.4 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.

10.5.1.5 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.

10.5.1.6 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.

10.5.1.7 TEE_AllocatePersistentObjectEnumerator() `TEE_Result` TEE_AllocatePersistentObjectEnumerator (

`TEE_ObjectEnumHandle` * *objectEnumerator*)

10.5.1.8 TEE_AllocatePropertyEnumerator() `TEE_Result` TEE_AllocatePropertyEnumerator (

`TEE_PropSetHandle` * *enumerator*)

10.5.1.9 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.

10.5.1.10 TEE_AsymmetricDecrypt() `TEE_Result` TEE_AsymmetricDecrypt (
 `TEE.OperationHandle` operation,
 const `TEE.Attribute` * params,
 uint32_t paramCount,
 const void * srcData,
 uint32_t srcLen,
 void * destData,
 uint32_t * destLen)

10.5.1.11 TEE_AsymmetricEncrypt() `TEE_Result` TEE_AsymmetricEncrypt (
 `TEE.OperationHandle` operation,
 const `TEE.Attribute` * params,
 uint32_t paramCount,
 const void * srcData,
 uint32_t srcLen,
 void * destData,
 uint32_t * destLen)

10.5.1.12 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

10.5.1.13 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.

10.5.1.14 TEE_BigIntAdd() `void TEE_BigIntAdd (`
`TEE_BigInt * dest,`
`const TEE_BigInt * op1,`
`const TEE_BigInt * op2)`

10.5.1.15 TEE_BigIntAddMod() `void TEE_BigIntAddMod (`
`TEE_BigInt * dest,`
`const TEE_BigInt * op1,`
`const TEE_BigInt * op2,`
`const TEE_BigInt * n)`

10.5.1.16 TEE_BigIntCmp() `int32_t TEE_BigIntCmp (`
`const TEE_BigInt * op1,`
`const TEE_BigInt * op2)`

10.5.1.17 TEE_BigIntCmpS32() `int32_t TEE_BigIntCmpS32 (`
 `const TEE_BigInt * op,`
 `int32_t shortVal)`

10.5.1.18 TEE_BigIntComputeExtendedGcd() `void TEE_BigIntComputeExtendedGcd (`
 `TEE_BigInt * gcd,`
 `TEE_BigInt * u,`
 `TEE_BigInt * v,`
 `const TEE_BigInt * op1,`
 `const TEE_BigInt * op2)`

10.5.1.19 TEE_BigIntComputeFMM() `void TEE_BigIntComputeFMM (`
 `TEE_BigIntFMM * dest,`
 `const TEE_BigIntFMM * op1,`
 `const TEE_BigIntFMM * op2,`
 `const TEE_BigInt * n,`
 `const TEE_BigIntFMMContext * context)`

10.5.1.20 TEE_BigIntConvertFromFMM() `void TEE_BigIntConvertFromFMM (`
 `TEE_BigInt * dest,`
 `const TEE_BigIntFMM * src,`
 `const TEE_BigInt * n,`
 `const TEE_BigIntFMMContext * context)`

10.5.1.21 TEE_BigIntConvertFromOctetString() `TEE_Result TEE_BigIntConvertFromOctetString (`
 `TEE_BigInt * dest,`
 `const uint8_t * buffer,`
 `uint32_t bufferLen,`
 `int32_t sign)`

10.5.1.22 TEE_BigIntConvertFromS32() `void TEE_BigIntConvertFromS32 (`
 `TEE_BigInt * dest,`
 `int32_t shortVal)`

10.5.1.23 TEE_BigIntConvertToFMM() `void TEE_BigIntConvertToFMM (`
 `TEE_BigIntFMM * dest,`
 `const TEE_BigInt * src,`
 `const TEE_BigInt * n,`
 `const TEE_BigIntFMMContext * context)`

10.5.1.24 TEE_BigIntConvertToOctetString() `TEE_Result TEE_BigIntConvertToOctetString (`
 `uint8_t * buffer,`
 `uint32_t * bufferLen,`
 `const TEE_BigInt * bigInt)`

10.5.1.25 TEE_BigIntConvertToS32() `TEE_Result TEE_BigIntConvertToS32 (`
 `int32_t * dest,`
 `const TEE_BigInt * src)`

10.5.1.26 TEE_BigIntDiv() `void TEE_BigIntDiv (`
 `TEE_BigInt * dest_q,`
 `TEE_BigInt * dest_r,`
 `const TEE_BigInt * op1,`
 `const TEE_BigInt * op2)`

10.5.1.27 TEE_BigIntFMMContextSizeInU32() `uint32_t TEE_BigIntFMMContextSizeInU32 (`
 `uint32_t modulusSizeInBits)`

10.5.1.28 TEE_BigIntFMMConvertToBigInt() `void TEE_BigIntFMMConvertToBigInt (`
 `TEE_BigInt * dest,`
 `const TEE_BigIntFMM * src,`
 `const TEE_BigInt * n,`
 `const TEE_BigIntFMMContext * context)`

10.5.1.29 TEE_BigIntFMMSizeInU32() `uint32_t TEE_BigIntFMMSizeInU32 (`
 `uint32_t modulusSizeInBits)`

10.5.1.30 TEE_BigIntGetBit() `bool TEE_BigIntGetBit (`
 `const TEE_BigInt * src,`
 `uint32_t bitIndex)`

10.5.1.31 TEE_BigIntGetBitCount() `uint32_t TEE_BigIntGetBitCount (`
 `const TEE_BigInt * src)`

10.5.1.32 TEE_BigIntInit() void TEE_BigIntInit (
 TEE_BigInt * *bigInt*,
 uint32_t *len*)

10.5.1.33 TEE_BigIntInitFMM() void TEE_BigIntInitFMM (
 TEE_BigIntFMM * *bigIntFMM*,
 uint32_t *len*)

10.5.1.34 TEE_BigIntInitFMMContext() void TEE_BigIntInitFMMContext (
 TEE_BigIntFMMContext * *context*,
 uint32_t *len*,
 const TEE_BigInt * *modulus*)

10.5.1.35 TEE_BigIntInvMod() void TEE_BigIntInvMod (
 TEE_BigInt * *dest*,
 const TEE_BigInt * *op*,
 const TEE_BigInt * *n*)

10.5.1.36 TEE_BigIntIsProbablePrime() int32_t TEE_BigIntIsProbablePrime (
 const TEE_BigInt * *op*,
 uint32_t *confidenceLevel*)

10.5.1.37 TEE_BigIntMod() void TEE_BigIntMod (
 TEE_BigInt * *dest*,
 const TEE_BigInt * *op*,
 const TEE_BigInt * *n*)

10.5.1.38 TEE_BigIntMul() void TEE_BigIntMul (
 TEE_BigInt * *dest*,
 const TEE_BigInt * *op1*,
 const TEE_BigInt * *op2*)

10.5.1.39 TEE_BigIntMulMod() void TEE_BigIntMulMod (
 TEE_BigInt * *dest*,
 const TEE_BigInt * *op1*,
 const TEE_BigInt * *op2*,
 const TEE_BigInt * *n*)

10.5.1.40 TEE_BigIntNeg() void TEE_BigIntNeg (
 TEE_BigInt * dest,
 const TEE_BigInt * op)

10.5.1.41 TEE_BigIntRelativePrime() bool TEE_BigIntRelativePrime (
 const TEE_BigInt * op1,
 const TEE_BigInt * op2)

10.5.1.42 TEE_BigIntShiftRight() void TEE_BigIntShiftRight (
 TEE_BigInt * dest,
 const TEE_BigInt * op,
 size_t bits)

10.5.1.43 TEE_BigIntSquare() void TEE_BigIntSquare (
 TEE_BigInt * dest,
 const TEE_BigInt * op)

10.5.1.44 TEE_BigIntSquareMod() void TEE_BigIntSquareMod (
 TEE_BigInt * dest,
 const TEE_BigInt * op,
 const TEE_BigInt * n)

10.5.1.45 TEE_BigIntSub() void TEE_BigIntSub (
 TEE_BigInt * dest,
 const TEE_BigInt * op1,
 const TEE_BigInt * op2)

10.5.1.46 TEE_BigIntSubMod() void TEE_BigIntSubMod (
 TEE_BigInt * dest,
 const TEE_BigInt * op1,
 const TEE_BigInt * op2,
 const TEE_BigInt * n)

10.5.1.47 TEE.CheckMemoryAccessRights() `TEE_Result` TEE.CheckMemoryAccessRights (
 uint32_t *accessFlags*,
 void * *buffer*,
 uint32_t *size*)

10.5.1.48 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

10.5.1.49 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

10.5.1.50 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.

10.5.1.51 TEE_CloseAndDeletePersistentObject() `void TEE_CloseAndDeletePersistentObject (`
 `TEE_ObjectHandle object)`

10.5.1.52 TEE_CloseAndDeletePersistentObject1() `TEE_Result TEE_CloseAndDeletePersistentObject1 (`
 `TEE_ObjectHandle object)`

10.5.1.53 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.

[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.

10.5.1.54 TEE_CloseTASession() `void TEE_CloseTASession (`
`TEE_TASessionHandle session)`

10.5.1.55 TEE_CopyObjectAttributes() `void TEE_CopyObjectAttributes (`
`TEE_ObjectHandle destObject,`
`TEE_ObjectHandle srcObject)`

10.5.1.56 TEE_CopyObjectAttributes1() `TEE_Result TEE_CopyObjectAttributes1 (`
`TEE_ObjectHandle destObject,`
`TEE_ObjectHandle srcObject)`

10.5.1.57 TEE_CopyOperation() void TEE_CopyOperation (
 TEE_OperationHandle dstOperation,
 TEE_OperationHandle srcOperation)

10.5.1.58 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.

[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.
Paramter list continued on next page	

<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.

10.5.1.59 TEE.DeriveKey() void TEE.DeriveKey (
 TEE.OperationHandle operation,
 const TEE.Attribute * params,
 uint32_t paramCount,
 TEE.ObjectHandle derivedKey)

10.5.1.60 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.

10.5.1.61 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.

10.5.1.62 TEE.Free() void TEE.Free (
 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.
---------------	--

10.5.1.63 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.

10.5.1.64 TEE_FreePersistentObjectEnumerator() void TEE_FreePersistentObjectEnumerator (
[TEE_ObjectEnumHandle](#) objectEnumerator)

10.5.1.65 TEE_FreePropertyEnumerator() void TEE_FreePropertyEnumerator (
[TEE_PropSetHandle](#) enumerator)

10.5.1.66 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.
---------------	-------------------------------

10.5.1.67 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.
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.

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

Crypto, common.

[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 bufferlength 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

10.5.1.69 TEE_GetCancellationFlag() `bool TEE_GetCancellationFlag (void)`

10.5.1.70 TEE_GetInstanceData() `const void* TEE_GetInstanceData (void)`

10.5.1.71 TEE_GetNextPersistentObject() `TEE_Result TEE_GetNextPersistentObject (TEE_ObjectEnumHandle objectEnumerator, TEE_ObjectInfo * objectInfo, void * objectID, uint32_t * objectIDLen)`

10.5.1.72 TEE_GetNextProperty() `TEE_Result TEE_GetNextProperty (TEE_PropSetHandle enumerator)`

10.5.1.73 TEE_GetObjectBufferAttribute() `TEE_Result TEE_GetObjectBufferAttribute (TEE_ObjectHandle object, uint32_t attributeID, void * buffer, uint32_t * size)`

10.5.1.74 TEE_GetObjectInfo() `void TEE_GetObjectInfo (`
 `TEE_ObjectHandle object,`
 `TEE_ObjectInfo * objectInfo)`

10.5.1.75 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.

[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.

10.5.1.76 TEE_GetObjectValueAttribute() `TEE_Result TEE_GetObjectValueAttribute (`
 `TEE_ObjectHandle object,`
 `uint32_t attributeID,`
 `uint32_t * a,`
 `uint32_t * b)`

10.5.1.77 TEE_GetOperationInfo() `void TEE_GetOperationInfo (`
 `TEE_OperationHandle operation,`
 `TEE_OperationInfo * operationInfo)`

10.5.1.78 TEE_GetOperationInfoMultiple() `TEE_Result TEE_GetOperationInfoMultiple (`
 `TEE_OperationHandle operation,`
 `TEE_OperationInfoMultiple * operationInfoMultiple,`
 `uint32_t * operationSize)`

10.5.1.79 TEE_GetPropertyAsBinaryBlock() `TEE_Result TEE_GetPropertyAsBinaryBlock (`
 `TEE_PropSetHandle propsetOrEnumerator,`
 `const char * name,`
 `void * valueBuffer,`
 `uint32_t * valueBufferLen)`

10.5.1.80 TEE_GetPropertyAsBool() `TEE_Result TEE_GetPropertyAsBool (`
 `TEE_PropSetHandle propsetOrEnumerator,`
 `const char * name,`
 `bool * value)`

10.5.1.81 TEE_GetPropertyAsIdentity() `TEE_Result TEE_GetPropertyAsIdentity (`
 `TEE_PropSetHandle propsetOrEnumerator,`
 `const char * name,`
 `TEE_Identity * value)`

10.5.1.82 TEE_GetPropertyAsString() `TEE_Result TEE_GetPropertyAsString (`
 `TEE_PropSetHandle propsetOrEnumerator,`
 `const char * name,`
 `char * valueBuffer,`
 `uint32_t * valueBufferLen)`

10.5.1.83 TEE_GetPropertyAsU32() `TEE_Result TEE_GetPropertyAsU32 (`
 `TEE_PropSetHandle propsetOrEnumerator,`
 `const char * name,`
 `uint32_t * value)`

10.5.1.84 TEE_GetPropertyAsUUID() `TEE_Result TEE_GetPropertyAsUUID (`
`TEE_PropSetHandle propsetOrEnumerator,`
`const char * name,`
`TEE_UUID * value)`

10.5.1.85 TEE_GetPropertyName() `TEE_Result TEE_GetPropertyName (`
`TEE_PropSetHandle enumerator,`
`void * nameBuffer,`
`uint32_t * nameBufferLen)`

10.5.1.86 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
-------------	--

[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.
-------------	---

10.5.1.87 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
-------------	--

[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.
-------------	---

10.5.1.88 TEE_GetTAPersistentTime() `TEE_Result TEE_GetTAPersistentTime (TEE_Time * time)`

10.5.1.89 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.

10.5.1.90 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

10.5.1.91 TEE_InvokeTACommand() `TEE_Result` TEE_InvokeTACommand (
 `TEE_TASessionHandle` session,
 uint32_t cancellationRequestTimeout,
 uint32_t commandID,
 uint32_t paramTypes,
 `TEE_Param` params[`TEE_NUM_PARAMS`],
 uint32_t * returnOrigin)

10.5.1.92 TEE_IsAlgorithmSupported() `TEE_Result` TEE_IsAlgorithmSupported (
 uint32_t algId,
 uint32_t element)

10.5.1.93 TEE_MACCompareFinal() `TEE_Result` TEE_MACCompareFinal (
 `TEE.OperationHandle` operation,
 const void * message,
 uint32_t messageLen,
 const void * mac,
 uint32_t macLen)

10.5.1.94 TEE_MACComputeFinal() `TEE_Result` TEE_MACComputeFinal (
 `TEE.OperationHandle` operation,
 const void * message,
 uint32_t messageLen,
 void * mac,
 uint32_t * macLen)

10.5.1.95 TEE_MACInit() `void TEE_MACInit (`
 `TEE.OperationHandle operation,`
 `const void * IV,`
 `uint32_t IVLen)`

10.5.1.96 TEE_MACUpdate() `void TEE_MACUpdate (`
 `TEE.OperationHandle operation,`
 `const void * chunk,`
 `uint32_t chunkSize)`

10.5.1.97 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.

10.5.1.98 TEE_MaskCancellation() `bool TEE_MaskCancellation (`
 `void)`

10.5.1.99 TEE_MemCompare() `int32_t TEE_MemCompare (`
 `const void * buffer1,`
 `const void * buffer2,`
 `uint32_t size)`

10.5.1.100 TEE_MemFill() void* TEE_MemFill (
 void * *buff*,
 uint32_t *x*,
 uint32_t *size*)

10.5.1.101 TEE_MemMove() void* TEE_MemMove (
 void * *dest*,
 const void * *src*,
 uint32_t *size*)

10.5.1.102 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.

[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.

10.5.1.103 TEE_OpenTASession() `TEE_Result TEE_OpenTASession (`
 `const TEE_UUID * destination,`
 `uint32_t cancellationRequestTimeout,`
 `uint32_t paramTypes,`
 `TEE_Param params[TEE_NUM_PARAMS],`
 `TEE_TASessionHandle * session,`
 `uint32_t * returnOrigin)`

10.5.1.104 TEE_Panic() `void TEE_Panic (`
 `TEE_Result panicCode)`

10.5.1.105 TEE_PopulateTransientObject() `TEE_Result TEE_PopulateTransientObject (`
 `TEE_ObjectHandle object,`
 `const TEE_Attribute * attrs,`
 `uint32_t attrCount)`

10.5.1.106 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-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-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.

10.5.1.107 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.

- 10.5.1.108 TEE_RenamePersistentObject()** `TEE_Result` TEE_RenamePersistentObject (
 `TEE_ObjectHandle` *object*,
 `const void *` *newObjectID*,
 `uint32_t` *newObjectIDLen*)
- 10.5.1.109 TEE_ResetOperation()** `void` TEE_ResetOperation (
 `TEE_OperationHandle` *operation*)
- 10.5.1.110 TEE_ResetPersistentObjectEnumerator()** `void` TEE_ResetPersistentObjectEnumerator (
 `TEE_ObjectEnumHandle` *objectEnumerator*)
- 10.5.1.111 TEE_ResetPropertyEnumerator()** `void` TEE_ResetPropertyEnumerator (
 `TEE_PropSetHandle` *enumerator*)
- 10.5.1.112 TEE_ResetTransientObject()** `void` TEE_ResetTransientObject (
 `TEE_ObjectHandle` *object*)
- 10.5.1.113 TEE_RestrictObjectUsage()** `void` TEE_RestrictObjectUsage (
 `TEE_ObjectHandle` *object*,
 `uint32_t` *objectUsage*)
- 10.5.1.114 TEE_RestrictObjectUsage1()** `TEE_Result` TEE_RestrictObjectUsage1 (
 `TEE_ObjectHandle` *object*,
 `uint32_t` *objectUsage*)
- 10.5.1.115 TEE_SeekObjectData()** `TEE_Result` TEE_SeekObjectData (
 `TEE_ObjectHandle` *object*,
 `int32_t` *offset*,
 `TEE_Whence` *whence*)
- 10.5.1.116 TEE_SetInstanceData()** `void` TEE_SetInstanceData (
 `const void *` *instanceData*)

10.5.1.117 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.

10.5.1.118 TEE_SetOperationKey2() `TEE_Result TEE_SetOperationKey2 (`
`TEE.OperationHandle operation,`
`TEE.ObjectHandle key1,`
`TEE.ObjectHandle key2)`

10.5.1.119 TEE_SetTAPersistentTime() `TEE_Result TEE_SetTAPersistentTime (`
`const TEE.Time * time)`

10.5.1.120 TEE_StartPersistentObjectEnumerator() `TEE_Result TEE_StartPersistentObjectEnumerator (`
`TEE.ObjectEnumHandle objectEnumerator,`
`uint32_t storageID)`

10.5.1.121 TEE_StartPropertyEnumerator() `void TEE_StartPropertyEnumerator (`
`TEE.PropSetHandle enumerator,`
`TEE.PropSetHandle propSet)`

10.5.1.122 TEE_TruncateObjectData() `TEE_Result TEE_TruncateObjectData (`
 `TEE_ObjectHandle object,`
 `uint32_t size)`

10.5.1.123 TEE_UnmaskCancellation() `bool TEE_UnmaskCancellation (`
 `void)`

10.5.1.124 TEE_Wait() `TEE_Result TEE_Wait (`
 `uint32_t timeout)`

10.5.1.125 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.

[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

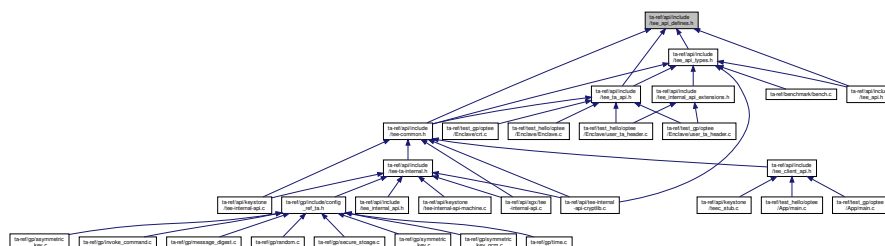
<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write

Returns

TEE_SUCCESS if success else error occurred.

10.6 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)0xFFFFFFFF
- #define TEE_PROPSET_CURRENT_CLIENT (TEE_PropSetHandle)0xFFFFFFFFE
- #define TEE_PROPSET_CURRENT_TA (TEE_PropSetHandle)0xFFFFFFFF
- #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

10.6.1 Macro Definition Documentation

10.6.1.1 TEE_ALG_AES_CBC_MAC_NOPAD #define TEE_ALG_AES_CBC_MAC_NOPAD 0x30000110

10.6.1.2 TEE_ALG_AES_CBC_MAC_PKCS5 #define TEE_ALG_AES_CBC_MAC_PKCS5 0x30000510

10.6.1.3 TEE_ALG_AES_CBC_NOPAD #define TEE_ALG_AES_CBC_NOPAD 0x10000110

10.6.1.4 TEE_ALG_AES_CCM #define TEE_ALG_AES_CCM 0x40000710

10.6.1.5 TEE_ALG_AES_CMAC #define TEE_ALG_AES_CMAC 0x30000610

10.6.1.6 TEE_ALG_AES_CTR #define TEE_ALG_AES_CTR 0x10000210

10.6.1.7 TEE_ALG_AES_CTS #define TEE_ALG_AES_CTS 0x10000310

10.6.1.8 TEE_ALG_AES_ECB_NOPAD #define TEE_ALG_AES_ECB_NOPAD 0x10000010

10.6.1.9 TEE_ALG_AES_GCM #define TEE_ALG_AES_GCM 0x40000810

10.6.1.10 TEE_ALG_AES_XTS #define TEE_ALG_AES_XTS 0x10000410

10.6.1.11 TEE_ALG_DES3_CBC_MAC_NOPAD #define TEE_ALG_DES3_CBC_MAC_NOPAD 0x30000113

10.6.1.12 TEE_ALG_DES3_CBC_MAC_PKCS5 #define TEE_ALG_DES3_CBC_MAC_PKCS5 0x30000513

10.6.1.13 TEE_ALG_DES3_CBC_NOPAD #define TEE_ALG_DES3_CBC_NOPAD 0x10000113

10.6.1.14 TEE_ALG_DES3_ECB_NOPAD `#define TEE_ALG_DES3_ECB_NOPAD 0x10000013`

10.6.1.15 TEE_ALG_DES_CBC_MAC_NOPAD `#define TEE_ALG_DES_CBC_MAC_NOPAD 0x30000111`

10.6.1.16 TEE_ALG_DES_CBC_MAC_PKCS5 `#define TEE_ALG_DES_CBC_MAC_PKCS5 0x30000511`

10.6.1.17 TEE_ALG_DES_CBC_NOPAD `#define TEE_ALG_DES_CBC_NOPAD 0x10000111`

10.6.1.18 TEE_ALG_DES_ECB_NOPAD `#define TEE_ALG_DES_ECB_NOPAD 0x10000011`

10.6.1.19 TEE_ALG_DH_DERIVE_SHARED_SECRET `#define TEE_ALG_DH_DERIVE_SHARED_SECRET 0x80000032`

10.6.1.20 TEE_ALG_DSA_SHA1 `#define TEE_ALG_DSA_SHA1 0x70002131`

10.6.1.21 TEE_ALG_DSA_SHA224 `#define TEE_ALG_DSA_SHA224 0x70003131`

10.6.1.22 TEE_ALG_DSA_SHA256 `#define TEE_ALG_DSA_SHA256 0x70004131`

10.6.1.23 TEE_ALG_ECDH_P192 `#define TEE_ALG_ECDH_P192 0x80001042`

10.6.1.24 TEE_ALG_ECDH_P224 `#define TEE_ALG_ECDH_P224 0x80002042`

10.6.1.25 TEE_ALG_ECDH_P256 #define TEE_ALG_ECDH_P256 0x80003042

10.6.1.26 TEE_ALG_ECDH_P384 #define TEE_ALG_ECDH_P384 0x80004042

10.6.1.27 TEE_ALG_ECDH_P521 #define TEE_ALG_ECDH_P521 0x80005042

10.6.1.28 TEE_ALG_ECDSA_P192 #define TEE_ALG_ECDSA_P192 0x70001041

10.6.1.29 TEE_ALG_ECDSA_P224 #define TEE_ALG_ECDSA_P224 0x70002041

10.6.1.30 TEE_ALG_ECDSA_P256 #define TEE_ALG_ECDSA_P256 0x70003041

10.6.1.31 TEE_ALG_ECDSA_P384 #define TEE_ALG_ECDSA_P384 0x70004041

10.6.1.32 TEE_ALG_ECDSA_P521 #define TEE_ALG_ECDSA_P521 0x70005041

10.6.1.33 TEE_ALG_HMAC_MD5 #define TEE_ALG_HMAC_MD5 0x30000001

10.6.1.34 TEE_ALG_HMAC_SHA1 #define TEE_ALG_HMAC_SHA1 0x30000002

10.6.1.35 TEE_ALG_HMAC_SHA224 #define TEE_ALG_HMAC_SHA224 0x30000003

10.6.1.36 TEE_ALG_HMAC_SHA256 #define TEE_ALG_HMAC_SHA256 0x30000004

10.6.1.37 TEE_ALG_HMAC_SHA384 #define TEE_ALG_HMAC_SHA384 0x30000005

10.6.1.38 TEE_ALG_HMAC_SHA512 #define TEE_ALG_HMAC_SHA512 0x30000006

10.6.1.39 TEE_ALG_MD5 #define TEE_ALG_MD5 0x50000001

10.6.1.40 TEE_ALG_MD5SHA1 #define TEE_ALG_MD5SHA1 0x5000000F

10.6.1.41 TEE_ALG_RSA_NOPAD #define TEE_ALG_RSA_NOPAD 0x60000030

10.6.1.42 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1 0x60210230

10.6.1.43 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_↔
SHA224 0x60310230

10.6.1.44 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_↔
SHA256 0x60410230

10.6.1.45 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_↔
SHA384 0x60510230

10.6.1.46 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_↔
SHA512 0x60610230

10.6.1.47 TEE_ALG_RSAES_PKCS1_V1_5 #define TEE_ALG_RSAES_PKCS1_V1_5 0x60000130

10.6.1.48 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1 0x70212930

10.6.1.49 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_↔
SHA224 0x70313930

10.6.1.50 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_↔
SHA256 0x70414930

10.6.1.51 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_↔
SHA384 0x70515930

10.6.1.52 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_↔
SHA512 0x70616930

10.6.1.53 TEE_ALG_RSASSA_PKCS1_V1_5_MD5 #define TEE_ALG_RSASSA_PKCS1_V1_5_MD5 0x70001830

10.6.1.54 TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1 #define TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1 0x7000↔
F830

10.6.1.55 TEE_ALG_RSASSA_PKCS1_V1_5_SHA1 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA1 0x70002830

10.6.1.56 TEE_ALG_RSASSA_PKCS1_V1_5_SHA224 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA224 0x70003830

10.6.1.57 TEE_ALG_RSASSA_PKCS1_V1_5_SHA256 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA256 0x70004830

10.6.1.58 TEE_ALG_RSASSA_PKCS1_V1_5_SHA384 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA384 0x70005830

10.6.1.59 TEE_ALG_RSASSA_PKCS1_V1_5_SHA512 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA512 0x70006830

10.6.1.60 TEE_ALG_SHA1 #define TEE_ALG_SHA1 0x50000002

10.6.1.61 TEE_ALG_SHA224 #define TEE_ALG_SHA224 0x50000003

10.6.1.62 TEE_ALG_SHA256 #define TEE_ALG_SHA256 0x50000004

10.6.1.63 TEE_ALG_SHA384 #define TEE_ALG_SHA384 0x50000005

10.6.1.64 TEE_ALG_SHA512 #define TEE_ALG_SHA512 0x50000006

10.6.1.65 TEE_ATTR_BIT_PROTECTED #define TEE_ATTR_BIT_PROTECTED (1 << 28)

10.6.1.66 TEE_ATTR_BIT_VALUE #define TEE_ATTR_BIT_VALUE (1 << 29)

10.6.1.67 TEE_ATTR_DH_BASE #define TEE_ATTR_DH_BASE 0xD0001232

10.6.1.68 TEE_ATTR_DH_PRIME #define TEE_ATTR_DH_PRIME 0xD0001032

10.6.1.69 TEE_ATTR_DH_PRIVATE_VALUE #define TEE_ATTR_DH_PRIVATE_VALUE 0xC0000232

10.6.1.70 TEE_ATTR_DH_PUBLIC_VALUE #define TEE_ATTR_DH_PUBLIC_VALUE 0xD0000132

10.6.1.71 TEE_ATTR_DH_SUBPRIME #define TEE_ATTR_DH_SUBPRIME 0xD0001132

10.6.1.72 TEE_ATTR_DH_X_BITS #define TEE_ATTR_DH_X_BITS 0xF0001332

10.6.1.73 TEE_ATTR_DSA_BASE #define TEE_ATTR_DSA_BASE 0xD0001231

10.6.1.74 TEE_ATTR_DSA_PRIME #define TEE_ATTR_DSA_PRIME 0xD0001031

10.6.1.75 TEE_ATTR_DSA_PRIVATE_VALUE #define TEE_ATTR_DSA_PRIVATE_VALUE 0xC0000231

10.6.1.76 TEE_ATTR_DSA_PUBLIC_VALUE #define TEE_ATTR_DSA_PUBLIC_VALUE 0xD0000131

10.6.1.77 TEE_ATTR_DSA_SUBPRIME #define TEE_ATTR_DSA_SUBPRIME 0xD0001131

10.6.1.78 TEE_ATTR_ECC_CURVE #define TEE_ATTR_ECC_CURVE 0xF0000441

10.6.1.79 TEE_ATTR_ECC_PRIVATE_VALUE `#define TEE_ATTR_ECC_PRIVATE_VALUE 0xC0000341`

10.6.1.80 TEE_ATTR_ECC_PUBLIC_VALUE_X `#define TEE_ATTR_ECC_PUBLIC_VALUE_X 0xD0000141`

10.6.1.81 TEE_ATTR_ECC_PUBLIC_VALUE_Y `#define TEE_ATTR_ECC_PUBLIC_VALUE_Y 0xD0000241`

10.6.1.82 TEE_ATTR_RSA_COEFFICIENT `#define TEE_ATTR_RSA_COEFFICIENT 0xC0000830`

10.6.1.83 TEE_ATTR_RSA_EXPONENT1 `#define TEE_ATTR_RSA_EXPONENT1 0xC0000630`

10.6.1.84 TEE_ATTR_RSA_EXPONENT2 `#define TEE_ATTR_RSA_EXPONENT2 0xC0000730`

10.6.1.85 TEE_ATTR_RSA_MODULUS `#define TEE_ATTR_RSA_MODULUS 0xD0000130`

10.6.1.86 TEE_ATTR_RSA_OAEP_LABEL `#define TEE_ATTR_RSA_OAEP_LABEL 0xD0000930`

10.6.1.87 TEE_ATTR_RSA_PRIME1 `#define TEE_ATTR_RSA_PRIME1 0xC0000430`

10.6.1.88 TEE_ATTR_RSA_PRIME2 `#define TEE_ATTR_RSA_PRIME2 0xC0000530`

10.6.1.89 TEE_ATTR_RSA_PRIVATE_EXPONENT `#define TEE_ATTR_RSA_PRIVATE_EXPONENT 0xC0000330`

10.6.1.90 TEE_ATTR_RSA_PSS_SALT_LENGTH #define TEE_ATTR_RSA_PSS_SALT_LENGTH 0xF0000A30

10.6.1.91 TEE_ATTR_RSA_PUBLIC_EXPONENT #define TEE_ATTR_RSA_PUBLIC_EXPONENT 0xD0000230

10.6.1.92 TEE_ATTR_SECRET_VALUE #define TEE_ATTR_SECRET_VALUE 0xC0000000

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

10.6.1.94 TEE_DATA_FLAG_ACCESS_READ #define TEE_DATA_FLAG_ACCESS_READ 0x00000001

10.6.1.95 TEE_DATA_FLAG_ACCESS_WRITE #define TEE_DATA_FLAG_ACCESS_WRITE 0x00000002

10.6.1.96 TEE_DATA_FLAG_ACCESS_WRITE_META #define TEE_DATA_FLAG_ACCESS_WRITE_META 0x00000004

10.6.1.97 TEE_DATA_FLAG_OVERWRITE #define TEE_DATA_FLAG_OVERWRITE 0x00000400

10.6.1.98 TEE_DATA_FLAG_SHARE_READ #define TEE_DATA_FLAG_SHARE_READ 0x00000010

10.6.1.99 TEE_DATA_FLAG_SHARE_WRITE #define TEE_DATA_FLAG_SHARE_WRITE 0x00000020

10.6.1.100 TEE_DATA_MAX_POSITION #define TEE_DATA_MAX_POSITION 0xFFFFFFFF

10.6.1.101 TEE_ECC_CURVE_NIST_P192 #define TEE_ECC_CURVE_NIST_P192 0x00000001

10.6.1.102 TEE_ECC_CURVE_NIST_P224 #define TEE_ECC_CURVE_NIST_P224 0x00000002

10.6.1.103 TEE_ECC_CURVE_NIST_P256 #define TEE_ECC_CURVE_NIST_P256 0x00000003

10.6.1.104 TEE_ECC_CURVE_NIST_P384 #define TEE_ECC_CURVE_NIST_P384 0x00000004

10.6.1.105 TEE_ECC_CURVE_NIST_P521 #define TEE_ECC_CURVE_NIST_P521 0x00000005

10.6.1.106 TEE_ERROR_ACCESS_CONFLICT #define TEE_ERROR_ACCESS_CONFLICT 0xFFFF0003

10.6.1.107 TEE_ERROR_ACCESS_DENIED #define TEE_ERROR_ACCESS_DENIED 0xFFFF0001

10.6.1.108 TEE_ERROR_BAD_FORMAT #define TEE_ERROR_BAD_FORMAT 0xFFFF0005

10.6.1.109 TEE_ERROR_BAD_PARAMETERS #define TEE_ERROR_BAD_PARAMETERS 0xFFFF0006

10.6.1.110 TEE_ERROR_BAD_STATE #define TEE_ERROR_BAD_STATE 0xFFFF0007

10.6.1.111 TEE_ERROR_BUSY #define TEE_ERROR_BUSY 0xFFFF000D

10.6.1.112 TEE_ERROR_CANCEL #define TEE_ERROR_CANCEL 0xFFFF0002

10.6.1.113 TEE_ERROR_COMMUNICATION #define TEE_ERROR_COMMUNICATION 0xFFFF000E

10.6.1.114 TEE_ERROR_CORRUPT_OBJECT #define TEE_ERROR_CORRUPT_OBJECT 0xF0100001

10.6.1.115 TEE_ERROR_CORRUPT_OBJECT_2 #define TEE_ERROR_CORRUPT_OBJECT_2 0xF0100002

10.6.1.116 TEE_ERROR_EXCESS_DATA #define TEE_ERROR_EXCESS_DATA 0xFFFF0004

10.6.1.117 TEE_ERROR_EXTERNAL_CANCEL #define TEE_ERROR_EXTERNAL_CANCEL 0xFFFF0011

10.6.1.118 TEE_ERROR_GENERIC #define TEE_ERROR_GENERIC 0xFFFF0000

10.6.1.119 TEE_ERROR_ITEM_NOT_FOUND #define TEE_ERROR_ITEM_NOT_FOUND 0xFFFF0008

10.6.1.120 TEE_ERROR_MAC_INVALID #define TEE_ERROR_MAC_INVALID 0xFFFF3071

10.6.1.121 TEE_ERROR_NO_DATA #define TEE_ERROR_NO_DATA 0xFFFF000B

10.6.1.122 TEE_ERROR_NOT_IMPLEMENTED #define TEE_ERROR_NOT_IMPLEMENTED 0xFFFF0009

10.6.1.123 TEE_ERROR_NOT_SUPPORTED #define TEE_ERROR_NOT_SUPPORTED 0xFFFF000A

10.6.1.124 TEE_ERROR_OUT_OF_MEMORY #define TEE_ERROR_OUT_OF_MEMORY 0xFFFF000C

10.6.1.125 TEE_ERROR_OVERFLOW #define TEE_ERROR_OVERFLOW 0xFFFF300F

10.6.1.126 TEE_ERROR_SECURITY #define TEE_ERROR_SECURITY 0xFFFF000F

10.6.1.127 TEE_ERROR_SHORT_BUFFER #define TEE_ERROR_SHORT_BUFFER 0xFFFF0010

10.6.1.128 TEE_ERROR_SIGNATURE_INVALID #define TEE_ERROR_SIGNATURE_INVALID 0xFFFF3072

10.6.1.129 TEE_ERROR_STORAGE_NO_SPACE #define TEE_ERROR_STORAGE_NO_SPACE 0xFFFF3041

10.6.1.130 TEE_ERROR_STORAGE_NOT_AVAILABLE #define TEE_ERROR_STORAGE_NOT_AVAILABLE 0x↔
F0100003

10.6.1.131 TEE_ERROR_STORAGE_NOT_AVAILABLE 2 #define TEE_ERROR_STORAGE_NOT_AVAILABLE_2 0x↔
F0100004

10.6.1.132 TEE_ERROR_TARGET_DEAD #define TEE_ERROR_TARGET_DEAD 0xFFFF3024

10.6.1.133 TEE_ERROR_TIME_NEEDS_RESET #define TEE_ERROR_TIME_NEEDS_RESET 0xFFFF5001

10.6.1.134 TEE_ERROR_TIME_NOT_SET #define TEE_ERROR_TIME_NOT_SET 0xFFFF5000

10.6.1.135 TEE_HANDLE_FLAG_EXPECT_TWO_KEYS #define TEE_HANDLE_FLAG_EXPECT_TWO_KEYS 0x00080000

10.6.1.136 TEE_HANDLE_FLAG_INITIALIZED #define TEE_HANDLE_FLAG_INITIALIZED 0x00020000

10.6.1.137 TEE_HANDLE_FLAG_KEY_SET #define TEE_HANDLE_FLAG_KEY_SET 0x00040000

10.6.1.138 TEE_HANDLE_FLAG_PERSISTENT #define TEE_HANDLE_FLAG_PERSISTENT 0x00010000

10.6.1.139 TEE_HANDLE_NULL #define TEE_HANDLE_NULL 0

10.6.1.140 TEE_INT_CORE_API_SPEC_VERSION #define TEE_INT_CORE_API_SPEC_VERSION 0x0000000A

10.6.1.141 TEE_LOGIN_APPLICATION #define TEE_LOGIN_APPLICATION 0x00000004

10.6.1.142 TEE_LOGIN_APPLICATION_GROUP #define TEE_LOGIN_APPLICATION_GROUP 0x00000006

10.6.1.143 TEE_LOGIN_APPLICATION_USER #define TEE_LOGIN_APPLICATION_USER 0x00000005

10.6.1.144 TEE_LOGIN_GROUP #define TEE_LOGIN_GROUP 0x00000002

10.6.1.145 TEE_LOGIN_PUBLIC `#define TEE_LOGIN_PUBLIC 0x00000000`

10.6.1.146 TEE_LOGIN_TRUSTED_APP `#define TEE_LOGIN_TRUSTED_APP 0xF0000000`

10.6.1.147 TEE_LOGIN_USER `#define TEE_LOGIN_USER 0x00000001`

10.6.1.148 TEE_MALLOC_FILL_ZERO `#define TEE_MALLOC_FILL_ZERO 0x00000000`

10.6.1.149 TEE_MEMORY_ACCESS_ANY_OWNER `#define TEE_MEMORY_ACCESS_ANY_OWNER 0x00000004`

10.6.1.150 TEE_MEMORY_ACCESS_READ `#define TEE_MEMORY_ACCESS_READ 0x00000001`

10.6.1.151 TEE_MEMORY_ACCESS_WRITE `#define TEE_MEMORY_ACCESS_WRITE 0x00000002`

10.6.1.152 TEE_NUM_PARAMS `#define TEE_NUM_PARAMS 4`

10.6.1.153 TEE_OBJECT_ID_MAX_LEN `#define TEE_OBJECT_ID_MAX_LEN 64`

10.6.1.154 TEE_OPERATION_AE `#define TEE_OPERATION_AE 4`

10.6.1.155 TEE_OPERATION_ASYMMETRIC_CIPHER `#define TEE_OPERATION_ASYMMETRIC_CIPHER 6`

10.6.1.156 TEE_OPERATION_ASYMMETRIC_SIGNATURE #define TEE_OPERATION_ASYMMETRIC_SIGNATURE 7

10.6.1.157 TEE_OPERATION_CIPHER #define TEE_OPERATION_CIPHER 1

10.6.1.158 TEE_OPERATION_DIGEST #define TEE_OPERATION_DIGEST 5

10.6.1.159 TEE_OPERATION_KEY_DERIVATION #define TEE_OPERATION_KEY_DERIVATION 8

10.6.1.160 TEE_OPERATION_MAC #define TEE_OPERATION_MAC 3

10.6.1.161 TEE_OPERATION_STATE_ACTIVE #define TEE_OPERATION_STATE_ACTIVE 0x00000001

10.6.1.162 TEE_OPERATION_STATE_INITIAL #define TEE_OPERATION_STATE_INITIAL 0x00000000

10.6.1.163 TEE_ORIGIN_API #define TEE_ORIGIN_API 0x00000001

10.6.1.164 TEE_ORIGIN_COMMS #define TEE_ORIGIN_COMMS 0x00000002

10.6.1.165 TEE_ORIGIN_TEE #define TEE_ORIGIN_TEE 0x00000003

10.6.1.166 TEE_ORIGIN_TRUSTED_APP #define TEE_ORIGIN_TRUSTED_APP 0x00000004

10.6.1.167 TEE_PANIC_ID_TA_CLOSESESSIONENTRYPOINT #define TEE_PANIC_ID_TA_CLOSESESSIONENTRYPOINT 0x00000100

10.6.1.168 TEE_PANIC_ID_TA_CREATEENTRYPOINT #define TEE_PANIC_ID_TA_CREATEENTRYPOINT 0x00000102

10.6.1.169 TEE_PANIC_ID_TA_DESTROYENTRYPOINT #define TEE_PANIC_ID_TA_DESTROYENTRYPOINT 0x00000103

10.6.1.170 TEE_PANIC_ID_TA_INVOKECOMMANDENTRYPOINT #define TEE_PANIC_ID_TA_INVOKECOMMANDENTRYPOINT 0x00000104

10.6.1.171 TEE_PANIC_ID_TA_OPENSESSIONENTRYPOINT #define TEE_PANIC_ID_TA_OPENSESSIONENTRYPOINT 0x00000105

10.6.1.172 TEE_PANIC_ID_TEE_AEDECRIPTFINAL #define TEE_PANIC_ID_TEE_AEDECRIPTFINAL 0x00001001

10.6.1.173 TEE_PANIC_ID_TEE_AEENCRYPTFINAL #define TEE_PANIC_ID_TEE_AEENCRYPTFINAL 0x00001002

10.6.1.174 TEE_PANIC_ID_TEE_AEINIT #define TEE_PANIC_ID_TEE_AEINIT 0x00001003

10.6.1.175 TEE_PANIC_ID_TEE_AEUPDATE #define TEE_PANIC_ID_TEE_AEUPDATE 0x00001004

10.6.1.176 TEE_PANIC_ID_TEE_AEUPDATEAAD #define TEE_PANIC_ID_TEE_AEUPDATEAAD 0x00001005

10.6.1.177 TEE_PANIC_ID_TEE_ALLOCATEOPERATION #define TEE_PANIC_ID_TEE_ALLOCATEOPERATION 0x00000106
C01

10.6.1.178 TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR #define TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR 0x00000A01

10.6.1.179 TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERATOR #define TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERATOR 0x00000B01

10.6.1.180 TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT #define TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT 0x00000C01

10.6.1.181 TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT #define TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT 0x00001101

10.6.1.182 TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT #define TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT 0x00001102

10.6.1.183 TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST #define TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST 0x00001103

10.6.1.184 TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST #define TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST 0x00001104

10.6.1.185 TEE_PANIC_ID_TEE_BIGINTADD #define TEE_PANIC_ID_TEE_BIGINTADD 0x00001901

10.6.1.186 TEE_PANIC_ID_TEE_BIGINTADDMOD #define TEE_PANIC_ID_TEE_BIGINTADDMOD 0x00001A01

10.6.1.187 TEE_PANIC_ID_TEE_BIGINTCMP #define TEE_PANIC_ID_TEE_BIGINTCMP 0x00001801

10.6.1.188 TEE_PANIC_ID_TEE_BIGINTCMPS32 #define TEE_PANIC_ID_TEE_BIGINTCMPS32 0x00001802

10.6.1.189 TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD `#define TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD 0xB01`

10.6.1.190 TEE_PANIC_ID_TEE_BIGINTCOMPUTE_FMM `#define TEE_PANIC_ID_TEE_BIGINTCOMPUTE_FMM 0x00001C01`

10.6.1.191 TEE_PANIC_ID_TEE_BIGINTCONVERTFROM_FMM `#define TEE_PANIC_ID_TEE_BIGINTCONVERTFROM_FMM 0x00001C02`

10.6.1.192 TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING `#define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING 0x00001701`

10.6.1.193 TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32 `#define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32 0x00001702`

10.6.1.194 TEE_PANIC_ID_TEE_BIGINTCONVERTTO_FMM `#define TEE_PANIC_ID_TEE_BIGINTCONVERTTO_FMM 0x00001C03`

10.6.1.195 TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING `#define TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING 0x00001703`

10.6.1.196 TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32 `#define TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32 0x00001704`

10.6.1.197 TEE_PANIC_ID_TEE_BIGINTDIV `#define TEE_PANIC_ID_TEE_BIGINTDIV 0x00001902`

10.6.1.198 TEE_PANIC_ID_TEE_BIGINTFMMCONTEXT_SIZE_IN_U32 `#define TEE_PANIC_ID_TEE_BIGINTFMMCONTEXT_SIZE_IN_U32 0x00001903`

10.6.1.199 TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32 #define TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32 0x00001502

10.6.1.200 TEE_PANIC_ID_TEE_BIGINTGETBIT #define TEE_PANIC_ID_TEE_BIGINTGETBIT 0x00001803

10.6.1.201 TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT #define TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT 0x00001804

10.6.1.202 TEE_PANIC_ID_TEE_BIGINTINIT #define TEE_PANIC_ID_TEE_BIGINTINIT 0x00001601

10.6.1.203 TEE_PANIC_ID_TEE_BIGINTINITFMM #define TEE_PANIC_ID_TEE_BIGINTINITFMM 0x00001602

10.6.1.204 TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT #define TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT 0x00001603

10.6.1.205 TEE_PANIC_ID_TEE_BIGINTINVMOD #define TEE_PANIC_ID_TEE_BIGINTINVMOD 0x00001A02

10.6.1.206 TEE_PANIC_ID_TEE_BIGINTISPROBABLEPRIME #define TEE_PANIC_ID_TEE_BIGINTISPROBABLEPRIME 0x00001A02
B02

10.6.1.207 TEE_PANIC_ID_TEE_BIGINTMOD #define TEE_PANIC_ID_TEE_BIGINTMOD 0x00001A03

10.6.1.208 TEE_PANIC_ID_TEE_BIGINTMUL #define TEE_PANIC_ID_TEE_BIGINTMUL 0x00001903

10.6.1.209 TEE_PANIC_ID_TEE_BIGINTMULMOD #define TEE_PANIC_ID_TEE_BIGINTMULMOD 0x00001A04

10.6.1.210 TEE_PANIC_ID_TEE_BIGINTNEG #define TEE_PANIC_ID_TEE_BIGINTNEG 0x00001904

10.6.1.211 TEE_PANIC_ID_TEE_BIGINTRELATIVEPRIME #define TEE_PANIC_ID_TEE_BIGINTRELATIVEPRIME 0x00001↔
B03

10.6.1.212 TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT #define TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT 0x00001805

10.6.1.213 TEE_PANIC_ID_TEE_BIGINTSQUARE #define TEE_PANIC_ID_TEE_BIGINTSQUARE 0x00001905

10.6.1.214 TEE_PANIC_ID_TEE_BIGINTSQUAREMOD #define TEE_PANIC_ID_TEE_BIGINTSQUAREMOD 0x00001↔
A05

10.6.1.215 TEE_PANIC_ID_TEE_BIGINTSUB #define TEE_PANIC_ID_TEE_BIGINTSUB 0x00001906

10.6.1.216 TEE_PANIC_ID_TEE_BIGINTSUBMOD #define TEE_PANIC_ID_TEE_BIGINTSUBMOD 0x00001A06

10.6.1.217 TEE_PANIC_ID_TEE_CHECKMEMORYACCESSRIGHTS #define TEE_PANIC_ID_TEE_CHECKMEMORYACCESSRIGHTS 0x0

10.6.1.218 TEE_PANIC_ID_TEE_CIPHERDOFINAL #define TEE_PANIC_ID_TEE_CIPHERDOFINAL 0x00000E01

10.6.1.219 TEE_PANIC_ID_TEE_CIPHERINIT #define TEE_PANIC_ID_TEE_CIPHERINIT 0x00000E02

10.6.1.220 TEE_PANIC_ID_TEE_CIPHERUPDATE #define TEE_PANIC_ID_TEE_CIPHERUPDATE 0x00000E03

10.6.1.221 TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT #define TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT 0x00000901

10.6.1.222 TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT1 #define TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT1 0x00000905

10.6.1.223 TEE_PANIC_ID_TEE_CLOSEOBJECT #define TEE_PANIC_ID_TEE_CLOSEOBJECT 0x00000701

10.6.1.224 TEE_PANIC_ID_TEE_CLOSETASESSION #define TEE_PANIC_ID_TEE_CLOSETASESSION 0x00000401

10.6.1.225 TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES #define TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES 0x00000802

10.6.1.226 TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES1 #define TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES1 0x00000803

10.6.1.227 TEE_PANIC_ID_TEE_COPYOPERATION #define TEE_PANIC_ID_TEE_COPYOPERATION 0x00000C02

10.6.1.228 TEE_PANIC_ID_TEE_CREATEPERSISTENTOBJECT #define TEE_PANIC_ID_TEE_CREATEPERSISTENTOBJECT 0x00000D00

10.6.1.229 TEE_PANIC_ID_TEE_DERIVEKEY #define TEE_PANIC_ID_TEE_DERIVEKEY 0x00001201

10.6.1.230 TEE_PANIC_ID_TEE_DIGESTDOFINAL #define TEE_PANIC_ID_TEE_DIGESTDOFINAL 0x00000D01

10.6.1.231 TEE_PANIC_ID_TEE_DIGESTUPDATE #define TEE_PANIC_ID_TEE_DIGESTUPDATE 0x00000D02

10.6.1.232 TEE_PANIC_ID_TEE_FREE #define TEE_PANIC_ID_TEE_FREE 0x00000602

10.6.1.233 TEE_PANIC_ID_TEE_FREEOPERATION #define TEE_PANIC_ID_TEE_FREEOPERATION 0x00000C03

10.6.1.234 TEE_PANIC_ID_TEE_FREEPERSISTENTOBJECTENUMERATOR #define TEE_PANIC_ID_TEE_FREEPERSISTENTOBJECTENUMERATOR 0x00000A02

10.6.1.235 TEE_PANIC_ID_TEE_FREEPROPERTYENUMERATOR #define TEE_PANIC_ID_TEE_FREEPROPERTYENUMERATOR 0x00000804

10.6.1.236 TEE_PANIC_ID_TEE_FREETRANSIENTOBJECT #define TEE_PANIC_ID_TEE_FREETRANSIENTOBJECT 0x00000803

10.6.1.237 TEE_PANIC_ID_TEE_GENERATEKEY #define TEE_PANIC_ID_TEE_GENERATEKEY 0x00000804

10.6.1.238 TEE_PANIC_ID_TEE_GENERATERANDOM #define TEE_PANIC_ID_TEE_GENERATERANDOM 0x00001301

10.6.1.239 TEE_PANIC_ID_TEE_GETCANCELLATIONFLAG #define TEE_PANIC_ID_TEE_GETCANCELLATIONFLAG 0x00000501

10.6.1.240 TEE_PANIC_ID_TEE_GETINSTANCEDATA #define TEE_PANIC_ID_TEE_GETINSTANCEDATA 0x00000603

10.6.1.241 TEE_PANIC_ID_TEE_GETNEXTPERSISTENTOBJECT #define TEE_PANIC_ID_TEE_GETNEXTPERSISTENTOBJECT 0x00000A03

10.6.1.242 TEE_PANIC_ID_TEE_GETNEXTPROPERTY #define TEE_PANIC_ID_TEE_GETNEXTPROPERTY 0x00000203

- 10.6.1.243 TEE_PANIC_ID_TEE_GETOBJECTBUFFERATTRIBUTE** #define TEE_PANIC_ID_TEE_GETOBJECTBUFFERATTRIBUTE 0x00000702
- 10.6.1.244 TEE_PANIC_ID_TEE_GETOBJECTINFO** #define TEE_PANIC_ID_TEE_GETOBJECTINFO 0x00000703
- 10.6.1.245 TEE_PANIC_ID_TEE_GETOBJECTINFO1** #define TEE_PANIC_ID_TEE_GETOBJECTINFO1 0x00000706
- 10.6.1.246 TEE_PANIC_ID_TEE_GETOBJECTVALUEATTRIBUTE** #define TEE_PANIC_ID_TEE_GETOBJECTVALUEATTRIBUTE 0x00000707
- 10.6.1.247 TEE_PANIC_ID_TEE_GETOPERATIONINFO** #define TEE_PANIC_ID_TEE_GETOPERATIONINFO 0x00000C04↵
C04
- 10.6.1.248 TEE_PANIC_ID_TEE_GETOPERATIONINFOMULTIPLE** #define TEE_PANIC_ID_TEE_GETOPERATIONINFOMULTIPLE 0x00000C08↵
C08
- 10.6.1.249 TEE_PANIC_ID_TEE_GETPROPERTYASBINARYBLOCK** #define TEE_PANIC_ID_TEE_GETPROPERTYASBINARYBLOCK 0x00000C09↵
C09
- 10.6.1.250 TEE_PANIC_ID_TEE_GETPROPERTYASBOOL** #define TEE_PANIC_ID_TEE_GETPROPERTYASBOOL 0x00000205
- 10.6.1.251 TEE_PANIC_ID_TEE_GETPROPERTYASIDENTITY** #define TEE_PANIC_ID_TEE_GETPROPERTYASIDENTITY 0x00000206
- 10.6.1.252 TEE_PANIC_ID_TEE_GETPROPERTYASSTRING** #define TEE_PANIC_ID_TEE_GETPROPERTYASSTRING 0x00000207
- 10.6.1.253 TEE_PANIC_ID_TEE_GETPROPERTYASU32** #define TEE_PANIC_ID_TEE_GETPROPERTYASU32 0x00000208
-

- 10.6.1.254 TEE_PANIC_ID_TEE_GETPROPERTYASUUID** #define TEE_PANIC_ID_TEE_GETPROPERTYASUUID 0x00000209
- 10.6.1.255 TEE_PANIC_ID_TEE_GETPROPERTYNAME** #define TEE_PANIC_ID_TEE_GETPROPERTYNAME 0x0000020A
- 10.6.1.256 TEE_PANIC_ID_TEE_GETREETIME** #define TEE_PANIC_ID_TEE_GETREETIME 0x00001401
- 10.6.1.257 TEE_PANIC_ID_TEE_GETSYSTEMTIME** #define TEE_PANIC_ID_TEE_GETSYSTEMTIME 0x00001402
- 10.6.1.258 TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME** #define TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME 0x00001403
- 10.6.1.259 TEE_PANIC_ID_TEE_INITREFATTRIBUTE** #define TEE_PANIC_ID_TEE_INITREFATTRIBUTE 0x00000805
- 10.6.1.260 TEE_PANIC_ID_TEE_INITVALUEATTRIBUTE** #define TEE_PANIC_ID_TEE_INITVALUEATTRIBUTE 0x00000806
- 10.6.1.261 TEE_PANIC_ID_TEE_INVOKETACOMMAND** #define TEE_PANIC_ID_TEE_INVOKETACOMMAND 0x00000402
- 10.6.1.262 TEE_PANIC_ID_TEE_MACCOMPAREFINAL** #define TEE_PANIC_ID_TEE_MACCOMPAREFINAL 0x00000↔
F01
- 10.6.1.263 TEE_PANIC_ID_TEE_MACCOMPUTEFINAL** #define TEE_PANIC_ID_TEE_MACCOMPUTEFINAL 0x00000↔
F02
- 10.6.1.264 TEE_PANIC_ID_TEE_MACINIT** #define TEE_PANIC_ID_TEE_MACINIT 0x00000F03

10.6.1.265 TEE_PANIC_ID_TEE_MACUPDATE #define TEE_PANIC_ID_TEE_MACUPDATE 0x00000F04

10.6.1.266 TEE_PANIC_ID_TEE_MALLOC #define TEE_PANIC_ID_TEE_MALLOC 0x00000604

10.6.1.267 TEE_PANIC_ID_TEE_MASKANCELLATION #define TEE_PANIC_ID_TEE_MASKANCELLATION 0x00000502

10.6.1.268 TEE_PANIC_ID_TEE_MEMCOMPARE #define TEE_PANIC_ID_TEE_MEMCOMPARE 0x00000605

10.6.1.269 TEE_PANIC_ID_TEE_MEMFILL #define TEE_PANIC_ID_TEE_MEMFILL 0x00000606

10.6.1.270 TEE_PANIC_ID_TEE_MEMMOVE #define TEE_PANIC_ID_TEE_MEMMOVE 0x00000607

10.6.1.271 TEE_PANIC_ID_TEE_OPENPERSISTENTOBJECT #define TEE_PANIC_ID_TEE_OPENPERSISTENTOBJECT 0x00000903

10.6.1.272 TEE_PANIC_ID_TEE_OPENTASESSION #define TEE_PANIC_ID_TEE_OPENTASESSION 0x00000403

10.6.1.273 TEE_PANIC_ID_TEE_PANIC #define TEE_PANIC_ID_TEE_PANIC 0x00000301

10.6.1.274 TEE_PANIC_ID_TEE_POPULATETRANSIENTOBJECT #define TEE_PANIC_ID_TEE_POPULATETRANSIENTOBJECT 0x00000000

10.6.1.275 TEE_PANIC_ID_TEE_READOBJECTDATA #define TEE_PANIC_ID_TEE_READOBJECTDATA 0x000000↔
B01

10.6.1.276 TEE_PANIC_ID_TEE_REALLOC #define TEE_PANIC_ID_TEE_REALLOC 0x00000608

10.6.1.277 TEE_PANIC_ID_TEE_RENAMEPERSISTENTOBJECT #define TEE_PANIC_ID_TEE_RENAMEPERSISTENTOBJECT 0x0000

10.6.1.278 TEE_PANIC_ID_TEE_RESETOPERATION #define TEE_PANIC_ID_TEE_RESETOPERATION 0x00000↵
C05

10.6.1.279 TEE_PANIC_ID_TEE_RESETPERSISTENTOBJECTENUMERATOR #define TEE_PANIC_ID_TEE_↵
RESETPERSISTENTOBJECTENUMERATOR 0x00000A04

10.6.1.280 TEE_PANIC_ID_TEE_RESETPROPERTYENUMERATOR #define TEE_PANIC_ID_TEE_RESETPROPERTYENUMERATOR 0x0

10.6.1.281 TEE_PANIC_ID_TEE_RESETTRANSIENTOBJECT #define TEE_PANIC_ID_TEE_RESETTRANSIENTOBJECT 0x00000808

10.6.1.282 TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE #define TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE 0x00000705

10.6.1.283 TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE1 #define TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE1 0x00000707

10.6.1.284 TEE_PANIC_ID_TEE_SEEKOBJECTDATA #define TEE_PANIC_ID_TEE_SEEKOBJECTDATA 0x00000↵
B02

10.6.1.285 TEE_PANIC_ID_TEE_SETINSTANCEDATA #define TEE_PANIC_ID_TEE_SETINSTANCEDATA 0x00000609

10.6.1.286 TEE_PANIC_ID_TEE_SETOPERATIONKEY #define TEE_PANIC_ID_TEE_SETOPERATIONKEY 0x00000↵
C06

10.6.1.287 TEE_PANIC_ID_TEE_SETOPERATIONKEY2 #define TEE_PANIC_ID_TEE_SETOPERATIONKEY2 0x00000↵
C07

10.6.1.288 TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME #define TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME 0x00001404

10.6.1.289 TEE_PANIC_ID_TEE_STARTPERSISTENTOBJECTENUMERATOR #define TEE_PANIC_ID_TEE_↵
STARTPERSISTENTOBJECTENUMERATOR 0x00000A05

10.6.1.290 TEE_PANIC_ID_TEE_STARTPROPERTYENUMERATOR #define TEE_PANIC_ID_TEE_STARTPROPERTYENUMERATOR 0x0

10.6.1.291 TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA #define TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA 0x00000↵
B03

10.6.1.292 TEE_PANIC_ID_TEE_UNMASKCANCELLATION #define TEE_PANIC_ID_TEE_UNMASKCANCELLATION 0x00000503

10.6.1.293 TEE_PANIC_ID_TEE_WAIT #define TEE_PANIC_ID_TEE_WAIT 0x00001405

10.6.1.294 TEE_PANIC_ID_TEE_WRITEOBJECTDATA #define TEE_PANIC_ID_TEE_WRITEOBJECTDATA 0x00000↵
B04

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

10.6.1.296 TEE_PARAM_TYPE_MEMREF_INOUT #define TEE_PARAM_TYPE_MEMREF_INOUT 7

10.6.1.297 TEE_PARAM_TYPE_MEMREF_INPUT `#define TEE_PARAM_TYPE_MEMREF_INPUT 5`

10.6.1.298 TEE_PARAM_TYPE_MEMREF_OUTPUT `#define TEE_PARAM_TYPE_MEMREF_OUTPUT 6`

10.6.1.299 TEE_PARAM_TYPE_NONE `#define TEE_PARAM_TYPE_NONE 0`

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

10.6.1.301 TEE_PARAM_TYPE_VALUE_INOUT `#define TEE_PARAM_TYPE_VALUE_INOUT 3`

10.6.1.302 TEE_PARAM_TYPE_VALUE_INPUT `#define TEE_PARAM_TYPE_VALUE_INPUT 1`

10.6.1.303 TEE_PARAM_TYPE_VALUE_OUTPUT `#define TEE_PARAM_TYPE_VALUE_OUTPUT 2`

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

10.6.1.305 TEE_PROPSET_CURRENT_CLIENT `#define TEE_PROPSET_CURRENT_CLIENT (TEE_PropSetHandle) 0x↔
FFFFFFFF`

10.6.1.306 TEE_PROPSET_CURRENT_TA `#define TEE_PROPSET_CURRENT_TA (TEE_PropSetHandle) 0x↔
FFFFFFFF`

10.6.1.307 TEE_PROPSET_TEE_IMPLEMENTATION #define TEE_PROPSET_TEE_IMPLEMENTATION (TEE_PropSetHandle) 0x←
FFFFFFFFD

10.6.1.308 TEE_STORAGE_PRIVATE #define TEE_STORAGE_PRIVATE 0x00000001

10.6.1.309 TEE_SUCCESS #define TEE_SUCCESS 0x00000000

10.6.1.310 TEE_TIMEOUT_INFINITE #define TEE_TIMEOUT_INFINITE 0xFFFFFFFF

10.6.1.311 TEE_TYPE_AES #define TEE_TYPE_AES 0xA0000010

10.6.1.312 TEE_TYPE_CORRUPTED_OBJECT #define TEE_TYPE_CORRUPTED_OBJECT 0xA00000BE

10.6.1.313 TEE_TYPE_DATA #define TEE_TYPE_DATA 0xA00000BF

10.6.1.314 TEE_TYPE_DES #define TEE_TYPE_DES 0xA0000011

10.6.1.315 TEE_TYPE_DES3 #define TEE_TYPE_DES3 0xA0000013

10.6.1.316 TEE_TYPE_DH_KEYPAIR #define TEE_TYPE_DH_KEYPAIR 0xA1000032

10.6.1.317 TEE_TYPE_DSA_KEYPAIR #define TEE_TYPE_DSA_KEYPAIR 0xA1000031

10.6.1.318 TEE_TYPE_DSA_PUBLIC_KEY #define TEE_TYPE_DSA_PUBLIC_KEY 0xA0000031

10.6.1.319 TEE_TYPE_ECDH_KEYPAIR #define TEE_TYPE_ECDH_KEYPAIR 0xA1000042

10.6.1.320 TEE_TYPE_ECDH_PUBLIC_KEY #define TEE_TYPE_ECDH_PUBLIC_KEY 0xA0000042

10.6.1.321 TEE_TYPE_ECDSA_KEYPAIR #define TEE_TYPE_ECDSA_KEYPAIR 0xA1000041

10.6.1.322 TEE_TYPE_ECDSA_PUBLIC_KEY #define TEE_TYPE_ECDSA_PUBLIC_KEY 0xA0000041

10.6.1.323 TEE_TYPE_GENERIC_SECRET #define TEE_TYPE_GENERIC_SECRET 0xA0000000

10.6.1.324 TEE_TYPE_HMAC_MD5 #define TEE_TYPE_HMAC_MD5 0xA0000001

10.6.1.325 TEE_TYPE_HMAC_SHA1 #define TEE_TYPE_HMAC_SHA1 0xA0000002

10.6.1.326 TEE_TYPE_HMAC_SHA224 #define TEE_TYPE_HMAC_SHA224 0xA0000003

10.6.1.327 TEE_TYPE_HMAC_SHA256 #define TEE_TYPE_HMAC_SHA256 0xA0000004

10.6.1.328 TEE_TYPE_HMAC_SHA384 #define TEE_TYPE_HMAC_SHA384 0xA0000005

10.6.1.329 TEE_TYPE_HMAC_SHA512 #define TEE_TYPE_HMAC_SHA512 0xA0000006

10.6.1.330 TEE_TYPE_RSA_KEYPAIR #define TEE_TYPE_RSA_KEYPAIR 0xA1000030

10.6.1.331 TEE_TYPE_RSA_PUBLIC_KEY #define TEE_TYPE_RSA_PUBLIC_KEY 0xA0000030

10.6.1.332 TEE_USAGE_DECRYPT #define TEE_USAGE_DECRYPT 0x00000004

10.6.1.333 TEE_USAGE_DERIVE #define TEE_USAGE_DERIVE 0x00000040

10.6.1.334 TEE_USAGE_ENCRYPT #define TEE_USAGE_ENCRYPT 0x00000002

10.6.1.335 TEE_USAGE_EXTRACTABLE #define TEE_USAGE_EXTRACTABLE 0x00000001

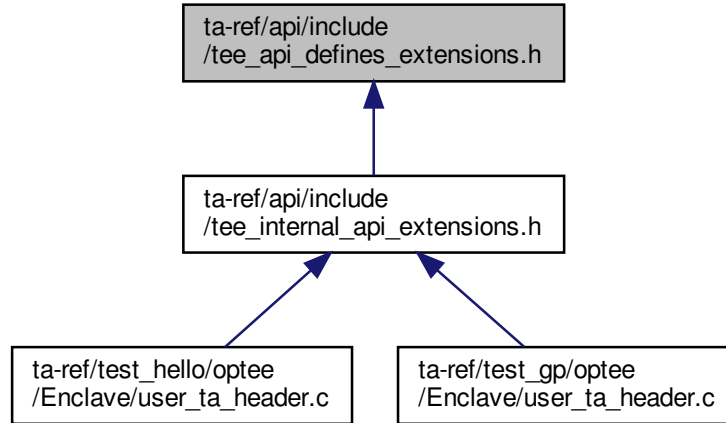
10.6.1.336 TEE_USAGE_MAC #define TEE_USAGE_MAC 0x00000008

10.6.1.337 TEE_USAGE_SIGN #define TEE_USAGE_SIGN 0x00000010

10.6.1.338 TEE_USAGE_VERIFY #define TEE_USAGE_VERIFY 0x00000020

10.7 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

10.7.1 Macro Definition Documentation

10.7.1.1 TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY `#define TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY 0x800020C1`

10.7.1.2 TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY `#define TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY 0x800030C1`

10.7.1.3 TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY `#define TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY 0x800040C1`

10.7.1.4 TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY `#define TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY 0x800050C1`

10.7.1.5 TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY `#define TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY 0x800060C1`

10.7.1.6 TEE_ALG_HKDF_MD5_DERIVE_KEY `#define TEE_ALG_HKDF_MD5_DERIVE_KEY 0x800010C0`

10.7.1.7 TEE_ALG_HKDF_SHA1_DERIVE_KEY `#define TEE_ALG_HKDF_SHA1_DERIVE_KEY 0x800020C0`

10.7.1.8 TEE_ALG_HKDF_SHA224_DERIVE_KEY `#define TEE_ALG_HKDF_SHA224_DERIVE_KEY 0x800030C0`

10.7.1.9 TEE_ALG_HKDF_SHA256_DERIVE_KEY `#define TEE_ALG_HKDF_SHA256_DERIVE_KEY 0x800040C0`

10.7.1.10 TEE_ALG_HKDF_SHA384_DERIVE_KEY #define TEE_ALG_HKDF_SHA384_DERIVE_KEY 0x800050C0

10.7.1.11 TEE_ALG_HKDF_SHA512_DERIVE_KEY #define TEE_ALG_HKDF_SHA512_DERIVE_KEY 0x800060C0

10.7.1.12 TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY #define TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY 0x800020C2

10.7.1.13 TEE_ATTR_CONCAT_KDF_DKM_LENGTH #define TEE_ATTR_CONCAT_KDF_DKM_LENGTH 0xF00003C1

10.7.1.14 TEE_ATTR_CONCAT_KDF_OTHER_INFO #define TEE_ATTR_CONCAT_KDF_OTHER_INFO 0xD00002C1

10.7.1.15 TEE_ATTR_CONCAT_KDF_Z #define TEE_ATTR_CONCAT_KDF_Z 0xC00001C1

10.7.1.16 TEE_ATTR_HKDF_IKM #define TEE_ATTR_HKDF_IKM 0xC00001C0

10.7.1.17 TEE_ATTR_HKDF_INFO #define TEE_ATTR_HKDF_INFO 0xD00003C0

10.7.1.18 TEE_ATTR_HKDF_OKM_LENGTH #define TEE_ATTR_HKDF_OKM_LENGTH 0xF00004C0

10.7.1.19 TEE_ATTR_HKDF_SALT #define TEE_ATTR_HKDF_SALT 0xD00002C0

10.7.1.20 TEE_ATTR_PBKDF2_DKM_LENGTH #define TEE_ATTR_PBKDF2_DKM_LENGTH 0xF00004C2

10.7.1.21 TEE_ATTR_PBKDF2_ITERATION_COUNT #define TEE_ATTR_PBKDF2_ITERATION_COUNT 0x↵
F00003C2

10.7.1.22 TEE_ATTR_PBKDF2_PASSWORD #define TEE_ATTR_PBKDF2_PASSWORD 0xC00001C2

10.7.1.23 TEE_ATTR_PBKDF2_SALT #define TEE_ATTR_PBKDF2_SALT 0xD00002C2

10.7.1.24 TEE_MEMORY_ACCESS_NONSECURE #define TEE_MEMORY_ACCESS_NONSECURE 0x10000000

10.7.1.25 TEE_MEMORY_ACCESS_SECURE #define TEE_MEMORY_ACCESS_SECURE 0x20000000

10.7.1.26 TEE_STORAGE_PRIVATE_REE #define TEE_STORAGE_PRIVATE_REE 0x80000000

10.7.1.27 TEE_STORAGE_PRIVATE_RPMB #define TEE_STORAGE_PRIVATE_RPMB 0x80000100

10.7.1.28 TEE_STORAGE_PRIVATE_SQL_RESERVED #define TEE_STORAGE_PRIVATE_SQL_RESERVED 0x80000200

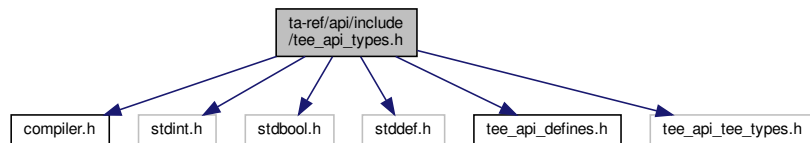
10.7.1.29 TEE_TYPE_CONCAT_KDF_Z #define TEE_TYPE_CONCAT_KDF_Z 0xA10000C1

10.7.1.30 TEE_TYPE_HKDF_IKM #define TEE_TYPE_HKDF_IKM 0xA10000C0

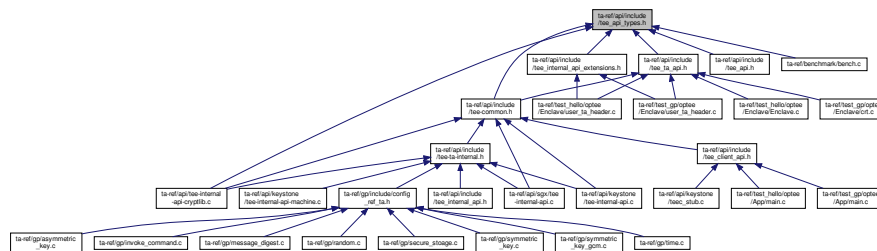
10.7.1.31 TEE_TYPE_PBKDF2_PASSWORD #define TEE_TYPE_PBKDF2_PASSWORD 0xA10000C2

10.8 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_tee_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_SEReaderProperties](#)
- 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 nfdst`
- `typedef uint32_t 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 }`

10.8.1 Macro Definition Documentation

10.8.1.1 DMREQ_FINISH `#define DMREQ_FINISH 0`

10.8.1.2 DMREQ_WRITE `#define DMREQ_WRITE 1`

10.8.1.3 TEE_MEM_INPUT `#define TEE_MEM_INPUT 0x00000001`

10.8.1.4 TEE_MEM_OUTPUT `#define TEE_MEM_OUTPUT 0x00000002`

10.8.1.5 TEE_MEMREF_0_USED `#define TEE_MEMREF_0_USED 0x00000001`

10.8.1.6 TEE_MEMREF_1_USED `#define TEE_MEMREF_1_USED 0x00000002`

10.8.1.7 TEE_MEMREF_2_USED `#define TEE_MEMREF_2_USED 0x00000004`

10.8.1.8 TEE_MEMREF_3_USED `#define TEE_MEMREF_3_USED 0x00000008`

10.8.1.9 TEE_SE_READER_NAME_MAX `#define TEE_SE_READER_NAME_MAX 20`

10.8.2 Typedef Documentation

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

10.8.2.2 nfds_t `typedef unsigned long int nfds_t`

10.8.2.3 socklen_t `typedef uint32_t socklen_t`

10.8.2.4 TEE_BigInt `typedef uint32_t TEE_BigInt`

10.8.2.5 TEE.BigIntFMM `typedef uint32_t TEE.BigIntFMM`

10.8.2.6 TEE.ErrorOrigin `typedef uint32_t TEE.ErrorOrigin`

10.8.2.7 TEE.ObjectEnumHandle `typedef struct __TEE_ObjectEnumHandle* TEE.ObjectEnumHandle`

10.8.2.8 TEE.ObjectHandle `typedef struct __TEE_ObjectHandle* TEE.ObjectHandle`

10.8.2.9 TEE.ObjectType `typedef uint32_t TEE.ObjectType`

10.8.2.10 TEE.OperationHandle `typedef struct __TEE_OperationHandle* TEE.OperationHandle`

10.8.2.11 TEE.PropSetHandle `typedef struct __TEE_PropSetHandle* TEE_PropSetHandle`

10.8.2.12 TEE.Result `typedef uint32_t TEE.Result`

10.8.2.13 TEE.SEChannelHandle `typedef struct __TEE_SEChannelHandle* TEE_SEChannelHandle`

10.8.2.14 TEE.SEReaderHandle `typedef struct __TEE_SEReaderHandle* TEE.SEReaderHandle`

10.8.2.15 TEE.SEServiceHandle `typedef struct __TEE_SEServiceHandle* TEE.SEServiceHandle`

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

10.8.2.17 TEE_Session typedef void* [TEE_Session](#)

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

10.8.3 Enumeration Type Documentation

10.8.3.1 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	

10.8.3.2 TEE_Whence enum [TEE_Whence](#)

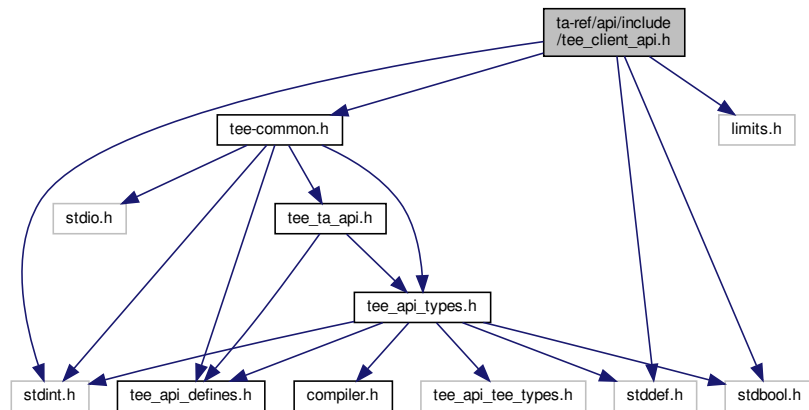
Enumerator

TEE_DATA_SEEK_SET	
TEE_DATA_SEEK_CUR	
TEE_DATA_SEEK_END	

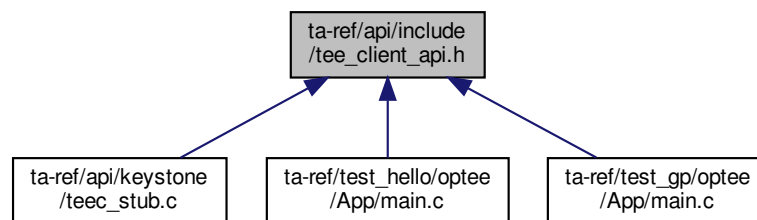
10.9 ta-ref/api/include/tee_client_api.h File Reference

```
#include <stdint.h>
#include <stddef.h>
#include <stdbool.h>
#include <limits.h>
#include "tee-common.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`)

10.9.1 Macro Definition Documentation

10.9.1.1 TEEC_CONFIG_PAYLOAD_REF_COUNT `#define TEEC_CONFIG_PAYLOAD_REF_COUNT 4`

10.9.1.2 TEEC_CONFIG_SHAREDMEM_MAX_SIZE `#define TEEC_CONFIG_SHAREDMEM_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.

10.9.1.3 TEEC_ERROR_ACCESS_CONFLICT `#define TEEC_ERROR_ACCESS_CONFLICT 0xFFFF0003`

10.9.1.4 TEEC_ERROR_ACCESS_DENIED `#define TEEC_ERROR_ACCESS_DENIED 0xFFFF0001`

10.9.1.5 TEEC_ERROR_BAD_FORMAT `#define TEEC_ERROR_BAD_FORMAT 0xFFFF0005`

10.9.1.6 TEEC_ERROR_BAD_PARAMETERS `#define TEEC_ERROR_BAD_PARAMETERS 0xFFFF0006`

10.9.1.7 TEEC_ERROR_BAD_STATE `#define TEEC_ERROR_BAD_STATE 0xFFFF0007`

10.9.1.8 TEEC_ERROR_BUSY #define TEEC_ERROR_BUSY 0xFFFF000D

10.9.1.9 TEEC_ERROR_CANCEL #define TEEC_ERROR_CANCEL 0xFFFF0002

10.9.1.10 TEEC_ERROR_COMMUNICATION #define TEEC_ERROR_COMMUNICATION 0xFFFF000E

10.9.1.11 TEEC_ERROR_EXCESS_DATA #define TEEC_ERROR_EXCESS_DATA 0xFFFF0004

10.9.1.12 TEEC_ERROR_EXTERNAL_CANCEL #define TEEC_ERROR_EXTERNAL_CANCEL 0xFFFF0011

10.9.1.13 TEEC_ERROR_GENERIC #define TEEC_ERROR_GENERIC 0xFFFF0000

10.9.1.14 TEEC_ERROR_ITEM_NOT_FOUND #define TEEC_ERROR_ITEM_NOT_FOUND 0xFFFF0008

10.9.1.15 TEEC_ERROR_NO_DATA #define TEEC_ERROR_NO_DATA 0xFFFF000B

10.9.1.16 TEEC_ERROR_NOT_IMPLEMENTED #define TEEC_ERROR_NOT_IMPLEMENTED 0xFFFF0009

10.9.1.17 TEEC_ERROR_NOT_SUPPORTED #define TEEC_ERROR_NOT_SUPPORTED 0xFFFF000A

10.9.1.18 TEEC_ERROR_OUT_OF_MEMORY #define TEEC_ERROR_OUT_OF_MEMORY 0xFFFF000C

10.9.1.19 TEEC_ERROR_SECURITY `#define TEEC_ERROR_SECURITY 0xFFFF000F`

10.9.1.20 TEEC_ERROR_SHORT_BUFFER `#define TEEC_ERROR_SHORT_BUFFER 0xFFFF0010`

10.9.1.21 TEEC_ERROR_TARGET_DEAD `#define TEEC_ERROR_TARGET_DEAD 0xFFFF3024`

10.9.1.22 TEEC_LOGIN_APPLICATION `#define TEEC_LOGIN_APPLICATION 0x00000004`

10.9.1.23 TEEC_LOGIN_GROUP `#define TEEC_LOGIN_GROUP 0x00000002`

10.9.1.24 TEEC_LOGIN_GROUP_APPLICATION `#define TEEC_LOGIN_GROUP_APPLICATION 0x00000006`

10.9.1.25 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.

10.9.1.26 TEEC_LOGIN_USER `#define TEEC_LOGIN_USER 0x00000001`

10.9.1.27 TEEC_LOGIN_USER_APPLICATION `#define TEEC_LOGIN_USER_APPLICATION 0x00000005`

10.9.1.28 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.

10.9.1.29 TEEC_MEM_OUTPUT `#define TEEC_MEM_OUTPUT 0x00000002`**10.9.1.30 TEEC_MEMREF_PARTIAL_INOUT** `#define TEEC_MEMREF_PARTIAL_INOUT 0x0000000F`**10.9.1.31 TEEC_MEMREF_PARTIAL_INPUT** `#define TEEC_MEMREF_PARTIAL_INPUT 0x0000000D`**10.9.1.32 TEEC_MEMREF_PARTIAL_OUTPUT** `#define TEEC_MEMREF_PARTIAL_OUTPUT 0x0000000E`**10.9.1.33 TEEC_MEMREF_TEMP_INOUT** `#define TEEC_MEMREF_TEMP_INOUT 0x00000007`**10.9.1.34 TEEC_MEMREF_TEMP_INPUT** `#define TEEC_MEMREF_TEMP_INPUT 0x00000005`**10.9.1.35 TEEC_MEMREF_TEMP_OUTPUT** `#define TEEC_MEMREF_TEMP_OUTPUT 0x00000006`**10.9.1.36 TEEC_MEMREF_WHOLE** `#define TEEC_MEMREF_WHOLE 0x0000000C`

10.9.1.37 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.

10.9.1.38 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.

10.9.1.39 TEEC.ORIGIN_COMMS `#define TEEC_ORIGIN_COMMS 0x00000002`

10.9.1.40 TEEC.ORIGIN_TEE `#define TEEC_ORIGIN_TEE 0x00000003`

10.9.1.41 TEEC.ORIGIN_TRUSTED_APP `#define TEEC_ORIGIN_TRUSTED_APP 0x00000004`

10.9.1.42 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-th parameter to get the type for.

10.9.1.43 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.

10.9.1.44 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.

10.9.1.45 TEEC_VALUE_INOUT `#define TEEC_VALUE_INOUT 0x00000003`

10.9.1.46 TEEC_VALUE_INPUT `#define TEEC_VALUE_INPUT 0x00000001`

10.9.1.47 TEEC_VALUE_OUTPUT `#define TEEC_VALUE_OUTPUT 0x00000002`

10.9.2 Typedef Documentation

10.9.2.1 TEEC_Result `typedef uint32_t TEEC_Result`

10.9.3 Function Documentation

10.9.3.1 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.

10.9.3.2 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.
----------------	------------------------------

10.9.3.3 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.
----------------	------------------------------

10.9.3.4 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.

10.9.3.5 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.

10.9.3.6 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.

10.9.3.7 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.

10.9.3.8 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.
------------------	---

10.9.3.9 TEEC_RequestCancellation() `void TEEC_RequestCancellation (`
`TEEC_Operation * operation)`

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

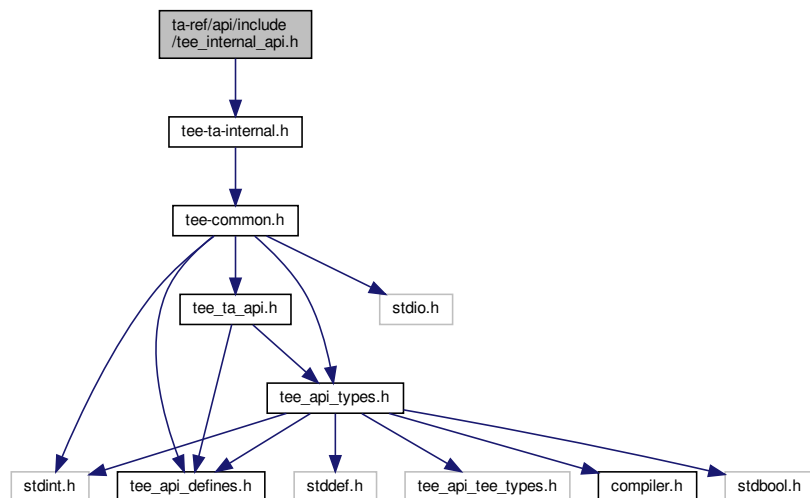
Parameters

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

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

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

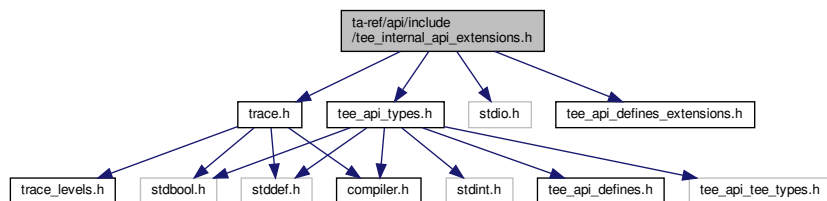
Include dependency graph for tee_internal_api.h:



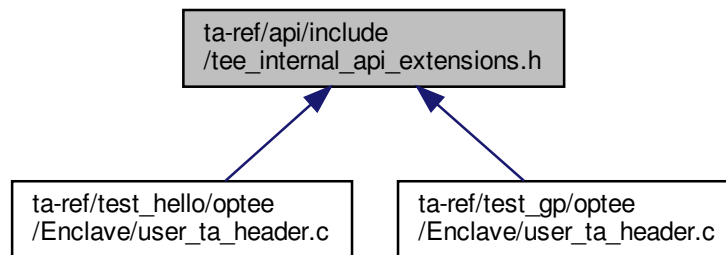
10.11 ta-ref/api/include/tee_internal_api_extensions.h File Reference

```
#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:



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



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)

10.11.1 Macro Definition Documentation

10.11.1.1 TEE_USER_MEM_HINT_NO_FILL_ZERO `#define TEE_USER_MEM_HINT_NO_FILL_ZERO 0x80000000`

10.11.2 Function Documentation

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

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

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

10.11.2.4 tee_map_zi() void* tee_map_zi (
 size_t *len*,
 uint32_t *flags*)

10.11.2.5 tee_unmap() `TEE_Result` tee_unmap (
 void * *buf*,
 size_t *len*)

10.11.2.6 tee_user_mem_check_heap() size_t tee_user_mem_check_heap (
 void)

10.11.2.7 tee_user_mem_mark_heap() void tee_user_mem_mark_heap (
 void)

10.11.2.8 tee_uuid_from_str() `TEE_Result` tee_uuid_from_str (
 `TEE_UUID` * *uuid*,
 const char * *s*)

10.12.2 Function Documentation

10.12.2.1 TA_CloseSessionEntryPoint() `void TA_EXPORT TA_CloseSessionEntryPoint (void * sessionContext)`

10.12.2.2 TA_CreateEntryPoint() `TEE_Result TA_EXPORT TA_CreateEntryPoint (void)`

[TA_CreateEntryPoint\(\)](#) - Trusted application creates the entry point.

TA_CreateEntryPoint function is the Trusted Application's constructor, which the framework calls when it creates a new instance of the Trusted Application.

Returns

TEE_SUCCESS If success, else error occurred.

[TA_CreateEntryPoint\(\)](#) - The function creates the entry point of TA(Trusted Application).

This function is to be called when the instance of the TA is created. This is the first call in the TA and the displayed message should be "has been called".

Returns

TEE_SUCCESS If the command is successfully executed, else error occurred.

10.12.2.3 TA_DestroyEntryPoint() `void TA_EXPORT TA_DestroyEntryPoint (void)`

[TA_DestroyEntryPoint\(\)](#) - The function TA_DestroyEntryPoint is the Trusted Application's destructor, which the Framework calls when the instance is being destroyed.

[TA_DestroyEntryPoint\(\)](#) - Destroy entry point with TA.

This function is to be called, when the instance of the TA is destroyed. This is the last call in the TA and the displayed message should be "has been called".

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

```

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

```

10.13 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

10.13.1 Variable Documentation

10.13.1.1 `_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
}

```

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

10.13.1.3 `_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
}

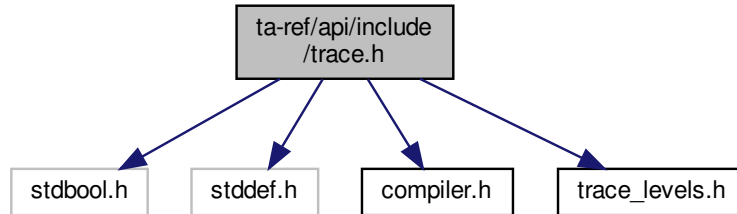
```

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

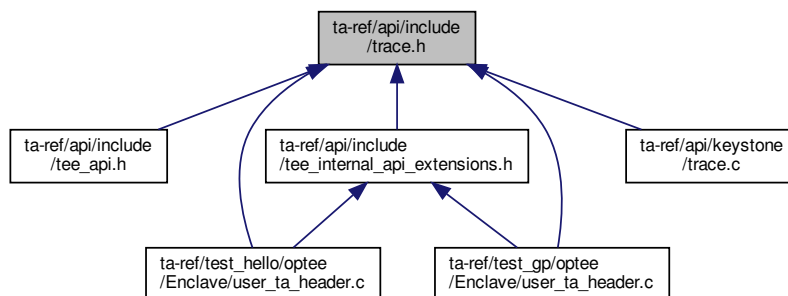
10.14 ta-ref/api/include/trace.h File Reference

```
#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` []

10.14.1 Macro Definition Documentation

10.14.1.1 DHEXDUMP #define DHEXDUMP (
 buf,
 len)

Value:

```
dhex_dump(__func__, __LINE__, TRACE_DEBUG, \
buf, len)
```

10.14.1.2 DMSG #define DMSG (
 ...) `trace_printf_helper` (TRACE_DEBUG, true, __VA_ARGS__)

10.14.1.3 DMSG_RAW #define DMSG_RAW (
 ...) `trace_printf_helper_raw` (TRACE_DEBUG, true, __VA_ARGS__)

10.14.1.4 DPRINT_STACK `#define DPRINT_STACK() (void)0`

10.14.1.5 EMSG `#define EMSG(
...) trace_printf_helper(TRACE_ERROR, true, __VA_ARGS__)`

10.14.1.6 EMSG_RAW `#define EMSG_RAW(
...) trace_printf_helper_raw(TRACE_ERROR, true, __VA_ARGS__)`

10.14.1.7 EPRINT_STACK `#define EPRINT_STACK() (void)0`

10.14.1.8 FMSG `#define FMSG(
...) trace_printf_helper(TRACE_FLOW, true, __VA_ARGS__)`

10.14.1.9 FMSG_RAW `#define FMSG_RAW(
...) trace_printf_helper_raw(TRACE_FLOW, true, __VA_ARGS__)`

10.14.1.10 FPRINT_STACK `#define FPRINT_STACK() (void)0`

10.14.1.11 IMSG `#define IMSG(
...) trace_printf_helper(TRACE_INFO, true, __VA_ARGS__)`

10.14.1.12 IMSG_RAW `#define IMSG_RAW(
...) trace_printf_helper_raw(TRACE_INFO, true, __VA_ARGS__)`

10.14.1.13 INMSG `#define INMSG(
...) FMSG("> " __VA_ARGS__)`

10.14.1.14 IPRINT_STACK `#define IPRINT_STACK() (void)0`

10.14.1.15 MAX_FUNC_PRINT_SIZE `#define MAX_FUNC_PRINT_SIZE 32`

10.14.1.16 MAX_PRINT_SIZE `#define MAX_PRINT_SIZE 256`

10.14.1.17 MSG `#define MSG(
...) (void)0`

10.14.1.18 MSG_RAW `#define MSG_RAW(
...) (void)0`

10.14.1.19 OUTMSG `#define OUTMSG(
...) FMSG("< " __VA_ARGS__)`

10.14.1.20 OUTRMSG `#define OUTRMSG(
r)`

Value:

```
do {  
    OUTMSG("r=[%x]", r);  
    return r;  
} while (0)
```

10.14.1.21 SMSG `#define SMSG(
...) (void)0`

10.14.1.22 TRACE_LEVEL `#define TRACE_LEVEL TRACE_MAX`

10.14.1.23 trace_printf_helper #define trace_printf_helper(
 level,
 level_ok,
 ...)

Value:

```
trace_printf(__func__, __LINE__, (level), (level_ok), \  
    __VA_ARGS__)
```

10.14.1.24 trace_printf_helper_raw #define trace_printf_helper_raw(
 level,
 level_ok,
 ...) trace_printf(NULL, 0, (level), (level_ok), __VA_ARGS__)

10.14.2 Function Documentation

10.14.2.1 dhex_dump() void dhex_dump (
 const char * *function*,
 int *line*,
 int *level*,
 const void * *buf*,
 int *len*)

10.14.2.2 trace_ext_get_thread_id() int trace_ext_get_thread_id (
 void)

10.14.2.3 trace_ext_puts() void trace_ext_puts (
 const char * *str*)

10.14.2.4 trace_get_level() int trace_get_level (
 void)

10.14.2.5 trace_printf() void trace_printf (
 const char * *func*,
 int *line*,
 int *level*,
 bool *level_ok*,
 const char * *fmt*,
 ...)

10.14.2.6 trace_set_level() `void trace_set_level (`
`int level)`

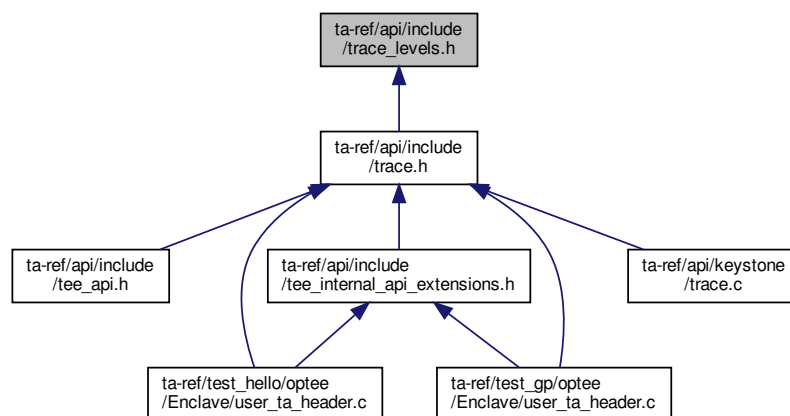
10.14.3 Variable Documentation

10.14.3.1 trace_ext_prefix `const char trace_ext_prefix[]` [extern]

10.14.3.2 trace_level `int trace_level` [extern]

10.15 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`

10.15.1 Macro Definition Documentation

10.15.1.1 TRACE_DEBUG `#define TRACE_DEBUG 3`

10.15.1.2 TRACE_ERROR `#define TRACE_ERROR TRACE_MIN`

10.15.1.3 TRACE_FLOW `#define TRACE_FLOW 4`

10.15.1.4 TRACE_INFO `#define TRACE_INFO 2`

10.15.1.5 TRACE_MAX `#define TRACE_MAX TRACE_FLOW`

10.15.1.6 TRACE_MIN `#define TRACE_MIN 1`

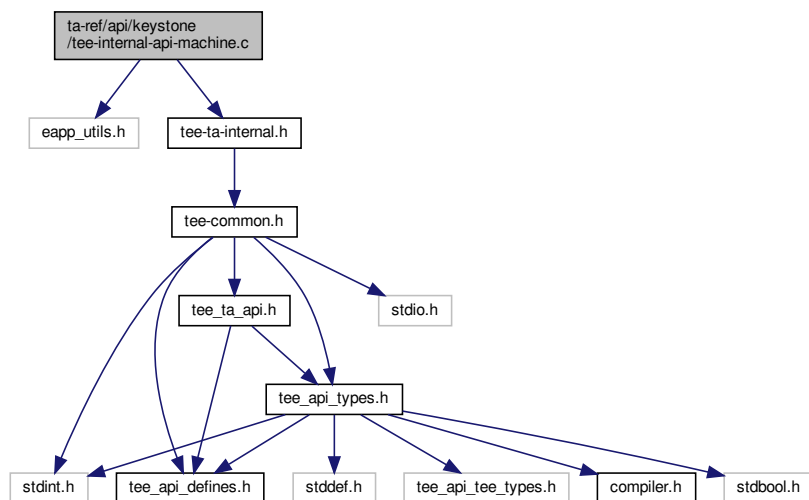
10.15.1.7 TRACE_PRINTF_LEVEL `#define TRACE_PRINTF_LEVEL TRACE_ERROR`

10.16 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))`

10.16.1 Function Documentation

10.16.1.1 `__attribute__((noreturn))` 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

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

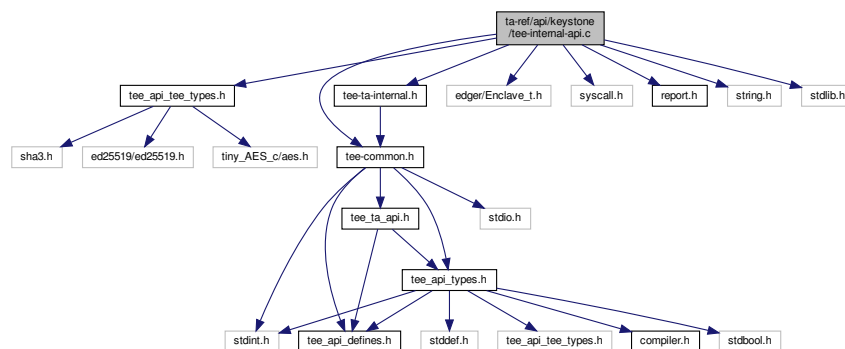
Returns

panic code will be returned.

10.17 ta-ref/api/keystone/tee-internal-api.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.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.

10.17.1 Macro Definition Documentation

10.17.1.1 FPERMS `#define FPERMS 0600`

10.17.1.2 O_CREAT `#define O_CREAT 00100`

10.17.1.3 O_EXCL `#define O_EXCL 00200`

10.17.1.4 O_RDONLY `#define O_RDONLY 0`

10.17.1.5 O_RDWR `#define O_RDWR 00002`

10.17.1.6 O_TRUNC `#define O_TRUNC 01000`

10.17.1.7 O_WRONLY `#define O_WRONLY 00001`

10.17.2 Function Documentation

10.17.2.1 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.

10.17.2.2 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

10.17.2.3 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

10.17.2.4 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.

```

10.17.2.5 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.

10.17.2.6 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.

10.17.2.7 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.

10.17.2.8 TEE_Free() `void TEE_Free (`
`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.
---------------	--

10.17.2.9 TEE_GenerateRandom() `void TEE_GenerateRandom (`
`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

10.17.2.10 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.

10.17.2.11 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
-------------	--

10.17.2.12 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
-------------	--

10.17.2.13 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.

10.17.2.14 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.

10.17.2.15 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.

10.17.2.16 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.

10.17.2.17 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

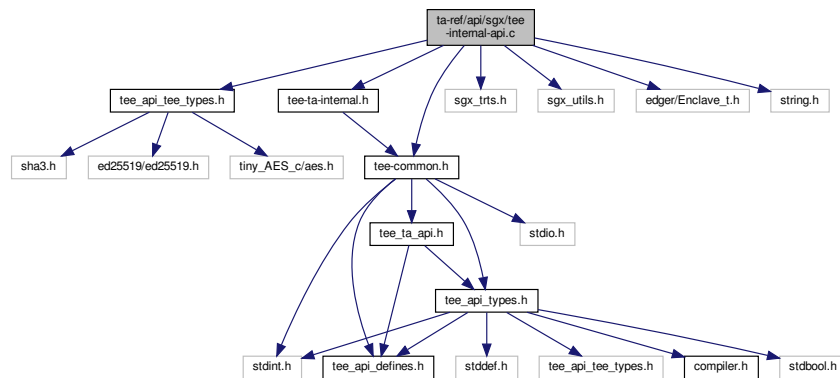
Returns

TEE_SUCCESS if success else error occurred.

10.18 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.

10.18.1 Macro Definition Documentation

10.18.1.1 FPERMS `#define FPERMS 0600`

10.18.1.2 O_CREAT `#define O_CREAT 00100`

10.18.1.3 O_EXCL `#define O_EXCL 00200`

10.18.1.4 O_RDONLY `#define O_RDONLY 0`

10.18.1.5 O_RDWR `#define O_RDWR 00002`

10.18.1.6 O_TRUNC `#define O_TRUNC 01000`

10.18.1.7 O_WRONLY `#define O_WRONLY 00001`

10.18.2 Function Documentation

10.18.2.1 __attribute__((noreturn)) `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.
-----------	--

10.18.2.2 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.

10.18.2.3 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

10.18.2.4 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.

10.18.2.5 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.
<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.

10.18.2.6 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.

10.18.2.7 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.

10.18.2.8 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.

10.18.2.9 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

10.18.2.10 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.

10.18.2.11 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.
-------------	---

10.18.2.12 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.
-------------	---

10.18.2.13 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.

10.18.2.14 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.

10.18.2.15 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

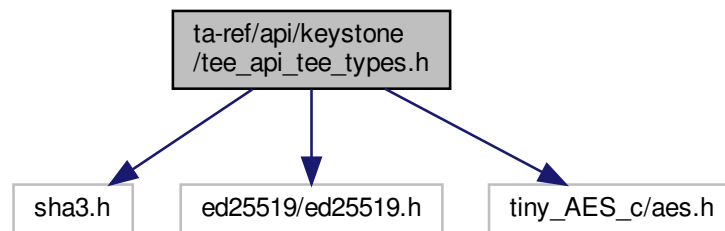
<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write

Returns

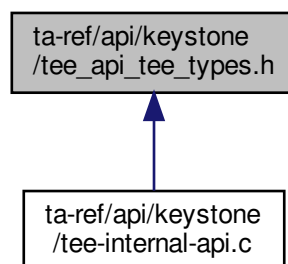
TEE_SUCCESS if success else error occurred.

10.19 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

10.19.1 Macro Definition Documentation

10.19.1.1 AES256 #define AES256 1

10.19.1.2 MBEDCRYPT #define MBEDCRYPT 1

10.19.1.3 SHA_LENGTH #define SHA_LENGTH (256/8)

10.19.1.4 TEE_OBJECT_AAD_SIZE #define TEE_OBJECT_AAD_SIZE 16

10.19.1.5 TEE_OBJECT_KEY_SIZE #define TEE_OBJECT_KEY_SIZE 32

10.19.1.6 TEE_OBJECT_NONCE_SIZE #define TEE_OBJECT_NONCE_SIZE 16

10.19.1.7 TEE_OBJECT_SKEY_SIZE #define TEE_OBJECT_SKEY_SIZE 64

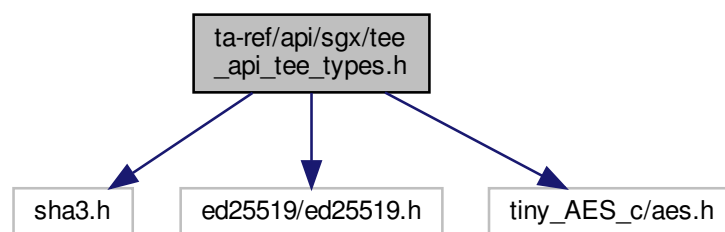
10.19.1.8 TEE_OBJECT_TAG_SIZE #define TEE_OBJECT_TAG_SIZE 16

10.19.1.9 WOLFCRYPT `#define WOLFCRYPT 2`

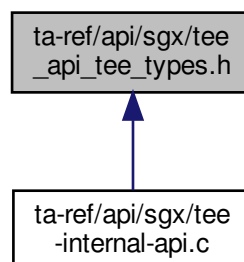
10.20 ta-ref/api/optee/tee_api_tee_types.h File Reference

10.21 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

10.21.1 Macro Definition Documentation

10.21.1.1 AES256 #define AES256 1

10.21.1.2 MBEDCRYPT #define MBEDCRYPT 1

10.21.1.3 SHA_LENGTH #define SHA_LENGTH (256/8)

10.21.1.4 TEE_HANDLE_NULL #define TEE_HANDLE_NULL 0

10.21.1.5 TEE_OBJECT_AAD_SIZE #define TEE_OBJECT_AAD_SIZE 16

10.21.1.6 TEE_OBJECT_KEY_SIZE #define TEE_OBJECT_KEY_SIZE 32

10.21.1.7 TEE_OBJECT_NONCE_SIZE #define TEE_OBJECT_NONCE_SIZE 16

10.21.1.8 TEE_OBJECT_KEY_SIZE `#define TEE_OBJECT_KEY_SIZE 64`

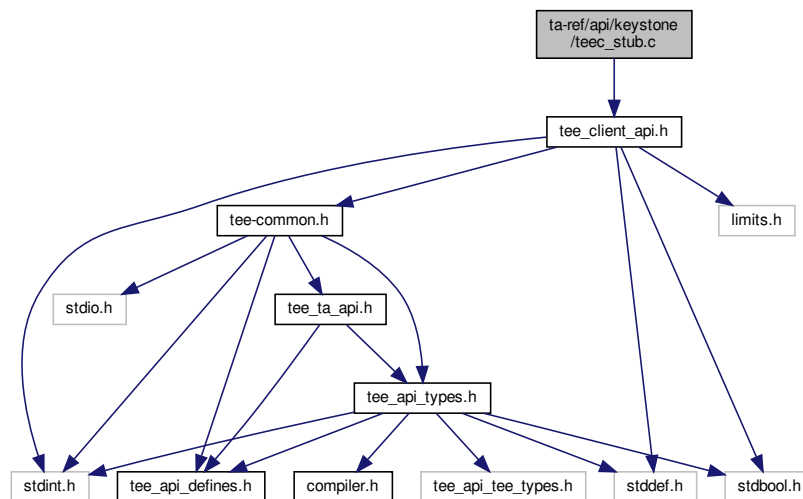
10.21.1.9 TEE_OBJECT_TAG_SIZE `#define TEE_OBJECT_TAG_SIZE 16`

10.21.1.10 WOLFCRYPT `#define WOLFCRYPT 2`

10.22 ta-ref/api/keystone/teeec_stub.c File Reference

```
#include <tee-client-api.h>
```

Include dependency graph for teeec_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)

10.22.1 Function Documentation

10.22.1.1 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.

10.22.1.2 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.
----------------	------------------------------

10.22.1.3 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.
----------------	------------------------------

10.22.1.4 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.

10.22.1.5 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.
<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.

10.22.1.6 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.

10.22.1.7 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.
------------------	---

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

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

Parameters

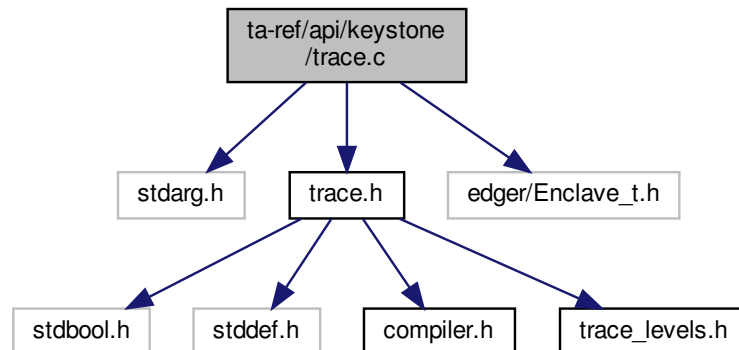
<i>operation</i>	Pointer to an operation previously passed to open session or invoke.
------------------	--

10.23 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,...)

10.23.1 Function Documentation

10.23.1.1 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.

```
10.23.1.2 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

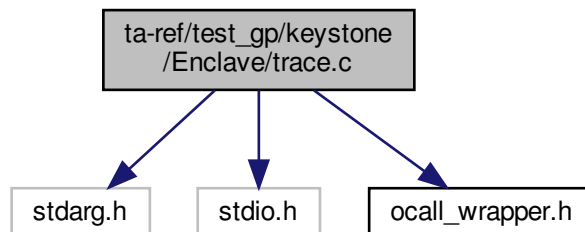
<i>func</i>	Pointer to a buffer where the resulting C-string is stored.
<i>line</i>	integer type of line
<i>level_ok</i>	boolean value
<i>fmt</i>	C string that contains a format string
<i>ap</i>	A value identifying a variable arguments list

Returns

buf The total number of characters written is returned.

10.24 ta-ref/test_gp/keystone/Enclave/trace.c File Reference

```
#include <stdarg.h>
#include <stdio.h>
#include "ocall_wrapper.h"
Include dependency graph for trace.c:
```



Functions

- static unsigned int `_strlen` (const char *str)
- int `tee_printf` (const char *fmt,...)

10.24.1 Function Documentation

10.24.1.1 `_strlen()` static unsigned int `_strlen` (
const char * *str*) [inline], [static]

`_strlen()` - calculate the length of characters in str.

Parameters

<i>str</i>	str is argument of type pointer.
------------	----------------------------------

Returns

string string length.

10.24.1.2 `tee_printf()` int `tee_printf` (
const char * *fmt*,
...)

`tee_printf()` - For trace GP API.

Initializes ap variable. Formats data under control of the format control string and stores the result in buf and ends the processing of ap. Finally prints the buffer value.

Parameters

<i>fmt</i>	fmt is constant character argument of type pointer.
------------	---

Returns

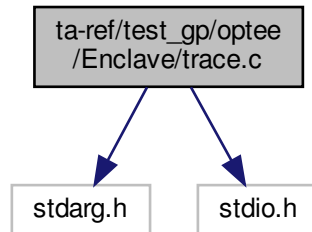
res Based on the condition check it will return string length else returns 0.

10.25 ta-ref/test_gp/optee/Enclave/trace.c File Reference

```
#include <stdarg.h>
```

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

Include dependency graph for trace.c:



Functions

- int [tee_printf](#) (const char *fmt,...)

10.25.1 Function Documentation

10.25.1.1 tee_printf() `int tee_printf (`
 `const char * fmt,`
 `...)`

[tee_printf\(\)](#) - Printing the formatted output in to a character array.

In this function the "@param ap" variable is initialized by calling `va_start()` and then formatted data will send to a string using argument list by calling [vsnprintf\(\)](#) and finally the string length will be stored in `res`.

Parameters

<i>fmt</i>	A string that specifies the format of the output.
------------	---

Returns

result If success, else error occurred.

[tee_printf\(\)](#) - For trace GP API.

Initializes `ap` variable. Formats data under control of the format control string and stores the result in `buf` and ends the processing of `ap`. Finally prints the buffer value.

Parameters

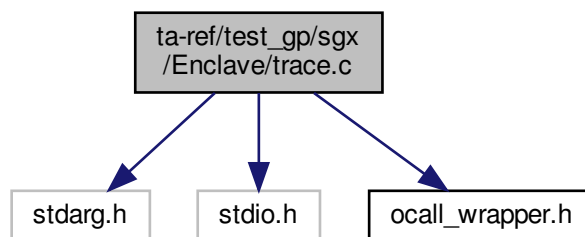
<i>fmt</i>	fmt is constant character argument of type pointer.
------------	---

Returns

res Based on the condition check it will return string length else returns 0.

10.26 ta-ref/test_gp/sgx/Enclave/trace.c File Reference

```
#include <stdarg.h>
#include <stdio.h>
#include "ocall_wrapper.h"
Include dependency graph for trace.c:
```

**Functions**

- static unsigned int [_strlen](#) (const char *str)
- int [tee_printf](#) (const char *fmt,...)

10.26.1 Function Documentation

10.26.1.1 [_strlen\(\)](#) static unsigned int [_strlen](#) (
const char * *str*) [inline], [static]

[_strlen\(\)](#) - calculate the length of characters in a str.

Parameters

<i>str</i>	str is an argument of type pointer.
------------	-------------------------------------

Returns

string length on success.

10.26.1.2 tee_printf() `int tee_printf (`
 `const char * fmt,`
 `...)`

[tee_printf\(\)](#) - For tracing GP API.

Initializes ap variable. Formats data under control of the format control string and stores the result in buf and ends the processing of ap. Finally print the buffer value.

Parameters

<i>fmt</i>	fmt is a constant character argument of type pointer.
------------	---

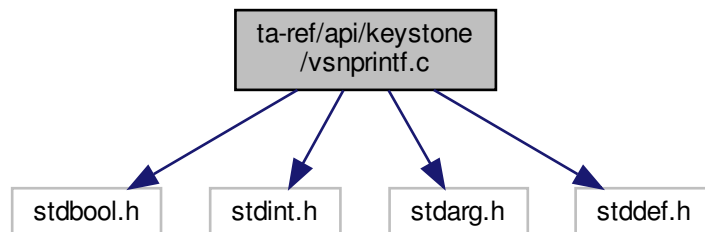
Returns

buffer If successfully executed, else error occurred.

10.27 ta-ref/api/keystone/vsnprintf.c File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include <stdarg.h>
#include <stddef.h>
```

Include dependency graph for vsnprintf.c:



Classes

- struct [out_fct_wrap_type](#)

Macros

- #define [PRINTF_NTOA_BUFFER_SIZE](#) 32U
- #define [PRINTF_FTOA_BUFFER_SIZE](#) 32U
- #define [PRINTF_SUPPORT_FLOAT](#)
- #define [PRINTF_SUPPORT_LONG_LONG](#)
- #define [PRINTF_SUPPORT_PTRDIFF_T](#)
- #define [FLAGS_ZEROPAD](#) (1U << 0U)
- #define [FLAGS_LEFT](#) (1U << 1U)
- #define [FLAGS_PLUS](#) (1U << 2U)
- #define [FLAGS_SPACE](#) (1U << 3U)
- #define [FLAGS_HASH](#) (1U << 4U)
- #define [FLAGS_UPPERCASE](#) (1U << 5U)
- #define [FLAGS_CHAR](#) (1U << 6U)
- #define [FLAGS_SHORT](#) (1U << 7U)
- #define [FLAGS_LONG](#) (1U << 8U)
- #define [FLAGS_LONG_LONG](#) (1U << 9U)
- #define [FLAGS_PRECISION](#) (1U << 10U)
- #define [_putchar](#) putchar

Typedefs

- typedef void(* [out_fct_type](#)) (char character, void *buffer, size_t idx, size_t maxlen)

Functions

- int [putchar](#) (char ch)
- static void [_out_buffer](#) (char character, void *buffer, size_t idx, size_t maxlen)
- static void [_out_null](#) (char character, void *buffer, size_t idx, size_t maxlen)
- static void [_out_char](#) (char character, void *buffer, size_t idx, size_t maxlen)
- static void [_out_fct](#) (char character, void *buffer, size_t idx, size_t maxlen)
- static unsigned int [_strlen](#) (const char *str)
- static bool [_is_digit](#) (char ch)
- static unsigned int [_atoi](#) (const char **str)
- static size_t [_ntoa_format](#) ([out_fct_type](#) out, char *buffer, size_t idx, size_t maxlen, char *buf, size_t len, bool negative, unsigned int base, unsigned int prec, unsigned int width, unsigned int flags)
- static size_t [_ntoa_long](#) ([out_fct_type](#) out, char *buffer, size_t idx, size_t maxlen, unsigned long value, bool negative, unsigned long base, unsigned int prec, unsigned int width, unsigned int flags)
- static size_t [_ntoa_long_long](#) ([out_fct_type](#) out, char *buffer, size_t idx, size_t maxlen, unsigned long long value, bool negative, unsigned long long base, unsigned int prec, unsigned int width, unsigned int flags)
- static size_t [_ftoa](#) ([out_fct_type](#) out, char *buffer, size_t idx, size_t maxlen, double value, unsigned int prec, unsigned int width, unsigned int flags)
- static int [_vsnprintf](#) ([out_fct_type](#) out, char *buffer, const size_t maxlen, const char *format, va_list va)
- int [sprintf](#) (char *buffer, const char *format,...)
- int [snprintf](#) (char *buffer, size_t count, const char *format,...)
- int [vsnprintf](#) (char *buffer, size_t count, const char *format, va_list va)
- int [fctprintf](#) (void(*out)(char character, void *arg), void *arg, const char *format,...)

10.27.1 Macro Definition Documentation

10.27.1.1 `_putchar` `#define _putchar putchar`

10.27.1.2 `FLAGS.CHAR` `#define FLAGS_CHAR (1U << 6U)`

10.27.1.3 `FLAGS.HASH` `#define FLAGS_HASH (1U << 4U)`

10.27.1.4 `FLAGS.LEFT` `#define FLAGS_LEFT (1U << 1U)`

10.27.1.5 `FLAGS.LONG` `#define FLAGS_LONG (1U << 8U)`

10.27.1.6 `FLAGS.LONG_LONG` `#define FLAGS_LONG_LONG (1U << 9U)`

10.27.1.7 `FLAGS.PLUS` `#define FLAGS_PLUS (1U << 2U)`

10.27.1.8 `FLAGS.PRECISION` `#define FLAGS_PRECISION (1U << 10U)`

10.27.1.9 `FLAGS.SHORT` `#define FLAGS_SHORT (1U << 7U)`

10.27.1.10 `FLAGS.SPACE` `#define FLAGS_SPACE (1U << 3U)`

10.27.1.11 **FLAGS_UPPERCASE** `#define FLAGS_UPPERCASE (1U << 5U)`

10.27.1.12 **FLAGS_ZEROPAD** `#define FLAGS_ZEROPAD (1U << 0U)`

10.27.1.13 **PRINTF_FTOA_BUFFER_SIZE** `#define PRINTF_FTOA_BUFFER_SIZE 32U`

10.27.1.14 **PRINTF_NTOA_BUFFER_SIZE** `#define PRINTF_NTOA_BUFFER_SIZE 32U`

10.27.1.15 **PRINTF_SUPPORT_FLOAT** `#define PRINTF_SUPPORT_FLOAT`

10.27.1.16 **PRINTF_SUPPORT_LONG_LONG** `#define PRINTF_SUPPORT_LONG_LONG`

10.27.1.17 **PRINTF_SUPPORT_PTRDIFF_T** `#define PRINTF_SUPPORT_PTRDIFF_T`

10.27.2 Typedef Documentation

10.27.2.1 **out_fct_type** `typedef void(* out_fct_type) (char character, void *buffer, size_t idx, size_t maxlen)`

10.27.3 Function Documentation

10.27.3.1 **_atoi()** `static unsigned int _atoi (const char ** str) [static]`

10.27.3.2 `_ftoa()` `static size_t _ftoa (`
 `out.fct_type out,`
 `char * buffer,`
 `size_t idx,`
 `size_t maxlen,`
 `double value,`
 `unsigned int prec,`
 `unsigned int width,`
 `unsigned int flags) [static]`

10.27.3.3 `_is_digit()` `static bool _is_digit (`
 `char ch) [inline], [static]`

10.27.3.4 `_ntoa_format()` `static size_t _ntoa_format (`
 `out.fct_type out,`
 `char * buffer,`
 `size_t idx,`
 `size_t maxlen,`
 `char * buf,`
 `size_t len,`
 `bool negative,`
 `unsigned int base,`
 `unsigned int prec,`
 `unsigned int width,`
 `unsigned int flags) [static]`

10.27.3.5 `_ntoa_long()` `static size_t _ntoa_long (`
 `out.fct_type out,`
 `char * buffer,`
 `size_t idx,`
 `size_t maxlen,`
 `unsigned long value,`
 `bool negative,`
 `unsigned long base,`
 `unsigned int prec,`
 `unsigned int width,`
 `unsigned int flags) [static]`

10.27.3.6 `_ntoa_long_long()` `static size_t _ntoa_long_long (`
 `out.fct_type out,`
 `char * buffer,`
 `size_t idx,`
 `size_t maxlen,`
 `unsigned long long value,`
 `bool negative,`
 `unsigned long long base,`
 `unsigned int prec,`
 `unsigned int width,`
 `unsigned int flags) [static]`

10.27.3.7 `_out_buffer()` `static void _out_buffer (`
 `char character,`
 `void * buffer,`
 `size_t idx,`
 `size_t maxlen) [inline], [static]`

10.27.3.8 `_out_char()` `static void _out_char (`
 `char character,`
 `void * buffer,`
 `size_t idx,`
 `size_t maxlen) [inline], [static]`

10.27.3.9 `_out_fct()` `static void _out_fct (`
 `char character,`
 `void * buffer,`
 `size_t idx,`
 `size_t maxlen) [inline], [static]`

10.27.3.10 `_out_null()` `static void _out_null (`
 `char character,`
 `void * buffer,`
 `size_t idx,`
 `size_t maxlen) [inline], [static]`

10.27.3.11 `_strlen()` `static unsigned int _strlen (`
 `const char * str) [inline], [static]`

10.27.3.12 `_vsnprintf()` `static int _vsnprintf (`
 `out.fct.type out,`
 `char * buffer,`
 `const size_t maxlen,`
 `const char * format,`
 `va_list va) [static]`

10.27.3.13 `fctprintf()` `int fctprintf (`
 `void(*) (char character, void *arg) out,`
 `void * arg,`
 `const char * format,`
 `...)`

10.27.3.14 putchar() `int putchar (`
 `char ch)`

10.27.3.15 snprintf() `int snprintf (`
 `char * buffer,`
 `size_t count,`
 `const char * format,`
 `...)`

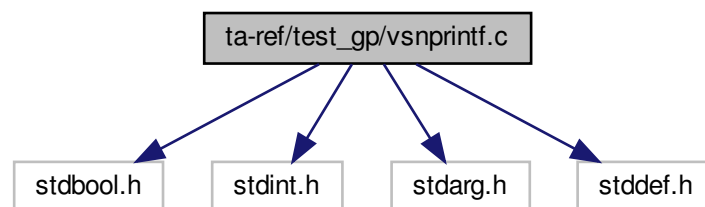
10.27.3.16 sprintf() `int sprintf (`
 `char * buffer,`
 `const char * format,`
 `...)`

10.27.3.17 vsnprintf() `int vsnprintf (`
 `char * buffer,`
 `size_t count,`
 `const char * format,`
 `va_list va)`

10.28 ta-ref/test_gp/vsnprintf.c File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include <stdarg.h>
#include <stddef.h>
```

Include dependency graph for vsnprintf.c:



Classes

- struct [out_fct_wrap_type](#)

Macros

- #define `PRINTF_NTOA_BUFFER_SIZE` 32U
- #define `PRINTF_FTOA_BUFFER_SIZE` 32U
- #define `PRINTF_SUPPORT_FLOAT`
- #define `PRINTF_SUPPORT_LONG_LONG`
- #define `PRINTF_SUPPORT_PTRDIFF_T`
- #define `FLAGS_ZEROPAD` (1U << 0U)
- #define `FLAGS_LEFT` (1U << 1U)
- #define `FLAGS_PLUS` (1U << 2U)
- #define `FLAGS_SPACE` (1U << 3U)
- #define `FLAGS_HASH` (1U << 4U)
- #define `FLAGS_UPPERCASE` (1U << 5U)
- #define `FLAGS_CHAR` (1U << 6U)
- #define `FLAGS_SHORT` (1U << 7U)
- #define `FLAGS_LONG` (1U << 8U)
- #define `FLAGS_LONG_LONG` (1U << 9U)
- #define `FLAGS_PRECISION` (1U << 10U)
- #define `_putchar` putchar

Typedefs

- typedef void(* `out_fct_type`) (char character, void *buffer, size_t idx, size_t maxlen)

Functions

- int `putchar` (char ch)
- static void `_out_buffer` (char character, void *buffer, size_t idx, size_t maxlen)
- static void `_out_null` (char character, void *buffer, size_t idx, size_t maxlen)
- static void `_out_char` (char character, void *buffer, size_t idx, size_t maxlen)
- static void `_out_fct` (char character, void *buffer, size_t idx, size_t maxlen)
- static unsigned int `_strlen` (const char *str)
- static bool `_is_digit` (char ch)
- static unsigned int `_atoi` (const char **str)
- static size_t `_ntoa_format` (`out_fct_type` out, char *buffer, size_t idx, size_t maxlen, char *buf, size_t len, bool negative, unsigned int base, unsigned int prec, unsigned int width, unsigned int flags)
- static size_t `_ntoa_long` (`out_fct_type` out, char *buffer, size_t idx, size_t maxlen, unsigned long value, bool negative, unsigned long base, unsigned int prec, unsigned int width, unsigned int flags)
- static size_t `_ntoa_long_long` (`out_fct_type` out, char *buffer, size_t idx, size_t maxlen, unsigned long long value, bool negative, unsigned long long base, unsigned int prec, unsigned int width, unsigned int flags)
- static size_t `_ftoa` (`out_fct_type` out, char *buffer, size_t idx, size_t maxlen, double value, unsigned int prec, unsigned int width, unsigned int flags)
- static int `_vsnprintf` (`out_fct_type` out, char *buffer, const size_t maxlen, const char *format, va_list va)
- int `sprintf` (char *buffer, const char *format,...)
- int `snprintf` (char *buffer, size_t count, const char *format,...)
- int `vsnprintf` (char *buffer, size_t count, const char *format, va_list va)
- int `fctprintf` (void(*out)(char character, void *arg), void *arg, const char *format,...)

10.28.1 Macro Definition Documentation

10.28.1.1 `_putchar` `#define _putchar putchar`

10.28.1.2 `FLAGS.CHAR` `#define FLAGS.CHAR (1U << 6U)`

10.28.1.3 `FLAGS.HASH` `#define FLAGS.HASH (1U << 4U)`

10.28.1.4 `FLAGS.LEFT` `#define FLAGS.LEFT (1U << 1U)`

10.28.1.5 `FLAGS.LONG` `#define FLAGS.LONG (1U << 8U)`

10.28.1.6 `FLAGS.LONG.LONG` `#define FLAGS.LONG.LONG (1U << 9U)`

10.28.1.7 `FLAGS.PLUS` `#define FLAGS.PLUS (1U << 2U)`

10.28.1.8 `FLAGS.PRECISION` `#define FLAGS.PRECISION (1U << 10U)`

10.28.1.9 `FLAGS.SHORT` `#define FLAGS.SHORT (1U << 7U)`

10.28.1.10 `FLAGS.SPACE` `#define FLAGS.SPACE (1U << 3U)`

10.28.1.11 `FLAGS.UPPERCASE` `#define FLAGS.UPPERCASE (1U << 5U)`

10.28.1.12 **FLAGS_ZEROPAD** `#define FLAGS_ZEROPAD (1U << 0U)`

10.28.1.13 **PRINTF_FTOA_BUFFER_SIZE** `#define PRINTF_FTOA_BUFFER_SIZE 32U`

10.28.1.14 **PRINTF_NTOA_BUFFER_SIZE** `#define PRINTF_NTOA_BUFFER_SIZE 32U`

10.28.1.15 **PRINTF_SUPPORT_FLOAT** `#define PRINTF_SUPPORT_FLOAT`

10.28.1.16 **PRINTF_SUPPORT_LONG_LONG** `#define PRINTF_SUPPORT_LONG_LONG`

10.28.1.17 **PRINTF_SUPPORT_PTRDIFF_T** `#define PRINTF_SUPPORT_PTRDIFF_T`

10.28.2 Typedef Documentation

10.28.2.1 **out_fct_type** `typedef void(* out_fct_type) (char character, void *buffer, size_t idx, size_t maxlen)`

10.28.3 Function Documentation

10.28.3.1 **_atoi()** `static unsigned int _atoi (const char ** str) [static]`

[_atoi\(\)](#) - Converts the internal ASCII string into an unsigned integer.

This function is to convert the internal ASCII string into unsigned integer.

Parameters

<i>str</i>	string representation of an integral number.
------------	--

Returns

i unsigned integer value.

10.28.3.2 `_ftoa()` `static size_t _ftoa (`
 `out.fct.type out,`
 `char * buffer,`
 `size_t idx,`
 `size_t maxlen,`
 `double value,`
 `unsigned int prec,`
 `unsigned int width,`
 `unsigned int flags) [static]`

`_ftoa()` - Converts a given floating-point number or a double to a string with the use of standard library functions.

This function checks whether the value is negative or not, then it checks with if condition default precision to 6, if it not set it will set explicitly. Using the while loop it limits the precision to 9, because it causes a overflow error when precision crosses above 10. Using the if condition rollover or round If the precison value is greater than 0.5 up the precision value.it round up to

1. Using the while_loop condition adding extra zeros and append decimal value to the lenthth. Finally using the conditional statement executes pad leading zeros, handling the hash value, padding spaces up to given width and reverses the string.

Parameters

<i>out</i>	type of out.fct.type
<i>buffer</i>	Pointer to a character string to write the result.
<i>idx</i>	idx bytes of size_t
<i>maxlen</i>	Maximum number of characters to write.
<i>negative</i>	boolean type
<i>base</i>	an unsigned long data type
<i>prec</i>	an unsigned integral data type
<i>width</i>	an unsigned integral data type
<i>flags</i>	an unsigned integral data type

Returns

non integer value if success else error occur

10.28.3.3 `_is_digit()` `static bool _is_digit (`
 `char ch) [inline], [static]`

`_is_digit()` - Is for the internal test if char is a digit from 0 to 9

Parameters

<i>ch</i>	This is the character to be checked.
-----------	--------------------------------------

Returns

true if *char* is a digit and internal test if *char* is a digit from 0 to 9

10.28.3.4 `_ntoa_format()` `static size_t _ntoa_format (`
`out.fct.type out,`
`char * buffer,`
`size_t idx,`
`size_t maxlen,`
`char * buf,`
`size_t len,`
`bool negative,`
`unsigned int base,`
`unsigned int prec,`
`unsigned int width,`
`unsigned int flags) [static]`

`_ntoa_format()` - Converts the string into the defined format structure.

This function uses the while condition for padding the leading zeroes and also applies the if conditions to handle the hash. Using the if condition pad spaces up to given width what specifies in that. It reverse the string and again append pad spaces up to given width.

Parameters

<i>out</i>	type of out.fct.type
<i>buffer</i>	Pointer to a character string to write the result.
<i>idx</i>	idx bytes of size_t
<i>maxlen</i>	Maximum number of characters to write.
<i>negative</i>	boolean type
<i>base</i>	an unsigned long data type
<i>prec</i>	an unsigned integer data type
<i>width</i>	an unsigned integer data type
<i>flags</i>	an unsigned integer data type

Returns

idx non integer value if success else error occur.

```

10.28.3.5 _ntoa_long() static size_t _ntoa_long (
    out.fct.type out,
    char * buffer,
    size_t idx,
    size_t maxlen,
    unsigned long value,
    bool negative,
    unsigned long base,
    unsigned int prec,
    unsigned int width,
    unsigned int flags ) [static]

```

`_ntoa_long()` - Converts string into long value.

This function begins with an if condition value then it assigns ~FLAGS.HASH into flags & value. Later it uses the if condition and do while write if precision not equal to zero and value is not equals to zero.

Parameters

<i>out</i>	type of out.fct.type
<i>buffer</i>	Pointer to a character string to write the result.
<i>id</i>	idx bytes of size_t
<i>maxlen</i>	Maximum number of characters to write.
<i>negative</i>	boolean type
<i>base</i>	an unsigned long data type
<i>prec</i>	an unsigned integral data type
<i>width</i>	an unsigned integral data type
<i>flags</i>	an unsigned integral data type

Returns

idx non integer value if success else error occur.

```

10.28.3.6 _ntoa_long_long() static size_t _ntoa_long_long (
    out.fct.type out,
    char * buffer,
    size_t idx,
    size_t maxlen,
    unsigned long long value,
    bool negative,
    unsigned long long base,
    unsigned int prec,
    unsigned int width,
    unsigned int flags ) [static]

```

`_ntoa_long_long()` - Function to convert string to long value.

This function begins with an if condition then it assigns ~FLAGS.HASH into flags & value. Later it uses the if condition and do while write if precision not equal to zero and value is not equals to zero.

Parameters

<i>out</i>	type of out.fct_type
<i>buffer</i>	Pointer to a character string to write the result.
<i>idx</i>	idx bytes of size_t
<i>maxlen</i>	Maximum number of characters to write.
<i>negative</i>	boolean type
<i>base</i>	an unsigned long data type
<i>prec</i>	an unsigned integral data type
<i>width</i>	an unsigned integral data type
<i>flag</i>	an unsigned integral data type

Returns

idx non integer value if success else error occur.

10.28.3.7 `_out_buffer()` static void _out_buffer (
 char *character*,
 void * *buffer*,
 size_t *idx*,
 size_t *maxlen*) [inline], [static]

[_out_buffer\(\)](#) - Internal buffer output

This function compares the idx and maxlen, If "idx" is less than "maxlen" then it will assign "character" value into the typecasting char "buffer[idx]"

Parameters

<i>character</i>	character type string
<i>buffer</i>	Pointer to a character string to write the result.
<i>idx</i>	bytes of size_t
<i>maxlen</i>	Maximum number of characters to write.

10.28.3.8 `_out_char()` static void _out_char (
 char *character*,
 void * *buffer*,
 size_t *idx*,
 size_t *maxlen*) [inline], [static]

[_out_char\(\)](#) - Internal putchar wrapper

The typecasting of arguments with void is to avoid unused variable warnings in some compilers. Checks the character value once the if condition is success then [putchar\(\)](#) writes a character into stdout.

Parameters

<i>character</i>	character type string
<i>buffer</i>	Pointer to a character string to write the result.
<i>idx</i>	bytes of size_t
<i>maxlen</i>	Maximum number of characters to write.

10.28.3.9 `_out_fct()` `static void _out_fct (`
 `char character,`
 `void * buffer,`
 `size_t idx,`
 `size_t maxlen) [inline], [static]`

[_out_fct\(\)](#) - Internal output function wrapper

This function typecasting `idx` and `maxlen` arguments is to avoid compiler error. And then output function wrapper and the buffer is the output fct pointer.

Parameters

<i>character</i>	character type string
<i>buffer</i>	Pointer to a character string to write the result.
<i>idx</i>	bytes of size_t
<i>maxlen</i>	Maximum number of characters to write.

10.28.3.10 `_out_null()` `static void _out_null (`
 `char character,`
 `void * buffer,`
 `size_t idx,`
 `size_t maxlen) [inline], [static]`

[_out_null\(\)](#) - Internal null output.

The typecasting of arguments with void is applied to avoid unused variable warnings in some compilers.

Parameters

<i>character</i>	character type string
<i>buffer</i>	Pointer to a character string to write the result.
<i>idx</i>	bytes of size_t
<i>maxlen</i>	Maximum number of characters to write.

10.28.3.11 `_strlen()` `static unsigned int _strlen (`
`const char * str) [inline], [static]`

`_strlen()` - calculates the length of the string.

Parameters

<i>str</i>	str is an argument of type pointer.
------------	-------------------------------------

Returns

string length if successfully executed, else error occurred.

10.28.3.12 `_vsnprintf()` `static int _vsnprintf (`
`out_fct_type out,`
`char * buffer,`
`const size_t maxlen,`
`const char * format,`
`va_list va) [static]`

`_vsnprintf()` - Function writes formatted output to a character array, up to a maximum number of characters.

The `_vsnprintf` function firstly initializes the variables of format specifiers like flags, width, precision in this they evaluate all the specifiers individually. First it checks the buffer equal to zero or not for null output function. After that flags evaluation will start using the switch case, then width field evaluation takes process using if condition.

Parameters

<i>out</i>	type of out.fct.type.
<i>buffer</i>	pointer to the buffer where you want to function to store the formatted string.
<i>maxlen</i>	maximum number of characters to store in the buffer.
<i>format</i>	string that specifies the format of the output.
<i>va</i>	variable-argument list of the additional argument.

Returns

Its return the typecasted int of idx if success otherwise error occurred.

10.28.3.13 `fctprintf()` `int fctprintf (`
`void(*) (char character, void *arg) out,`
`void * arg,`
`const char * format,`
`...)`

`fcprintf()` - Function is using the library macros of variable arguments like `vstart` and `vaend`.

This function initializes the `va_list` variable and invokes the `va_start()`. Invokes `_vsprintf` function and stores the value into `ret`. It applies the functions `va_start` and `va_end` on `va` and returns `ret`.

Parameters

<i>out</i>	An output function which takes one character and an argument pointer.
<i>arg</i>	An argument pointer for user data passed to output function.
<i>format</i>	A string that specifies the format of the output.

Returns

The number of characters that are sent to the output function, not counting the terminating null character.

10.28.3.14 putchar() `int putchar (`
 `char ch)`

10.28.3.15 snprintf() `int snprintf (`
 `char * buffer,`
 `size_t count,`
 `const char * format,`
 `...)`

[snprintf\(\)](#) - Places the generated output into the character array pointed to by buf, instead of writing it to a file

This function initializes the va_list variable and invokes the va_start(). Invokes _vsnprintf function and stores the value into ret. It applies the functions va_start and va_end on va and returns ret.

Parameters

<i>buffer</i>	pointer to buffer where you want to function to store the formatted string.
<i>count</i>	maximum number of characters to store in the buffer.
<i>format</i>	string that specifies the format of the output.

Returns

ret returns the ret value as an integer type.

10.28.3.16 sprintf() `int sprintf (`
 `char * buffer,`
 `const char * format,`
 `...)`

[sprintf\(\)](#) - Sends formatted output to a string pointed to by the argument buffer.

This function initialize the `va_list` variable and invokes the `va_start()`. Invokes `_vsnprintf` function and store the value into `ret`. It applies the functions `va_start` and `va_end` on `va` and returns `ret`.

Parameters

<i>buffer</i>	pointer to an array of char elements resulting string will store.
<i>format</i>	string that contains the text to be written to buffer.

Returns

ret It returns the ret value as an integer type.

10.28.3.17 vsnprintf() `int vsnprintf (`
 `char * buffer,`
 `size_t count,`
 `const char * format,`
 `va_list va)`

[vsnprintf\(\)](#) - Invokes another function called [_vsnprintf\(\)](#). with some arguments.

Parameters

<i>buffer</i>	Pointer to the buffer where you want to function to store the formatted string.
<i>count</i>	maximum number of characters to store in the buffer.
<i>format</i>	string that specifies the format of the output.

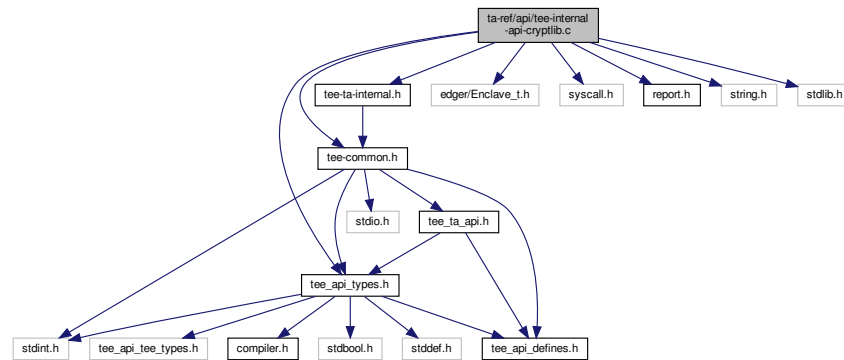
Returns

Its return the typecasted int of idx if success otherwise error occurred.

10.29 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.

10.29.1 Macro Definition Documentation

10.29.1.1 GCM_ST_AAD #define GCM_ST_AAD 2

10.29.1.2 GCM_ST_ACTIVE #define GCM_ST_ACTIVE 3

10.29.1.3 GCM_ST_FINAL #define GCM_ST_FINAL 4

10.29.1.4 GCM_ST_INIT #define GCM_ST_INIT 1

10.29.1.5 SIG_LENGTH `#define SIG_LENGTH 64`

10.29.2 Function Documentation

10.29.2.1 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

10.29.2.2 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.

10.29.2.3 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.

10.29.2.4 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.

10.29.2.5 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.

10.29.2.6 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.

10.29.2.7 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.

10.29.2.8 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

10.29.2.9 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.

10.29.2.10 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

10.29.2.11 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

10.29.2.12 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>	ouput 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.

10.29.2.13 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>	lenth of the mesaage hash.

Returns

0 on success

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

10.29.2.14 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.

10.29.2.15 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.

10.29.2.16 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.
---------------	-------------------------------

10.29.2.17 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 success

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

10.29.2.18 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.

10.29.2.19 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

10.29.2.20 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.

10.29.2.21 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.
----------	--

10.29.2.22 wolfSSL_Malloc() `void* wolfSSLMalloc (`
`size_t n)`

[wolfSSL_Malloc\(\)](#) - Allocates the requested memory and returns a pointer to it.

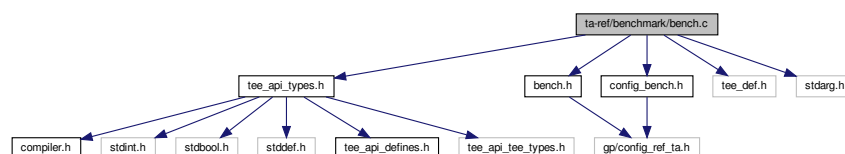
Parameters

<i>n</i>	size of the memory block.
----------	---------------------------

10.30 ta-ref/benchmark/bench.c File Reference

```
#include "tee_api_types.h"
#include "bench.h"
#include "config_bench.h"
#include "tee_def.h"
#include <stdarg.h>
```

Include dependency graph for bench.c:



Functions

- static void `benchmark` (int type, int unit)
- static uint64_t `NO_PERF time_to_millis` (TEE_Time *time)
- static uint64_t `NO_PERF time_diff` (TEE_Time *t1, TEE_Time *t2)
- void `NO_PERF init` ()
- void `time_test` (char type, TEE_Time *time, int idx)
- void `NO_PERF tee_time_tests` (int type, TEE_Time *time, int size)
- void `NO_PERF record` (int type, TEE_Time *start, TEE_Time *end, int size, int unit)

Variables

- static char `labels` [][256]

10.30.1 Function Documentation

10.30.1.1 `benchmark()` static void benchmark (
 int type,
 int unit) [static]

`benchmark()` - It invokes the benchmark function using the switch case.

This function starts with for_loop, The loop condition is based on the "@param unit" for each iteration it will go through the switch case if the switch statement matches with the type it will invoke the respective function. If it is not matched executes the default case.

Parameters

<i>type</i>	The integer type argument for switch case.
<i>unit</i>	The integer type argument for loop.

10.30.1.2 `init()` void `NO_PERF init` ()

`init()` - It Writes memory input and output to write benchmark.

This function invokes `tee_init()` and using the for_loop based on the BUFF_SIZE assigns the typecasting character value of "i&255" to the "buf[i]"

10.30.1.3 `record()` void `NO_PERF record` (
 int type,
 TEE_Time * start,
 TEE_Time * end,
 int size,
 int unit)

[record\(\)](#) - It records the execution time taken by [benchmark\(\)](#) by using the [TEE_GetREETime\(\)](#).

First this function iterates `for_loop` which invokes `TEE_GetREETime(start)`, [benchmark\(\)](#) and `TEE_GetREETime(end)`. It iterates and records the start and end time of the benchmark execution, and [test_printf\(\)](#) prints the values using `for_loop`.

Parameters

<i>type</i>	The integer type argument of memory benchmark.
<i>start</i>	The pointer type argument of TEE.Time .
<i>end</i>	The pointer type argument of TEE.Time .
<i>size</i>	The maximum size to be recorded.
<i>unit</i>	The integer type argument of memory benchmark.

10.30.1.4 tee_time_tests() `void NO_PERF tee_time_tests (`
`int type,`
`TEE.Time * time,`
`int size)`

[tee_time_tests\(\)](#) - It gets the values and prints the values using [test_printf\(\)](#).

This function iterates `for_loop` which invokes [time_test\(\)](#) to get values like type and time. Then prints the gathered information using the [test_printf\(\)](#).

Parameters

<i>type</i>	The integer type for switch case
<i>time</i>	The pointer type of TEE.Time
<i>size</i>	The maximum size to be stored.

10.30.1.5 time_diff() `static uint64_t NO_PERF time_diff (`
`TEE.Time * t1,`
`TEE.Time * t2) [static]`

[time_diff\(\)](#) - To get time difference between time *t1 and time *t2.

This function returns the time difference between the two given times.

Parameters

<i>t1</i>	The pointer type argument of TEE.Time
<i>t2</i>	The pointer type argument of TEE.Time

Returns

It will return the difference time between t1, t2.

10.30.1.6 time_test() `void time_test (`
 `char type,`
 `TEE_Time * time,`
 `int idx)`

[time_test\(\)](#) - It has two switch case statements, both contains time functions.

This function contains two switch case statements, One is to call [TEE_GetSystemTime\(\)](#) and another one is to call [TEE_GetREETime\(\)](#).

Parameters

<i>type</i>	The character type argument for switch case
<i>time</i>	The pointer type of TEE_Time
<i>idx</i>	The integer type of time_t

10.30.1.7 time_to_millis() `static uint64_t NO_PERF time_to_millis (`
 `TEE_Time * time) [static]`

[time_to_millis\(\)](#) - To get time value in milliseconds.

This function returns the conversion of time values into milliseconds.

Parameters

<i>time</i>	The pointer type argument of TEE_Time .
-------------	---

Returns

It will return time value as a milliseconds.

10.30.2 Variable Documentation

10.30.2.1 labels `char labels[][256] [static]`

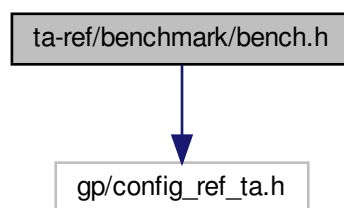
Initial value:

```
= {
    "TEE.GetREETime",
    "TEE.GetSystemTime",
    "cpu sensitive",
    "memory sensitive",
    "io sensitive",
}
```

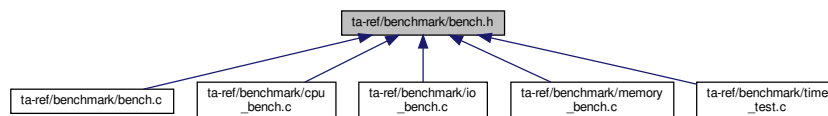
10.31 ta-ref/benchmark/bench.h File Reference

```
#include "gp/config_ref_ta.h"
```

Include dependency graph for bench.h:



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



Macros

- `#define NO_PERF __attribute__((no_instrument_function))`

Functions

- void `NO_PERF ree_time_test` (void)
- void `NO_PERF system_time_test` (void)
- void `NO_PERF cpu_int_benchmark` (void)
- void `NO_PERF cpu_double_benchmark` (void)
- void `NO_PERF io_read_benchmark` (char *buf, char *fname, int size)
- void `NO_PERF io_write_benchmark` (char *buf, char *fname, int size)
- void `NO_PERF random_memory_benchmark` (char *buf, int size)
- void `NO_PERF sequential_memory_benchmark` (char *buf, int size)

10.31.1 Macro Definition Documentation

10.31.1.1 NO_PERF `#define NO_PERF __attribute__((no_instrument_function))`

10.31.2 Function Documentation

10.31.2.1 cpu_double_benchmark() `void NO_PERF cpu_double_benchmark (`
`void)`

[cpu_double_benchmark\(\)](#) - TO check the processing of cpu double benchmark

This function invokes a for_loop based on the condition of OFFSET+MULT.SIZE values. Another for_loop is invoked inside that loop with same condition. Then the variable c gets incremented until the loop condition gets satisfied.

10.31.2.2 cpu_int_benchmark() `void NO_PERF cpu_int_benchmark (`
`void)`

[cpu_int_benchmark\(\)](#) - TO check the processing of cpu integer benchmark

This function invokes a for_loop based on the condition of OFFSET+MULT.SIZE values. Another for_loop is invoked inside that loop with same condition. Then the variable c gets incremented until the loop condition gets satisfied.

10.31.2.3 io_read_benchmark() `void NO_PERF io_read_benchmark (`
`char * buf,`
`char * fname,`
`int size)`

[io_read_benchmark\(\)](#) - About input and output read benchmark.

This function creates a persistent object with initial attributes and an initial data stream content, and optionally returns either a handle on the created object or TEE_HANDLE_NULL upon failure. Using the for_loop based on the SPLITS value it will read the object data. TEE_ReadObjectData function reads "size/SPLITS" bytes from the "b" pointed to by buffer to the data stream associated with the open object handle object. Finally it will close the object.

Parameters

<i>buf</i>	A pointer to a buffer which will be written to the file.
<i>fname</i>	The pointer type argument for filename
<i>size</i>	The length of the buffer.

10.31.2.4 io_write_benchmark() void NO_PERF io_write_benchmark (
char * buf,
char * fname,
int size)

[io_write_benchmark\(\)](#) - About input and output write benchmark.

This function creates a persistent object with initial attributes and an initial data stream content, and optionally returns either a handle on the created object or TEE_HANDLE_NULL upon failure. Using the for_loop based on the SPLITS value it will write the object data. TEE_WriteObjectData function writes "size/SPLITS" bytes from the "b" pointed to by buffer to the data stream associated with the open object handle object. Finally it will close the object.

Parameters

<i>buf</i>	A pointer to a buffer which will be written to the file.
<i>fname</i>	The pointer type argument for filename
<i>size</i>	The length of the buffer.

10.31.2.5 random_memory_benchmark() void NO_PERF random_memory_benchmark (
char * buf,
int size)

[random_memory_benchmark\(\)](#) - Mainly focusing on read and write of memory benchmark in random.

This function invokes a for_loop for memory write, it iterates upto size -1. Then assigns typecasting character value of "i&255" into "buf[idx]" along with "idx+INC" assigned to idx for each iteration. For read memory another for_loop is initiated with same condition, Here "sum" is incremented by value of "buf[idx]"

Parameters

<i>buf</i>	A pointer to the buffer in the process of read and write
<i>size</i>	The size of the buffer.

10.31.2.6 ree_time_test() void NO_PERF ree_time_test (
void)

The [ree_time_test\(\)](#) - Invokes [TEE_GetREETime\(\)](#) to get ree time

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

10.31.2.7 sequential_memory_benchmark() void NO_PERF sequential_memory_benchmark (
char * buf,
int size)

[sequential_memory_benchmark\(\)](#) - Mainly focusing on read and write of memory benchmark in sequence.

This function invokes a for_loop for memory write, it iterates upto size -1. Then assigns typecasting character value of "i&255" into "buf[idx]" For read memory another for_loop is initiated with same condition, Here "sum" is incremented by value of "buf[i]"

Parameters

<i>buf</i>	A pointer to the buffer in the process of read and write
<i>size</i>	The size of the buffer.

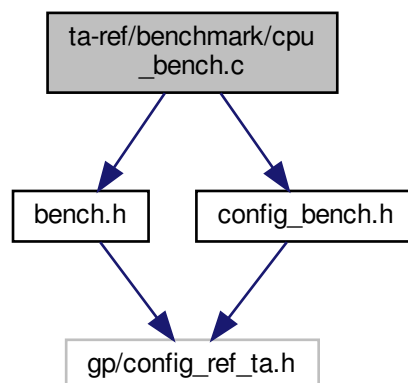
10.31.2.8 system_time_test() void NO_PERF system_time_test (
void)

The [system_time_test\(\)](#) - Invokes the [TEE_GetSystemTime\(\)](#) to get system time.

This function declares time variable and it retrieves the current system time.

10.32 ta-ref/benchmark/cpu_bench.c File Reference

```
#include "bench.h"
#include "config_bench.h"
Include dependency graph for cpu_bench.c:
```



Functions

- void [NO_PERF cpu_int_benchmark](#) (void)
- void [NO_PERF cpu_double_benchmark](#) (void)

10.32.1 Function Documentation

10.32.1.1 [cpu_double_benchmark\(\)](#) void [NO_PERF](#) [cpu_double_benchmark](#) (
void)

[cpu_double_benchmark\(\)](#) - TO check the processing of cpu double benchmark

This function invokes a for_loop based on the condition of OFFSET+MULT.SIZE values. Another for_loop is invoked inside that loop with same condition. Then the variable c gets incremented until the loop condition gets satisfied.

10.32.1.2 [cpu_int_benchmark\(\)](#) void [NO_PERF](#) [cpu_int_benchmark](#) (
void)

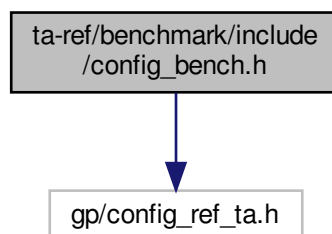
[cpu_int_benchmark\(\)](#) - TO check the processing of cpu integer benchmark

This function invokes a for_loop based on the condition of OFFSET+MULT.SIZE values. Another for_loop is invoked inside that loop with same condition. Then the variable c gets incremented until the loop condition gets satisfied.

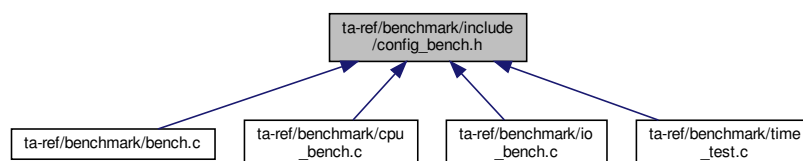
10.33 ta-ref/benchmark/include/config_bench.h File Reference

```
#include "gp/config_ref_ta.h"
```

Include dependency graph for config_bench.h:



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



Macros

- #define `OFFSET` (uint64_t)0x0102030405060708
- #define `DOUBLE_OFFSET` (double)1234567890.123456789
- #define `MULT_SIZE` 5000
- #define `BUF_SIZE` 1048576

Enumerations

- enum `BENCH_TYPE` {
 `REE_TIME_TEST` , `SYSTEM_TIME_TEST` , `CPU_INT_SENSITIVE` , `CPU_DOUBLE_SENSITIVE` ,
 `SEQUENTIAL_MEMORY_SENSITIVE` , `RANDOM_MEMORY_SENSITIVE` , `IO_WRITE_SENSITIVE` ,
 `IO_READ_SENSITIVE` }

Functions

- void `record` (int type, `TEE_Time` *start, `TEE_Time` *end, int size, int unit)

10.33.1 Macro Definition Documentation

10.33.1.1 `BUF_SIZE` #define `BUF_SIZE` 1048576

10.33.1.2 `DOUBLE_OFFSET` #define `DOUBLE_OFFSET` (double)1234567890.123456789

10.33.1.3 `MULT_SIZE` #define `MULT_SIZE` 5000

10.33.1.4 `OFFSET` #define `OFFSET` (uint64_t)0x0102030405060708

10.33.2 Enumeration Type Documentation

10.33.2.1 `BENCH_TYPE` enum `BENCH_TYPE`

Enumerator

REE_TIME_TEST	
SYSTEM_TIME_TEST	
CPU_INT_SENSITIVE	
CPU_DOUBLE_SENSITIVE	
SEQUENTIAL_MEMORY_SENSITIVE	
RANDOM_MEMORY_SENSITIVE	
IO_WRITE_SENSITIVE	
IO_READ_SENSITIVE	

10.33.3 Function Documentation

10.33.3.1 record() void record (

```

    int type,
    TEE_Time * start,
    TEE_Time * end,
    int size,
    int unit )
```

[record\(\)](#) - It records the execution time taken by [benchmark\(\)](#) by using the [TEE_GetREETime\(\)](#).

First this function iterates for_loop which invokes [TEE_GetREETime\(start\)](#), [benchmark\(\)](#) and [TEE_GetREETime\(end\)](#). It iterates and records the start and end time of the benchmark execution, and [test_printf\(\)](#) prints the values using for_loop.

Parameters

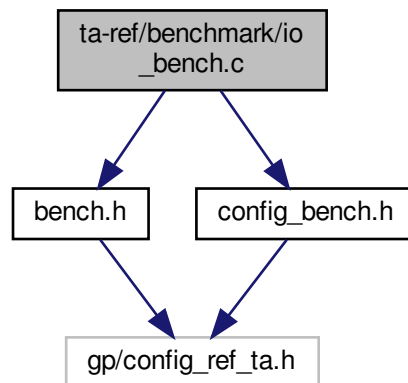
<i>type</i>	The integer type argument of memory benchmark.
<i>start</i>	The pointer type argument of TEE_Time .
<i>end</i>	The pointer type argument of TEE_Time .
<i>size</i>	The maximum size to be recorded.
<i>unit</i>	The integer type argument of memory benchmark.

10.34 ta-ref/benchmark/io_bench.c File Reference

```

#include "bench.h"
#include "config_bench.h"
```

Include dependency graph for io_bench.c:



Macros

- #define `SPLITS` 32

Functions

- void `NO_PERF io_write_benchmark` (char *buf, char *fname, int size)
- void `NO_PERF io_read_benchmark` (char *buf, char *fname, int size)

10.34.1 Macro Definition Documentation

10.34.1.1 `SPLITS` #define `SPLITS` 32

10.34.2 Function Documentation

10.34.2.1 `io_read_benchmark()` void `NO_PERF io_read_benchmark` (
 char * buf,
 char * fname,
 int size)

`io_read_benchmark()` - About input and output read benchmark.

This function creates a persistent object with initial attributes and an initial data stream content, and optionally returns either a handle on the created object or TEE_HANDLE_NULL upon failure. Using the for_loop based on the SPLITS value it will read the object data. TEE.ReadObjectData function reads "size/SPLITS" bytes from the "b" pointed to by buffer to the data stream associated with the open object handle. Finally it will close the object.

Parameters

<i>buf</i>	A pointer to a buffer which will be written to the file.
<i>fname</i>	The pointer type argument for filename
<i>size</i>	The length of the buffer.

10.34.2.2 io.write_benchmark() void [NO_PERF](#) io.write_benchmark (
char * *buf*,
char * *fname*,
int *size*)

[io.write_benchmark\(\)](#) - About input and output write benchmark.

This function creates a persistent object with initial attributes and an initial data stream content, and optionally returns either a handle on the created object or TEE_HANDLE_NULL upon failure. Using the for_loop based on the SPLITS value it will write the object data. TEE_WriteObjectData function writes "size/SPLITS" bytes from the "b" pointed to by buffer to the data stream associated with the open object handle object. Finally it will close the object.

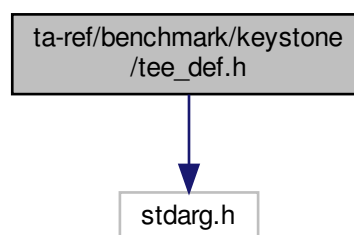
Parameters

<i>buf</i>	A pointer to a buffer which will be written to the file.
<i>fname</i>	The pointer type argument for filename
<i>size</i>	The length of the buffer.

10.35 ta-ref/benchmark/keystone/tee_def.h File Reference

```
#include <stdarg.h>
```

Include dependency graph for tee_def.h:



Functions

- static void `NO_PERF tee_init` ()
- static int `NO_PERF test_printf` (const char *fmt,...)

Variables

- static int `buf_flag` = 1
- static char * `buf`

10.35.1 Function Documentation

10.35.1.1 tee_init() static void `NO_PERF tee_init` (
void) [static]

10.35.1.2 test_printf() static int `NO_PERF test_printf` (
const char * *fmt*,
...) [static]

10.35.2 Variable Documentation

10.35.2.1 buf char* `buf` [static]

10.35.2.2 buf_flag int `buf_flag` = 1 [static]

10.36 ta-ref/benchmark/optee/tee_def.h File Reference

Macros

- #define `test_printf tee_printf`

Functions

- static void `NO_PERF tee_init` (void)

Variables

- static char `buf` [`BUF_SIZE`]
- static int `buf_flag` = 1

10.36.1 Macro Definition Documentation

10.36.1.1 test_printf `#define test_printf tee_printf`

10.36.2 Function Documentation

10.36.2.1 tee_init() `static void NO_PERF tee_init (`
`void) [static]`

10.36.3 Variable Documentation

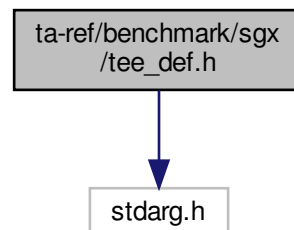
10.36.3.1 buf `char buf[BUF_SIZE] [static]`

10.36.3.2 buf_flag `int buf_flag = 1 [static]`

10.37 ta-ref/benchmark/sgx/tee_def.h File Reference

```
#include <stdarg.h>
```

Include dependency graph for `tee_def.h`:



Functions

- static void `NO_PERF tee_init` (void)
- static int `NO_PERF test_printf` (const char *fmt,...)

Variables

- static char `buf` [`BUF_SIZE`]
- static int `buf_flag` = 1

10.37.1 Function Documentation

10.37.1.1 tee_init() static void `NO_PERF tee_init` (
void) [static]

10.37.1.2 test_printf() static int `NO_PERF test_printf` (
const char * *fmt*,
...) [static]

10.37.2 Variable Documentation

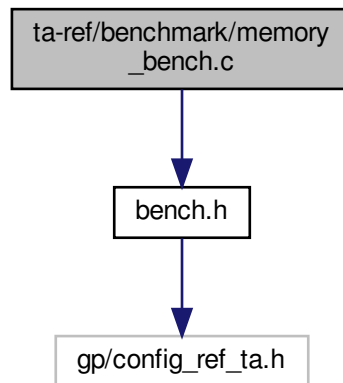
10.37.2.1 buf char `buf`[`BUF_SIZE`] [static]

10.37.2.2 buf_flag int `buf_flag` = 1 [static]

10.38 ta-ref/benchmark/memory_bench.c File Reference

```
#include "bench.h"
```

Include dependency graph for memory_bench.c:



Macros

- `#define INC 390625`

Functions

- void `NO_PERF random_memory_benchmark` (char *buf, int size)
- void `NO_PERF sequential_memory_benchmark` (char *buf, int size)

10.38.1 Macro Definition Documentation

10.38.1.1 INC `#define INC 390625`

10.38.2 Function Documentation

10.38.2.1 `random_memory_benchmark()` void `NO_PERF random_memory_benchmark` (

```

char * buf,
int size )

```

`random_memory_benchmark()` - Mainly focusing on read and write of memory benchmark in random.

This function invokes a for_loop for memory write, it iterates upto size -1. Then assigns typecasting character value of "i&255" into "buf[idx]" along with "idx+INC" assigned to idx for each iteration. For read memory another for_loop is initiated with same condition, Here "sum" is incremented by value of "buf[idx]"

Parameters

<i>buf</i>	A pointer to the buffer in the process of read and write
<i>size</i>	The size of the buffer.

10.38.2.2 sequential_memory_benchmark() void NO_PERF sequential_memory_benchmark (
char * *buf*,
int *size*)

[sequential_memory_benchmark\(\)](#) - Mainly focusing on read and write of memory benchmark in sequence.

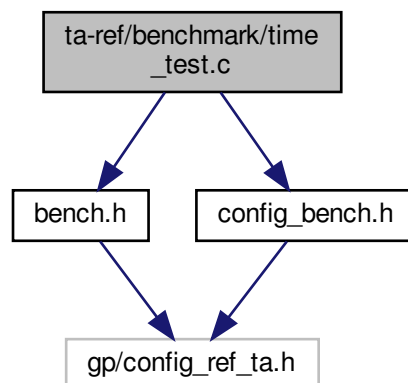
This function invokes a for_loop for memory write, it iterates upto size -1. Then assigns typecasting character value of "i&255" into "buf[idx]" For read memory another for_loop is initiated with same condition, Here "sum" is incremented by value of "buf[i]"

Parameters

<i>buf</i>	A pointer to the buffer in the process of read and write
<i>size</i>	The size of the buffer.

10.39 ta-ref/benchmark/time_test.c File Reference

```
#include "bench.h"
#include "config_bench.h"
Include dependency graph for time_test.c:
```



Functions

- void [NO_PERF ree_time_test](#) (void)
- void [NO_PERF system_time_test](#) (void)

10.39.1 Function Documentation

10.39.1.1 ree_time_test() void [NO_PERF](#) ree_time_test (void)

The [ree_time_test\(\)](#) - Invokes [TEE_GetREETime\(\)](#) to get ree time

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

10.39.1.2 system_time_test() void [NO_PERF](#) system_time_test (void)

The [system_time_test\(\)](#) - Invokes the [TEE_GetSystemTime\(\)](#) to get system time.

This function declares time variable and it retrieves the current system time.

10.40 ta-ref/docs/building.md File Reference

10.41 ta-ref/docs/gp_api.md File Reference

10.42 ta-ref/docs/how_to_program_on_ta-ref.md File Reference

10.43 ta-ref/docs/overview_of_ta-ref.md File Reference

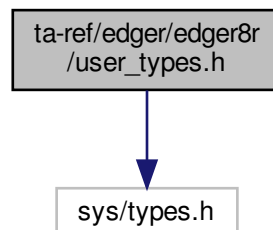
10.44 ta-ref/docs/preparation.md File Reference

10.45 ta-ref/docs/running_on_dev_boards.md File Reference

10.46 ta-ref/edger/edger8r/user_types.h File Reference

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

Include dependency graph for user_types.h:



Macros

- `#define LOOPS_PER_THREAD 500`

Typedefs

- `typedef void * buffer_t`
- `typedef int array_t[10]`

10.46.1 Macro Definition Documentation

10.46.1.1 LOOPS_PER_THREAD `#define LOOPS_PER_THREAD 500`

10.46.2 Typedef Documentation

10.46.2.1 array_t `typedef int array_t[10]`

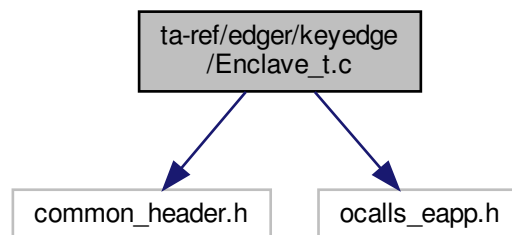
10.46.2.2 buffer_t `typedef void* buffer_t`

10.47 ta-ref/edger/keyedge/Enclave_t.c File Reference

```
#include "common_header.h"
```

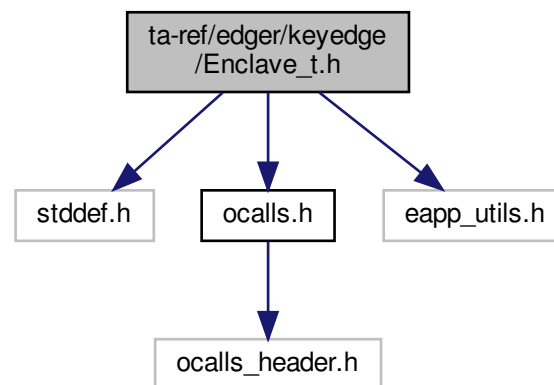
```
#include "ocalls_eapp.h"
```

Include dependency graph for Enclave_t.c:



10.48 ta-ref/edger/keyedge/Enclave_t.h File Reference

```
#include <stddef.h>
#include "ocalls.h"
#include "eapp_utils.h"
Include dependency graph for Enclave_t.h:
```



Functions

- `int ocall_file_read` (int fd, void *buf, size_t count)
- `int ocall_file_write` (int fd, const void *buf, size_t count)
- `int ocall_file_read_full` (int fd, void *buf, size_t count)
- `int ocall_file_write_full` (int fd, const void *buf, size_t count)

10.48.1 Function Documentation

10.48.1.1 ocall_file_read() `int ocall_file_read (`
 `int fd,`
 `void * buf,`
 `size_t count)`

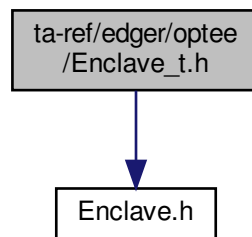
10.48.1.2 ocall_file_read_full() `int ocall_file_read_full (`
 `int fd,`
 `void * buf,`
 `size_t count)`

10.48.1.3 ocall_file_write() `int ocall_file_write (`
 `int fd,`
 `const void * buf,`
 `size_t count)`

10.48.1.4 ocall_file_write_full() `int ocall_file_write_full (`
 `int fd,`
 `const void * buf,`
 `size_t count)`

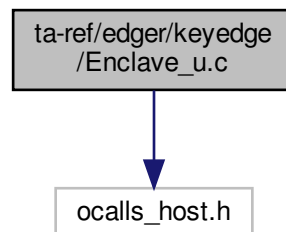
10.49 ta-ref/edger/optee/Enclave.t.h File Reference

`#include "Enclave.h"`
Include dependency graph for Enclave.t.h:



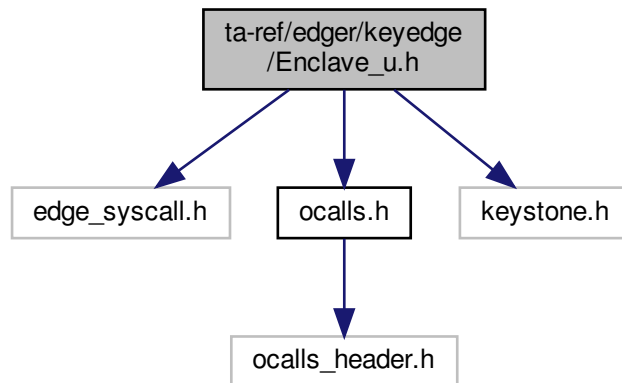
10.50 ta-ref/edger/keyedge/Enclave_u.c File Reference

`#include "ocalls_host.h"`
Include dependency graph for Enclave_u.c:



10.51 ta-ref/edger/keyedge/Enclave_u.h File Reference

```
#include "edge_syscall.h"
#include "ocalls.h"
#include "keystone.h"
Include dependency graph for Enclave_u.h:
```



Macros

- `#define` [EDGE_EXTERN_BEGIN](#)
- `#define` [EDGE_EXTERN_END](#)

Functions

- void [register_functions](#) ()
- void [__wrapper_ocall_close_file](#) (void *buffer)

10.51.1 Macro Definition Documentation

10.51.1.1 `EDGE_EXTERN_BEGIN` `#define` `EDGE_EXTERN_BEGIN`

10.51.1.2 `EDGE_EXTERN_END` `#define` `EDGE_EXTERN_END`

10.51.2 Function Documentation

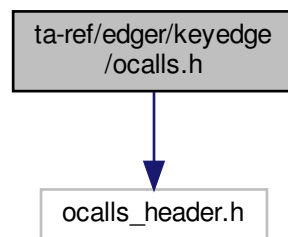
10.51.2.1 `__wrapper_ocall_close_file()` `void __wrapper_ocall_close_file (`
`void * buffer)`

10.51.2.2 `register_functions()` `void register_functions ()`

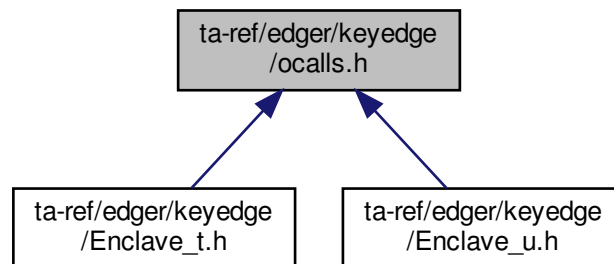
10.52 ta-ref/edger/keyedge/ocalls.h File Reference

```
#include "ocalls_header.h"
```

Include dependency graph for ocalls.h:



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



Classes

- struct [ree_time_t](#)
- struct [ob16_t](#)
- struct [ob196_t](#)
- struct [invoke_command_param_t](#)
- struct [param_buffer_t](#)
- struct [invoke_command_t](#)
- struct [ob256_t](#)

Macros

- #define [EDGE_OUT_WITH_STRUCTURE](#)
- #define [O_RDONLY](#) 0
- #define [O_WRONLY](#) 00001
- #define [O_RDWR](#) 00002
- #define [O_CREAT](#) 00100
- #define [O_EXCL](#) 00200
- #define [O_TRUNC](#) 01000

Typedefs

- typedef struct [ree_time_t](#) [ree_time_t](#)
- typedef struct [ob16_t](#) [ob16_t](#)
- typedef struct [ob196_t](#) [ob196_t](#)
- typedef struct [invoke_command_param_t](#) [invoke_command_param_t](#)
- typedef struct [param_buffer_t](#) [param_buffer_t](#)
- typedef struct [invoke_command_t](#) [invoke_command_t](#)
- typedef struct [ob256_t](#) [ob256_t](#)

Functions

- unsigned int [ocall_print_string](#) (keyedge_str const char *str)
- [ree_time_t](#) [ocall_ree_time](#) (void)
- [ob16_t](#) [ocall_getrandom16](#) (unsigned int flags)
- [ob196_t](#) [ocall_getrandom196](#) (unsigned int flags)
- [invoke_command_t](#) [ocall_pull_invoke_command](#) ()
- void [ocall_write_invoke_param](#) (int index, size_t offset, keyedge_size size_t [size](#), keyedge_vla const char *buf)
- [param_buffer_t](#) [ocall_read_invoke_param](#) (int index, size_t offset)
- void [ocall_put_invoke_command_result](#) ([invoke_command_t](#) cmd, unsigned int [result](#))
- int [ocall_open_file](#) (keyedge_str const char *str, int flags, int perm)
- int [ocall_close_file](#) (int fd)
- [ob256_t](#) [ocall_read_file256](#) (int fd, unsigned int count)
- int [ocall_write_file256](#) (int fd, keyedge_vla const char *buf, keyedge_size unsigned int count)
- int [ocall_unlink](#) (keyedge_str const char *path)
- int [ocall_fstat_size](#) (int fd)

10.52.1 Macro Definition Documentation

10.52.1.1 [EDGE_OUT_WITH_STRUCTURE](#) #define [EDGE_OUT_WITH_STRUCTURE](#)

10.52.1.2 [O_CREAT](#) #define [O_CREAT](#) 00100

10.52.1.3 O_EXCL `#define O_EXCL 00200`

10.52.1.4 O_RDONLY `#define O_RDONLY 0`

10.52.1.5 O_RDWR `#define O_RDWR 00002`

10.52.1.6 O_TRUNC `#define O_TRUNC 01000`

10.52.1.7 O_WRONLY `#define O_WRONLY 00001`

10.52.2 Typedef Documentation

10.52.2.1 `invoke_command_param_t` `typedef struct invoke_command_param_t invoke_command_param_t`

10.52.2.2 `invoke_command_t` `typedef struct invoke_command_t invoke_command_t`

10.52.2.3 `ob16_t` `typedef struct ob16_t ob16_t`

10.52.2.4 `ob196_t` `typedef struct ob196_t ob196_t`

10.52.2.5 `ob256_t` `typedef struct ob256_t ob256_t`

10.52.2.6 `param_buffer_t` `typedef struct param_buffer_t param_buffer_t`

10.52.2.7 `ree_time_t` `typedef struct ree_time_t ree_time_t`

10.52.3 Function Documentation

10.52.3.1 `ocall_close_file()` `int ocall_close_file (
 int desc)`

[ocall_close_file\(\)](#) - To close a file.

Parameters

<i>fdesc</i>	file descriptor.
--------------	------------------

Returns

integer value If success

[ocall_close_file\(\)](#) - To close a file.

Parameters

<i>desc</i>	file descriptor.
-------------	------------------

Returns

integer value If success

[ocall_close_file\(\)](#) - Frees the file descriptor in the process.

Parameters

<i>fdesc</i>	fdesc is a file descriptor of the type integer.
--------------	---

Returns

rtn on success,-1 on failure.

[ocall_close_file\(\)](#) - Used for closing a file

Parameters

<i>desc</i>	File descriptor.
-------------	------------------

Returns

file descripto If success, else error occured.

10.52.3.2 ocall_fstat_size() `int ocall_fstat_size (int fd)`

10.52.3.3 ocall_getrandom16() `ob16_t` ocall_getrandom16 (
 unsigned int *flags*)

10.52.3.4 ocall_getrandom196() `ob196_t` ocall_getrandom196 (
 unsigned int *flags*)

10.52.3.5 ocall_open_file() int ocall_open_file (
 keyedge_str const char * *str*,
 int *flags*,
 int *perm*)

10.52.3.6 ocall_print_string() unsigned int ocall_print_string (
 keyedge_str const char * *str*)

10.52.3.7 ocall_pull_invoke_command() `invoke_command_t` ocall_pull_invoke_command ()

10.52.3.8 ocall_put_invoke_command_result() void ocall_put_invoke_command_result (
 `invoke_command_t` *cmd*,
 unsigned int *result*)

10.52.3.9 ocall_read_file256() `ob256_t` ocall_read_file256 (
 int *fd*,
 unsigned int *count*)

10.52.3.10 ocall_read_invoke_param() `param_buffer_t` ocall_read_invoke_param (
 int *index*,
 size_t *offset*)

10.52.3.11 ocall_ree_time() `ree_time_t` ocall_ree_time (
 void)

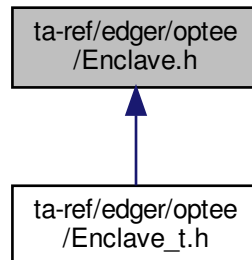
10.52.3.12 ocall_unlink() `int ocall_unlink (`
`keyedge_str const char * path)`

10.52.3.13 ocall_write_file256() `int ocall_write_file256 (`
`int fd,`
`keyedge_vla const char * buf,`
`keyedge_size unsigned int count)`

10.52.3.14 ocall_write_invoke_param() `void ocall_write_invoke_param (`
`int index,`
`size_t offset,`
`keyedge_size size_t size,`
`keyedge_vla const char * buf)`

10.53 ta-ref/edger/optee/Enclave.h File Reference

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



Macros

- `#define TA_REF_UUID { 0xa6f77c1e, 0x96fe, 0x4a0e, { 0x9e, 0x74, 0x26, 0x25, 0x82, 0xa4, 0xc8, 0xf1 } }`
- `#define TA_REF_RUN_ALL 0`

10.53.1 Macro Definition Documentation

10.53.1.1 TA_REF_RUN_ALL `#define TA_REF_RUN_ALL 0`

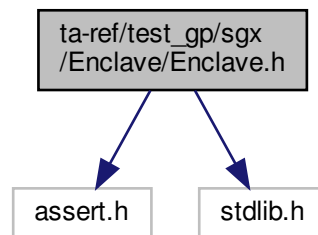
10.53.1.2 TA_REF_UUID #define TA_REF_UUID { 0xa6f77c1e, 0x96fe, 0x4a0e, { 0x9e, 0x74, 0x26, 0x25, 0x82, 0xa4, 0xc8, 0xf1}}

10.54 ta-ref/test_gp/sgx/Enclave/Enclave.h File Reference

```
#include <assert.h>
```

```
#include <stdlib.h>
```

Include dependency graph for Enclave.h:



Functions

- void [gp_random_test](#) (void)
- void [gp_ree_time_test](#) (void)
- void [gp_trusted_time_test](#) (void)
- void [gp_secure_storage_test](#) (void)
- void [gp_message_digest_test](#) (void)
- void [gp_symmetric_key_enc_verify_test](#) (void)
- void [gp_symmetric_key_gcm_verify_test](#) (void)
- void [gp_asymmetric_key_sign_test](#) (void)

10.54.1 Function Documentation

10.54.1.1 gp_asymmetric_key_sign_test() void gp_asymmetric_key_sign_test (void)

[gp_asymmetric_key_sign_test\(\)](#) - Cryptographic Operations for API Message Digest Functions.

[TEE_AllocateOperation\(\)](#) function allocates a handle for a new cryptographic operation and sets the mode([TEE_MODE_DIGEST](#)) and algorithm type ([TEE_ALG_SHA256](#)). If this function does not return with [TEE_SUCCESS](#) then there is no valid handle value. [TEE_DigestUpdate\(\)](#) function accumulates message data for hashing. The message does not have to be block aligned. Subsequent calls to this function are possible. [TEE_DigestDoFinal\(\)](#) finalizes the message digest operation and produces the message hash. Afterwards the Message Digest operation is reset to initial state and can be reused. [TEE_FreeOperation\(\)](#) function deallocates all resources associated with an operation handle. after that print the dump hashed data and allocate handle for a Sign hashed data with the generated keys and allocates an uninitialized transient object, i.e. a container for attributes. Transient objects are used to hold a cryptographic object(key or keypair) and generates a random key or a key-pair and populates a transient key object with the generated key material and The key material is copied from the key object handle into the operation and signs a message digest within an asymmetric operation and deallocates all resources associated with an operation handle, print the dump signature and verifies a message digest signature within an asymmetric operation and Free Transient Object finally check the TEE Result if it success it will print the verify ok otherwise verify fails.

10.54.1.2 gp_message_digest_test() void gp_message_digest_test (void)

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

The function performs many operations to achieve message data hash techniques to allocate a handle for a new cryptographic operation, to finalize the message digest operation and to produce the message hash. The hashed message is printed to check the output.

10.54.1.3 gp_random_test() void gp_random_test (void)

[gp_random_test\(\)](#) - Generates the random data from the method.

Generates the random data and finally print the generated random data.

10.54.1.4 gp_ree_time_test() void gp_ree_time_test (void)

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

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

10.54.1.5 gp_secure_storage_test() void gp_secure_storage_test (void)

[gp_secure_storage_test\(\)](#) - Create persistent object for read and write the object data.

Creates a persistent object with initial attributes and an initial data stream content, and optionally returns either a handle on the created object, or TEE_HANDLE_NULL upon failure and TEE_STORAGE_PRIVATE parameter indicates which is the Trusted Storage Space to access. TEE_DATA_FLAG_ACCESS_WRITE object is opened with the write access right. This allows the Trusted Application to call the functions TEE_WriteObjectData and TEE_TruncateObjectData. TEE_DATA_FLAG_OVERWRITE The flags which determine the settings under which the object is opened and copies data length from data to buf. writes DATA_LENGTH bytes from the buffer pointed to by data to the data stream associated with the open object handle object, finally close the object and clear the buffer. Create the persistent object for reading the object data and once completed it will close the object. otherwise it will error message like TEE_ReadObjectData fails and finally it will Compare read data with written data if it is success it will print the verify ok, otherwise verify fails.

10.54.1.6 gp_symmetric_key_enc.verify_test() void gp_symmetric_key_enc.verify_test (void)

[gp_symmetric_key_enc.verify_test\(\)](#) - starts the symmetric cipher operation.

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.

10.54.1.7 gp_symmetric_key_gcm_verify_test() void gp_symmetric_key_gcm_verify_test (void)

[gp_symmetric_key_gcm_verify_test\(\)](#) - Encrypt and Decrypt the test data.

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 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.

10.54.1.8 gp_trusted_time_test() void gp_trusted_time_test (void)

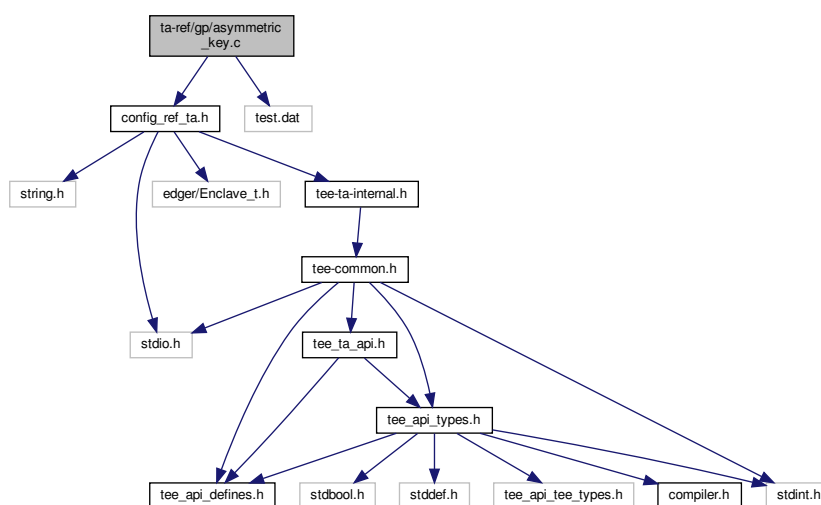
[gp_trusted_time_test\(\)](#) - Retrieves the current system time.

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

10.55 ta-ref/gp/asymmetric_key.c File Reference

```
#include "config_ref_ta.h"
#include "test.dat"
```

Include dependency graph for asymmetric_key.c:



Macros

- #define [SHA_LENGTH](#) (256/8)
- #define [SIG_LENGTH](#) 64

Functions

- void [gp_asymmetric_key_sign_test](#) (void)

10.55.1 Macro Definition Documentation

10.55.1.1 SHA_LENGTH `#define SHA_LENGTH (256/8)`

10.55.1.2 SIG_LENGTH `#define SIG_LENGTH 64`

10.55.2 Function Documentation

10.55.2.1 gp_asymmetric_key_sign_test() `void gp_asymmetric_key_sign_test (void)`

[gp_asymmetric_key_sign_test\(\)](#) - Cryptographic Operations for API Message Digest Functions.

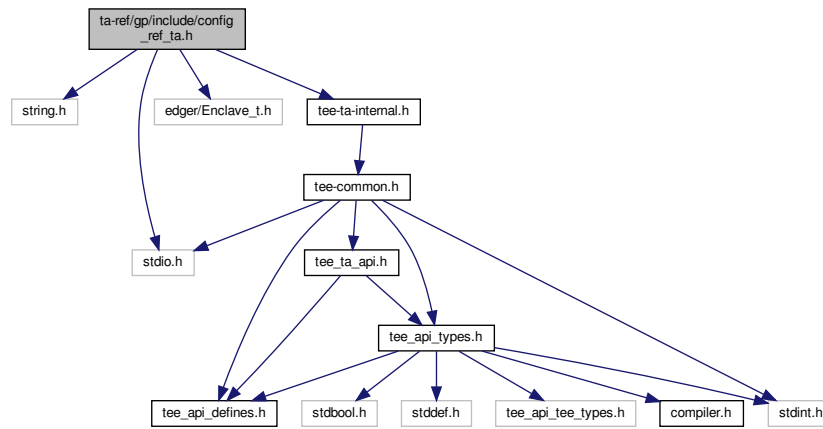
[TEE.AllocateOperation\(\)](#) function allocates a handle for a new cryptographic operation and sets the mode([TEE.MODE_DIGEST](#)) and algorithm type ([TEE.ALG_SHA256](#)). If this function does not return with [TEE.SUCCESS](#) then there is no valid handle value. [TEE.DigestUpdate\(\)](#) function accumulates message data for hashing. The message does not have to be block aligned. Subsequent calls to this function are possible. [TEE.DigestDoFinal\(\)](#) finalizes the message digest operation and produces the message hash. Afterwards the Message Digest operation is reset to initial state and can be reused. [TEE.FreeOperation\(\)](#) function deallocates all resources associated with an operation handle. after that print the dump hashed data and allocate handle for a Sign hashed data with the generated keys and allocates an uninitialized transient object, i.e. a container for attributes. Transient objects are used to hold a cryptographic object (key or keypair) and generates a random key or a key-pair and populates a transient key object with the generated key material and The key material is copied from the key object handle into the operation and signs a message digest within an asymmetric operation and deallocates all resources associated with an operation handle, print the dump signature and verifies a message digest signature within an asymmetric operation and Free Transient Object finally check the TEE Result if it success it will print the verify ok otherwise verify fails.

10.56 ta-ref/gp/include/config_ref_ta.h File Reference

```
#include <string.h>
#include <stdio.h>
#include "edger/Enclave_t.h"
```

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

Include dependency graph for config_ref_ta.h:



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



Macros

- `#define GP_ASSERT(rv, msg)`

Functions

- `int tee_printf(const char *fmt,...)`

10.56.1 Macro Definition Documentation

10.56.1.1 GP_ASSERT `#define GP_ASSERT(`
`rv,`
`msg)`

10.56.2 Function Documentation

10.56.2.1 tee_printf() `int tee_printf (`
`const char * fmt,`
`...)`

[tee_printf\(\)](#) - Printing the formatted output in to a character array.

In this function the "@param ap" variable is initialized by calling `va_start()` and then formatted data will send to a string using argument list by calling [vsprintf\(\)](#) and finally the string length will be stored in `res`.

Parameters

<i>fmt</i>	A string that specifies the format of the output.
------------	---

Returns

result If success, else error occurred.

[tee_printf\(\)](#) - For trace GP API.

Initializes `ap` variable. Formats data under control of the format control string and stores the result in `buf` and ends the processing of `ap`. Finally prints the buffer value.

Parameters

<i>fmt</i>	<code>fmt</code> is constant character argument of type pointer.
------------	--

Returns

`res` Based on the condition check it will return string length else returns 0.

[tee_printf\(\)](#) - For tracing GP API.

Initializes `ap` variable. Formats data under control of the format control string and stores the result in `buf` and ends the processing of `ap`. Finally print the buffer value.

Parameters

<i>fmt</i>	<code>fmt</code> is a constant character argument of type pointer.
------------	--

Returns

buffer If successfully executed, else error occurred.

10.57 ta-ref/gp/include/gp_test.h File Reference

Functions

- void [gp_random_test](#) (void)

- void [gp_ree_time_test](#) (void)
- void [gp_trusted_time_test](#) (void)
- void [gp_secure_storage_test](#) (void)
- void [gp_message_digest_test](#) (void)
- void [gp_symmetric_key_enc_verify_test](#) (void)
- void [gp_symmetric_key_gcm_verify_test](#) (void)
- void [gp_asymmetric_key_sign_test](#) (void)
- void [gp_invokecommand_test](#) (void)

10.57.1 Function Documentation

10.57.1.1 [gp_asymmetric_key_sign_test\(\)](#) void [gp_asymmetric_key_sign_test](#) (void)

[gp_asymmetric_key_sign_test\(\)](#) - Cryptographic Operations for API Message Digest Functions.

[TEE.AllocateOperation\(\)](#) function allocates a handle for a new cryptographic operation and sets the mode([TEE.MODE_DIGEST](#)) and algorithm type ([TEE.ALG_SHA256](#)). If this function does not return with [TEE.SUCCESS](#) then there is no valid handle value. [TEE.DigestUpdate\(\)](#) function accumulates message data for hashing. The message does not have to be block aligned. Subsequent calls to this function are possible. [TEE.DigestDoFinal\(\)](#) finalizes the message digest operation and produces the message hash. Afterwards the Message Digest operation is reset to initial state and can be reused. [TEE.FreeOperation\(\)](#) function deallocates all resources associated with an operation handle. After that print the dump hashed data and allocate handle for a Sign hashed data with the generated keys and allocates an uninitialized transient object, i.e. a container for attributes. Transient objects are used to hold a cryptographic object (key or keypair) and generates a random key or a key-pair and populates a transient key object with the generated key material. The key material is copied from the key object handle into the operation and signs a message digest within an asymmetric operation and deallocates all resources associated with an operation handle, print the dump signature and verifies a message digest signature within an asymmetric operation and Free Transient Object finally check the TEE Result if it success it will print the verify ok otherwise verify fails.

10.57.1.2 [gp_invokecommand_test\(\)](#) void [gp_invokecommand_test](#) (void)

10.57.1.3 [gp_message_digest_test\(\)](#) void [gp_message_digest_test](#) (void)

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

The function performs many operations to achieve message data hash techniques to allocate a handle for a new cryptographic operation, to finalize the message digest operation and to produce the message hash. The hashed message is printed to check the output.

10.57.1.4 [gp_random_test\(\)](#) void [gp_random_test](#) (void)

[gp_random_test\(\)](#) - Generates the random data from the method.

Generates the random data and finally print the generated random data.

10.57.1.5 gp_ree_time_test() void gp_ree_time_test (void)

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

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

10.57.1.6 gp_secure_storage_test() void gp_secure_storage_test (void)

[gp_secure_storage_test\(\)](#) - Create persistent object for read and write the object data.

Creates a persistent object with initial attributes and an initial data stream content, and optionally returns either a handle on the created object, or TEE_HANDLE_NULL upon failure and TEE_STORAGE_PRIVATE parameter indicates which is the Trusted Storage Space to access. TEE_DATA_FLAG_ACCESS_WRITE object is opened with the write access right. This allows the Trusted Application to call the functions TEE_WriteObjectData and TEE_TruncateObjectData. TEE_DATA_FLAG_OVERWRITE The flags which determine the settings under which the object is opened and copies data length from data to buf. writes DATA_LENGTH bytes from the buffer pointed to by data to the data stream associated with the open object handle object, finally close the object and clear the buffer. Create the persistent object for reading the object data and once completed it will close the object. otherwise it will error message like TEE_ReadObjectData fails and finally it will Compare read data with written data if it is success it will print the verify ok, otherwise verify fails.

10.57.1.7 gp_symmetric_key_enc.verify_test() void gp_symmetric_key_enc.verify_test (void)

[gp_symmetric_key_enc.verify_test\(\)](#) - starts the symmetric cipher operation.

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.

10.57.1.8 gp_symmetric_key_gcm.verify_test() void gp_symmetric_key_gcm.verify_test (void)

[gp_symmetric_key_gcm.verify_test\(\)](#) - Encrypt and Decrypt the test data.

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 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.

10.57.1.9 gp_trusted_time_test() void gp_trusted_time_test (void)

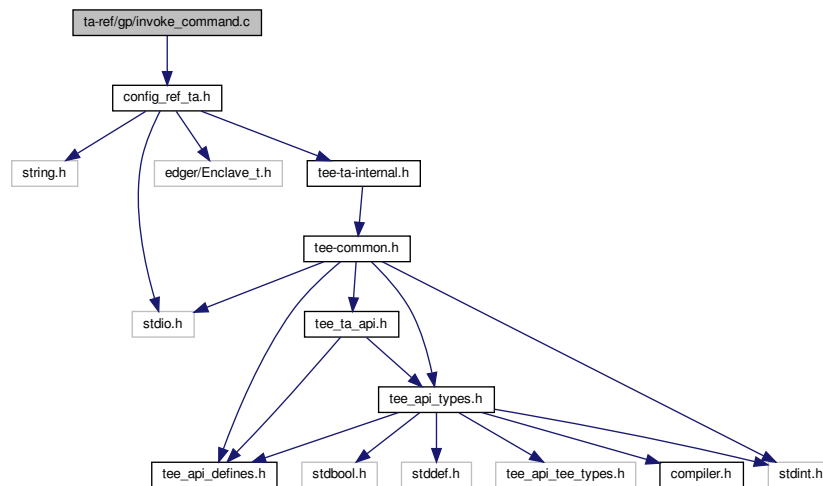
[gp_trusted_time_test\(\)](#) - Retrieves the current system time.

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

10.58 ta-ref/gp/invoke_command.c File Reference

```
#include "config_ref_ta.h"
```

Include dependency graph for invoke_command.c:



Macros

- `#define TA_MAX_SIZE 32768`
- `#define TEEP_AGENT_TA_NONE 0`
- `#define TEEP_AGENT_TA_EXIT 999`
- `#define TEEP_AGENT_TA_LOAD 1`
- `#define TEEP_AGENT_TA_INSTALL 2`
- `#define TEEP_AGENT_TA_DELETE 3`

10.58.1 Macro Definition Documentation

10.58.1.1 TA_MAX_SIZE `#define TA_MAX_SIZE 32768`

10.58.1.2 TEEP_AGENT_TA_DELETE `#define TEEP_AGENT_TA_DELETE 3`

10.58.1.3 TEEP_AGENT_TA_EXIT `#define TEEP_AGENT_TA_EXIT 999`

10.58.1.4 TEEP_AGENT_TA_INSTALL `#define TEEP_AGENT_TA_INSTALL 2`

10.58.1.5 TEEP_AGENT_TA_LOAD `#define TEEP_AGENT_TA_LOAD 1`

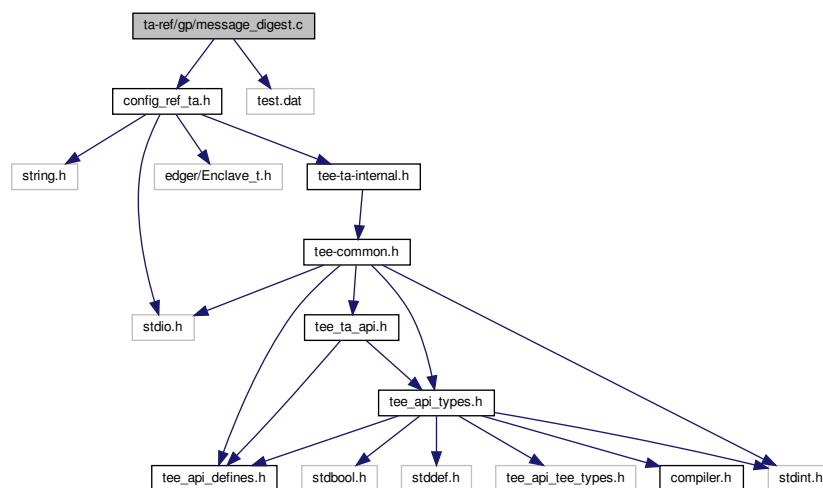
10.58.1.6 TEEP_AGENT_TA_NONE `#define TEEP_AGENT_TA_NONE 0`

10.59 ta-ref/gp/message_digest.c File Reference

```
#include "config_ref_ta.h"
```

```
#include "test.dat"
```

Include dependency graph for message_digest.c:



Macros

- `#define` [SHA_LENGTH](#) (256/8)
- `#define` [SIG_LENGTH](#) 64

Functions

- void [gp_message_digest_test](#) (void)

10.59.1 Macro Definition Documentation

10.59.1.1 SHA_LENGTH `#define SHA_LENGTH (256/8)`

10.59.1.2 SIG_LENGTH `#define SIG_LENGTH 64`

10.59.2 Function Documentation

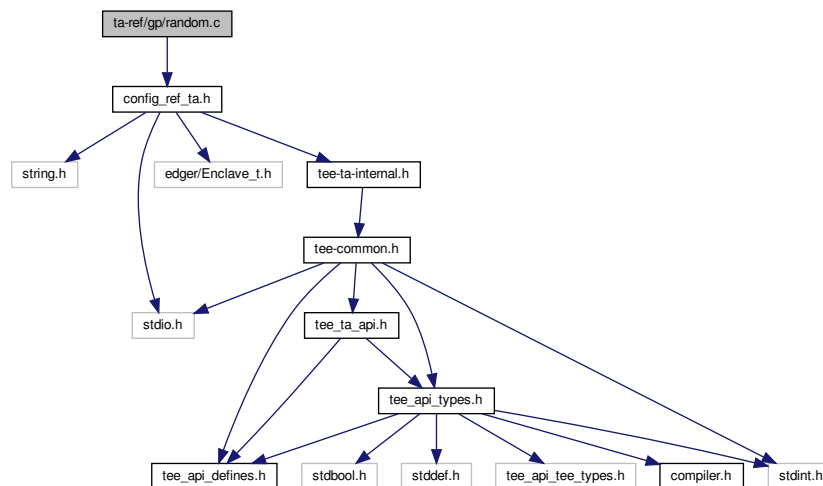
10.59.2.1 gp_message_digest_test() `void gp_message_digest_test (`
`void)`

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

The function performs many operations to achieve message data hash techniques to allocate a handle for a new cryptographic operation, to finalize the message digest operation and to produce the message hash. The hashed message is printed to check the output.

10.60 ta-ref/gp/random.c File Reference

`#include "config_ref_ta.h"`
 Include dependency graph for random.c:



Functions

- `void` [gp_random_test](#) (`void`)

10.60.1 Function Documentation

10.60.1.1 `gp_random_test()` `void gp_random_test (`
`void)`

`gp_random_test()` - Generates the random data from the method.

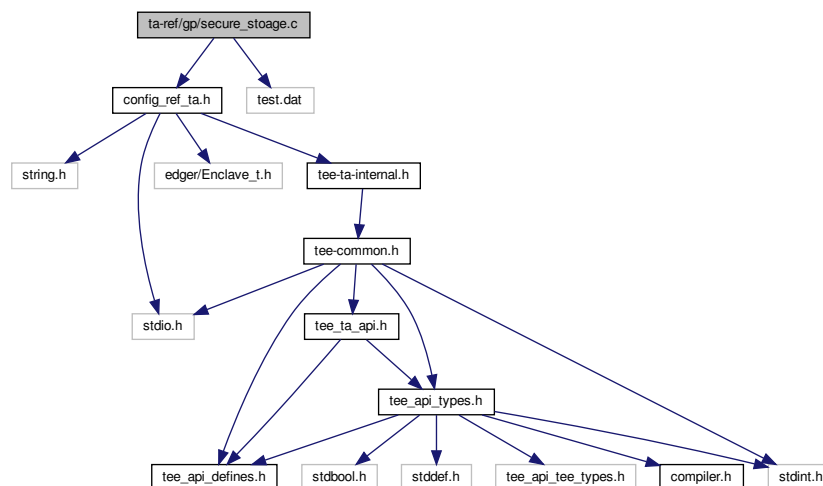
Generates the random data and finally print the generated random data.

10.61 ta-ref/gp/secure_stoage.c File Reference

```
#include "config_ref_ta.h"
```

```
#include "test.dat"
```

Include dependency graph for `secure_stoage.c`:



Macros

- `#define DATA_LENGTH 256`

Functions

- `void gp_secure_storage_test (void)`

10.61.1 Macro Definition Documentation

10.61.1.1 DATA_LENGTH `#define DATA_LENGTH 256`

10.61.2 Function Documentation

10.61.2.1 `gp_secure_storage_test()` `void gp_secure_storage_test (`
`void)`

`gp_secure_storage_test()` - Create persistent object for read and write the object data.

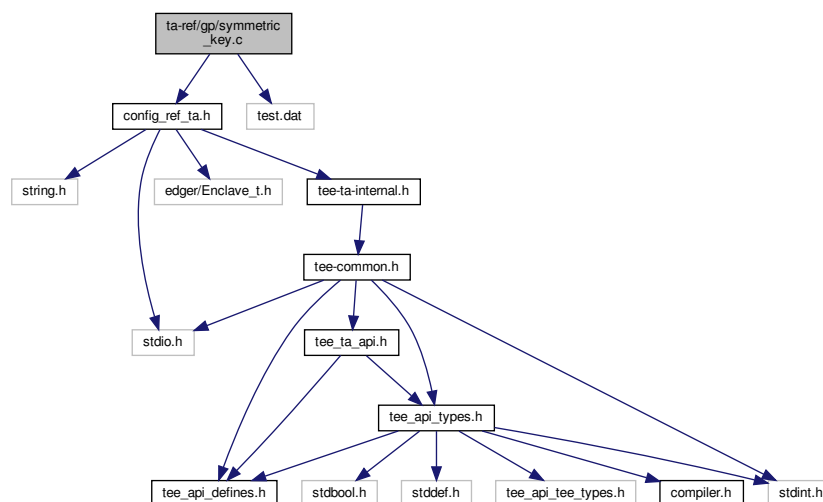
Creates a persistent object with initial attributes and an initial data stream content, and optionally returns either a handle on the created object, or `TEE_HANDLE_NULL` upon failure and `TEE_STORAGE_PRIVATE` parameter indicates which is the Trusted Storage Space to access. `TEE_DATA_FLAG_ACCESS_WRITE` object is opened with the write access right. This allows the Trusted Application to call the functions `TEE_WriteObjectData` and `TEE_TruncateObjectData`. `TEE_DATA_FLAG_OVERWRITE` The flags which determine the settings under which the object is opened and copies data length from data to buf. writes `DATA_LENGTH` bytes from the buffer pointed to by data to the data stream associated with the open object handle object, finally close the object and clear the buffer. Create the persistent object for reading the object data and once completed it will close the object. otherwise it will error message like `TEE_ReadObjectData` fails and finally it will Compare read data with written data if it is success it will print the verify ok, otherwise verify fails.

10.62 ta-ref/gp/symmetric_key.c File Reference

```
#include "config_ref_ta.h"
```

```
#include "test.dat"
```

Include dependency graph for `symmetric_key.c`:



Macros

- `#define CIPHER_LENGTH 256`

Functions

- void [gp_symmetric_key_enc_verify_test](#) (void)

10.62.1 Macro Definition Documentation

10.62.1.1 CIPHER_LENGTH `#define CIPHER_LENGTH 256`

10.62.2 Function Documentation

10.62.2.1 `gp_symmetric_key_enc_verify_test()` `void gp_symmetric_key_enc_verify_test (void)`

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

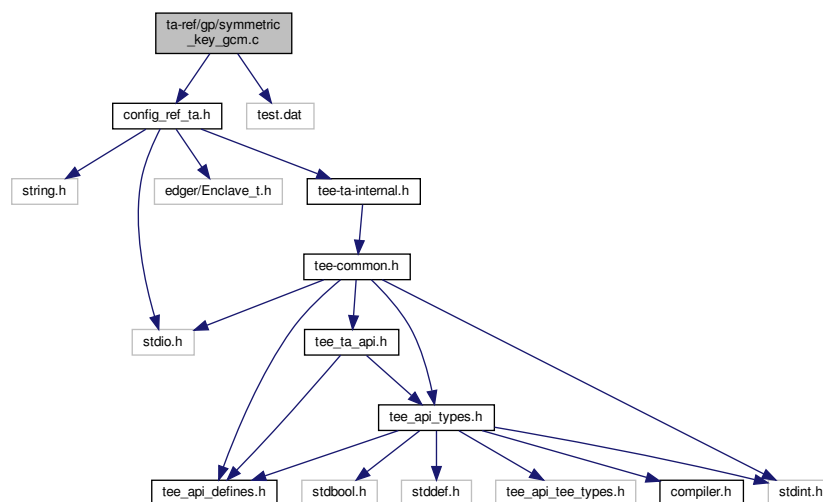
This function allocates an uninitialized transient object, i.e. a container for attributes. Transient objects are used to hold a cryptographic object (key or key-pair). With the generation of a key, a new cryptographic operation for encrypt and decrypt data is initiated with a symmetric cipher operation.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

10.63 ta-ref/gp/symmetric_key_gcm.c File Reference

```
#include "config_ref_ta.h"
```

```
#include "test.dat"
```

Include dependency graph for symmetric_key_gcm.c:



Macros

- `#define CIPHER_LENGTH 256`

Functions

- `void gp_symmetric_key_gcm_verify_test (void)`

10.63.1 Macro Definition Documentation

10.63.1.1 CIPHER_LENGTH `#define CIPHER_LENGTH 256`

10.63.2 Function Documentation

10.63.2.1 `gp_symmetric_key_gcm_verify_test()` `void gp_symmetric_key_gcm_verify_test (void)`

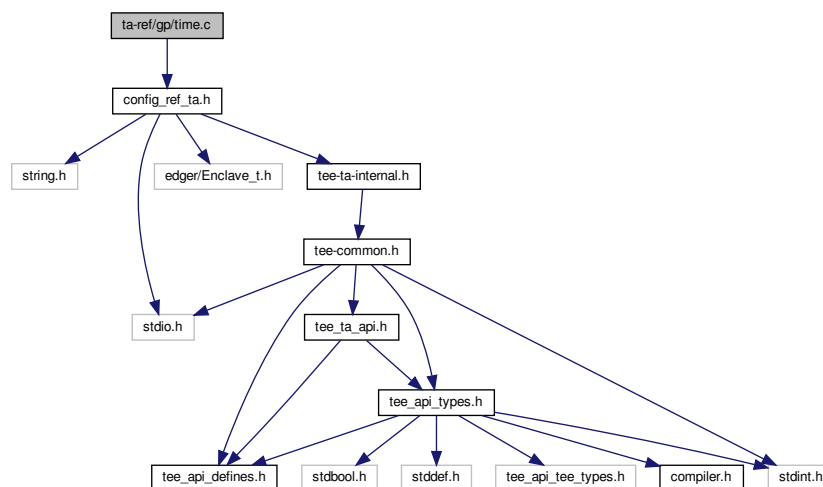
`gp_symmetric_key_gcm_verify_test()` - Encrypt and Decrypt the test data.

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 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.

10.64 ta-ref/gp/time.c File Reference

```
#include "config_ref.ta.h"
```

Include dependency graph for time.c:



Functions

- void [gp_ree_time_test](#) (void)
- void [gp_trusted_time_test](#) (void)

10.64.1 Function Documentation

10.64.1.1 [gp_ree_time_test\(\)](#) void gp_ree_time_test (
void)

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

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

10.64.1.2 [gp_trusted_time_test\(\)](#) void gp_trusted_time_test (
void)

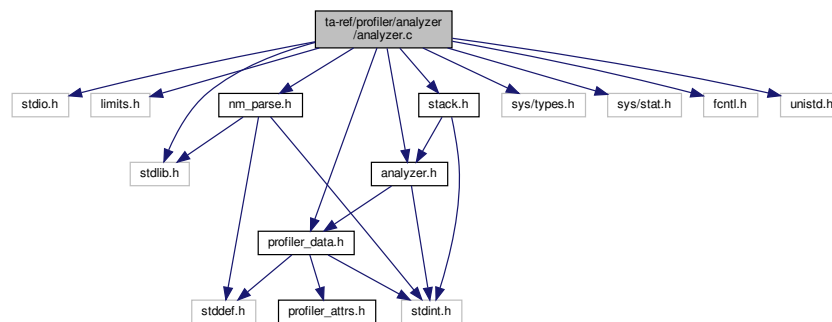
[gp_trusted_time_test\(\)](#) - Retrieves the current system time.

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

10.65 ta-ref/profiler/analyzer/analyzer.c File Reference

```
#include <stdio.h>
#include <limits.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include "profiler_data.h"
#include "stack.h"
#include "analyzer.h"
#include "nm_parse.h"
```

Include dependency graph for analyzer.c:



Macros

- #define `BUF_MAX` 65536
- #define `COLS` "id, `idx`, start_core_id, end_core_id, depth, addr, funcname, `start`[clocks], end, duration"
- #define `FORMAT` "%03d,%03d,%d,%d,%ld,0x%08lx,%s,%ld,%ld,%ld\n"

Functions

- int `main` (int argc, char *argv[])

10.65.1 Macro Definition Documentation

10.65.1.1 `BUF_MAX` `#define BUF_MAX 65536`

10.65.1.2 `COLS` `#define COLS "id, idx, start_core_id, end_core_id, depth, addr, funcname, start[clocks], end, duration"`

10.65.1.3 `FORMAT` `#define FORMAT "%03d,%03d,%d,%d,%ld,0x%08lx,%s,%ld,%ld,%ld\n"`

10.65.2 Function Documentation

10.65.2.1 `main()` `int main (`
 `int argc,`
 `char * argv[])`

`main()` - Opens the log file, reads and performs the print operation.

This function opens the log file and read the data inside the log file. `for_loop` starts to print the column one by one and hence it shows the complete log details.

Parameters

<code>argc</code>	Argument Count is int and stores number of command-line arguments passed by the user including the name of the program.
<code>argv</code>	Argument Vector is array of character pointers listing all the arguments.

Returns

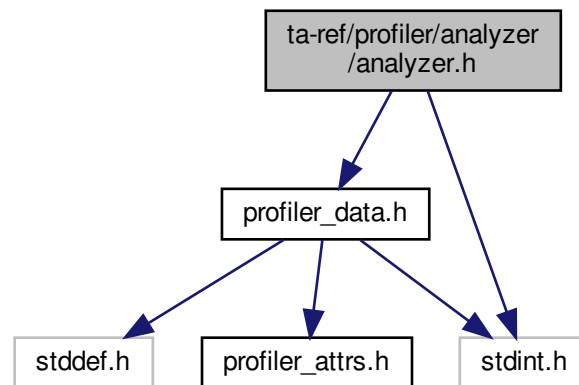
0 If success, else error occurred.

10.66 ta-ref/profiler/analyzer/analyzer.h File Reference

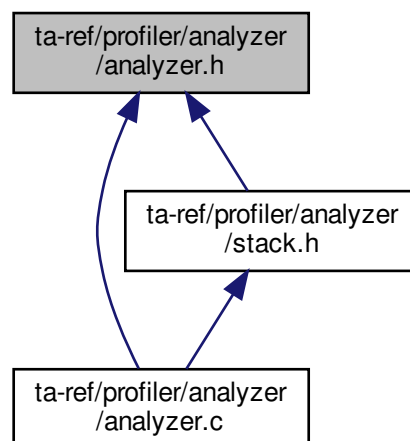
```
#include "profiler_data.h"
```

```
#include <stdint.h>
```

Include dependency graph for analyzer.h:



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



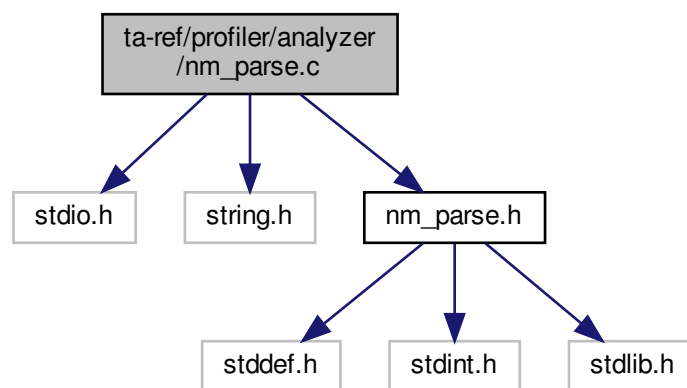
Classes

- struct [result](#)

10.67 ta-ref/profiler/analyzer/nm_parse.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "nm_parse.h"
```

Include dependency graph for nm_parse.c:



Macros

- #define [BUF_SIZE](#) 512
- #define [POOL_SIZE](#) 30000
- #define [MAX_ADDR](#) 0xFFFFFFFF

Functions

- static struct [list](#) * [create_htable](#) (void)
- static size_t [get_key](#) (unsigned long addr)
- const char * [get_func_name](#) (struct [list](#) *table, unsigned long addr)
- static void [insert_nm](#) (struct [list](#) *table, unsigned long addr, struct [nm_info](#) *nm)
- struct [list](#) * [parse_nm](#) (const char *fname)

Variables

- static struct [nm_info](#) [nm_pool](#) [[POOL_SIZE](#)]
- static int [idx](#) = 0

10.67.1 Macro Definition Documentation

10.67.1.1 BUF_SIZE `#define BUF_SIZE 512`

10.67.1.2 MAX_ADDR `#define MAX_ADDR 0xFFFFFFFF`

10.67.1.3 POOL_SIZE `#define POOL_SIZE 30000`

10.67.2 Function Documentation

10.67.2.1 create_htable() `static struct list* create_htable (`
`void) [static]`

[create_htable\(\)](#) - Creates the hash table which stores data in an associative manner.

This function returns the hash table where the data is stored in an array format.

Returns

list Updated structure list returns if success, else error occurred.

10.67.2.2 get_func_name() `const char* get_func_name (`
`struct list * table,`
`unsigned long addr)`

[get_func_name\(\)](#) - Returns the function name by assigning elements to it.

This function returns func.name if the element of address is equal to address of the get.key else returns NULL.

Parameters

<i>table</i>	It's an object of struct list.
<i>addr</i>	Address to find the key value.

Returns

String length If success, else error occurred.

```
10.67.2.3  get_key()  static size_t get_key (
                unsigned long addr )  [static]
```

[get_key\(\)](#) - Returns the address of the hash key.

This function it returns the modulo operator of address and hash size of the pointer.

Parameters

<i>addr</i>	Address of the key value.
-------------	---------------------------

Returns

Address of the hash key If success, else error occurred.

```
10.67.2.4  insert_nm()  static void insert_nm (
                struct list * table,
                unsigned long addr,
                struct nm_info * nm )  [static]
```

[insert_nm\(\)](#) - Inserts the element into the list.

This function is to insert the element inside the list.

Parameters

<i>table</i>	It's an object of struct list.
<i>addr</i>	Address of the key value.
<i>nm</i>	Name of the information of struct nm_info

```
10.67.2.5  parse_nm()  struct list* parse_nm (
                const char * fname )
```

[parse_nm\(\)](#) - Returns the table of the list structure.

This function opens the file and checks if the file is empty or not. If the file is not empty then it reads a line from the file pointer(fp) and stores it into the line. Function name copies to the network pool, and then inserts the network monitor.

Parameters

<i>fname</i>	File name.
--------------	------------

Returns

Updated structure list If success, else error occurred.

10.67.3 Variable Documentation

10.67.3.1 idx `int idx = 0 [static]`

10.67.3.2 nm_pool `struct nm_info nm_pool[POOL_SIZE] [static]`

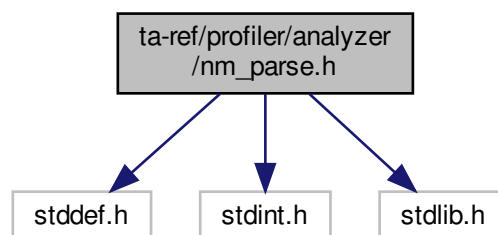
10.68 ta-ref/profiler/analyzer/nm_parse.h File Reference

```
#include <stddef.h>
```

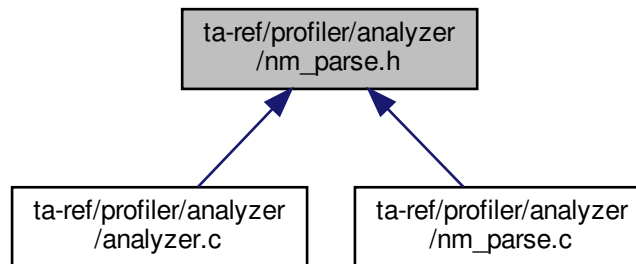
```
#include <stdint.h>
```

```
#include <stdlib.h>
```

Include dependency graph for nm_parse.h:



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



Classes

- struct [nm_info](#)
- struct [list](#)

Macros

- #define [HASH_SIZE](#) 65536

Functions

- const char * [get_func_name](#) (struct [list](#) *table, uintptr_t addr)
- struct [list](#) * [parse_nm](#) (const char *fname)

10.68.1 Macro Definition Documentation

10.68.1.1 [HASH_SIZE](#) #define [HASH_SIZE](#) 65536

10.68.2 Function Documentation

10.68.2.1 [get_func_name\(\)](#) const char* [get_func_name](#) (
 struct [list](#) * *table*,
 uintptr_t *addr*)

10.68.2.2 [parse_nm\(\)](#) struct [list](#)* [parse_nm](#) (
 const char * *fname*)

[parse_nm\(\)](#) - Returns the table of the list structure.

This function opens the file and checks if the file is empty or not. If the file is not empty then it reads a line from the file pointer(fp) and stores it into the line. Function name copies to the network pool, and then inserts the network monitor.

Parameters

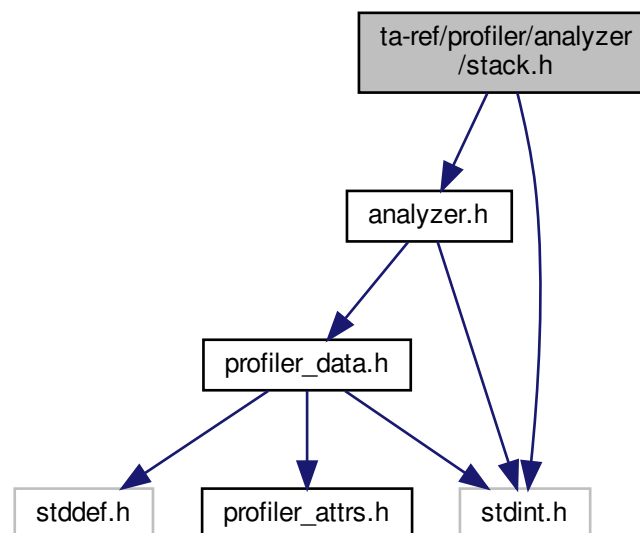
<i>fname</i>	File name.
--------------	------------

Returns

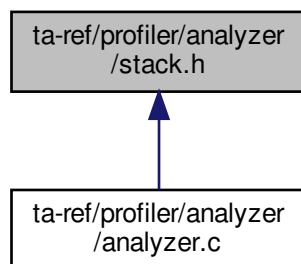
Updated structure list If success, else error occurred.

10.69 ta-ref/profiler/analyzer/stack.h File Reference

```
#include "analyzer.h"  
#include <stdint.h>  
Include dependency graph for stack.h:
```



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



Macros

- `#define STACK_SIZE 100`

Functions

- `struct result pop` (void)
- `void push` (struct result data)
- `char is_empty` (void)

Variables

- `static uint64_t pos = 0`
- `static struct result stack [STACK_SIZE]`

10.69.1 Macro Definition Documentation

10.69.1.1 STACK_SIZE `#define STACK_SIZE 100`

10.69.2 Function Documentation

10.69.2.1 is_empty() `char is_empty (void)`

10.69.2.2 pop() `struct result pop (`
`void)`

10.69.2.3 push() `void push (`
`struct result data)`

10.69.3 Variable Documentation

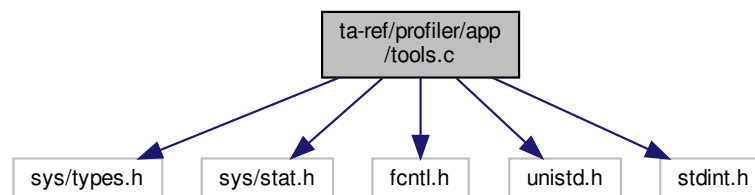
10.69.3.1 pos `uint64_t pos = 0 [static]`

10.69.3.2 stack `struct result stack[STACK_SIZE] [static]`

10.70 ta-ref/profiler/app/tools.c File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdint.h>
```

Include dependency graph for tools.c:



Functions

- `int profiler_write (void *ptr, uint64_t sz)`

10.70.1 Function Documentation

10.70.1.1 profiler_write() `int profiler_write (`
`void * ptr,`
`uint64_t sz)`

`profiler_write()` - Performs the file operations like open, write and close.

This function performs the three actions - opens the log file, writes into file and closes the file. It returns 0 when the file performance is done. Upon the failure of file it returns -1.

Parameters

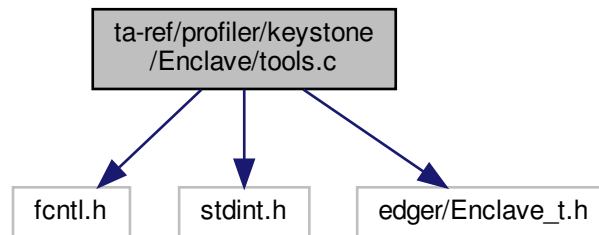
<i>ptr</i>	This is the pointer to the array of elements to be written.
<i>sz</i>	This is the size in bytes of each element to be written.

Returns

0 If success, else error occurred.

10.71 ta-ref/profiler/keystone/Enclave/tools.c File Reference

```
#include <fcntl.h>
#include <stdint.h>
#include "edger/Enclave_t.h"
Include dependency graph for tools.c:
```

**Functions**

- int [profiler.write](#) (void *ptr, uint64_t sz)

10.71.1 Function Documentation

10.71.1.1 profiler.write() int profiler.write (

```
void * ptr,
uint64_t sz )
```

[profiler.write\(\)](#) - Performs the file operations like open, write and close.

This function performs the three actions - open the log file, write into the file, and closes the file. It returns 0 when the file performance is done. Upon the failure of file it returns -1.

Parameters

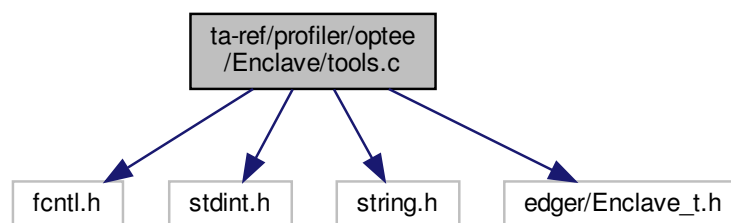
<i>ptr</i>	This is the pointer to the array of elements to be written.
<i>sz</i>	This is the size in bytes of each element to be written.

Returns

0 If success, else error occurred.

10.72 ta-ref/profiler/optee/Enclave/tools.c File Reference

```
#include <fcntl.h>
#include <stdint.h>
#include <string.h>
#include "edger/Enclave_t.h"
Include dependency graph for tools.c:
```

**Functions**

- int [profiler.write](#) (char *buf, void *ptr, uint64_t sz)

10.72.1 Function Documentation

10.72.1.1 profiler.write() int profiler.write (

```

    char * buf,
    void * ptr,
    uint64_t sz )
```

[profiler.write\(\)](#) - Copies the size of the pointer into the buffer.

This function calls the memmove(), where a block of memory is copied from one location to another.

Parameters

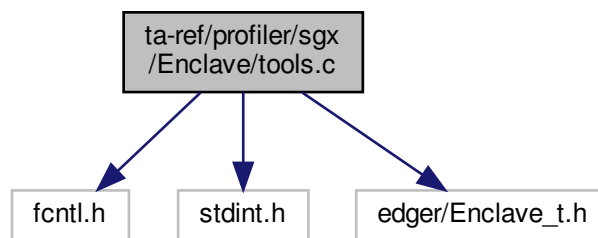
<i>buf</i>	This is a pointer to the destination array where the content is to be copied,
<i>ptr</i>	This is a pointer to the source of data to be copied,
<i>sz</i>	This is the number of bytes to be copied.

Returns

0 If success, else error occurred.

10.73 ta-ref/profiler/sgx/Enclave/tools.c File Reference

```
#include <fcntl.h>
#include <stdint.h>
#include "edger/Enclave_t.h"
Include dependency graph for tools.c:
```

**Functions**

- int [profiler.write](#) (void *ptr, uint64_t sz)

10.73.1 Function Documentation

10.73.1.1 profiler.write() int profiler.write (

```
void * ptr,
uint64_t sz )
```

[profiler.write\(\)](#) - Write out the profiled data to an output file.

This function used for the open the file and writing the file and close the file operation performed.

Parameters

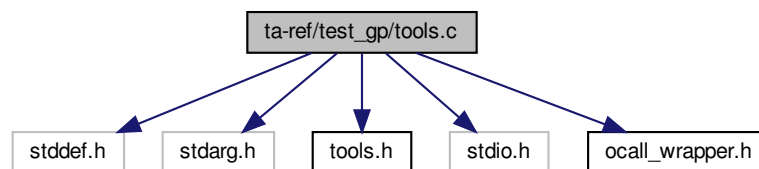
<i>ptr</i>	This is the pointer to the array of elements to be written.
<i>sz</i>	This is the size in bytes of each element to be written.

Returns

0 If success, else error ocured.

10.74 ta-ref/test_gp/tools.c File Reference

```
#include <stddef.h>
#include <stdarg.h>
#include "tools.h"
#include <stdio.h>
#include "ocall_wrapper.h"
Include dependency graph for tools.c:
```



Functions

- static unsigned int [_strlen](#) (const char *str)
- int [puts](#) (const char *s)
- int [putchar](#) (int c)
- int [printf](#) (const char *fmt,...)

10.74.1 Function Documentation

10.74.1.1 [_strlen\(\)](#) static unsigned int [_strlen](#) (
const char * *str*) [inline], [static]

10.74.1.2 [printf\(\)](#) int [printf](#) (
const char * *fmt*,
...)

[printf\(\)](#) - Function sends formatted output to stdout.

format can optionally contain embedded format tags that are replaced by the values specified in subsequent additional arguments and formatted as requested.

Parameters

<i>fm</i>	This is the string that contains the text to be written to stdout.
-----------	--

Returns

string length If success.

0 Error occurred.

10.74.1.3 putchar() `int putchar (`
`int c)`

[putchar\(\)](#) - Function writes a character (an unsigned char) specified by the argument char to stdout.

This function returns the character written as an unsigned char cast to an int or EOF on error.

Parameters

<i>c</i>	This is the character to be written. This is passed as its int promotion.
----------	---

Returns

size If success.

0 Error occurred.

10.74.1.4 puts() `int puts (`
`const char * s)`

[puts\(\)](#) - Function writes a string to stdout up to but not including the null character.

A newline character is appended to the output by calling [putchar\(\)](#). Compiler may replace simple printf to puts and putchar.

Parameters

<i>s</i>	This is the C string to be written
----------	------------------------------------

Returns

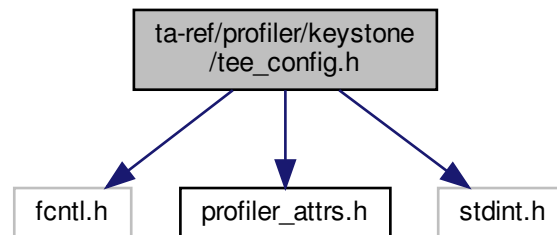
size If success.

0 Error occurred.

10.75 ta-ref/profiler/keystone/tee_config.h File Reference

```
#include <fcntl.h>
#include "profiler_attrs.h"
#include <stdint.h>
```

Include dependency graph for tee_config.h:



Functions

- static uint64_t [NO_PERF](#) [tee_rdtscp](#) (uint8_t *id)

Variables

- static uintptr_t [__ImageBase](#) = 0
- static char [PERF_SECTION](#) [perf_buffer](#) [[PERF_SIZE](#)]

10.75.1 Function Documentation

10.75.1.1 tee_rdtscp() static uint64_t [NO_PERF](#) tee_rdtscp (uint8_t * id) [inline], [static]

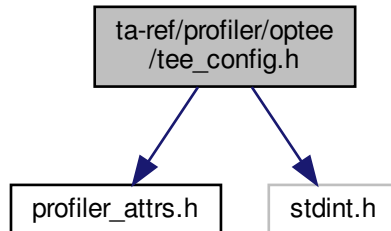
10.75.2 Variable Documentation

10.75.2.1 __ImageBase uintptr_t __ImageBase = 0 [static]

10.75.2.2 perf_buffer char [PERF_SECTION](#) perf_buffer[[PERF_SIZE](#)] [static]

10.76 ta-ref/profiler/optee/tee_config.h File Reference

```
#include "profiler_attrs.h"
#include <stdint.h>
Include dependency graph for tee_config.h:
```



Macros

- `#define` `COMMAND` `"mrs %0, cntpct_el0"`

Functions

- `static uint64_t` `NO_PERF tee_rdtscp` (`uint8_t *id`)

Variables

- `uintptr_t` `__ImageBase` []
- `static char` `perf_buffer` [`PERF_SIZE`]

10.76.1 Macro Definition Documentation

10.76.1.1 COMMAND `#define COMMAND "mrs %0, cntpct_el0"`

10.76.2 Function Documentation

10.76.2.1 tee_rdtscp() `static uint64_t NO_PERF tee_rdtscp (uint8_t * id) [inline], [static]`

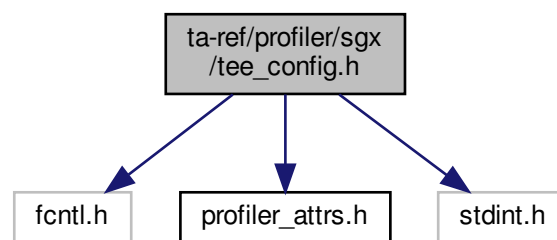
10.76.3 Variable Documentation

10.76.3.1 `__ImageBase` `uintptr_t __ImageBase[]` [extern]

10.76.3.2 `perf_buffer` `char perf_buffer[PERF_SIZE]` [static]

10.77 ta-ref/profiler/sgx/tee_config.h File Reference

```
#include <fcntl.h>
#include "profiler_attrs.h"
#include <stdint.h>
Include dependency graph for tee_config.h:
```



Functions

- static uint64_t `tee_rdtscp` (uint8_t *id)

Variables

- uintptr_t `__ImageBase` []
- static char `perf_buffer` [PERF_SIZE]

10.77.1 Function Documentation

10.77.1.1 `tee_rdtscp()` `static uint64_t tee_rdtscp (uint8_t * id)` [inline], [static]

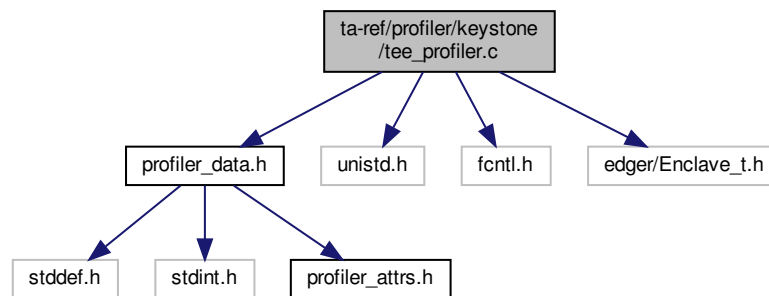
10.77.2 Variable Documentation

10.77.2.1 `__ImageBase` `uintptr_t __ImageBase[]` [extern]

10.77.2.2 `perf_buffer` `char perf_buffer[PERF_SIZE]` [static]

10.78 ta-ref/profiler/keystone/tee_profiler.c File Reference

```
#include "profiler_data.h"
#include <unistd.h>
#include <fcntl.h>
#include "edger/Enclave_t.h"
Include dependency graph for tee_profiler.c:
```



Functions

- `int profiler_write` (`void *ptr`, `uint64_t sz`)
- `void NO_PERF __profiler_unmap_info` (`void`)

Variables

- `struct __profiler_header * __profiler_head`

10.78.1 Function Documentation

10.78.1.1 `__profiler_unmap_info()` void NO_PERF __profiler_unmap_info (
void)

`__profiler_unmap_info()` - Write out the profiled data to an output file.

If the `__profiler_head` is not null then it returns the output file.

10.78.1.2 `profiler_write()` int profiler_write (
void * ptr,
uint64_t sz)

`profiler_write()` - Performs the file operations like open, write and close.

This function performs the three actions - opens the log file, writes into file and closes the file. It returns 0 when the file performance is done. Upon the failure of file it returns -1.

Parameters

<i>ptr</i>	This is the pointer to the array of elements to be written.
<i>sz</i>	This is the size in bytes of each element to be written.

Returns

0 If success, else error occurred.

`profiler_write()` - Performs the file operations like open, write and close.

This function performs the three actions - open the log file, write into the file, and closes the file. It returns 0 when the file performance is done. Upon the failure of file it returns -1.

Parameters

<i>ptr</i>	This is the pointer to the array of elements to be written.
<i>sz</i>	This is the size in bytes of each element to be written.

Returns

0 If success, else error occurred.

`profiler_write()` - Write out the profiled data to an output file.

This function used for the open the file and writing the file and close the file operation performed.

Parameters

<i>ptr</i>	This is the pointer to the array of elements to be written.
<i>sz</i>	This is the size in bytes of each element to be written.

Returns

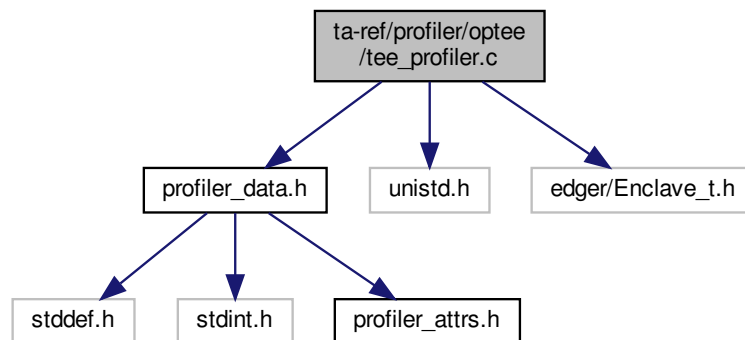
0 If success, else error occurred.

10.78.2 Variable Documentation

10.78.2.1 `__profiler_head` `struct __profiler_header* __profiler_head` [extern]

10.79 ta-ref/profiler/optee/tee_profiler.c File Reference

```
#include "profiler_data.h"
#include <unistd.h>
#include "edger/Enclave_t.h"
Include dependency graph for tee_profiler.c:
```



Functions

- `int profiler_write` (`char *buf`, `void *ptr`, `uint64_t sz`)
- `void NO_PERF __profiler_unmap_info` (`char *buf`, `size_t *size`)

Variables

- `struct __profiler_header * __profiler_head`

10.79.1 Function Documentation

10.79.1.1 `__profiler_unmap_info()` `void NO_PERF __profiler_unmap_info (`
`char * buf,`
`size_t * size)`

`__profiler_unmap_info()` - Write out the profiled data to an output file.

If the `__profiler_head` is not null then returns the output file.

Parameters

<i>buf</i>	It copies the read string into the buffer <i>buf</i>
<i>size</i>	This is the size in bytes of each element to be written.

10.79.1.2 profiler_write() `int profiler_write (`
 `char * buf,`
 `void * ptr,`
 `uint64_t sz)`

[profiler_write\(\)](#) - Copies the size of the pointer into the buffer.

This function calls the `memmove()`, where a block of memory is copied from one location to another.

Parameters

<i>buf</i>	This is a pointer to the destination array where the content is to be copied,
<i>ptr</i>	This is a pointer to the source of data to be copied,
<i>sz</i>	This is the number of bytes to be copied.

Returns

0 If success, else error occurred.

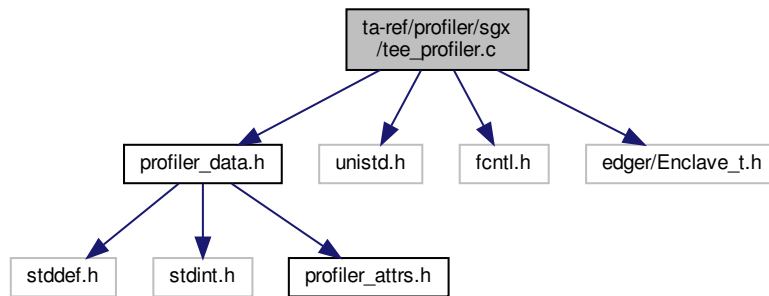
10.79.2 Variable Documentation

10.79.2.1 __profiler_head `struct __profiler_header* __profiler_head [extern]`

10.80 ta-ref/profiler/sgx/tee_profiler.c File Reference

```
#include "profiler_data.h"
#include <unistd.h>
#include <fcntl.h>
```

```
#include "edger/Enclave_t.h"
Include dependency graph for tee_profiler.c:
```



Functions

- int [profiler.write](#) (void *ptr, uint64_t sz)
- void [NO_PERF __profiler_unmap_info](#) (void)

Variables

- struct [__profiler_header](#) * [__profiler_head](#)

10.80.1 Function Documentation

10.80.1.1 [__profiler_unmap_info\(\)](#) void [NO_PERF __profiler_unmap_info](#) (void)

[__profiler_unmap_info\(\)](#) - Unmap the profile.

This function used for find the size of file and writing the updated file.

10.80.1.2 [profiler.write\(\)](#) int [profiler.write](#) (void * ptr, uint64_t sz)

[profiler.write\(\)](#) - Performs the file operations like open, write and close.

This function performs the three actions - opens the log file, writes into file and closes the file. It returns 0 when the file performance is done. Upon the failure of file it returns -1.

Parameters

<i>ptr</i>	This is the pointer to the array of elements to be written.
<i>sz</i>	This is the size in bytes of each element to be written.

Returns

0 If success, else error occurred.

[profiler_write\(\)](#) - Performs the file operations like open, write and close.

This function performs the three actions - open the log file, write into the file, and closes the file. It returns 0 when the file performance is done. Upon the failure of file it returns -1.

Parameters

<i>ptr</i>	This is the pointer to the array of elements to be written.
<i>sz</i>	This is the size in bytes of each element to be written.

Returns

0 If success, else error occurred.

[profiler_write\(\)](#) - Write out the profiled data to an output file.

This function used for the open the file and writing the file and close the file operation performed.

Parameters

<i>ptr</i>	This is the pointer to the array of elements to be written.
<i>sz</i>	This is the size in bytes of each element to be written.

Returns

0 If success, else error occurred.

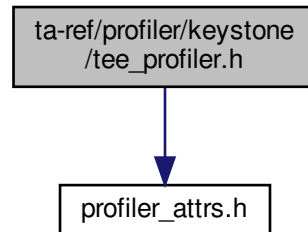
10.80.2 Variable Documentation

10.80.2.1 `__profiler_head` `struct __profiler_header* __profiler_head [extern]`

10.81 ta-ref/profiler/keystone/tee_profiler.h File Reference

```
#include "profiler_attrs.h"
```

Include dependency graph for tee_profiler.h:



Functions

- void [NO_PERF __profiler_unmap_info](#) (void)

10.81.1 Function Documentation

10.81.1.1 [__profiler_unmap_info\(\)](#) void [NO_PERF __profiler_unmap_info](#) (
void)

[__profiler_unmap_info\(\)](#) - Write out the profiled data to an output file.

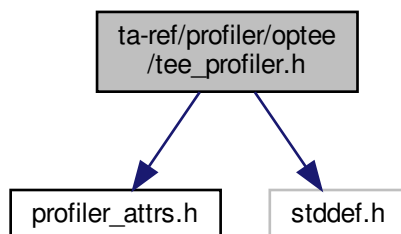
If the `__profiler_head` is not null then it returns the output file.

[__profiler_unmap_info\(\)](#) - Unmap the profile.

This function used for find the size of file and writing the updated file.

10.82 ta-ref/profiler/optee/tee_profiler.h File Reference

```
#include "profiler_attrs.h"
#include <stddef.h>
Include dependency graph for tee_profiler.h:
```



Functions

- void `NO_PERF __profiler_unmap_info` (char *buf, size_t *size)

10.82.1 Function Documentation

10.82.1.1 `__profiler_unmap_info()` void `NO_PERF __profiler_unmap_info` (

```
char * buf,
size_t * size )
```

`__profiler_unmap_info()` - Write out the profiled data to an output file.

If the `__profiler_head` is not null then returns the output file.

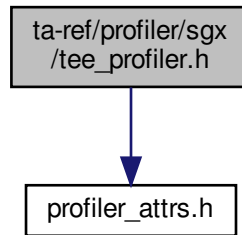
Parameters

<i>buf</i>	It copies the read string into the buffer buf
<i>size</i>	This is the size in bytes of each element to be written.

10.83 ta-ref/profiler/sgx/tee_profiler.h File Reference

```
#include "profiler_attrs.h"
```

Include dependency graph for tee_profiler.h:



Functions

- void [NO_PERF __profiler_unmap_info](#) (void)

10.83.1 Function Documentation

10.83.1.1 [__profiler_unmap_info\(\)](#) void [NO_PERF __profiler_unmap_info](#) (
void)

[__profiler_unmap_info\(\)](#) - Write out the profiled data to an output file.

If the `__profiler_head` is not null then it returns the output file.

[__profiler_unmap_info\(\)](#) - Unmap the profile.

This function used for find the size of file and writing the updated file.

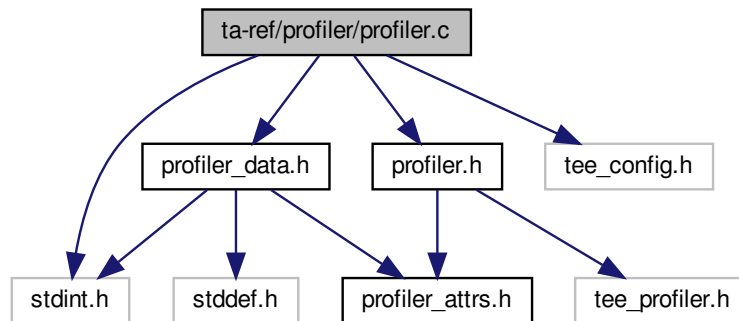
10.84 ta-ref/profiler/profiler.c File Reference

```
#include <stdint.h>
#include "profiler.h"
#include "profiler_data.h"
```



```
#include "tee_config.h"
```

Include dependency graph for profiler.c:



Functions

- static void [NO_PERF __cyg_profile_func](#) (void *const this_fn, enum [direction_t](#) const dir)
- static struct [__profiler_data](#) *const [NO_PERF __profiler_get_data_ptr](#) (void)
- void [NO_PERF __profiler_map_info](#) (void)
- void [NO_PERF USED __cyg_profile_func_enter](#) (void *this_fn, void *call_site)
- void [NO_PERF USED __cyg_profile_func_exit](#) (void *this_fn, void *call_site)

Variables

- struct [__profiler_header](#) * [__profiler_head](#) = NULL

10.84.1 Function Documentation

10.84.1.1 [__cyg_profile_func\(\)](#) static void [NO_PERF __cyg_profile_func](#) (
 void *const *this_fn*,
 enum [direction_t](#) const *dir*) [inline], [static]

[__cyg_profile_func\(\)](#) - Defines the function for the entry and exit function operations.

Parameters

<i>this↔_fn</i>	A keyword that refers to the current instance of the class.
<i>dir</i>	An enumeration constant.

10.84.1.2 `__cyg_profile_func_enter()` `void NO_PERF USED __cyg_profile_func_enter (`
 `void * this_fn,`
 `void * call_site)`

[__cyg_profile_func_enter\(\)](#) - Performs entry operation

This function is called after entering the function [__cyg_profile_func\(\)](#).

Parameters

<i>this_fn</i>	A keyword that refers to the current instance of the class.
<i>call_site</i>	It means which operation performs for calling, start etc.

10.84.1.3 `__cyg_profile_func_exit()` `void NO_PERF USED __cyg_profile_func_exit (`
 `void * this_fn,`
 `void * call_site)`

[__cyg_profile_func_exit\(\)](#) - Performs exit operation.

This function is called after exiting from the function [__cyg_profile_func\(\)](#).

Parameters

<i>this_fn</i>	A keyword that refers to the current instance of the class.
<i>call_site</i>	It means which operation performs calling, stop etc.

10.84.1.4 `__profiler_get_data_ptr()` `static struct __profiler_data* const NO_PERF __profiler_get_data_ptr (`
 `void) [inline], [static]`

[__profiler_get_data_ptr\(\)](#) - Gets the profiler data from an output file.

Returns

Result If success.

10.84.1.5 `__profiler_map_info()` `void NO_PERF __profiler_map_info (`
 `void)`

[__profiler_map_info\(\)](#) - Maps the profile information.

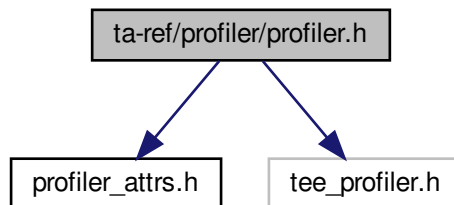
This function creates the new data value in the header of profiler.

10.84.2 Variable Documentation

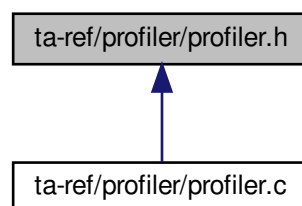
10.84.2.1 `__profiler_head` `struct __profiler_header* __profiler_head = NULL`

10.85 ta-ref/profiler/profiler.h File Reference

```
#include "profiler_attrs.h"
#include "tee_profiler.h"
Include dependency graph for profiler.h:
```



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



Functions

- void `NO_PERF __profiler_map_info` (void)

10.85.1 Function Documentation

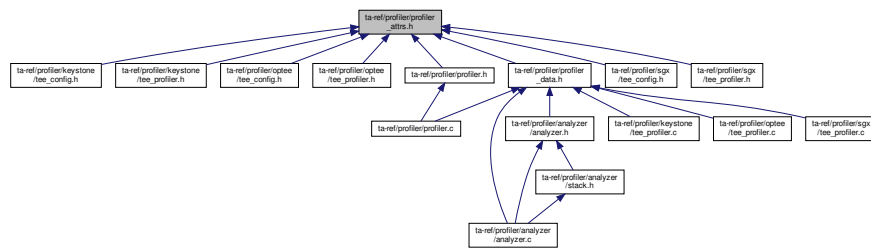
10.85.1.1 `__profiler_map_info()` void NO_PERF __profiler_map_info (void)

`__profiler_map_info()` - Maps the profile information.

This function creates the new data value in the header of profiler.

10.86 ta-ref/profiler/profiler_attr.h File Reference

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



Macros

- `#define NO_PERF __attribute__((no_instrument_function,hot))`
- `#define PERF_SECTION __attribute__((section(".perf_region")))`
- `#define USED __attribute__((used))`

10.86.1 Macro Definition Documentation

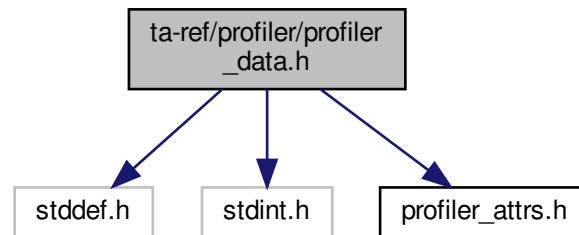
10.86.1.1 NO_PERF `#define NO_PERF __attribute__((no_instrument_function,hot))`

10.86.1.2 PERF_SECTION `#define PERF_SECTION __attribute__((section(".perf_region")))`

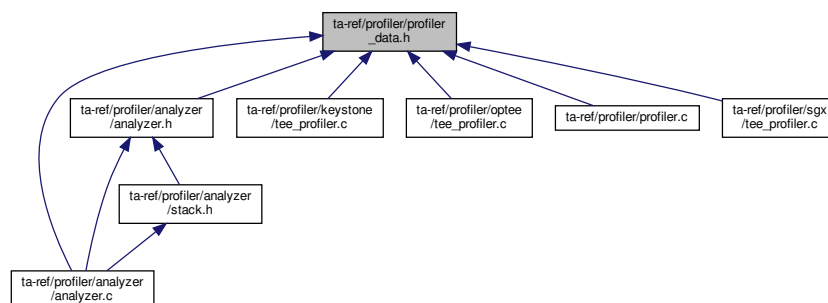
10.86.1.3 USED `#define USED __attribute__((used))`

10.87 ta-ref/profiler/profiler_data.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "profiler_attrs.h"
Include dependency graph for profiler_data.h:
```



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



Classes

- struct __profiler_data
- struct __profiler_header

Macros

- #define LOG_FILE "/root"
- #define PERF_SIZE 8192

Typedefs

- typedef uint64_t __profiler_nsec_t

Enumerations

- enum `direction_t` { `START` = 0 , `CALL` = 1 , `RET` = 2 }

Functions

- struct `__profiler_header __attribute__((packed, aligned(8)))`

Variables

- uint64_t `size`
- uint64_t `idx`
- uintptr_t `start`

10.87.1 Macro Definition Documentation

10.87.1.1 LOG_FILE `#define LOG_FILE "/root"`

10.87.1.2 PERF_SIZE `#define PERF_SIZE 8192`

10.87.2 Typedef Documentation

10.87.2.1 __profiler_nsec_t `typedef uint64_t __profiler_nsec_t`

10.87.3 Enumeration Type Documentation

10.87.3.1 direction.t `enum direction_t`

Enumerator

START	
CALL	
RET	

10.87.4 Function Documentation

10.87.4.1 `__attribute__()` `struct __profiler_header __attribute__ (`
`(packed, aligned(8)))`

10.87.5 Variable Documentation

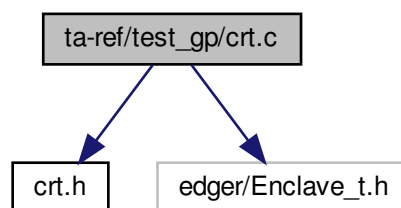
10.87.5.1 `idx` `uint64_t idx`

10.87.5.2 `size` `uint64_t size`

10.87.5.3 `start` `uintptr_t start`

10.88 ta-ref/test_gp/crt.c File Reference

```
#include "crt.h"  
#include "edger/Enclave_t.h"  
Include dependency graph for crt.c:
```



Functions

- void `crt_end` (void)

Variables

- static void(*const [init_array](#) [])() [__attribute__\(\(section\(".init_array"\)\)\)](#)
- static void(*const [aligned](#) []) (sizeof(void *))
- static void(*const [fini_array](#) [])() [__attribute__\(\(section\(".fini_array"\)\)\)](#)
- void(* [__init_array_start](#) []) (void)

10.88.1 Function Documentation

10.88.1.1 [crt_end\(\)](#) void crt_end (void)

[crt_end\(\)](#) - Ends the certification.

It compares `__fini_array_start` and `__fini_array_end`; and then it the loops through the file pointer.

10.88.2 Variable Documentation

10.88.2.1 [__init_array_start](#) void(* [__init_array_start](#)[]) (void) (void) [extern]

[crt_begin\(\)](#) - Commences the certification.

It compares `__init_array_start` and `__init_array_end`; and then it the loops through the file pointer.

10.88.2.2 [aligned](#) void(*const [aligned](#)[]) (sizeof(void *)) (sizeof(void *))

Initial value:

```
= {
}
```

10.88.2.3 [fini_array](#) void(*const [fini_array](#)[]) () [__attribute__\(\(section\(".fini_array"\)\)\)](#) () [static]

Termination array for the executable.

This section holds an array of function pointers that contributes to a single termination array for the executable or shared object containing the section and if defined is `PERF_ENABLE` then unmapping the profiler information.

Parameters

<i>fini_array[]</i>	constant array.
---------------------	-----------------

10.88.2.4 init_array void(*const init_array[])() `__attribute__((section(".init_array") () [static]`

Initialization array for the executable.

This section holds an array of function pointers that contributes to a single initialization array for the executable or shared object containing the section if defined is PERF_ENABLE then mapping the profiler information.

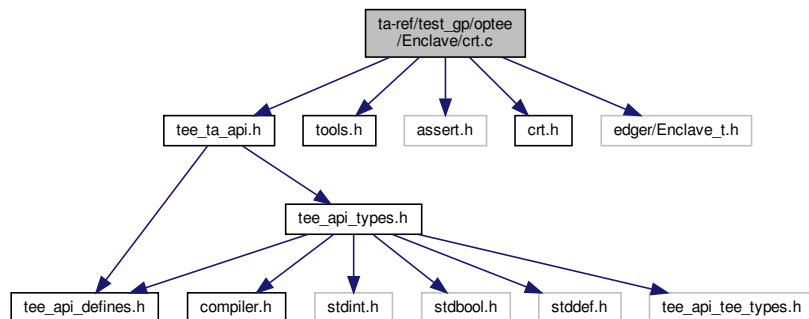
Parameters

<code>init_array[]</code>	constant array.
---------------------------	-----------------

10.89 ta-ref/test_gp/optee/Enclave/crt.c File Reference

```
#include <tee_ta_api.h>
#include "tools.h"
#include "assert.h"
#include "crt.h"
#include "edger/Enclave_t.h"
```

Include dependency graph for crt.c:



Macros

- `#define TEE_PARAM_TYPE0 TEE_PARAM_TYPE_NONE`
- `#define TEE_PARAM_TYPE1 TEE_PARAM_TYPE_NONE`

Functions

- int `tee_printf` (const char *fmt,...)
- `TEE_Result TA_CreateEntryPoint` (void)
- `TEE_Result TA_OpenSessionEntryPoint` (uint32_t `__unused` param_types, `TEE_Param` `__unused` params[4], void `__unused` **sess_ctx)
- void `TA_DestroyEntryPoint` (void)

- `TEE_Result run_all_test` (`uint32_t param_types`, `TEE_Param __maybe_unused params[4]`, `void __maybe_unused **sess_ctx`)
- `void TA_CloseSessionEntryPoint` (`void __maybe_unused *sess_ctx`)
- `TEE_Result TA_InvokeCommandEntryPoint` (`void *sess_ctx`, `uint32_t cmd_id`, `uint32_t param_types`, `TEE_Param params[4]`)

Variables

- `uintptr_t __ImageBase []`

10.89.1 Macro Definition Documentation

10.89.1.1 TEE_PARAM_TYPE0 `#define TEE_PARAM_TYPE0 TEE_PARAM_TYPE_NONE`

10.89.1.2 TEE_PARAM_TYPE1 `#define TEE_PARAM_TYPE1 TEE_PARAM_TYPE_NONE`

10.89.2 Function Documentation

10.89.2.1 run_all_test() `TEE_Result run_all_test (`
`uint32_t param_types,`
`TEE_Param __maybe_unused params[4],`
`void __maybe_unused ** sess_ctx)`

`run_all_test()` - Run all the tests in TA.

Verify the param types and if the defined macro is `PERF_ENABLE` then print the "enclave ELF address". If the defined macro is `ENCLAVE_VERBOSE`, print the message "ecall.ta_main() start" and invoke the `main()` function. If invoking the main function is a success, print the message "ecall.ta_main() end".

Parameters

<i>param_types</i>	The types of the four parameters.
<i>params[4]</i>	A pointer to an array of four parameters.
<i>sess_ctx</i>	A pointer to a variable that can be filled by the Trusted Application instance with an opaque void* data pointer

Returns

TEE_SUCCESS If the command is successfully executed, else error is occurred in the function.

10.89.2.2 TA_CloseSessionEntryPoint() `void TA_CloseSessionEntryPoint (`
`void __maybe_unused * sess_ctx)`

[TA_CloseSessionEntryPoint\(\)](#) - Closes the client session.

This function is to be called when a session is to be closed, The parameter to be passed is sess_ctx which holds the value assigned by [TA_OpenSessionEntryPoint\(\)](#). If the function succeeds in closing the session a message is printed as Goodbye!.

Parameters

<code>sess_ctx</code>	A pointer to a variable that can be filled by the Trusted Application instance with an opaque void* data pointer.
-----------------------	---

10.89.2.3 TA_CreateEntryPoint() `TEE_Result TA_CreateEntryPoint (`
`void)`

[TA_CreateEntryPoint\(\)](#) - The function creates the entry point of TA(Trusted Application).

This function is to be called when the instance of the TA is created. This is the first call in the TA and the displayed message should be "has been called".

Returns

TEE_SUCCESS If the command is successfully executed,else error occurred.

10.89.2.4 TA_DestroyEntryPoint() `void TA_DestroyEntryPoint (`
`void)`

[TA_DestroyEntryPoint\(\)](#) - Destroy entry point with TA.

This function is to be called, when the instance of the TA is destroyed. This is the last call in the TA and the displayed message should be "has been called".

10.89.2.5 TA_InvokeCommandEntryPoint() `TEE_Result TA_InvokeCommandEntryPoint (`
`void * sess_ctx,`
`uint32_t cmd_id,`
`uint32_t param_types,`
`TEE_Param params[4])`

[TA_InvokeCommandEntryPoint\(\)](#) - The Framework calls this function when the client invokes a command within the given session.

This function is to be called when a TA is invoked. When the client invokes the command within the given session and ,if switch case is TA_REF_RUN_ALL then invoke the [run_all_test\(\)](#) and sess.ctx holds the value assigned by [TA_OpenSessionEntryPoint\(\)](#). If the above operations are performed successfully by the function TEE_SUCCESS is returned.

Parameters

<i>param_types</i>	The types of the four parameters.
<i>params[4]</i>	A pointer to an array of four parameters.
<i>sess_ctx</i>	A pointer to a variable that can be filled by the Trusted Application instance with an opaque void* data pointer.

Returns

TEE_SUCCESS If the command is successfully executed,else error occurred.

10.89.2.6 TA_OpenSessionEntryPoint() `TEE_Result TA_OpenSessionEntryPoint (`
`uint32_t __unused param_types,`
`TEE_Param __unused params[4],`
`void __unused ** sess_ctx)`

[TA_OpenSessionEntryPoint\(\)](#) - The Framework calls this function when a client requests to open a session with the Trusted Application. This function takes parameters param_types and params used by the TA instance to transfer response data back to the client. If the reponse is tranferred successfully to the client TEE_SUCCESS is returned.

Parameters

<i>param_types</i>	This denotes the types of the four parameters.
<i>params[4]</i>	A pointer to an array of four parameters.
<i>sess_ctx</i>	A pointer to a variable that can be filled by the Trusted Application instance with an opaque void* data pointer

Returns

TEE_SUCCESS If the command is successfully executed, else error is occurred in the function.

```
10.89.2.7 tee_printf() int tee_printf (
    const char * fmt,
    ... )
```

[tee_printf\(\)](#) - Printing the formatted output in to a character array.

In this function the "@param ap" variable is initialized by calling `va_start()` and then formatted data will send to a string using argument list by calling [vsnprintf\(\)](#) and finally the string length will be stored in `res`.

Parameters

<i>fmt</i>	A string that specifies the format of the output.
------------	---

Returns

result If success, else error occurred.

[tee_printf\(\)](#) - For trace GP API.

Initializes `ap` variable. Formats data under control of the format control string and stores the result in `buf` and ends the processing of `ap`. Finally prints the buffer value.

Parameters

<i>fmt</i>	<code>fmt</code> is constant character argument of type pointer.
------------	--

Returns

`res` Based on the condition check it will return string length else returns 0.

[tee_printf\(\)](#) - Printing the formatted output in to a character array.

In this function the "@param ap" variable is initialized by calling `va_start()` and then formatted data will send to a string using argument list by calling [vsnprintf\(\)](#) and finally the string length will be stored in `res`.

Parameters

<i>fmt</i>	A string that specifies the format of the output.
------------	---

Returns

result If success, else error occurred.

[tee_printf\(\)](#) - For trace GP API.

Initializes `ap` variable. Formats data under control of the format control string and stores the result in `buf` and ends the processing of `ap`. Finally prints the buffer value.

Parameters

<i>fmt</i>	fmt is constant character argument of type pointer.
------------	---

Returns

res Based on the condition check it will return string length else returns 0.

[tee_printf\(\)](#) - For tracing GP API.

Initializes ap variable. Formats data under control of the format control string and stores the result in buf and ends the processing of ap. Finally print the buffer value.

Parameters

<i>fmt</i>	fmt is a constant character argument of type pointer.
------------	---

Returns

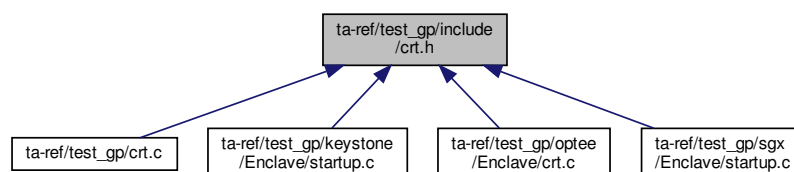
buffer If successfully executed, else error occurred.

10.89.3 Variable Documentation

10.89.3.1 `__ImageBase` `uintptr_t __ImageBase[]` [extern]

10.90 ta-ref/test_gp/include/crt.h File Reference

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

**Functions**

- void [crt_begin](#) (void)
- void [crt_end](#) (void)
- int [main](#) (void)

10.90.1 Function Documentation

10.90.1.1 crt_begin() `void crt_begin (`
`void)`

10.90.1.2 crt_end() `void crt_end (`
`void)`

[crt_end\(\)](#) - Ends the certification.

It compares `__fini_array_start` and `__fini_array_end`; and then it loops through the file pointer.

10.90.1.3 main() `int main (`
`void)`

[main\(\)](#) - To perform the TEEC operations for building TA inside TEE.

In this function the context is initialized for connecting to the TEE by calling [TEEC_InitializeContext\(\)](#). After initialization of context the session is opened on [TEEC_OpenSession\(\)](#) and then command is invoked in the TEE. Once the command is invoked the session is closed and the context is finalized. If the session is not opened properly, `session_failed` error appears.

Returns

0 If success, else displays error message.

This [main\(\)](#) function invokes the functions [gp_random_test\(\)](#) to generate random data [gp_ree_time_test\(\)](#) to retrieve the current REE system time [gp_trusted_time_test\(\)](#) to retrieve the current system time [gp_secure_storage_test\(\)](#) to create read and write the object data [gp_message_digest_test\(\)](#) to accumulate message data for hashing [gp_symmetric_key_enc_verify_test\(\)](#) to encrypt or decrypt input data [gp_symmetric_key_gcm_verify_test\(\)](#) to encrypt and decrypt in AE [gp_asymmetric_key_sign_test\(\)](#) for cryptographic Operations API message Digest Functions and returns the status as success when all the functions generates the same data.

Returns

return 0 for success.

[main\(\)](#) - Initializes a new TEE Context and opens a new Session.

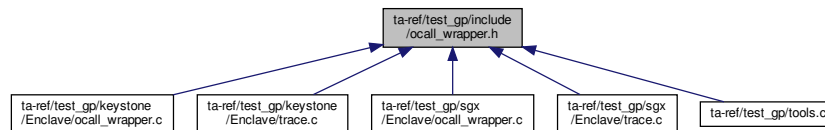
This function initializes a new TEE context and opens a new session between the client application and the specified trusted application. If initialization to a new TEE context and opening a new session are success then, first op(`↔` `TEEC.Operation`) characters of the string, are copied by the argument `&op`. If the macro is `PERF_ENABLE`, then assign the buffer and buffer size to `"params[0]"` and then open the log file for write. If the macro is `ENCLAVE_↔` `VERBOSE` then assign the buffer and buffer size to `"params[1]"`. Then print the "enclave log start" and "enclave log end". If macro is `APP_VERBOSE` then print the "start the invoke command" and invoke the [TEEC_InvokeCommand\(\)](#). If the [TEEC_InvokeCommand\(\)](#) is success then print the "TEEC.InvokeCommand succeeded!". If [TEEC_InvokeCommand\(\)](#) fails, Then print the message as "TEEC.InvokeCommand failed" with code message result and error origin. Finally close the session and destroy the initialized TEE context.

Returns

0 If the function is a success.

10.91 ta-ref/test_gp/include/ocall_wrapper.h File Reference

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



Functions

- unsigned int [ocall_print_string_wrapper](#) (const char *str)

10.91.1 Function Documentation

10.91.1.1 ocall_print_string_wrapper() unsigned int ocall_print_string_wrapper (const char * str)

[ocall_print_string_wrapper\(\)](#) - To print the argument string

This function invokes [ocall_print_string\(\)](#) to print the string.

Parameters

<i>str</i>	The string value for print.
------------	-----------------------------

Returns

string It prints the value of str by calling [ocall_print_string\(\)](#).

[ocall_print_string_wrapper\(\)](#) - To print the argument string

This function invokes [ocall_print_string\(\)](#) to print the string.

Parameters

<i>str</i>	The string value for print.
------------	-----------------------------

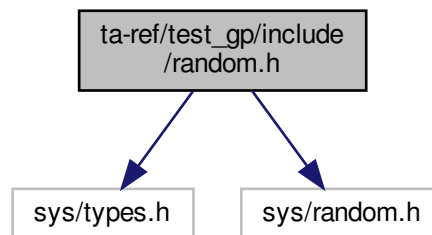
Returns

retval Its prints the value of str by calling `ocall_print_string()`.

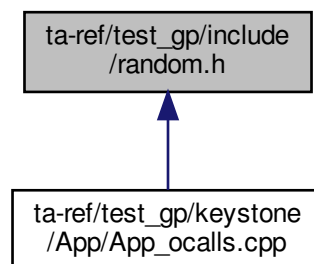
10.92 ta-ref/test_gp/include/random.h File Reference

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

Include dependency graph for random.h:

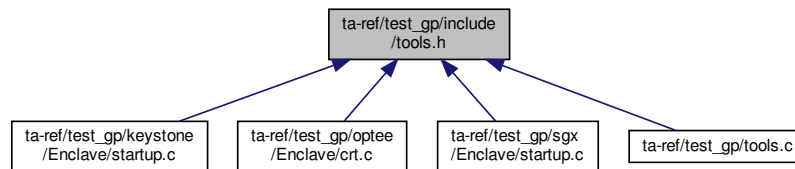


This graph shows which files directly or indirectly include this file:



10.93 ta-ref/test_gp/include/tools.h File Reference

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



Functions

- int [puts](#) (const char *s)
- int [putchar](#) (int c)
- int [printf](#) (const char *fmt,...)

10.93.1 Function Documentation

10.93.1.1 printf() int printf (
 const char * *fmt*,
 ...)

[printf\(\)](#) - Function sends formatted output to stdout.

format can optionally contain embedded format tags that are replaced by the values specified in subsequent additional arguments and formatted as requested.

Parameters

<i>fm</i>	This is the string that contains the text to be written to stdout.
-----------	--

Returns

string length If success.
 0 Error occured.

10.93.1.2 putchar() int putchar (
 int *c*)

[putchar\(\)](#) - Function writes a character (an unsigned char) specified by the argument char to stdout.

This function returns the character written as an unsigned char cast to an int or EOF on error.

Parameters

c	This is the character to be written. This is passed as its int promotion.
---	---

Returns

size If success.
0 Error occurred.

10.93.1.3 puts() `int puts (`
`const char * s)`

[puts\(\)](#) - Function writes a string to stdout up to but not including the null character.

A newline character is appended to the output by calling [putchar\(\)](#). Compiler may replace simple printf to puts and putchar.

Parameters

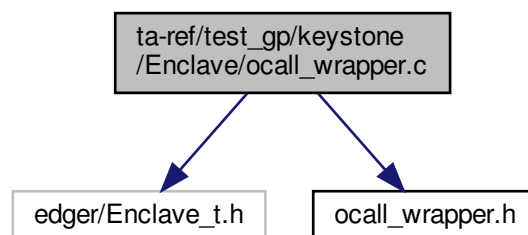
s	This is the C string to be written
---	------------------------------------

Returns

size If success.
0 Error occurred.

10.94 ta-ref/test_gp/keystone/Enclave/ocall_wrapper.c File Reference

```
#include "edger/Enclave_t.h"
#include "ocall_wrapper.h"
Include dependency graph for ocall_wrapper.c:
```



Functions

- unsigned int [ocall_print_string_wrapper](#) (const char *str)

10.94.1 Function Documentation

10.94.1.1 ocall_print_string_wrapper() unsigned int ocall_print_string_wrapper (
 const char * str)

[ocall_print_string_wrapper\(\)](#) - To print the argument string

This function invokes [ocall_print_string\(\)](#) to print the string.

Parameters

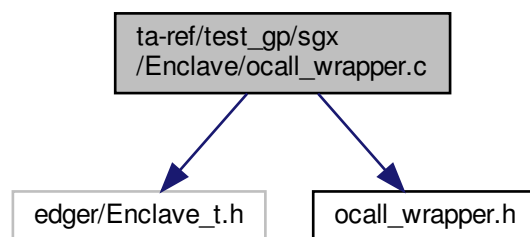
<i>str</i>	The string value for print.
------------	-----------------------------

Returns

string It prints the value of str by calling [ocall_print_string\(\)](#).

10.95 ta-ref/test_gp/sgx/Enclave/ocall_wrapper.c File Reference

```
#include "edger/Enclave_t.h"
#include "ocall_wrapper.h"
Include dependency graph for ocall_wrapper.c:
```



Functions

- unsigned int [ocall_print_string_wrapper](#) (const char *str)

10.95.1 Function Documentation

10.95.1.1 ocall_print_string_wrapper() unsigned int ocall_print_string_wrapper (
 const char * str)

[ocall_print_string_wrapper\(\)](#) - To print the argument string

This function invokes [ocall_print_string\(\)](#) to print the string.

Parameters

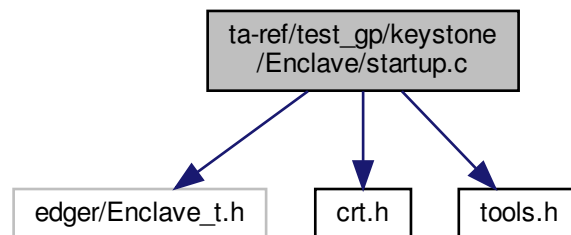
<i>str</i>	The string value for print.
------------	-----------------------------

Returns

retval Its prints the value of str by calling [ocall_print_string\(\)](#).

10.96 ta-ref/test_gp/keystone/Enclave/startup.c File Reference

```
#include "edger/Enclave_t.h"
#include "crt.h"
#include "tools.h"
Include dependency graph for startup.c:
```



Functions

- void EAPP_ENTRY [eapp_entry](#) ()

10.96.1 Function Documentation

10.96.1.1 `eapp_entry()` `void EAPP_ENTRY eapp_entry ()`

The `eapp_entry()` - It contains enclave verbose and invokes main function.

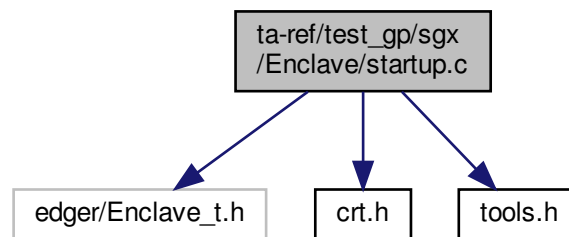
This function invokes `crt_begin()` if defined macro is `ENCLAVE_VERBOSE` then prints the main start and invokes `main()`. Once `main()` is completed prints the main end and invokes the `crt_end()`.

Returns

It will return `EAPP_RETURN(0)`.

10.97 `ta-ref/test_gp/sgx/Enclave/startup.c` File Reference

```
#include "edger/Enclave_t.h"
#include "crt.h"
#include "tools.h"
Include dependency graph for startup.c:
```



Functions

- void `ecall_ta_main` (void)

10.97.1 Function Documentation

10.97.1.1 `ecall_ta_main()` `void ecall_tamain (void)`

The `eapp_entry()` - It contains enclave verbose and invokes the main function.

This function invokes `crt_begin()` if defined macro is `ENCLAVE_VERBOSE` then prints the main start and invokes `main()`. Once `main()` is completed, it prints the main end and invokes the `crt_end()`.

Returns

It will return `EAPP_RETURN(0)`.

10.98 ta-ref/test_hello/keystone/App/App.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/random.h>
#include <fcntl.h>
#include <unistd.h>
#include <cstdio>
#include <string>
#include <cstring>
#include "edger/Enclave_u.h"
```

Include dependency graph for App.cpp:



Functions

- int [main](#) (int argc, char **argv)

Variables

- const char * [enc_path](#) = "Enclave.eapp_riscv"
- const char * [runtime_path](#) = "eyrie-rt"

10.98.1 Function Documentation

10.98.1.1 main() int main (int argc, char ** argv)

[main\(\)](#) - To start the enclave and run the enclave.

This function is to check the enclave initialization, if the enclave is not initialized then it prints the error message "unable to start enclave" and exit. If initialization is successful, it will go for the edge call initialization by calling `edge_call_init_internals()` before that the enclave must register the edge call handler and then the enclave will run and return 0.

Parameters

<i>argc</i>	Argument count is int and stores number of command-line arguments passed by the user including the name of the program.
<i>argv</i>	Argument Vector is array of character pointers listing all the arguments.

Returns

0 If success, else error occurred.

10.98.2 Variable Documentation

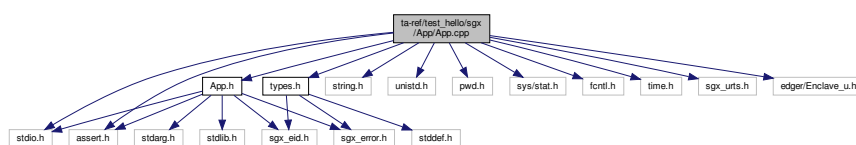
10.98.2.1 enc_path `const char* enc_path = "Enclave.eapp_riscv"`

10.98.2.2 runtime_path `const char* runtime_path = "eyrie-rt"`

10.99 ta-ref/test_hello/sgx/App/App.cpp File Reference

```
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include <unistd.h>
#include <pwd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>
#include "sgx_urts.h"
#include "App.h"
#include "edger/Enclave_u.h"
#include "types.h"
```

Include dependency graph for App.cpp:



Macros

- `#define MAX_PATH` `FILENAME_MAX`

Functions

- void `print_error_message` (sgx_status_t ret)
- int `initialize_enclave` (void)
- int `SGX_CDECL main` (int argc, char *argv[])

10.99.1 Macro Definition Documentation

10.99.1.1 MAX_PATH `#define MAX_PATH FILENAME_MAX`

10.99.2 Function Documentation

10.99.2.1 `initialize_enclave()` `int initialize_enclave (void)`

`initialize_enclave()` - Initializes an enclave by calling `sgx_create_enclave()`.

This function returns 0 on the success initialization of enclave. If enclave is not created properly then it will return -1 on error.

Returns

0 If success, else error occurred.

10.99.2.2 `main()` `int SGX_CDECL main (int argc, char * argv[])`

`main()` - Performs the enclave operation by creating and destroying enclave.

This function is used for initializing the enclave and calling TA inside the enclave. The enclave will destroy based on the success of TA.

Parameters

<code>argc</code>	Argument Count is int and stores number of command-line arguments passed by the user including the name of the program.
<code>argv</code>	Argument Vector is array of character pointers listing all the arguments.

Returns

0 If success, else error occurred.

10.99.2.3 print_error_message() void print_error_message (
 sgx_status_t ret)

[print_error_message\(\)](#) - Used for printing the error message.

This function prints the error message in sgx_errlist list and checks error conditions for loading enclave.

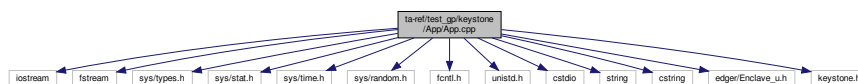
Parameters

<i>ret</i>	A list containing all possible values of sgx_status_t data type.
------------	--

10.100 ta-ref/test_gp/keystone/App/App.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/random.h>
#include <fcntl.h>
#include <unistd.h>
#include <cstdio>
#include <string>
#include <cstring>
#include "edger/Enclave_u.h"
#include "keystone.h"
```

Include dependency graph for App.cpp:



Functions

- int [main](#) (int argc, char **argv)

Variables

- const char * [enc_path](#) = "Enclave.eapp_riscv"
- const char * [runtime_path](#) = "eyrie-rt"

10.100.1 Function Documentation

```
10.100.1.1 main() int main (
                int argc,
                char ** argv )
```

`main()` - To start the enclave and run the enclave.

The function is to check the enclave initialization, If the enclave is not initialized then it will print the error message "unable to start enclave" and exit. If initialization is successful, it will go for the edge call initialization by calling `edge_call_init_internals()` and then the enclave will run and return 0.

Parameters

<i>argc</i>	Argument Count is int and stores number of command-line arguments passed by the user including the name of the program.
<i>argv</i>	Argument Vector is array of character pointers listing all the arguments.

Returns

0 If success, else error occurred.

10.100.2 Variable Documentation

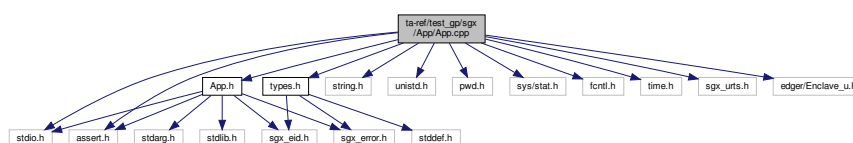
10.100.2.1 enc_path

10.100.2.2 runtime_path

10.101 ta-ref/test_gp/sgx/App/App.cpp File Reference

```
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include <unistd.h>
#include <pwd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>
#include "sgx_urts.h"
#include "App.h"
#include "edger/Enclave_u.h"
#include "types.h"
```

Include dependency graph for App.cpp:



Macros

- #define [MAX_PATH](#) FILENAME_MAX

Functions

- void [print_error_message](#) (sgx_status_t ret)
- int [initialize_enclave](#) (void)
- int SGX_CDECL [main](#) (int argc, char *argv[])

10.101.1 Macro Definition Documentation

10.101.1.1 [MAX_PATH](#) #define MAX_PATH FILENAME_MAX

10.101.2 Function Documentation

10.101.2.1 [initialize_enclave\(\)](#) int initialize_enclave (void)

[initialize_enclave\(\)](#) - Function initializes an enclave,

This function is used to create the enclave for sgx and if invoke's return value is equal to SGX_SUCCESS, then it will return the value zero, else it will print the error message.

Returns

0 If success else error occurred.

10.101.2.2 [main\(\)](#) int SGX_CDECL main (int argc, char * argv[])

[main\(\)](#) - Mapping and unmapping profile information.

If defined macro is APP_PERF_ENABLE then invoke the [__profiler_map_info\(\)](#) and [__profiler_unmap_info\(\)](#). It then initializes the enclave and Calls trusted application; if initialized enclave's return value is less than zero then it destroys the enclave.

- int [ocall_open_file](#) (const char *fname, int flags, int perm)
- int [ocall_close_file](#) (int fdesc)
- int [ocall_write_file](#) (int fdesc, const char *buf, unsigned int len)
- int [ocall_invoke_command_callback_write](#) (const char *str, const char *buf, unsigned int len)
- int [ocall_read_file](#) (int fdesc, char *buf, size_t len)
- int [ocall_ree_time](#) (struct [ree_time_t](#) *timep)
- ssize_t [ocall_getrandom](#) (char *buf, size_t len, unsigned int flags)
- [param_buffer_t](#) [ocall_read_invoke_param](#) (int index, unsigned int offset)
- void [ocall_write_invoke_param](#) (int index, unsigned int offset, unsigned int [size](#), const char *buf)
- void [ocall_put_invoke_command_result](#) ([invoke_command_t](#) cmd, unsigned int [result](#))

10.102.1 Function Documentation

10.102.1.1 [ocall_close_file\(\)](#) `int ocall_close_file (`
`int fdesc)`

[ocall_close_file\(\)](#) - To close a file.

Parameters

<i>fdesc</i>	file descriptor.
--------------	------------------

Returns

integer value If success

10.102.1.2 [ocall_getrandom\(\)](#) `ssize_t ocall_getrandom (`
`char * buf,`
`size_t len,`
`unsigned int flags)`

[ocall_getrandom\(\)](#) - To get random data.

Parameters

<i>buf</i>	Pointer of a buffer
<i>len</i>	length of buffer
<i>flags</i>	indicated permission.

Returns

integer value If success

10.102.1.3 ocall_invoke_command_callback_write() `int ocall_invoke_command_callback_write (`
 `const char * str,`
 `const char * buf,`
 `unsigned int len)`

[ocall_invoke_command_callback_write\(\)](#) - to write the invoke command for callback_write.

Parameters

<i>str</i>	pointer of a string.
<i>buf</i>	buffer to write data.
<i>len</i>	length of buffer.

Returns

integer value If success

10.102.1.4 ocall_open_file() `int ocall_open_file (`
 `const char * fname,`
 `int flags,`
 `int perm)`

[ocall_open_file\(\)](#) - To open a file.

Parameters

<i>fname</i>	name of the file.
<i>flags</i>	mode of the file.
<i>perm</i>	indicates permissions of a file.

Returns

integer If success

10.102.1.5 ocall_print_string() `EDGE_EXTERN_BEGIN unsigned int ocall_print_string (`
 `const char * str)`

[ocall_print_string\(\)](#) - To print the string and returns the length of string.

Parameters

<i>str</i>	The string to print.
------------	----------------------

Returns

str length of the string.

10.102.1.6 ocall_put_invoke_command_result() `void ocall_put_invoke_command_result (`
 `invoke_command_t cmd,`
 `unsigned int result)`

10.102.1.7 ocall_read_file() `int ocall_read_file (`
 `int fdesc,`
 `char * buf,`
 `size_t len)`

[ocall_read_file\(\)](#) - To read len bytes form file into the memory area indicated by buf.

Parameters

<i>fdesc</i>	file descriptor.
<i>buf</i>	buffer to write data.
<i>len</i>	length of buffer

Returns

integer value If success

10.102.1.8 ocall_read_invoke_param() `param_buffer_t ocall_read_invoke_param (`
 `int index,`
 `unsigned int offset)`

[ocall_read_file256\(\)](#) - To read a file of 256 bite.

Parameters

<i>fdesc</i>	File descriptor.
--------------	------------------

10.102.1.9 ocall_ree.time() `int ocall_ree_time (`
 `struct ree_time_t * timep)`

[ocall_ree.time\(\)](#) - gets the ree execution time.

Parameters

<i>timep</i>	pointer of time.
--------------	------------------

Returns

integer value If success

10.102.1.10 ocall.write_file() `int ocallwrite_file (`
`int fdesc,`
`const char * buf,`
`unsigned int len)`

[ocall.write_file\(\)](#) - To write data in to a file.

Parameters

<i>fdesc</i>	file descriptor.
<i>buf</i>	buffer to write data.
<i>len</i>	length of buffer.

Returns

integer value If success

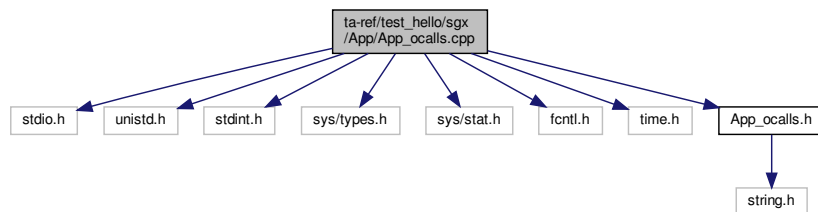
10.102.1.11 ocall.write.invoke_param() `void ocallwrite.invoke_param (`
`int index,`
`unsigned int offset,`
`unsigned int size,`
`const char * buf)`

10.103 ta-ref/test_hello/sgx/App/App_ocalls.cpp File Reference

```
#include <stdio.h>
#include <unistd.h>
#include <stdint.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>
```

```
#include "App_ocalls.h"
```

Include dependency graph for App_ocalls.cpp:



Functions

- unsigned int [ocall_print_string](#) (const char *str)
- int [ocall_open_file](#) (const char *fname, int flags, int perm)
- int [ocall_read_file](#) (int desc, char *buf, size_t len)
- int [ocall_write_file](#) (int desc, const char *buf, size_t len)
- int [ocall_close_file](#) (int desc)
- int [ocall_ree_time](#) (struct [ree_time_t](#) *time)

10.103.1 Function Documentation

10.103.1.1 ocall_close_file() `int ocall_close_file (int desc)`

[ocall_close_file\(\)](#) - To close a file.

Parameters

<i>desc</i>	file descriptor.
-------------	------------------

Returns

integer value If success

10.103.1.2 ocall_open_file() `int ocall_open_file (const char * fname, int flags, int perm)`

[ocall_open_file\(\)](#) - To open a file.

Parameters

<i>fname</i>	name of the file.
<i>flags</i>	mode of the file.
<i>perm</i>	indicates permissions of a file.

Returns

integer value If success

10.103.1.3 ocall_print_string() unsigned int ocall_print_string (
const char * *str*)

[ocall_print_string\(\)](#) - Prints the string.

This function invokes OCALL for displaying string type buffer inside the enclave.

Parameters

<i>str</i>	Pointer of the string.
------------	------------------------

Returns

length If success, else error occurred.

10.103.1.4 ocall_read_file() int ocall_read_file (
int *desc*,
char * *buf*,
size_t *len*)

[ocall_read_file\(\)](#) - To read len bytes form file into the memory area indicated by buf.

Parameters

<i>desc</i>	file descriptor.
<i>buf</i>	buffer to write data.
<i>len</i>	length of buffer

Returns

integer value If success

10.103.1.5 ocall_ree_time() `int ocall_ree_time (`
`struct ree_time_t * time)`

[ocall_ree_time\(\)](#) - gets the ree execution time.

Parameters

<i>time</i>	pointer of time.
-------------	------------------

Returns

integer value If success

10.103.1.6 ocall_write_file() `int ocall_write_file (`
`int desc,`
`const char * buf,`
`size_t len)`

[ocall_write_file\(\)](#) - To write data in to a file.

Parameters

<i>desc</i>	file descriptor.
<i>buf</i>	buffer to write data.
<i>len</i>	length of buffer.

Returns

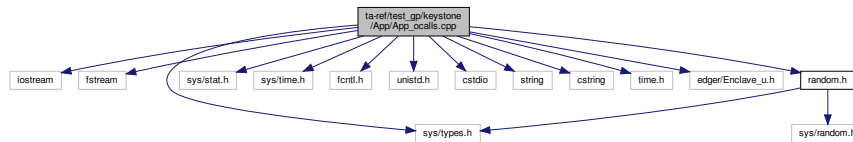
integer value If success

10.104 ta-ref/test_gp/keystone/App/App_ocalls.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <fcntl.h>
#include <unistd.h>
```

```
#include <stdio>
#include <string>
#include <cstring>
#include <time.h>
#include "edger/Enclave_u.h"
#include "random.h"
```

Include dependency graph for App_ocalls.cpp:



Macros

- `#define NO_PERF __attribute__((no_instrument_function))`

Functions

- `EDGE_EXTERN_C_BEGIN` unsigned int `NO_PERF ocall_print_string` (const char *str)
- int `ocall_open_file` (const char *fname, int flags, int perm)
- int `ocall_close_file` (int fdesc)
- int `ocall_write_file` (int fdesc, const char *buf, unsigned int len)
- int `ocall_invoke_command_callback_write` (const char *str, const char *buf, unsigned int len)
- int `ocall_read_file` (int fdesc, char *buf, size_t len)
- int `ocall_ree_time` (struct `ree_time_t` *timep)
- ssize_t `ocall_getrandom` (char *buf, size_t len, unsigned int flags)

10.104.1 Macro Definition Documentation

10.104.1.1 NO_PERF `#define NO_PERF __attribute__((no_instrument_function))`

10.104.2 Function Documentation

10.104.2.1 ocall_close_file() int `ocall_close_file` (
int *fdesc*)

`ocall_close_file()` - Frees the file descriptor in the process.

Parameters

<i>fdesc</i>	<i>fdesc</i> is a file descriptor of the type integer.
--------------	--

Returns

rtn on success,-1 on failure.

10.104.2.2 ocall_getrandom() `ssize_t ocall_getrandom (`
 `char * buf,`
 `size_t len,`
 `unsigned int flags)`

[ocall_getrandom\(\)](#) - System call fills the buffer pointed to by *buf* with up to *len* random bytes. These bytes can be used to seed user-space random number generators or for cryptographic purposes.

Parameters

<i>buf</i>	<i>buf</i> is a character datatype
<i>len</i>	<i>len</i> is a <code>size_t</code> datatype
<i>flags</i>	<i>flags</i> is a unsigned int datatype

Returns

the number of bytes stored in *buf*, -1 on failure.

10.104.2.3 ocall_invoke_command_callback_write() `int ocall_invoke_command_callback_write (`
 `const char * str,`
 `const char * buf,`
 `unsigned int len)`

[ocall_invoke_command_callback_write\(\)](#) -This function is invoked the `store_invoke_callback.file()` to store callback file.

Parameters

<i>str</i>	<i>str</i> is a constant character data type.
<i>buf</i>	<i>buf</i> is a constant character data type.
<i>len</i>	<i>len</i> is a unsigned int type.

Returns

0 on success else, error occurred.

10.104.2.4 ocall_open_file() `int ocall_open_file (`
 `const char * fname,`
 `int flags,`
 `int perm)`

[ocall_open_file\(\)](#) - opens a file name which shall be set according to the value of flag and determines the file permission mode.

Parameters

<i>fname</i>	file name is a constant character data type
<i>flags</i>	flags it is datatype of the integer
<i>perm</i>	permissions of the file if it is created

Returns

a nonnegative integer for success or -1 if an error occurred.

10.104.2.5 ocall_print_string() `EDGE_EXTERN_BEGIN unsigned int NO_PERF ocall_print_string (`
 `const char * str)`

[ocall_print_string\(\)](#) - To print the string and returns the length of string.

Parameters

<i>str</i>	The string to print.
------------	----------------------

Returns

str length of the string.

[ocall_print_string\(\)](#) - Prints the string.

This function invokes OCALL for displaying string type buffer inside the enclave.

Parameters

<i>str</i>	Pointer of the string.
------------	------------------------

Returns

length If success, else error occurred.

10.104.2.6 ocall_read_file() `int ocall_read_file (`
 `int fdesc,`
 `char * buf,`
 `size_t len)`

[ocall_read_file\(\)](#) - Reads a specified number of bytes into a buffer, through a file descriptor.

Parameters

<i>fdesc</i>	an open file descriptor
<i>buf</i>	buffer of at least size bytes
<i>len</i>	number of bytes to be read.

Returns

number of bytes read on success, -1 on failure.

10.104.2.7 ocall_ree_time() `int ocall_ree_time (`
 `struct ree_time_t * timep)`

[ocall_ree_time\(\)](#) - Function shall obtain the current time, expressed as seconds and microseconds.

Parameters

<i>timep</i>	timep is a structure type of ree_time_t
--------------	---

Returns

rtn value on success

10.104.2.8 ocall_write_file() `int ocall_write_file (`
 `int fdesc,`
 `const char * buf,`
 `unsigned int len)`

[ocall_write_file\(\)](#) - Writes the size bytes from buff to file specified by fdesc.

Parameters

<i>fdesc</i>	file descriptor
<i>buf</i>	buffer of at least size bytes
<i>len</i>	number of bytes to be write.

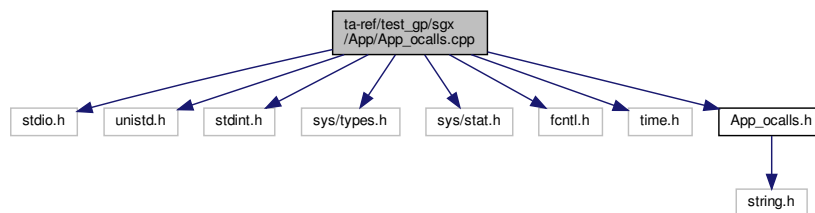
Returns

number of bytes written on success,-1 on failure.

10.105 ta-ref/test_gp/sgx/App/App_ocalls.cpp File Reference

```
#include <stdio.h>
#include <unistd.h>
#include <stdint.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>
#include "App_ocalls.h"
```

Include dependency graph for App_ocalls.cpp:



Macros

- #define `MAX_PATH` `FILENAME_MAX`
- #define `NO_PERF __attribute__((no_instrument_function))`

Functions

- unsigned int `NO_PERF ocall_print_string` (const char *str)
- int `ocall_open_file` (const char *fname, int flags, int perm)
- int `ocall_read_file` (int desc, char *buf, size_t len)
- int `ocall_write_file` (int desc, const char *buf, size_t len)
- int `ocall_close_file` (int desc)
- int `ocall_ree_time` (struct `ree_time_t` *time)

10.105.1 Macro Definition Documentation

10.105.1.1 MAX_PATH `#define MAX_PATH FILENAME_MAX`

10.105.1.2 NO_PERF `#define NO_PERF __attribute__((no_instrument_function))`

10.105.2 Function Documentation

10.105.2.1 ocall_close_file() `int ocall_close_file (`
 `int desc)`

[ocall_close_file\(\)](#) - Used for closing a file

Parameters

<i>desc</i>	File descriptor.
-------------	------------------

Returns

file descripto If success, else error ocured.

10.105.2.2 ocall_open_file() `int ocall_open_file (`
 `const char * fname,`
 `int flags,`
 `int perm)`

[ocall_open_file\(\)](#) - Used for opening a file.

Parameters

<i>fname</i>	File name
<i>flags</i>	Values for oflag are constructed by a bitwise-inclusive OR of flags from the following list.
<i>perm</i>	permission or mode

Returns

file descriptor If success, else error ocured

10.105.2.3 ocall_print_string() unsigned int NO_PERF ocall_print_string (
 const char * *str*)

[ocall_print_string\(\)](#) - To print the argument string message.

Parameters

<i>str</i>	Pointer of the string.
------------	------------------------

Returns

length If success, else error occurred.

10.105.2.4 ocall_read_file() int ocall_read_file (
 int *desc*,
 char * *buf*,
 size_t *len*)

[ocall_read_file\(\)](#) - Used to read from a file.

Parameters

<i>desc</i>	file descriptor
<i>buf</i>	pointer to a buffer
<i>len</i>	Size of elements

Returns

file descriptor If success, else error occurred

10.105.2.5 ocall_ree_time() int ocall_ree_time (
 struct ree_time_t * *time*)

[ocall_ree_time\(\)](#) - Used to fetch the current time.

Parameters

<i>time</i>	Pointer to a current time.
-------------	----------------------------

Returns

current time If success, else error occurred

10.105.2.6 ocall.write_file() `int ocall.write_file (`
 `int desc,`
 `const char * buf,`
 `size_t len)`

[ocall.write_file\(\)](#) - Used to write into a file.

Parameters

<i>desc</i>	file descriptor.
<i>buf</i>	pointer to a buffer.
<i>len</i>	Size of elements.

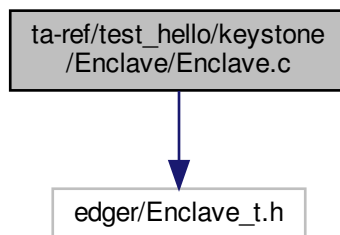
Returns

file descriptor If success, else error occurred.

10.106 ta-ref/test_hello/keystone/Enclave/Enclave.c File Reference

```
#include "edger/Enclave_t.h"
```

Include dependency graph for Enclave.c:



Macros

- `#define MESSAGE "hello world!\n"`

Functions

- `void EAPP_ENTRY eapp_entry ()`

10.106.1 Macro Definition Documentation

10.106.1.1 MESSAGE `#define MESSAGE "hello world!\n"`

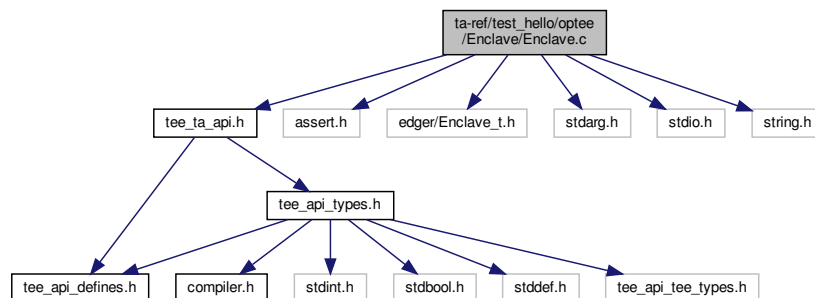
10.106.2 Function Documentation

10.106.2.1 eapp_entry() `void EAPP_ENTRY eapp_entry ()`

[eapp_entry\(\)](#) - This function is used for printing the Message.

10.107 ta-ref/test_hello/optee/Enclave/Enclave.c File Reference

```
#include <tee_ta_api.h>
#include "assert.h"
#include "edger/Enclave_t.h"
#include <stdarg.h>
#include <stdio.h>
#include <string.h>
Include dependency graph for Enclave.c:
```



Macros

- `#define BUF_SIZE 8192`
- `#define TEE_PARAM_TYPE1 TEE_PARAM_TYPE_MEMREF_OUTPUT`
- `#define MESSAGE "hello world!\n"`

Functions

- static unsigned int `_strlen` (const char *str)
- int `tee_printf` (const char *fmt,...)
- `TEE_Result TA_CreateEntryPoint` (void)
- `TEE_Result TA_OpenSessionEntryPoint` (uint32_t __unused param_types, `TEE_Param` __unused params[4], void __unused **sess_ctx)
- void `TA_DestroyEntryPoint` (void)
- `TEE_Result run_all_test` (uint32_t param_types, `TEE_Param` __maybe_unused params[4], void __maybe_unused **sess_ctx)
- void `TA_CloseSessionEntryPoint` (void __maybe_unused *sess_ctx)
- `TEE_Result TA_InvokeCommandEntryPoint` (void *sess_ctx, uint32_t cmd_id, uint32_t param_types, `TEE_Param` params[4])

Variables

- char `print_buf` [BUF_SIZE]
- size_t `print_pos`

10.107.1 Macro Definition Documentation

10.107.1.1 BUF_SIZE `#define BUF_SIZE 8192`

10.107.1.2 MESSAGE `#define MESSAGE "hello world!\n"`

10.107.1.3 TEE_PARAM_TYPE1 `#define TEE_PARAM_TYPE1 TEE_PARAM_TYPE_MEMREF_OUTPUT`

10.107.2 Function Documentation

10.107.2.1 _strlen() `static unsigned int _strlen (`
`const char * str) [inline], [static]`

`_strlen()` - returns the length of string.

This function is used for returning the length of the string "@param str".

Parameters

<i>str</i>	This is the string whose length is to be found.
------------	---

Returns

string length If success, else error occurred.

10.107.2.2 run_all_test() `TEE_Result run_all_test (`
 `uint32_t param_types,`
 `TEE_Param __maybe_unused params[4],`
 `void __maybe_unused ** sess_ctx)`

`run_all_test()` - Function is used for checking the test of "hello world" example.

This function prints the message and returns TEE_SUCCESS after completion of process.

Parameters

<i>param_types</i>	The types of the four parameters.
<i>params[4]</i>	A pointer to an array of four parameters.
<i>sess_ctx</i>	A pointer to a variable that can be filled by the Trusted Application instance with an opaque void* data pointer.

Returns

TEE_SUCCESS If success, else error occurred.

10.107.2.3 TA_CloseSessionEntryPoint() `void TA_CloseSessionEntryPoint (`
 `void __maybe_unused * sess_ctx)`

`TA_CloseSessionEntryPoint()` - The Framework calls to close a client session.

The Trusted Application function `TA_CloseSessionEntryPoint` implementation is responsible for freeing any resources consumed by the session being closed.

Parameters

<i>sess_ctx</i>	The value of the void* opaque data pointer set by the Trusted Application in this <code>TA_OpenSessionEntryPoint()</code> for this session.
-----------------	---

10.107.2.4 TA_CreateEntryPoint() `TEE_Result TA_CreateEntryPoint (`
`void)`

[TA_CreateEntryPoint\(\)](#) - Trusted application creates the entry point.

TA_CreateEntryPoint function is the Trusted Application's constructor, which the framework calls when it creates a new instance of the Trusted Application.

Returns

TEE_SUCCESS If success, else error occurred.

10.107.2.5 TA_DestroyEntryPoint() `void TA_DestroyEntryPoint (`
`void)`

[TA_DestroyEntryPoint\(\)](#) - The function TA_DestroyEntryPoint is the Trusted Application's destructor, which the Framework calls when the instance is being destroyed.

10.107.2.6 TA_InvokeCommandEntryPoint() `TEE_Result TA_InvokeCommandEntryPoint (`
`void * sess_ctx,`
`uint32_t cmd_id,`
`uint32_t param_types,`
`TEE_Param params[4])`

[TA_InvokeCommandEntryPoint\(\)](#) - The Framework calls the client invokes a command within the given session.

The Trusted Application function TA_InvokeCommandEntryPoint can access the parameters sent by the client through the paramTypes and params arguments. It can also use these arguments to transfer response data back to the client.

Parameters

<code>sess_ctx</code>	The value of the void* opaque data pointer set by the Trusted Application in the function TA_OpenSessionEntryPoint for this session.
-----------------------	--

Returns

TEE_SUCCESS If success, else error occurred.

10.107.2.7 TA_OpenSessionEntryPoint() `TEE_Result TA_OpenSessionEntryPoint (`
`uint32_t __unused param_types,`
`TEE_Param __unused params[4],`
`void __unused ** sess_ctx)`

[TA_OpenSessionEntryPoint\(\)](#) - Trusted application open the session entry point.

The Framework calls the function TA_OpenSessionEntryPoint when a client requests to open a session with the Trusted Application.

Parameters

<i>param_types</i>	The types of the four parameters.
<i>params</i>	A pointer to an array of four parameters.
<i>sess_ctx</i>	A pointer to a variable that can be filled by the Trusted Application instance with an opaque void* data pointer.

Returns

TEE_SUCCESS If success, else error occurred.

10.107.2.8 tee_printf() `int tee_printf (`
`const char * fmt,`
`...)`

[tee_printf\(\)](#) - Printing the formatted output in to a character array.

In this function the "@param ap" variable is initialized by calling `va.start()` and then formatted data will send to a string using argument list by calling [vsnprintf\(\)](#) and finally the string length will be stored in `res`.

Parameters

<i>fmt</i>	A string that specifies the format of the output.
------------	---

Returns

result If success, else error occurred.

10.107.3 Variable Documentation

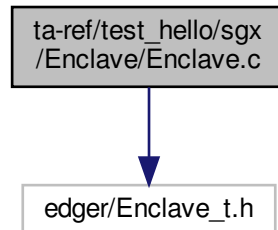
10.107.3.1 print_buf `char print_buf[BUF_SIZE]`

10.107.3.2 print_pos `size_t print_pos`

10.108 ta-ref/test_hello/sgx/Enclave/Enclave.c File Reference

```
#include "edger/Enclave_t.h"
```

Include dependency graph for Enclave.c:



Macros

- `#define MESSAGE "hello world!\n"`

Functions

- `void ecall_ta_main (void)`

10.108.1 Macro Definition Documentation

10.108.1.1 MESSAGE `#define MESSAGE "hello world!\n"`

10.108.2 Function Documentation

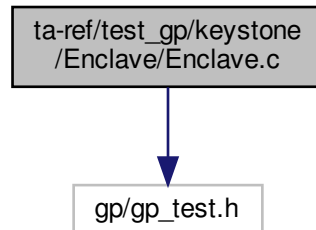
10.108.2.1 ecall_ta_main() `void ecall_ta_main (void)`

`ecall_ta_main()` - Prints the string and returns the number of string.

10.109 ta-ref/test_gp/keystone/Enclave/Enclave.c File Reference

```
#include "gp/gp_test.h"
```

Include dependency graph for Enclave.c:



Functions

- int [main](#) (void)

10.109.1 Function Documentation

10.109.1.1 main() `int main (`
`void)`

This [main\(\)](#) function invokes the functions [gp_random_test\(\)](#) to generate random data [gp_ree_time_test\(\)](#) to retrieve the current REE system time [gp_trusted_time_test\(\)](#) to retrieve the current system time [gp_secure_storage_test\(\)](#) to create read and write the object data [gp_message_digest_test\(\)](#) to accumulate message data for hashing [gp_symmetric_key_enc_verify_test\(\)](#) to encrypt or decrypt input data [gp_symmetric_key_gcm_verify_test\(\)](#) to encrypt and decrypt in AE [gp_asymmetric_key_sign_test\(\)](#) for cryptographic Operations API message Digest Functions and returns the status as success when all the functions generates the same data.

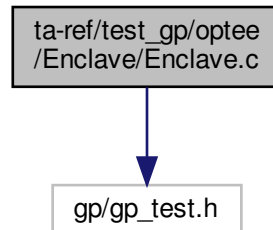
Returns

return 0 for success.

10.110 ta-ref/test_gp/optee/Enclave/Enclave.c File Reference

```
#include "gp/gp_test.h"
```

Include dependency graph for Enclave.c:



Functions

- int [main](#) (void)

10.110.1 Function Documentation

10.110.1.1 main() `int main (void)`

This [main\(\)](#) function invokes the functions [gp_random_test\(\)](#) to generate random data [gp_ree_time_test\(\)](#) to retrieve the current REE system time [gp_trusted_time_test\(\)](#) to retrieve the current system time [gp_secure_storage_test\(\)](#) to create read and write the object data [gp_message_digest_test\(\)](#) to accumulate message data for hashing [gp_symmetric_key_enc_verify_test\(\)](#) to encrypt or decrypt input data [gp_symmetric_key_gcm_verify_test\(\)](#) to encrypt and decrypt in AE [gp_asymmetric_key_sign_test\(\)](#) for cryptographic Operations API message Digest Functions and returns the status as success when all the functions generates the same data.

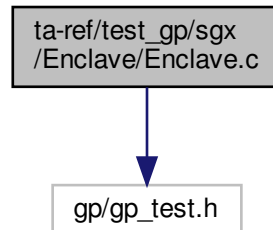
Returns

return 0 for success.

10.111 ta-ref/test_gp/sgx/Enclave/Enclave.c File Reference

```
#include "gp/gp_test.h"
```

Include dependency graph for Enclave.c:



Functions

- int [main](#) (void)

10.111.1 Function Documentation

10.111.1.1 main() `int main (void)`

This [main\(\)](#) function invokes the functions [gp_random_test\(\)](#) to generate random data [gp_ree_time_test\(\)](#) to retrieve the current REE system time [gp_trusted_time_test\(\)](#) to retrieve the current system time [gp_secure_storage_test\(\)](#) to create read and write the object data [gp_message_digest_test\(\)](#) to accumulate message data for hashing [gp_symmetric_key_enc_verify_test\(\)](#) to encrypt or decrypt input data [gp_symmetric_key_gcm_verify_test\(\)](#) to encrypt and decrypt in AE [gp_asymmetric_key_sign_test\(\)](#) for cryptographic Operations API message Digest Functions and returns the status as success when all the functions generates the same data.

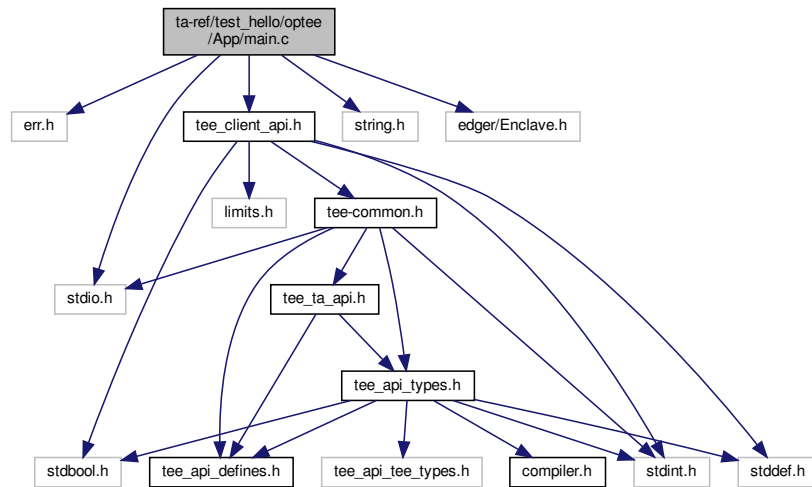
Returns

return 0 for success.

10.112 ta-ref/test_hello/optee/App/main.c File Reference

```
#include <err.h>
#include <stdio.h>
#include <string.h>
#include <tee_client_api.h>
```

```
#include <edger/Enclave.h>
Include dependency graph for main.c:
```



Macros

- `#define PRINT_BUF_SIZE 16384`
- `#define TEEC_PARAM_TYPE1 TEEC_MEMREF_TEMP_OUTPUT`

Functions

- `int main (void)`

Variables

- `static char print_buf [PRINT_BUF_SIZE]`

10.112.1 Macro Definition Documentation

10.112.1.1 PRINT_BUF_SIZE `#define PRINT_BUF_SIZE 16384`

10.112.1.2 TEEC_PARAM_TYPE1 `#define TEEC_PARAM_TYPE1 TEEC_MEMREF_TEMP_OUTPUT`

10.112.2 Function Documentation

10.112.2.1 main() `int main (`
`void)`

[main\(\)](#) -To perform the TEEC operations for building TA inside TEE.

In this function the context is initialized for connecting to the TEE by calling [TEEC.InitializeContext\(\)](#). After initialization of context the session is opened on [TEEC.OpenSession\(\)](#) and then command is invoked in the TEE. Once the command is invoked the session is closed and the context is finalized. If the session is not opened properly, `session_failed` error appears.

Returns

0 If success, else displays error message.

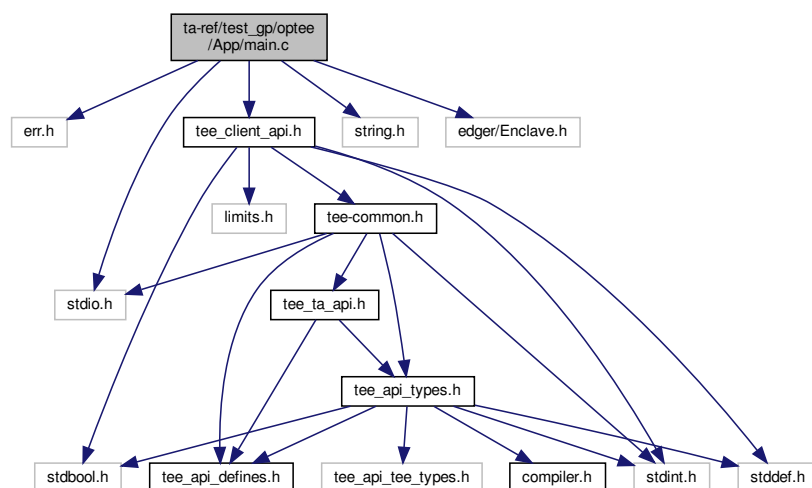
10.112.3 Variable Documentation

10.112.3.1 print_buf `char print_buf[PRINT_BUF_SIZE] [static]`

10.113 ta-ref/test_gp/optee/App/main.c File Reference

```
#include <err.h>
#include <stdio.h>
#include <string.h>
#include <tee_client_api.h>
#include <edger/Enclave.h>
```

Include dependency graph for main.c:



Macros

- #define [BUF_SIZE](#) 65536
- #define [PRINT_BUF_SIZE](#) 16384
- #define [TEEC_PARAM_TYPE0](#) [TEEC_NONE](#)
- #define [TEEC_PARAM_TYPE1](#) [TEEC_NONE](#)

Functions

- int [main](#) (void)

10.113.1 Macro Definition Documentation

10.113.1.1 BUF_SIZE #define BUF_SIZE 65536

10.113.1.2 PRINT_BUF_SIZE #define PRINT_BUF_SIZE 16384

10.113.1.3 TEEC_PARAM_TYPE0 #define TEEC_PARAM_TYPE0 [TEEC_NONE](#)

10.113.1.4 TEEC_PARAM_TYPE1 #define TEEC_PARAM_TYPE1 [TEEC_NONE](#)

10.113.2 Function Documentation

10.113.2.1 main() int main (
void)

[main\(\)](#) - Initializes a new TEE Context and opens a new Session.

This function initializes a new TEE context and opens a new session between the client application and the specified trusted application. If initialization to a new TEE context and opening a new session are success then, first op([↔](#) TEEC.Operation) characters of the string, are copied by the argument &op. If the macro is PERF_ENABLE, then assign the buffer and buffer size to "params[0]" and then open the log file for write. If the macro is ENCLAVE_ [↔](#) VERBOSE then assign the buffer and buffer size to "params[1]". Then print the "enclave log start" and "enclave log end". If macro is APP_VERBOSE then print the "start the invoke command" and invoke the [TEEC_InvokeCommand\(\)](#). If the [TEEC_InvokeCommand\(\)](#) is success then print the "TEEC.InvokeCommand succeeded!". If [TEEC_InvokeCommand\(\)](#) fails, Then print the message as "TEEC.InvokeCommand failed" with code message result and error origin. Finally close the session and destroy the initialized TEE context.

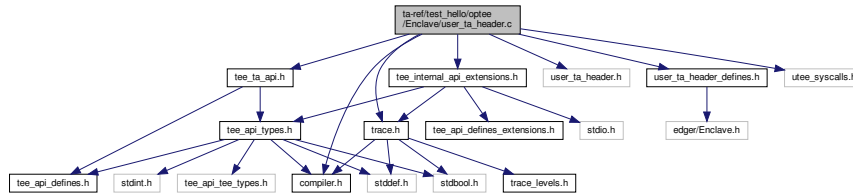
Returns

0 If the function is a success.

10.114 ta-ref/test_hello/optee/Enclave/user_ta_header.c File Reference

```
#include <compiler.h>
#include <tee_ta_api.h>
#include <tee_internal_api_extensions.h>
#include <trace.h>
#include <user_ta_header.h>
#include <user_ta_header_defines.h>
#include <utee_syscalls.h>
```

Include dependency graph for user_ta_header.c:



Macros

- `#define TA_VERSION` "Undefined version"
- `#define TA_DESCRIPTION` "Undefined description"
- `#define _C_FUNCTION(name)` name
- `#define TA_FRAMEWORK_STACK_SIZE` 2048

Functions

- `TEE_Result __utee_entry` (unsigned long func, unsigned long session_id, struct utee_params *up, unsigned long cmd_id)
- `void __noreturn _C_FUNCTION() __ta_entry` (unsigned long func, unsigned long session_id, struct utee_params *up, unsigned long cmd_id)
- `const struct ta_head ta_head __section` (".ta_head")
- `int tahead_get_trace_level` (void)

Variables

- `int trace_level` = `TRACE_LEVEL`
- `const char trace_ext_prefix []` = "TA"
- `uint8_t ta_heap [TA_DATA_SIZE]`
- `const size_t ta_heap_size` = `sizeof(ta_heap)`
- `const struct user_ta_property ta_props []`
- `const size_t ta_num_props` = `sizeof(ta_props) / sizeof(ta_props[0])`

10.114.1 Macro Definition Documentation

10.114.1.1 `_C_FUNCTION` `#define _C_FUNCTION(
 name) name`

10.114.1.2 `TA_DESCRIPTION` `#define TA_DESCRIPTION "Undefined description"`

10.114.1.3 `TA_FRAMEWORK_STACK_SIZE` `#define TA_FRAMEWORK_STACK_SIZE 2048`

10.114.1.4 `TA_VERSION` `#define TA_VERSION "Undefined version"`

10.114.2 Function Documentation

10.114.2.1 `__section()` `const struct ta_head ta_head __section (
 ".ta-head")`

10.114.2.2 `__ta_entry()` `void __noreturn _C_FUNCTION() __ta_entry (
 unsigned long func,
 unsigned long session_id,
 struct utee_params * up,
 unsigned long cmd_id)`

[`__ta_entry\(\)`](#) - The trusted application entry with no return value.

`__ta_entry` is the first TA API called from TEE core. As it being `__noreturn` API, we need to call `ftrace_return` in this API just before `utee_return` syscall to get proper `ftrace` call graph.

Parameters

<i>func</i>	Function
<i>session_id</i>	Session id
<i>up</i>	object of struct <code>utee_params</code>
<i>cmd_id</i>	command input id

10.114.2.3 `__utee_entry()` `TEE_Result __utee_entry (`

```

    unsigned long func,
    unsigned long session-id,
    struct utee_params * up,
    unsigned long cmd_id )

```

10.114.2.4 tahead_get_trace_level() `int tahead_get_trace_level (`
`void)`

[tahead_get_trace_level\(\)](#) - Store trace level in TA head structure, as ta_head. prop_tracelevel

Returns

Non-negative integer value if success, else error.

10.114.3 Variable Documentation

10.114.3.1 ta_heap `uint8_t ta_heap[TA_DATA_SIZE]`

10.114.3.2 ta_heap_size `const size_t ta_heap_size = sizeof(ta_heap)`

10.114.3.3 ta_num_props `const size_t ta_num_props = sizeof(ta_props) / sizeof(ta_props[0])`

10.114.3.4 ta_props `const struct user_ta_property ta_props[]`

Initial value:

```

= {
    {TA_PROP_STR_SINGLE_INSTANCE, USER_TA_PROP_TYPE_BOOL,
      &(const bool){(TA_FLAGS & TA_FLAG_SINGLE_INSTANCE) != 0}},
    {TA_PROP_STR_MULTI_SESSION, USER_TA_PROP_TYPE_BOOL,
      &(const bool){(TA_FLAGS & TA_FLAG_MULTI_SESSION) != 0}},
    {TA_PROP_STR_KEEP_ALIVE, USER_TA_PROP_TYPE_BOOL,
      &(const bool){(TA_FLAGS & TA_FLAG_INSTANCE_KEEP_ALIVE) != 0}},
    {TA_PROP_STR_DATA_SIZE, USER_TA_PROP_TYPE_U32,
      &(const uint32_t){TA_DATA_SIZE}},
    {TA_PROP_STR_STACK_SIZE, USER_TA_PROP_TYPE_U32,
      &(const uint32_t){TA_STACK_SIZE}},
    {TA_PROP_STR_VERSION, USER_TA_PROP_TYPE_STRING,
      TA_VERSION},
    {TA_PROP_STR_DESCRIPTION, USER_TA_PROP_TYPE_STRING,
      TA_DESCRIPTION},
}

```

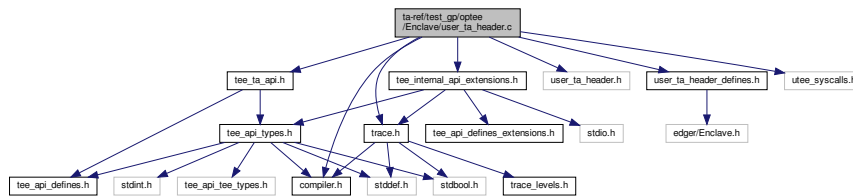
10.114.3.5 trace_ext_prefix `const char trace_ext_prefix[] = "TA"`

10.114.3.6 trace_level `int trace_level = TRACE_LEVEL`

10.115 ta-ref/test_gp/optee/Enclave/user_ta_header.c File Reference

```
#include <compiler.h>
#include <tee_ta_api.h>
#include <tee_internal_api_extensions.h>
#include <trace.h>
#include <user_ta_header.h>
#include <user_ta_header_defines.h>
#include <utee_syscalls.h>
```

Include dependency graph for user_ta_header.c:



Macros

- `#define TA_VERSION` "Undefined version"
- `#define TA_DESCRIPTION` "Undefined description"
- `#define _C_FUNCTION(name)` name
- `#define TA_FRAMEWORK_STACK_SIZE` 2048

Functions

- `TEE_Result __utee_entry` (unsigned long func, unsigned long session_id, struct utee_params *up, unsigned long cmd_id)
- `void __noreturn _C_FUNCTION() __ta_entry` (unsigned long func, unsigned long session_id, struct utee_params *up, unsigned long cmd_id)
- `const struct ta_head ta_head __section` ("ta_head")
- `int tahead_get_trace_level` (void)

Variables

- `int trace_level` = `TRACE_LEVEL`
- `const char trace_ext_prefix []` = "TA"
- `uint8_t ta_heap` [`TA_DATA_SIZE`]
- `const size_t ta_heap_size` = `sizeof(ta_heap)`
- `const struct user_ta_property ta_props []`
- `const size_t ta_num_props` = `sizeof(ta_props) / sizeof(ta_props[0])`

10.115.1 Macro Definition Documentation

10.115.1.1 `_C_FUNCTION` `#define _C_FUNCTION(
 name) name`

10.115.1.2 `TA_DESCRIPTION` `#define TA_DESCRIPTION "Undefined description"`

10.115.1.3 `TA_FRAMEWORK_STACK_SIZE` `#define TA_FRAMEWORK_STACK_SIZE 2048`

10.115.1.4 `TA_VERSION` `#define TA_VERSION "Undefined version"`

10.115.2 Function Documentation

10.115.2.1 `__section()` `const struct tahead tahead __section (
 ".tahead")`

10.115.2.2 `__ta_entry()` `void __noreturn _C_FUNCTION() __ta_entry (
 unsigned long func,
 unsigned long session_id,
 struct utee_params * up,
 unsigned long cmd_id)`

10.115.2.3 `__utee_entry()` `TEE_Result __utee_entry (
 unsigned long func,
 unsigned long session_id,
 struct utee_params * up,
 unsigned long cmd_id)`

[__utee_entry\(\)](#) - From libutee.

Receiving the session and command id and if defined macro is `CFG_FTRACE_SUPPORT` the function invokes the `ftrace_return()` in TA API just before the `utee_return` syscall to get proper ftrace call graph. The return of this function is `TEE_SUCCESS` when all the above functions are performed.

Parameters

<i>func</i>	func is the unsigned long data type.
<i>session_id</i>	session_id is the unsigned long data type.
<i>up</i>	up is the structure type of the utee.params.
<i>cmd_id</i>	cmd_id is the unsigned long data type.

Returns

TEE_SUCCESS If the command is successfully executed.

10.115.2.4 tahead_get_trace_level() `int tahead_get_trace_level (void)`

[tahead_get_trace_level\(\)](#) - Store trace level in TA head structure, as ta_head.prop_tracelevel.

Returns

trace level for success, else error occurred.

10.115.3 Variable Documentation

10.115.3.1 ta_heap `uint8_t ta_heap[TA_DATA_SIZE]`

10.115.3.2 ta_heap_size `const size_t ta_heap_size = sizeof(ta_heap)`

10.115.3.3 ta_num_props `const size_t ta_num_props = sizeof(ta_props) / sizeof(ta_props[0])`

10.115.3.4 ta_props `const struct user.ta.property ta_props[]`**Initial value:**

```

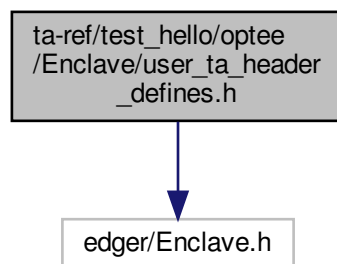
= {
    {TA_PROP_STR_SINGLE_INSTANCE, USER_TA_PROP_TYPE_BOOL,
      &(const bool){(TA_FLAGS & TA_FLAG_SINGLE_INSTANCE) != 0}},
    {TA_PROP_STR_MULTI_SESSION, USER_TA_PROP_TYPE_BOOL,
      &(const bool){(TA_FLAGS & TA_FLAG_MULTI_SESSION) != 0}},
    {TA_PROP_STR_KEEP_ALIVE, USER_TA_PROP_TYPE_BOOL,
      &(const bool){(TA_FLAGS & TA_FLAG_INSTANCE_KEEP_ALIVE) != 0}},
    {TA_PROP_STR_DATA_SIZE, USER_TA_PROP_TYPE_U32,
      &(const uint32_t){TA_DATA_SIZE}},
    {TA_PROP_STR_STACK_SIZE, USER_TA_PROP_TYPE_U32,
      &(const uint32_t){TA_STACK_SIZE}},
    {TA_PROP_STR_VERSION, USER_TA_PROP_TYPE_STRING,
      TA_VERSION},
    {TA_PROP_STR_DESCRIPTION, USER_TA_PROP_TYPE_STRING,
      TA_DESCRIPTION},
}

```

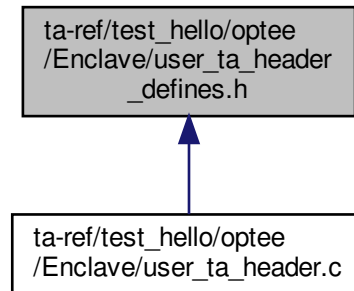
10.115.3.5 trace_ext_prefix `const char trace_ext_prefix[] = "TA"`**10.115.3.6 trace_level** `int trace_level = TRACE_LEVEL`**10.116 ta-ref/test.hello/optee/Enclave/user.ta.header.defines.h File Reference**

#include "edger/Enclave.h"

Include dependency graph for user.ta.header.defines.h:



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



Macros

- `#define TA_UUID TA_REF_UUID`
- `#define TA_FLAGS TA_FLAG_EXEC_DDR`
- `#define TA_STACK_SIZE (2 * 1024)`
- `#define TA_DATA_SIZE (32 * 1024)`
- `#define TA_VERSION "1.0"`
- `#define TA_DESCRIPTION "Example of OP-TEE TEST Trusted Application"`
- `#define TA_CURRENT_TA_EXT_PROPERTIES`

10.116.1 Macro Definition Documentation

10.116.1.1 TA_CURRENT_TA_EXT_PROPERTIES `#define TA_CURRENT_TA_EXT_PROPERTIES`

Value:

```

{ "org.linaro.optee.examples.test.property1", \
  USER_TA_PROP_TYPE_STRING, \
  "Some string" }, \
{ "org.linaro.optee.examples.test.property2", \
  USER_TA_PROP_TYPE_U32, &(const uint32_t){ 0x0010 } }

```

10.116.1.2 TA_DATA_SIZE `#define TA_DATA_SIZE (32 * 1024)`

10.116.1.3 TA_DESCRIPTION `#define TA_DESCRIPTION "Example of OP-TEE TEST Trusted Application"`

10.116.1.4 TA_FLAGS `#define TA_FLAGS TA_FLAG_EXEC_DDR`

10.116.1.5 TA_STACK_SIZE `#define TA_STACK_SIZE (2 * 1024)`

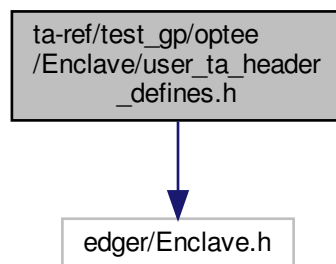
10.116.1.6 TA_UUID `#define TA_UUID TA_REF_UUID`

10.116.1.7 TA_VERSION `#define TA_VERSION "1.0"`

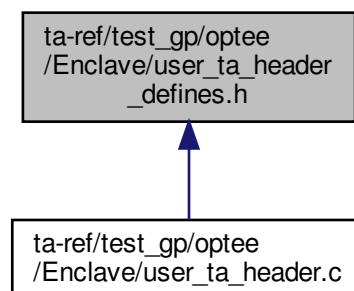
10.117 ta-ref/test_gp/optee/Enclave/user_ta_header_defines.h File Reference

```
#include "edger/Enclave.h"
```

Include dependency graph for user_ta_header_defines.h:



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



Macros

- `#define TA_UUID TA_REF_UUID`
- `#define TA_FLAGS TA_FLAG_EXEC_DDR`
- `#define TA_STACK_SIZE (2 * 1024)`
- `#define TA_DATA_SIZE (32 * 1024)`
- `#define TA_VERSION "1.0"`
- `#define TA_DESCRIPTION "Example of OP-TEE TEST Trusted Application"`
- `#define TA_CURRENT_TA_EXT_PROPERTIES`

10.117.1 Macro Definition Documentation

10.117.1.1 TA_CURRENT_TA_EXT_PROPERTIES `#define TA_CURRENT_TA_EXT_PROPERTIES`

Value:

```
{ "org.linaro.optee.examples.test.property1", \
  USER_TA_PROP_TYPE_STRING, \
  "Some string" }, \
{ "org.linaro.optee.examples.test.property2", \
  USER_TA_PROP_TYPE_U32, &(const uint32_t){ 0x0010 } }
```

10.117.1.2 TA_DATA_SIZE `#define TA_DATA_SIZE (32 * 1024)`

10.117.1.3 TA_DESCRIPTION `#define TA_DESCRIPTION "Example of OP-TEE TEST Trusted Application"`

10.117.1.4 TA_FLAGS `#define TA_FLAGS TA_FLAG_EXEC_DDR`

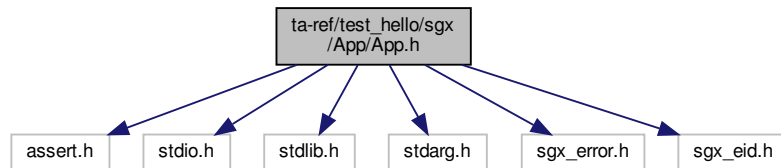
10.117.1.5 TA_STACK_SIZE `#define TA_STACK_SIZE (2 * 1024)`

10.117.1.6 TA_UUID `#define TA_UUID TA_REF_UUID`

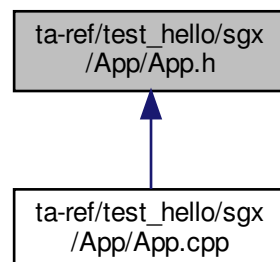
10.117.1.7 TA_VERSION `#define TA_VERSION "1.0"`

10.118 ta-ref/test_hello/sgx/App/App.h File Reference

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "sgx_error.h"
#include "sgx_eid.h"
Include dependency graph for App.h:
```



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



Macros

- #define `TRUE` 1
- #define `FALSE` 0
- #define `ENCLAVE_FILENAME` "enclave.signed.so"

Variables

- `sgx_enclave_id_t` `global_eid`

10.118.1 Macro Definition Documentation

10.118.1.1 ENCLAVE_FILENAME `#define ENCLAVE_FILENAME "enclave.signed.so"`

10.118.1.2 FALSE `#define FALSE 0`

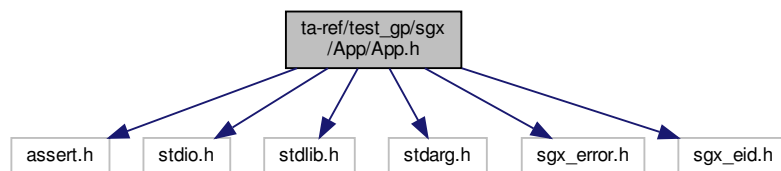
10.118.1.3 TRUE `#define TRUE 1`

10.118.2 Variable Documentation

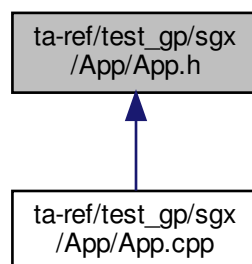
10.118.2.1 global_eid `sgx_enclave_id_t globaleid [extern]`

10.119 ta-ref/test_gp/sgx/App/App.h File Reference

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "sgx_error.h"
#include "sgx_eid.h"
Include dependency graph for App.h:
```



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



Macros

- #define `TRUE` 1
- #define `FALSE` 0
- #define `ENCLAVE_FILENAME` "enclave.signed.so"

Variables

- `sgx_enclave_id_t` `global_eid`

10.119.1 Macro Definition Documentation

10.119.1.1 ENCLAVE_FILENAME #define ENCLAVE_FILENAME "enclave.signed.so"

10.119.1.2 FALSE #define FALSE 0

10.119.1.3 TRUE #define TRUE 1

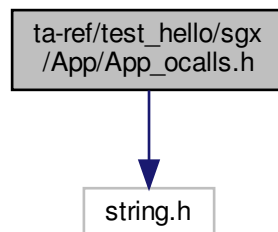
10.119.2 Variable Documentation

10.119.2.1 global_eid `sgx_enclave_id_t` `globaleid` [extern]

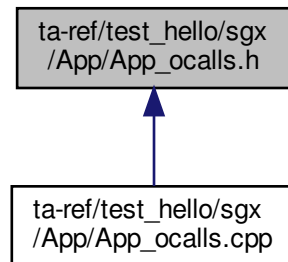
10.120 ta-ref/test_hello/sgx/App/App_ocalls.h File Reference

```
#include <string.h>
```

Include dependency graph for App_ocalls.h:



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



Classes

- struct [ree_time_t](#)

Typedefs

- typedef struct [ree_time_t](#) [ree_time_t](#)

Functions

- unsigned int [ocall_print_string](#) (const char *str)
- int [ocall_open_file](#) (const char *fname, int flags, int perm)
- int [ocall_read_file](#) (int desc, char *buf, size_t len)
- int [ocall_write_file](#) (int desc, const char *buf, size_t len)
- int [ocall_close_file](#) (int desc)
- int [ocall_ree_time](#) (struct [ree_time_t](#) *time)

10.120.1 Typedef Documentation

10.120.1.1 [ree_time_t](#) typedef struct [ree_time_t](#) [ree_time_t](#)

10.120.2 Function Documentation

10.120.2.1 [ocall_close_file\(\)](#) int [ocall_close_file](#) (
int desc)

[ocall_close_file\(\)](#) - To close a file.

Parameters

<i>fdesc</i>	file descriptor.
--------------	------------------

Returns

integer value If success

[ocall_close_file\(\)](#) - To close a file.

Parameters

<i>desc</i>	file descriptor.
-------------	------------------

Returns

integer value If success

[ocall_close_file\(\)](#) - Frees the file descriptor in the process.

Parameters

<i>fdesc</i>	fdesc is a file descriptor of the type integer.
--------------	---

Returns

rtn on success,-1 on failure.

[ocall_close_file\(\)](#) - Used for closing a file

Parameters

<i>desc</i>	File descriptor.
-------------	------------------

Returns

file descripto If success, else error occured.

10.120.2.2 ocall_open_file() `int ocall_open_file (`
 `const char * fname,`
 `int flags,`
 `int perm)`

[ocall_open_file\(\)](#) - To open a file.

Parameters

<i>fname</i>	name of the file.
<i>flags</i>	mode of the file.
<i>perm</i>	indicates permissions of a file.

Returns

integer If success

[ocall_open.file\(\)](#) - To open a file.

Parameters

<i>fname</i>	name of the file.
<i>flags</i>	mode of the file.
<i>perm</i>	indicates permissions of a file.

Returns

integer value If success

[ocall_open.file\(\)](#) - opens a file name which shall be set according to the value of flag and determines the file permission mode.

Parameters

<i>fname</i>	file name is a constant character data type
<i>flags</i>	flags it is datatype of the integer
<i>perm</i>	permissions of the file if it is created

Returns

a nonnegative integer for success or -1 if an error occurred.

[ocall_open.file\(\)](#) - Used for opening a file.

Parameters

<i>fname</i>	File name
<i>flags</i>	Values for oflag are constructed by a bitwise-inclusive OR of flags from the following list.
<i>perm</i>	permission or mode

Returns

file descriptor If success, else error occurred

10.120.2.3 ocall_print_string() `unsigned int ocall_print_string (const char * str)`

[ocall_print_string\(\)](#) - To print the string and returns the length of string.

Parameters

<i>str</i>	The string to print.
------------	----------------------

Returns

str length of the string.

[ocall_print_string\(\)](#) - Prints the string.

This function invokes OCALL for displaying string type buffer inside the enclave.

Parameters

<i>str</i>	Pointer of the string.
------------	------------------------

Returns

length If success, else error occurred.

[ocall_print_string\(\)](#) - To print the argument string message.

Parameters

<i>str</i>	Pointer of the string.
------------	------------------------

Returns

length If success, else error occurred.

10.120.2.4 ocall_read_file() `int ocall_read_file (int desc,`

```
char * buf,  
size_t len )
```

[ocall_read_file\(\)](#) - To read len bytes form file into the memory area indicated by buf.

Parameters

<i>fdesc</i>	file descripter.
<i>buf</i>	buffer to write data.
<i>len</i>	length of buffer

Returns

integer value If success

[ocall_read_file\(\)](#) - To read len bytes form file into the memory area indicated by buf.

Parameters

<i>desc</i>	file descripter.
<i>buf</i>	buffer to write data.
<i>len</i>	length of buffer

Returns

integer value If success

[ocall_read_file\(\)](#) - Reads a specified number of bytes into a buffer, through a file descriptior.

Parameters

<i>fdesc</i>	an open file descriptor
<i>buf</i>	buffer of at least size bytes
<i>len</i>	number of bytes to be read.

Returns

number of bytes read on success, -1 on failure.

[ocall_read_file\(\)](#) - Used to read from a file.

Parameters

<i>desc</i>	file descriptor
<i>buf</i>	pointer to a buffer
<i>len</i>	Size of elements

Returns

file descriptor If success, else error occurred

10.120.2.5 ocall_ree_time() `int ocall_ree_time (`
`struct ree_time_t * time)`

[ocall_ree_time\(\)](#) - gets the ree execution time.

Parameters

<i>timep</i>	pointer of time.
--------------	------------------

Returns

integer value If success

[ocall_ree_time\(\)](#) - gets the ree execution time.

Parameters

<i>time</i>	pointer of time.
-------------	------------------

Returns

integer value If success

[ocall_ree_time\(\)](#) - Function shall obtain the current time, expressed as seconds and microseconds.

Parameters

<i>timep</i>	timep is a structure type of ree_time_t
--------------	---

Returns

rtn value on success

[ocall_ree_time\(\)](#) - Used to fetch the current time.

Parameters

<i>time</i>	Pointer to a current time.
-------------	----------------------------

Returns

current time If success, else error occurred

10.120.2.6 ocall_write_file() `int ocall_write_file (`
 `int desc,`
 `const char * buf,`
 `size_t len)`

[ocall_write_file\(\)](#) - To write data in to a file.

Parameters

<i>desc</i>	file descriptor.
<i>buf</i>	buffer to write data.
<i>len</i>	length of buffer.

Returns

integer value If success

[ocall_write_file\(\)](#) - Used to write into a file.

Parameters

<i>desc</i>	file descriptor.
<i>buf</i>	pointer to a buffer.
<i>len</i>	Size of elements.

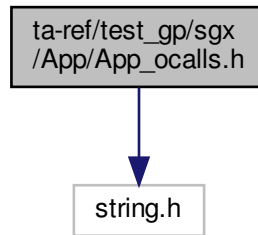
Returns

file descriptor If success, else error occurred.

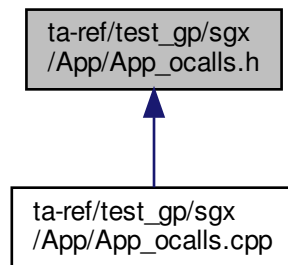
10.121 ta-ref/test_gp/sgx/App/App_ocalls.h File Reference

```
#include <string.h>
```

Include dependency graph for App_ocalls.h:



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



Classes

- struct [ree_time_t](#)

Typedefs

- typedef struct [ree_time_t](#) [ree_time_t](#)

Functions

- unsigned int [ocall_print_string](#) (const char *str)
- int [ocall_open_file](#) (const char *fname, int flags, int perm)
- int [ocall_read_file](#) (int desc, char *buf, size_t len)
- int [ocall_write_file](#) (int desc, const char *buf, size_t len)
- int [ocall_close_file](#) (int desc)
- int [ocall_ree_time](#) (struct [ree_time_t](#) *time)

10.121.1 Typedef Documentation

10.121.1.1 `ree_time_t` `typedef struct ree_time_t ree_time_t`

10.121.2 Function Documentation

10.121.2.1 `ocall_close_file()` `int ocall_close_file (`
`int desc)`

`ocall_close_file()` - To close a file.

Parameters

<i>fdesc</i>	file descriptor.
--------------	------------------

Returns

integer value If success

`ocall_close_file()` - To close a file.

Parameters

<i>desc</i>	file descriptor.
-------------	------------------

Returns

integer value If success

`ocall_close_file()` - Frees the file descriptor in the process.

Parameters

<i>fdesc</i>	fdesc is a file descriptor of the type integer.
--------------	---

Returns

rtn on success,-1 on failure.

`ocall_close_file()` - Used for closing a file

Parameters

<i>desc</i>	File descriptor.
-------------	------------------

Returns

file descripto If success, else error occured.

10.121.2.2 ocall_open_file() `int ocall_open_file (`
 `const char * fname,`
 `int flags,`
 `int perm)`

[ocall_open_file\(\)](#) - To open a file.

Parameters

<i>fname</i>	name of the file.
<i>flags</i>	mode of the file.
<i>perm</i>	indicates permissions of a file.

Returns

integer If success

[ocall_open_file\(\)](#) - To open a file.

Parameters

<i>fname</i>	name of the file.
<i>flags</i>	mode of the file.
<i>perm</i>	indicates permissions of a file.

Returns

integer value If success

[ocall_open_file\(\)](#) - opens a file name which shall be set according to the value of flag and determines the file permission mode.

Parameters

<i>fname</i>	file name is a constant character data type
<i>flags</i>	flags it is datatype of the integer
<i>perm</i>	permissions of the file if it is created

Returns

a nonnegative integer for success or -1 if an error occurred.

[ocall_open_file\(\)](#) - Used for opening a file.

Parameters

<i>fname</i>	File name
<i>flags</i>	Values for oflag are constructed by a bitwise-inclusive OR of flags from the following list.
<i>perm</i>	permission or mode

Returns

file descriptor If success, else error occurred

10.121.2.3 ocall_print_string() `unsigned int ocall_print_string (const char * str)`

[ocall_print_string\(\)](#) - To print the string and returns the length of string.

Parameters

<i>str</i>	The string to print.
------------	----------------------

Returns

str length of the string.

[ocall_print_string\(\)](#) - Prints the string.

This function invokes OCALL for displaying string type buffer inside the enclave.

Parameters

<i>str</i>	Pointer of the string.
------------	------------------------

Returns

length If success, else error occurred.

[ocall_print_string\(\)](#) - To print the string and returns the length of string.

Parameters

<i>str</i>	The string to print.
------------	----------------------

Returns

str length of the string.

[ocall_print_string\(\)](#) - Prints the string.

This function invokes OCALL for displaying string type buffer inside the enclave.

Parameters

<i>str</i>	Pointer of the string.
------------	------------------------

Returns

length If success, else error occurred.

[ocall_print_string\(\)](#) - To print the argument string message.

Parameters

<i>str</i>	Pointer of the string.
------------	------------------------

Returns

length If success, else error occurred.

10.121.2.4 ocall_read_file() `int ocall_read_file (`
 `int desc,`
 `char * buf,`
 `size_t len)`

[ocall_read_file\(\)](#) - To read len bytes form file into the memory area indicated by buf.

Parameters

<i>fdesc</i>	file descriptor.
<i>buf</i>	buffer to write data.
<i>len</i>	length of buffer

Returns

integer value If success

[ocall_read_file\(\)](#) - To read len bytes form file into the memory area indicated by buf.

Parameters

<i>desc</i>	file descriptor.
<i>buf</i>	buffer to write data.
<i>len</i>	length of buffer

Returns

integer value If success

[ocall_read_file\(\)](#) - Reads a specified number of bytes into a buffer, through a file descriptor.

Parameters

<i>fdesc</i>	an open file descriptor
<i>buf</i>	buffer of at least size bytes
<i>len</i>	number of bytes to be read.

Returns

number of bytes read on success, -1 on failure.

[ocall_read_file\(\)](#) - Used to read from a file.

Parameters

<i>desc</i>	file descriptor
<i>buf</i>	pointer to a buffer
<i>len</i>	Size of elements

Returns

file descriptor If success, else error occured

10.121.2.5 ocall_ree.time() `int ocall_ree_time (`
`struct ree_time_t * time)`

[ocall_ree.time\(\)](#) - gets the ree execution time.

Parameters

<i>timep</i>	pointer of time.
--------------	------------------

Returns

integer value If success

[ocall_ree_time\(\)](#) - gets the ree execution time.

Parameters

<i>time</i>	pointer of time.
-------------	------------------

Returns

integer value If success

[ocall_ree_time\(\)](#) - Function shall obtain the current time, expressed as seconds and microseconds.

Parameters

<i>timep</i>	timep is a structure type of ree_time_t
--------------	---

Returns

rtn value on success

[ocall_ree_time\(\)](#) - Used to fetch the current time.

Parameters

<i>time</i>	Pointer to a current time.
-------------	----------------------------

Returns

current time If success, else error occurred

10.121.2.6 ocall_write_file() `int ocall_write_file (`
`int desc,`

```
const char * buf,  
size_t len )
```

[ocall_write_file\(\)](#) - To write data in to a file.

Parameters

<i>desc</i>	file descriptor.
<i>buf</i>	buffer to write data.
<i>len</i>	length of buffer.

Returns

integer value If success

[ocall_write_file\(\)](#) - Used to write into a file.

Parameters

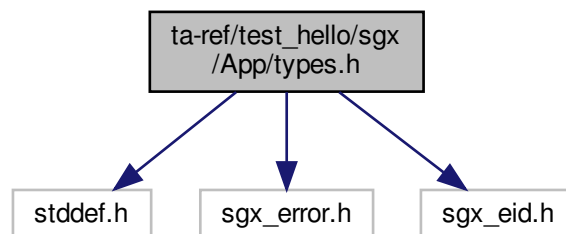
<i>desc</i>	file descriptor.
<i>buf</i>	pointer to a buffer.
<i>len</i>	Size of elements.

Returns

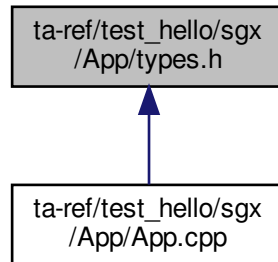
file descriptor If success, else error occurred.

10.122 ta-ref/test_hello/sgx/App/types.h File Reference

```
#include <stddef.h>
#include "sgx_error.h"
#include "sgx_eid.h"
Include dependency graph for types.h:
```



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



Classes

- struct `_sgx_errlist_t`

Typedefs

- typedef struct `_sgx_errlist_t` `sgx_errlist_t`

Variables

- `sgx_enclave_id_t` `global_eid` = 0
- static `sgx_errlist_t` `sgx_errlist` []

10.122.1 Typedef Documentation

10.122.1.1 `sgx_errlist_t` `typedef struct _sgx_errlist_t sgx_errlist_t`

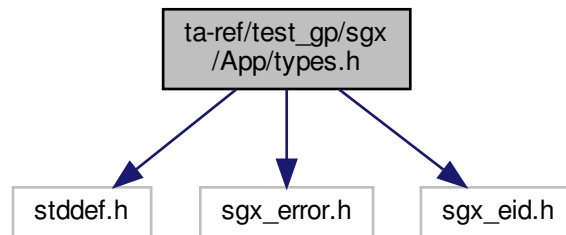
10.122.2 Variable Documentation

10.122.2.1 `global_eid` `sgx_enclave_id_t global_eid` = 0

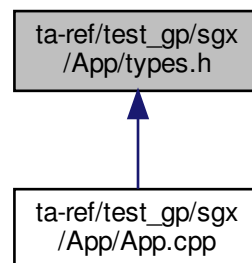
10.122.2.2 `sgx_errlist` `sgx_errlist_t sgx_errlist`[] [static]

10.123 ta-ref/test_gp/sgx/App/types.h File Reference

```
#include <stddef.h>
#include "sgx_error.h"
#include "sgx_eid.h"
Include dependency graph for types.h:
```



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



Classes

- struct [_sgx_errlist_t](#)

Typedefs

- typedef struct [_sgx_errlist_t](#) [sgx_errlist_t](#)

Variables

- [sgx_enclave_id_t](#) [global_eid](#) = 0
- static [sgx_errlist_t](#) [sgx_errlist](#) []

10.123.1 Typedef Documentation

10.123.1.1 `sgx_errlist_t` `typedef struct _sgx_errlist_t sgx_errlist_t`

10.123.2 Variable Documentation

10.123.2.1 `global_eid` `sgx_enclave_id_t global_eid = 0`

10.123.2.2 `sgx_errlist` `sgx_errlist_t sgx_errlist[] [static]`

Index

- `_C_FUNCTION`
 - `user_ta_header.c`, [405](#), [409](#)
- `__GCC_VERSION`
 - `compiler.h`, [73](#)
- `__INTOF_ADD`
 - `compiler.h`, [73](#)
- `__INTOF_ASSIGN`
 - `compiler.h`, [73](#)
- `__INTOF_HALF_MAX_SIGNED`
 - `compiler.h`, [73](#)
- `__INTOF_MAX`
 - `compiler.h`, [73](#)
- `__INTOF_MAX_SIGNED`
 - `compiler.h`, [73](#)
- `__INTOF_MIN`
 - `compiler.h`, [74](#)
- `__INTOF_MIN_SIGNED`
 - `compiler.h`, [74](#)
- `__INTOF_MUL`
 - `compiler.h`, [74](#)
- `__INTOF_SUB`
 - `compiler.h`, [75](#)
- `__ImageBase`
 - `crt.c`, [362](#)
 - `tee_config.h`, [337](#), [339](#), [340](#)
- `__TEE_ObjectHandle`, [35](#)
 - `desc`, [35](#)
 - `flags`, [36](#)
 - `persist_ctx`, [36](#)
 - `private_key`, [36](#)
 - `public_key`, [36](#)
 - `type`, [36](#)
- `__TEE_OperationHandle`, [36](#)
 - `aectx`, [36](#)
 - `aegcm.state`, [37](#)
 - `aeiv`, [37](#)
 - `aekey`, [37](#)
 - `alg`, [37](#)
 - `ctx`, [37](#)
 - `flags`, [37](#)
 - `mode`, [37](#)
 - `prikey`, [37](#)
 - `pubkey`, [37](#)
- `__aligned`
 - `compiler.h`, [71](#)
 - `tee_api_types.h`, [183](#)
- `__attr_const`
 - `compiler.h`, [71](#)
- `__attribute__`
 - `profiler_data.h`, [355](#)
 - `tee-internal-api-machine.c`, [211](#)
 - `tee-internal-api.c`, [224](#)
 - `tee-ta-internal.h`, [81](#)
- `__bss`
 - `compiler.h`, [71](#)
- `__cold`
 - `compiler.h`, [71](#)
- `__compiler_add_overflow`
 - `compiler.h`, [71](#)
- `__compiler_atomic_load`
 - `compiler.h`, [71](#)
- `__compiler_atomic_store`
 - `compiler.h`, [71](#)
- `__compiler_bswap16`
 - `compiler.h`, [72](#)
- `__compiler_bswap32`
 - `compiler.h`, [72](#)
- `__compiler_bswap64`
 - `compiler.h`, [72](#)
- `__compiler_compare_and_swap`
 - `compiler.h`, [72](#)
- `__compiler_mul_overflow`
 - `compiler.h`, [72](#)
- `__compiler_sub_overflow`
 - `compiler.h`, [72](#)
- `__cyg_profile_func`
 - `profiler.c`, [349](#)
- `__cyg_profile_func_enter`
 - `profiler.c`, [349](#)
- `__cyg_profile_func_exit`
 - `profiler.c`, [350](#)
- `__data`
 - `compiler.h`, [72](#)
- `__deprecated`
 - `compiler.h`, [72](#)
- `__early_ta`
 - `compiler.h`, [73](#)
- `__init_array_start`
 - `crt.c`, [356](#)
- `__intof_mul_a0`
 - `compiler.h`, [74](#)
- `__intof_mul_a1`
 - `compiler.h`, [74](#)
- `__intof_mul_b0`
 - `compiler.h`, [74](#)
- `__intof_mul_b1`
 - `compiler.h`, [74](#)
- `__intof_mul_hmask`
 - `compiler.h`, [75](#)
- `__intof_mul_hshift`
 - `compiler.h`, [75](#)
- `__intof_mul_negate`
 - `compiler.h`, [75](#)
- `__intof_mul_t`
 - `compiler.h`, [75](#)
- `__maybe_unused`
 - `compiler.h`, [75](#)
- `__must_check`
 - `compiler.h`, [75](#)
- `__noinline`

- compiler.h, 75
- __noprof
 - compiler.h, 76
- __noreturn
 - compiler.h, 76
- __packed
 - compiler.h, 76
- __printf
 - compiler.h, 76
- __profiler_data, 34
 - callee, 34
 - direction, 34
 - hartid, 34
 - nsec, 34
- __profiler_get_data_ptr
 - profiler.c, 350
- __profiler_head
 - profiler.c, 351
 - tee_profiler.c, 342, 343, 345
- __profiler_header, 35
 - idx, 35
 - size, 35
 - start, 35
- __profiler_map_info
 - profiler.c, 350
 - profiler.h, 351
- __profiler_nsec_t
 - profiler_data.h, 354
- __profiler_unmap_info
 - tee_profiler.c, 340, 342, 344
 - tee_profiler.h, 346–348
- __pure
 - compiler.h, 76
- __rodata
 - compiler.h, 76
- __rodata_unpaged
 - compiler.h, 76
- __section
 - compiler.h, 76
 - user_ta_header.c, 406, 409
- __ta_entry
 - user_ta_header.c, 406, 409
- __unused
 - compiler.h, 76
- __used
 - compiler.h, 76
- __utee_entry
 - user_ta_header.c, 406, 409
- __weak
 - compiler.h, 76
- __wrapper_ocall_close_file
 - Enclave_u.h, 299
- _atoi
 - vsnprintf.c, 247, 253
- _ftoa
 - vsnprintf.c, 247, 254
- _is_digit
 - vsnprintf.c, 248, 254
- _ntoa_format
 - vsnprintf.c, 248, 255
- _ntoa_long
 - vsnprintf.c, 248, 255
- _ntoa_long_long
 - vsnprintf.c, 248, 256
- _out_buffer
 - vsnprintf.c, 248, 257
- _out_char
 - vsnprintf.c, 249, 257
- _out_fct
 - vsnprintf.c, 249, 258
- _out_null
 - vsnprintf.c, 249, 258
- _putchar
 - vsnprintf.c, 246, 251
- _sanctum_dev_public_key
 - test_dev_key.h, 203
- _sanctum_dev_public_key_len
 - test_dev_key.h, 203
- _sanctum_dev_secret_key
 - test_dev_key.h, 203
- _sanctum_dev_secret_key_len
 - test_dev_key.h, 203
- _sgx_errlist_t, 38
 - err, 38
 - msg, 38
 - sug, 38
- _strlen
 - Enclave.c, 394
 - tools.c, 335
 - trace.c, 241, 243
 - vsnprintf.c, 249, 258
- _vsnprintf
 - vsnprintf.c, 249, 259
- a
 - invoke_command_param_t, 41
 - TEE_Attribute, 50
 - TEE_Param, 57
 - TEEC_Value, 69
- addr
 - list, 42
- addrinfo, 38
 - ai_addr, 39
 - ai_addrlen, 39
 - ai_canonname, 39
 - ai_family, 39
 - ai_flags, 39
 - ai_next, 39
 - ai_protocol, 39
 - ai_socktype, 39
- aectx
 - __TEE_OperationHandle, 36
- aegcm_state
 - __TEE_OperationHandle, 37
- aeiv
 - __TEE_OperationHandle, 37
- aekey

- __TEE_OperationHandle, 37
- AES256
 - tee_api_tee_types.h, 232, 234
- ai_addr
 - addrinfo, 39
- ai_addrlen
 - addrinfo, 39
- ai_canonname
 - addrinfo, 39
- ai_family
 - addrinfo, 39
- ai_flags
 - addrinfo, 39
- ai_next
 - addrinfo, 39
- ai_protocol
 - addrinfo, 39
- ai_socktype
 - addrinfo, 39
- alg
 - __TEE_OperationHandle, 37
- algorithm
 - TEE_OperationInfo, 54
 - TEE_OperationInfoMultiple, 56
- aligned
 - crt.c, 356
- allocated_size
 - TEEC_SharedMemory, 66
- analyzer.c
 - BUF_MAX, 322
 - COLS, 322
 - FORMAT, 322
 - main, 322
- App.cpp
 - enc_path, 372, 375
 - initialize_enclave, 373, 376
 - main, 371, 373, 374, 376
 - MAX_PATH, 373, 376
 - print_error_message, 373, 377
 - runtime_path, 372, 375
- App.h
 - ENCLAVE_FILENAME, 415, 417
 - FALSE, 416, 417
 - global_eid, 416, 417
 - TRUE, 416, 417
- App_ocalls.cpp
 - MAX_PATH, 389
 - NO_PERF, 385, 390
 - ocall_close_file, 378, 382, 385, 390
 - ocall_getrandom, 378, 386
 - ocall_invoke_command_callback_write, 378, 386
 - ocall_open_file, 379, 382, 387, 390
 - ocall_print_string, 379, 383, 387, 390
 - ocall_put_invoke_command_result, 380
 - ocall_read_file, 380, 383, 388, 391
 - ocall_read_invoke_param, 380
 - ocall_ree_time, 380, 384, 388, 391
 - ocall_write_file, 381, 384, 388, 392
 - ocall_write_invoke_param, 381
- App_ocalls.h
 - ocall_close_file, 418, 426
 - ocall_open_file, 419, 427
 - ocall_print_string, 421, 428
 - ocall_read_file, 421, 429
 - ocall_ree_time, 423, 430
 - ocall_write_file, 424, 431
 - ree_time_t, 418, 426
- arg
 - out_fct_wrap_type, 45
- array_t
 - user_types.h, 296
- asymmetric_key.c
 - gp_asymmetric_key_sign_test, 309
 - SHA_LENGTH, 309
 - SIG_LENGTH, 309
- ATTEST_DATA_MAXLEN
 - report.h, 77
- attributeID
 - TEE_Attribute, 50
- b
 - invoke_command_param_t, 41
 - ob16_t, 44
 - ob196_t, 44
 - ob256_t, 45
 - TEE_Attribute, 50
 - TEE_Param, 57
 - TEEC_Value, 69
- bench.c
 - benchmark, 278
 - init, 278
 - labels, 280
 - record, 278
 - tee_time_tests, 279
 - time_diff, 279
 - time_test, 280
 - time_to_millis, 280
- bench.h
 - cpu_double_benchmark, 282
 - cpu_int_benchmark, 282
 - io_read_benchmark, 282
 - io_write_benchmark, 282
 - NO_PERF, 282
 - random_memory_benchmark, 283
 - ree_time_test, 283
 - sequential_memory_benchmark, 283
 - system_time_test, 284
- BENCH_TYPE
 - config_bench.h, 286
- benchmark
 - bench.c, 278
- buf
 - param_buffer_t, 46
 - tee_def.h, 290–292
- buf_flag
 - tee_def.h, 290–292
- BUF_MAX

- analyzer.c, 322
- BUF_SIZE
 - config_bench.h, 286
 - Enclave.c, 394
 - main.c, 404
 - nm_parse.c, 325
- buffer
 - TEE_Attribute, 50
 - TEE_Param, 57
 - TEE_SEAID, 58
 - TEEC_SharedMemory, 66
 - TEEC_TempMemoryReference, 67
- buffer_allocated
 - TEEC_SharedMemory, 66
- buffer_t
 - user_types.h, 296
- bufferLen
 - TEE_SEAID, 58
- CALL
 - profiler_data.h, 354
- callee
 - __profiler_data, 34
 - result, 48
- CIPHER_LENGTH
 - symmetric_key.c, 319
 - symmetric_key_gcm.c, 320
- clockSeqAndNode
 - TEE_UUID, 60
 - TEEC_UUID, 68
- COLS
 - analyzer.c, 322
- COMMAND
 - tee_config.h, 338
- commandID
 - invoke_command_t, 42
- compiler.h
 - __GCC_VERSION, 73
 - __INTOF_ADD, 73
 - __INTOF_ASSIGN, 73
 - __INTOF_HALF_MAX_SIGNED, 73
 - __INTOF_MAX, 73
 - __INTOF_MAX_SIGNED, 73
 - __INTOF_MIN, 74
 - __INTOF_MIN_SIGNED, 74
 - __INTOF_MUL, 74
 - __INTOF_SUB, 75
 - __aligned, 71
 - __attr_const, 71
 - __bss, 71
 - __cold, 71
 - __compiler_add_overflow, 71
 - __compiler_atomic_load, 71
 - __compiler_atomic_store, 71
 - __compiler_bswap16, 72
 - __compiler_bswap32, 72
 - __compiler_bswap64, 72
 - __compiler_compare_and_swap, 72
 - __compiler_mul_overflow, 72
 - __compiler_sub_overflow, 72
 - __data, 72
 - __deprecated, 72
 - __early_ta, 73
 - __intof_mul_a0, 74
 - __intof_mul_a1, 74
 - __intof_mul_b0, 74
 - __intof_mul_b1, 74
 - __intof_mul_hmask, 75
 - __intof_mul_hshift, 75
 - __intof_mul_negate, 75
 - __intof_mul_t, 75
 - __maybe_unused, 75
 - __must_check, 75
 - __noinline, 75
 - __noprof, 76
 - __noreturn, 76
 - __packed, 76
 - __printf, 76
 - __pure, 76
 - __rodata, 76
 - __rodata_unpaged, 76
 - __section, 76
 - __unused, 76
 - __used, 76
 - __weak, 76
- config_bench.h
 - BENCH_TYPE, 286
 - BUF_SIZE, 286
 - CPU_DOUBLE_SENSITIVE, 287
 - CPU_INT_SENSITIVE, 287
 - DOUBLE_OFFSET, 286
 - IO_READ_SENSITIVE, 287
 - IO_WRITE_SENSITIVE, 287
 - MULT_SIZE, 286
 - OFFSET, 286
 - RANDOM_MEMORY_SENSITIVE, 287
 - record, 287
 - REE_TIME_TEST, 287
 - SEQUENTIAL_MEMORY_SENSITIVE, 287
 - SYSTEM_TIME_TEST, 287
- config_ref_ta.h
 - GP_ASSERT, 310
 - tee_printf, 310
- content
 - TEE_Attribute, 51
- cpu_bench.c
 - cpu_double_benchmark, 285
 - cpu_int_benchmark, 285
- cpu_double_benchmark
 - bench.h, 282
 - cpu_bench.c, 285
- CPU_DOUBLE_SENSITIVE
 - config_bench.h, 287
- cpu_int_benchmark
 - bench.h, 282
 - cpu_bench.c, 285
- CPU_INT_SENSITIVE

- config_bench.h, 287
- create_htable
 - nm_parse.c, 325
- crt.c
 - __ImageBase, 362
 - __init_array_start, 356
 - aligned, 356
 - crt_end, 356
 - fini_array, 356
 - init_array, 357
 - run_all_test, 358
 - TA_CloseSessionEntryPoint, 359
 - TA_CreateEntryPoint, 359
 - TA_DestroyEntryPoint, 359
 - TA_InvokeCommandEntryPoint, 359
 - TA_OpenSessionEntryPoint, 360
 - TEE_PARAM_TYPE0, 358
 - TEE_PARAM_TYPE1, 358
 - tee_printf, 360
- crt.h
 - crt_begin, 363
 - crt_end, 363
 - main, 363
- crt_begin
 - crt.h, 363
- crt_end
 - crt.c, 356
 - crt.h, 363
- ctx
 - __TEE_OperationHandle, 37
 - TEEC_Session, 65
- data
 - enclave_report, 40
- data_len
 - enclave_report, 40
- DATA_LENGTH
 - secure_storage.c, 317
- dataPosition
 - TEE_ObjectInfo, 52
- dataSize
 - TEE_ObjectInfo, 53
- depth
 - result, 48
- desc
 - __TEE_ObjectHandle, 35
- dev_public_key
 - report, 48
- dhex_dump
 - trace.h, 208
- DHEXDUMP
 - trace.h, 205
- digestLength
 - TEE_OperationInfo, 54
 - TEE_OperationInfoMultiple, 56
- direction
 - __profiler_data, 34
- direction_t
 - profiler_data.h, 354
- DMREQ_FINISH
 - tee_api_types.h, 182
- DMREQ_WRITE
 - tee_api_types.h, 182
- DMSG
 - trace.h, 205
- DMSG_RAW
 - trace.h, 205
- DOUBLE_OFFSET
 - config_bench.h, 286
- DPRINT_STACK
 - trace.h, 205
- eapp_entry
 - Enclave.c, 393
 - startup.c, 369
- ecall_ta_main
 - Enclave.c, 398
 - startup.c, 370
- EDGE_EXTERN_BEGIN
 - Enclave_u.h, 299
- EDGE_EXTERN_END
 - Enclave_u.h, 299
- EDGE_OUT_WITH_STRUCTURE
 - ocalls.h, 301
- EMSG
 - trace.h, 206
- EMSG_RAW
 - trace.h, 206
- enc_path
 - App.cpp, 372, 375
- enclave
 - report, 48
- Enclave.c
 - _strlen, 394
 - BUF_SIZE, 394
 - eapp_entry, 393
 - ecall_ta_main, 398
 - main, 399–401
 - MESSAGE, 393, 394, 398
 - print_buf, 397
 - print_pos, 397
 - run_all_test, 395
 - TA_CloseSessionEntryPoint, 395
 - TA_CreateEntryPoint, 395
 - TA_DestroyEntryPoint, 396
 - TA_InvokeCommandEntryPoint, 396
 - TA_OpenSessionEntryPoint, 396
 - TEE_PARAM_TYPE1, 394
 - tee_printf, 397
- Enclave.h
 - gp_asymmetric_key_sign_test, 306
 - gp_message_digest_test, 306
 - gp_random_test, 307
 - gp_ree_time_test, 307
 - gp_secure_storage_test, 307
 - gp_symmetric_key_enc_verify_test, 307
 - gp_symmetric_key_gcm_verify_test, 307
 - gp_trusted_time_test, 308

- TA_REF_RUN_ALL, 305
- TA_REF_UUID, 305
- ENCLAVE_FILENAME
 - App.h, 415, 417
- enclave_report, 40
 - data, 40
 - data_len, 40
 - hash, 40
 - signature, 40
- Enclave_t.h
 - ocall_file_read, 297
 - ocall_file_read_full, 297
 - ocall_file_write, 297
 - ocall_file_write_full, 298
- Enclave_u.h
 - _wrapper_ocall_close_file, 299
 - EDGE_EXTERN_BEGIN, 299
 - EDGE_EXTERN_END, 299
 - register_functions, 300
- end
 - result, 49
- end_hartid
 - result, 49
- EPRINT_STACK
 - trace.h, 206
- err
 - _sgx_errlist_t, 38
- events
 - pollfd, 46
- FALSE
 - App.h, 416, 417
- fct
 - out_fct_wrap_type, 45
- fctprintf
 - vsnprintf.c, 249, 259
- fd
 - pollfd, 46
 - TEEC_Context, 60
- fini_array
 - crt.c, 356
- flags
 - _TEE_ObjectHandle, 36
 - _TEE_OperationHandle, 37
 - TEEC_SharedMemory, 66
- flags2flags
 - tee-internal-api.c, 213, 224
- FLAGS_CHAR
 - vsnprintf.c, 246, 252
- FLAGS_HASH
 - vsnprintf.c, 246, 252
- FLAGS_LEFT
 - vsnprintf.c, 246, 252
- FLAGS_LONG
 - vsnprintf.c, 246, 252
- FLAGS_LONG_LONG
 - vsnprintf.c, 246, 252
- FLAGS_PLUS
 - vsnprintf.c, 246, 252
- FLAGS_PRECISION
 - vsnprintf.c, 246, 252
- FLAGS_SHORT
 - vsnprintf.c, 246, 252
- FLAGS_SPACE
 - vsnprintf.c, 246, 252
- FLAGS_UPPERCASE
 - vsnprintf.c, 246, 252
- FLAGS_ZEROPAD
 - vsnprintf.c, 247, 252
- FMSG
 - trace.h, 206
- FMSG_RAW
 - trace.h, 206
- FORMAT
 - analyzer.c, 322
- FPERMS
 - tee-internal-api.c, 212, 223
- FPRINT_STACK
 - trace.h, 206
- func_name
 - nm_info, 43
- GCM_ST_AAD
 - tee-internal-api-cryptlib.c, 265
- GCM_ST_ACTIVE
 - tee-internal-api-cryptlib.c, 265
- GCM_ST_FINAL
 - tee-internal-api-cryptlib.c, 265
- GCM_ST_INIT
 - tee-internal-api-cryptlib.c, 265
- get_func_name
 - nm_parse.c, 325
 - nm_parse.h, 328
- get_key
 - nm_parse.c, 326
- GetRelTimeEnd
 - tee-internal-api.c, 214, 224
 - tee-ta-internal.h, 82
- GetRelTimeStart
 - tee-internal-api.c, 214, 225
 - tee-ta-internal.h, 83
- global_eid
 - App.h, 416, 417
 - types.h, 434, 436
- GP_ASSERT
 - config_ref_ta.h, 310
- gp_asymmetric_key_sign_test
 - asymmetric_key.c, 309
- Enclave.h, 306
- gp_test.h, 312
- gp_invokecommand_test
 - gp_test.h, 312
- gp_message_digest_test
 - Enclave.h, 306
 - gp_test.h, 312
 - message_digest.c, 316
- gp_random_test
 - Enclave.h, 307

- gp_test.h, 312
- random.c, 317
- gp_ree_time_test
 - Enclave.h, 307
 - gp_test.h, 312
 - time.c, 321
- gp_secure_storage_test
 - Enclave.h, 307
 - gp_test.h, 313
 - secure_storage.c, 318
- gp_symmetric_key_enc_verify_test
 - Enclave.h, 307
 - gp_test.h, 313
 - symmetric_key.c, 319
- gp_symmetric_key_gcm_verify_test
 - Enclave.h, 307
 - gp_test.h, 313
 - symmetric_key_gcm.c, 320
- gp_test.h
 - gp_asymmetric_key_sign_test, 312
 - gp_invokecommand_test, 312
 - gp_message_digest_test, 312
 - gp_random_test, 312
 - gp_ree_time_test, 312
 - gp_secure_storage_test, 313
 - gp_symmetric_key_enc_verify_test, 313
 - gp_symmetric_key_gcm_verify_test, 313
 - gp_trusted_time_test, 313
- gp_trusted_time_test
 - Enclave.h, 308
 - gp_test.h, 313
 - time.c, 321
- handleFlags
 - TEE_ObjectInfo, 53
- handleState
 - TEE_OperationInfo, 54
 - TEE_OperationInfoMultiple, 56
- hartid
 - _profiler_data, 34
- hash
 - enclave_report, 40
 - sm_report, 49
- HASH_SIZE
 - nm_parse.h, 328
- id
 - TEEC_SharedMemory, 66
- idx
 - _profiler_header, 35
 - nm_parse.c, 327
 - profiler_data.h, 355
 - result, 49
- IMSG
 - trace.h, 206
- IMSG_RAW
 - trace.h, 206
- INC
 - memory_bench.c, 293
- init
 - bench.c, 278
- init_array
 - crt.c, 357
- initialize_enclave
 - App.cpp, 373, 376
- INMSG
 - trace.h, 206
- insert_nm
 - nm_parse.c, 326
- invoke_command.c
 - TA_MAX_SIZE, 314
 - TEEP_AGENT_TA_DELETE, 314
 - TEEP_AGENT_TA_EXIT, 314
 - TEEP_AGENT_TA_INSTALL, 314
 - TEEP_AGENT_TA_LOAD, 315
 - TEEP_AGENT_TA_NONE, 315
- invoke_command_param_t, 40
 - a, 41
 - b, 41
 - ocalls.h, 302
 - size, 41
- invoke_command_t, 41
 - commandID, 42
 - ocalls.h, 302
 - params, 42
 - paramTypes, 42
- io_bench.c
 - io_read_benchmark, 288
 - io_write_benchmark, 289
 - SPLITS, 288
- io_read_benchmark
 - bench.h, 282
 - io_bench.c, 288
- IO_READ_SENSITIVE
 - config_bench.h, 287
- io_write_benchmark
 - bench.h, 282
 - io_bench.c, 289
- IO_WRITE_SENSITIVE
 - config_bench.h, 287
- IPRINT_STACK
 - trace.h, 206
- is_empty
 - stack.h, 330
- keyInformation
 - TEE_OperationInfoMultiple, 56
- keySize
 - TEE_ObjectInfo, 53
 - TEE_OperationInfo, 54
 - TEE_OperationInfoKey, 55
- labels
 - bench.c, 280
- length
 - TEE_Attribute, 51
- list, 42
 - addr, 42

- next, [43](#)
 - nm, [43](#)
- LOG_FILE
 - profiler_data.h, [354](#)
- login
 - TEE_Identity, [52](#)
- LOOPS_PER_THREAD
 - user_types.h, [296](#)
- main
 - analyzer.c, [322](#)
 - App.cpp, [371](#), [373](#), [374](#), [376](#)
 - crt.h, [363](#)
 - Enclave.c, [399–401](#)
 - main.c, [402](#), [404](#)
- main.c
 - BUF_SIZE, [404](#)
 - main, [402](#), [404](#)
 - print_buf, [403](#)
 - PRINT_BUF_SIZE, [402](#), [404](#)
 - TEEC_PARAM_TYPE0, [404](#)
 - TEEC_PARAM_TYPE1, [402](#), [404](#)
- MAX_ADDR
 - nm_parse.c, [325](#)
- MAX_FUNC_PRINT_SIZE
 - trace.h, [207](#)
- MAX_PATH
 - App.cpp, [373](#), [376](#)
 - App_ocalls.cpp, [389](#)
- MAX_PRINT_SIZE
 - trace.h, [207](#)
- maxKeySize
 - TEE_ObjectInfo, [53](#)
 - TEE_OperationInfo, [54](#)
 - TEE_OperationInfoMultiple, [56](#)
- maxObjectSize
 - TEE_ObjectInfo, [53](#)
- MBEDCRYPT
 - tee_api_tee_types.h, [232](#), [234](#)
- MDSIZE
 - report.h, [77](#)
- memory_bench.c
 - INC, [293](#)
 - random_memory_benchmark, [293](#)
 - sequential_memory_benchmark, [294](#)
- memref
 - TEE_Param, [57](#)
 - TEEC_Parameter, [63](#)
- MESSAGE
 - Enclave.c, [393](#), [394](#), [398](#)
- message_digest.c
 - gp_message_digest_test, [316](#)
 - SHA_LENGTH, [315](#)
 - SIG_LENGTH, [316](#)
- millis
 - ree_time_t, [47](#)
 - TEE_Time, [59](#)
- mode
 - __TEE_OperationHandle, [37](#)
 - TEE_OperationInfo, [54](#)
 - TEE_OperationInfoMultiple, [56](#)
- MSG
 - trace.h, [207](#)
- msg
 - _sgx_errlist_t, [38](#)
- MSG_RAW
 - trace.h, [207](#)
- MULT_SIZE
 - config_bench.h, [286](#)
- next
 - list, [43](#)
- nfds_t
 - tee_api_types.h, [183](#)
- nm
 - list, [43](#)
- nm_info, [43](#)
 - func_name, [43](#)
 - type, [43](#)
- nm_parse.c
 - BUF_SIZE, [325](#)
 - create_htable, [325](#)
 - get_func_name, [325](#)
 - get_key, [326](#)
 - idx, [327](#)
 - insert_nm, [326](#)
 - MAX_ADDR, [325](#)
 - nm_pool, [327](#)
 - parse_nm, [326](#)
 - POOL_SIZE, [325](#)
- nm_parse.h
 - get_func_name, [328](#)
 - HASH_SIZE, [328](#)
 - parse_nm, [328](#)
- nm_pool
 - nm_parse.c, [327](#)
- NO_PERF
 - App_ocalls.cpp, [385](#), [390](#)
 - bench.h, [282](#)
 - profiler_attrs.h, [352](#)
- nsec
 - __profiler_data, [34](#)
- numberOfKeys
 - TEE_OperationInfoMultiple, [56](#)
- O_CREAT
 - ocalls.h, [301](#)
 - tee-internal-api.c, [213](#), [223](#)
- O_EXCL
 - ocalls.h, [301](#)
 - tee-internal-api.c, [213](#), [223](#)
- O_RDONLY
 - ocalls.h, [302](#)
 - tee-internal-api.c, [213](#), [223](#)
- O_RDWR
 - ocalls.h, [302](#)
 - tee-internal-api.c, [213](#), [223](#)
- O_TRUNC

- ocalls.h, 302
 - tee-internal-api.c, 213, 223
- O_WRONLY
 - ocalls.h, 302
 - tee-internal-api.c, 213, 224
- ob16_t, 43
 - b, 44
 - ocalls.h, 302
 - ret, 44
- ob196_t, 44
 - b, 44
 - ocalls.h, 302
 - ret, 44
- ob256_t, 44
 - b, 45
 - ocalls.h, 302
 - ret, 45
- objectSize
 - TEE_ObjectInfo, 53
- objectType
 - TEE_ObjectInfo, 53
- objectUsage
 - TEE_ObjectInfo, 53
- ocall_close_file
 - App_ocalls.cpp, 378, 382, 385, 390
 - App_ocalls.h, 418, 426
 - ocalls.h, 302
- ocall_file_read
 - Enclave.t.h, 297
- ocall_file_read_full
 - Enclave.t.h, 297
- ocall_file_write
 - Enclave.t.h, 297
- ocall_file_write_full
 - Enclave.t.h, 298
- ocall_fstat_size
 - ocalls.h, 303
- ocall_getrandom
 - App_ocalls.cpp, 378, 386
- ocall_getrandom16
 - ocalls.h, 303
- ocall_getrandom196
 - ocalls.h, 304
- ocall_invoke_command_callback_write
 - App_ocalls.cpp, 378, 386
- ocall_open_file
 - App_ocalls.cpp, 379, 382, 387, 390
 - App_ocalls.h, 419, 427
 - ocalls.h, 304
- ocall_print_string
 - App_ocalls.cpp, 379, 383, 387, 390
 - App_ocalls.h, 421, 428
 - ocalls.h, 304
- ocall_print_string_wrapper
 - ocall_wrapper.c, 368, 369
 - ocall_wrapper.h, 364
- ocall_pull_invoke_command
 - ocalls.h, 304
- ocall_put_invoke_command_result
 - App_ocalls.cpp, 380
 - ocalls.h, 304
- ocall_read_file
 - App_ocalls.cpp, 380, 383, 388, 391
 - App_ocalls.h, 421, 429
- ocall_read_file256
 - ocalls.h, 304
- ocall_read_invoke_param
 - App_ocalls.cpp, 380
 - ocalls.h, 304
- ocall_ree_time
 - App_ocalls.cpp, 380, 384, 388, 391
 - App_ocalls.h, 423, 430
 - ocalls.h, 304
- ocall_unlink
 - ocalls.h, 304
- ocall_wrapper.c
 - ocall_print_string_wrapper, 368, 369
- ocall_wrapper.h
 - ocall_print_string_wrapper, 364
- ocall_write_file
 - App_ocalls.cpp, 381, 384, 388, 392
 - App_ocalls.h, 424, 431
- ocall_write_file256
 - ocalls.h, 305
- ocall_write_invoke_param
 - App_ocalls.cpp, 381
 - ocalls.h, 305
- ocalls.h
 - EDGE_OUT_WITH_STRUCTURE, 301
 - invoke_command_param_t, 302
 - invoke_command_t, 302
 - O_CREAT, 301
 - O_EXCL, 301
 - O_RDONLY, 302
 - O_RDWR, 302
 - O_TRUNC, 302
 - O_WRONLY, 302
 - ob16_t, 302
 - ob196_t, 302
 - ob256_t, 302
 - ocall_close_file, 302
 - ocall_fstat_size, 303
 - ocall_getrandom16, 303
 - ocall_getrandom196, 304
 - ocall_open_file, 304
 - ocall_print_string, 304
 - ocall_pull_invoke_command, 304
 - ocall_put_invoke_command_result, 304
 - ocall_read_file256, 304
 - ocall_read_invoke_param, 304
 - ocall_ree_time, 304
 - ocall_unlink, 304
 - ocall_write_file256, 305
 - ocall_write_invoke_param, 305
 - param_buffer_t, 302
 - ree_time_t, 302

- OFFSET
 - config_bench.h, 286
- offset
 - TEEC_RegisteredMemoryReference, 64
- OpenPersistentObject
 - tee-internal-api.c, 214, 225
- operationClass
 - TEE_OperationInfo, 54
 - TEE_OperationInfoMultiple, 56
- operationState
 - TEE_OperationInfoMultiple, 56
- out_fct_type
 - vsnprintf.c, 247, 253
- out_fct_wrap_type, 45
 - arg, 45
 - fct, 45
- OUTMSG
 - trace.h, 207
- OUTRMSG
 - trace.h, 207
- param_buffer_t, 45
 - buf, 46
 - ocalls.h, 302
 - size, 46
- params
 - invoke_command_t, 42
 - TEEC_Operation, 62
- paramTypes
 - invoke_command_t, 42
 - TEEC_Operation, 62
- parent
 - TEEC_RegisteredMemoryReference, 64
- parse_nm
 - nm_parse.c, 326
 - nm_parse.h, 328
- perf_buffer
 - tee_config.h, 337, 339, 340
- PERF_SECTION
 - profiler_attrs.h, 352
- PERF_SIZE
 - profiler_data.h, 354
- persist_ctx
 - __TEE_ObjectHandle, 36
- pollfd, 46
 - events, 46
 - fd, 46
 - revents, 46
- POOL_SIZE
 - nm_parse.c, 325
- pop
 - stack.h, 330
- pos
 - stack.h, 331
- pr_deb
 - tee-common.h, 79
- prikey
 - __TEE_OperationHandle, 37
- print_buf
 - Enclave.c, 397
 - main.c, 403
- PRINT_BUF_SIZE
 - main.c, 402, 404
- print_error_message
 - App.cpp, 373, 377
- print_pos
 - Enclave.c, 397
- printf
 - tools.c, 335
 - tools.h, 366
- PRINTF_FTOA_BUFFER_SIZE
 - vsnprintf.c, 247, 253
- PRINTF_NTOA_BUFFER_SIZE
 - vsnprintf.c, 247, 253
- PRINTF_SUPPORT_FLOAT
 - vsnprintf.c, 247, 253
- PRINTF_SUPPORT_LONG_LONG
 - vsnprintf.c, 247, 253
- PRINTF_SUPPORT_PTRDIFF_T
 - vsnprintf.c, 247, 253
- private_key
 - __TEE_ObjectHandle, 36
- profiler.c
 - __cyg_profile_func, 349
 - __cyg_profile_func_enter, 349
 - __cyg_profile_func_exit, 350
 - __profiler_get_data_ptr, 350
 - __profiler_head, 351
 - __profiler_map_info, 350
- profiler.h
 - __profiler_map_info, 351
- profiler_attrs.h
 - NO_PERF, 352
 - PERF_SECTION, 352
 - USED, 352
- profiler_data.h
 - __attribute__, 355
 - __profiler_nsec_t, 354
 - CALL, 354
 - direction_t, 354
 - idx, 355
 - LOG_FILE, 354
 - PERF_SIZE, 354
 - RET, 354
 - size, 355
 - START, 354
 - start, 355
- profiler_write
 - tee_profiler.c, 341, 343, 344
 - tools.c, 331–334
- pubkey
 - __TEE_OperationHandle, 37
- public_key
 - __TEE_ObjectHandle, 36
 - sm_report, 49
- PUBLIC_KEY_SIZE
 - report.h, 77

- push
 - stack.h, 331
- putchar
 - tools.c, 336
 - tools.h, 366
 - vsnprintf.c, 249, 261
- puts
 - tools.c, 336
 - tools.h, 367
- random.c
 - gp_random_test, 317
- random_memory_benchmark
 - bench.h, 283
 - memory_bench.c, 293
- RANDOM_MEMORY_SENSITIVE
 - config_bench.h, 287
- record
 - bench.c, 278
 - config_bench.h, 287
- ree_time_t, 47
 - App_ocalls.h, 418, 426
 - millis, 47
 - ocalls.h, 302
 - seconds, 47
- REE_TIME_TEST
 - config_bench.h, 287
- ree_time_test
 - bench.h, 283
 - time_test.c, 295
- ref
 - TEE_Attribute, 51
- reg_mem
 - TEEC_Context, 61
- register_functions
 - Enclave_u.h, 300
- registered_fd
 - TEEC_SharedMemory, 67
- report, 47
 - dev_public_key, 48
 - enclave, 48
 - sm, 48
- report.h
 - ATTEST_DATA_MAXLEN, 77
 - MDSIZE, 77
 - PUBLIC_KEY_SIZE, 77
 - SIGNATURE_SIZE, 78
- requiredKeyUsage
 - TEE_OperationInfo, 54
 - TEE_OperationInfoKey, 55
- result, 48
 - callee, 48
 - depth, 48
 - end, 49
 - end_hartid, 49
 - idx, 49
 - start, 49
 - start_hartid, 49
- RET
 - profiler_data.h, 354
- ret
 - ob16_t, 44
 - ob196_t, 44
 - ob256_t, 45
- revents
 - pollfd, 46
- run_all_test
 - crt.c, 358
 - Enclave.c, 395
- runtime_path
 - App.cpp, 372, 375
- seconds
 - ree_time_t, 47
 - TEE_Time, 59
- secure_storage.c
 - DATA_LENGTH, 317
 - gp_secure_storage_test, 318
- selectResponseEnable
 - TEE_SEReaderProperties, 58
- sePresent
 - TEE_SEReaderProperties, 59
- sequential_memory_benchmark
 - bench.h, 283
 - memory_bench.c, 294
- SEQUENTIAL_MEMORY_SENSITIVE
 - config_bench.h, 287
- session
 - TEEC_Operation, 62
- session_id
 - TEEC_Session, 65
- set_object_key
 - tee-internal-api.c, 215, 226
- sgx_errlist
 - types.h, 434, 436
- sgx_errlist_t
 - types.h, 434, 436
- SHA_LENGTH
 - asymmetric_key.c, 309
 - message_digest.c, 315
 - tee_api_tee_types.h, 232, 234
- shadow_buffer
 - TEEC_SharedMemory, 67
- SIG_LENGTH
 - asymmetric_key.c, 309
 - message_digest.c, 316
 - tee-internal-api-cryptlib.c, 265
- signature
 - enclave_report, 40
 - sm_report, 50
- SIGNATURE_SIZE
 - report.h, 78
- size
 - _profiler_header, 35
 - invoke_command_param_t, 41
 - param_buffer_t, 46
 - profiler_data.h, 355
 - TEE_Param, 57

- TEEC_RegisteredMemoryReference, [64](#)
- TEEC_SharedMemory, [67](#)
- TEEC_TempMemoryReference, [68](#)
- sm
 - report, [48](#)
- sm_report, [49](#)
 - hash, [49](#)
 - public_key, [49](#)
 - signature, [50](#)
- SMSG
 - trace.h, [207](#)
- snprintf
 - vsnprintf.c, [250](#), [261](#)
- socklen_t
 - tee_api_types.h, [183](#)
- SPLITS
 - io_bench.c, [288](#)
- sprintf
 - vsnprintf.c, [250](#), [261](#)
- stack
 - stack.h, [331](#)
- stack.h
 - is_empty, [330](#)
 - pop, [330](#)
 - pos, [331](#)
 - push, [331](#)
 - stack, [331](#)
 - STACK_SIZE, [330](#)
- STACK_SIZE
 - stack.h, [330](#)
- START
 - profiler_data.h, [354](#)
- start
 - _profiler_header, [35](#)
 - profiler_data.h, [355](#)
 - result, [49](#)
- start_hartid
 - result, [49](#)
- started
 - TEEC_Operation, [62](#)
- startup.c
 - eapp_entry, [369](#)
 - ecall_ta_main, [370](#)
- sug
 - _sgx_errlist_t, [38](#)
- symmetric.key.c
 - CIPHER_LENGTH, [319](#)
 - gp_symmetric_key_enc_verify_test, [319](#)
- symmetric.key_gcm.c
 - CIPHER_LENGTH, [320](#)
 - gp_symmetric_key_gcm_verify_test, [320](#)
- SYSTEM_TIME_TEST
 - config_bench.h, [287](#)
- system_time_test
 - bench.h, [284](#)
 - time_test.c, [295](#)
- ta-ref/api/include/compiler.h, [70](#)
- ta-ref/api/include/report.h, [77](#)
- ta-ref/api/include/tee-common.h, [78](#)
- ta-ref/api/include/tee-ta-internal.h, [79](#)
- ta-ref/api/include/tee_api.h, [103](#)
- ta-ref/api/include/tee_api_defines.h, [139](#)
- ta-ref/api/include/tee_api_defines_extensions.h, [177](#)
- ta-ref/api/include/tee_api_types.h, [181](#)
- ta-ref/api/include/tee_client_api.h, [185](#)
- ta-ref/api/include/tee_internal_api.h, [197](#)
- ta-ref/api/include/tee_internal_api_extensions.h, [198](#)
- ta-ref/api/include/tee_ta_api.h, [201](#)
- ta-ref/api/include/test_dev_key.h, [203](#)
- ta-ref/api/include/trace.h, [204](#)
- ta-ref/api/include/trace_levels.h, [209](#)
- ta-ref/api/keystone/tee-internal-api-machine.c, [210](#)
- ta-ref/api/keystone/tee-internal-api.c, [211](#)
- ta-ref/api/keystone/tee_api_tee_types.h, [231](#)
- ta-ref/api/keystone/teec_stub.c, [235](#)
- ta-ref/api/keystone/trace.c, [238](#)
- ta-ref/api/keystone/vsnprintf.c, [244](#)
- ta-ref/api/optee/tee_api_tee_types.h, [233](#)
- ta-ref/api/sgx/tee-internal-api.c, [222](#)
- ta-ref/api/sgx/tee_api_tee_types.h, [233](#)
- ta-ref/api/tee-internal-api-cryptlib.c, [263](#)
- ta-ref/benchmark/bench.c, [277](#)
- ta-ref/benchmark/bench.h, [281](#)
- ta-ref/benchmark/cpu_bench.c, [284](#)
- ta-ref/benchmark/include/config_bench.h, [285](#)
- ta-ref/benchmark/io_bench.c, [287](#)
- ta-ref/benchmark/keystone/tee_def.h, [289](#)
- ta-ref/benchmark/memory_bench.c, [293](#)
- ta-ref/benchmark/optee/tee_def.h, [290](#)
- ta-ref/benchmark/sgx/tee_def.h, [291](#)
- ta-ref/benchmark/time_test.c, [294](#)
- ta-ref/docs/building.md, [295](#)
- ta-ref/docs/gp_api.md, [295](#)
- ta-ref/docs/how_to_program_on_ta-ref.md, [295](#)
- ta-ref/docs/overview_of_ta-ref.md, [295](#)
- ta-ref/docs/preparation.md, [295](#)
- ta-ref/docs/running_on_dev_boards.md, [295](#)
- ta-ref/edger/edger8r/user_types.h, [295](#)
- ta-ref/edger/keyedge/Enclave_t.c, [296](#)
- ta-ref/edger/keyedge/Enclave_t.h, [297](#)
- ta-ref/edger/keyedge/Enclave_u.c, [298](#)
- ta-ref/edger/keyedge/Enclave_u.h, [299](#)
- ta-ref/edger/keyedge/ocalls.h, [300](#)
- ta-ref/edger/optee/Enclave.h, [305](#)
- ta-ref/edger/optee/Enclave_t.h, [298](#)
- ta-ref/gp/asymmetric.key.c, [308](#)
- ta-ref/gp/include/config_ref_ta.h, [309](#)
- ta-ref/gp/include/gp_test.h, [311](#)
- ta-ref/gp/invoke_command.c, [314](#)
- ta-ref/gp/message_digest.c, [315](#)
- ta-ref/gp/random.c, [316](#)
- ta-ref/gp/secure_storage.c, [317](#)
- ta-ref/gp/symmetric.key.c, [318](#)
- ta-ref/gp/symmetric.key_gcm.c, [319](#)
- ta-ref/gp/time.c, [320](#)
- ta-ref/profiler/analyzer/analyzer.c, [321](#)

- ta-ref/profiler/analyzer/analyzer.h, 323
- ta-ref/profiler/analyzer/nm_parse.c, 324
- ta-ref/profiler/analyzer/nm_parse.h, 327
- ta-ref/profiler/analyzer/stack.h, 329
- ta-ref/profiler/app/tools.c, 331
- ta-ref/profiler/keystone/Enclave/tools.c, 332
- ta-ref/profiler/keystone/tee_config.h, 337
- ta-ref/profiler/keystone/tee_profiler.c, 340
- ta-ref/profiler/keystone/tee_profiler.h, 346
- ta-ref/profiler/optee/Enclave/tools.c, 333
- ta-ref/profiler/optee/tee_config.h, 338
- ta-ref/profiler/optee/tee_profiler.c, 342
- ta-ref/profiler/optee/tee_profiler.h, 347
- ta-ref/profiler/profiler.c, 348
- ta-ref/profiler/profiler.h, 351
- ta-ref/profiler/profiler_attrs.h, 352
- ta-ref/profiler/profiler_data.h, 353
- ta-ref/profiler/sgx/Enclave/tools.c, 334
- ta-ref/profiler/sgx/tee_config.h, 339
- ta-ref/profiler/sgx/tee_profiler.c, 343
- ta-ref/profiler/sgx/tee_profiler.h, 347
- ta-ref/test_gp/crt.c, 355
- ta-ref/test_gp/include/crt.h, 362
- ta-ref/test_gp/include/ocall_wrapper.h, 364
- ta-ref/test_gp/include/random.h, 365
- ta-ref/test_gp/include/tools.h, 366
- ta-ref/test_gp/keystone/App/App.cpp, 374
- ta-ref/test_gp/keystone/App/App_ocalls.cpp, 384
- ta-ref/test_gp/keystone/Enclave/Enclave.c, 399
- ta-ref/test_gp/keystone/Enclave/ocall_wrapper.c, 367
- ta-ref/test_gp/keystone/Enclave/startup.c, 369
- ta-ref/test_gp/keystone/Enclave/trace.c, 240
- ta-ref/test_gp/optee/App/main.c, 403
- ta-ref/test_gp/optee/Enclave/crt.c, 357
- ta-ref/test_gp/optee/Enclave/Enclave.c, 400
- ta-ref/test_gp/optee/Enclave/trace.c, 241
- ta-ref/test_gp/optee/Enclave/user_ta_header.c, 408
- ta-ref/test_gp/optee/Enclave/user_ta_header_defines.h, 413
- ta-ref/test_gp/sgx/App/App.cpp, 375
- ta-ref/test_gp/sgx/App/App.h, 416
- ta-ref/test_gp/sgx/App/App_ocalls.cpp, 389
- ta-ref/test_gp/sgx/App/App_ocalls.h, 424
- ta-ref/test_gp/sgx/App/types.h, 435
- ta-ref/test_gp/sgx/Enclave/Enclave.c, 401
- ta-ref/test_gp/sgx/Enclave/Enclave.h, 306
- ta-ref/test_gp/sgx/Enclave/ocall_wrapper.c, 368
- ta-ref/test_gp/sgx/Enclave/startup.c, 370
- ta-ref/test_gp/sgx/Enclave/trace.c, 243
- ta-ref/test_gp/tools.c, 335
- ta-ref/test_gp/vsnprintf.c, 250
- ta-ref/test_hello/keystone/App/App.cpp, 371
- ta-ref/test_hello/keystone/App/App_ocalls.cpp, 377
- ta-ref/test_hello/keystone/Enclave/Enclave.c, 392
- ta-ref/test_hello/optee/App/main.c, 401
- ta-ref/test_hello/optee/Enclave/Enclave.c, 393
- ta-ref/test_hello/optee/Enclave/user_ta_header.c, 405
- ta-ref/test_hello/optee/Enclave/user_ta_header_defines.h, 411
- ta-ref/test_hello/sgx/App/App.cpp, 372
- ta-ref/test_hello/sgx/App/App.h, 415
- ta-ref/test_hello/sgx/App/App_ocalls.cpp, 381
- ta-ref/test_hello/sgx/App/App_ocalls.h, 417
- ta-ref/test_hello/sgx/App/types.h, 433
- ta-ref/test_hello/sgx/Enclave/Enclave.c, 398
- TA_CloseSessionEntryPoint
 - crt.c, 359
 - Enclave.c, 395
 - tee_ta_api.h, 202
- TA_CreateEntryPoint
 - crt.c, 359
 - Enclave.c, 395
 - tee_ta_api.h, 202
- TA_CURRENT_TA_EXT_PROPERTIES
 - user_ta_header_defines.h, 412, 414
- TA_DATA_SIZE
 - user_ta_header_defines.h, 412, 414
- TA_DESCRIPTION
 - user_ta_header.c, 406, 409
 - user_ta_header_defines.h, 412, 414
- TA_DestroyEntryPoint
 - crt.c, 359
 - Enclave.c, 396
 - tee_ta_api.h, 202
- TA_EXPORT
 - tee_ta_api.h, 201
- TA_FLAGS
 - user_ta_header_defines.h, 412, 414
- TA_FRAMEWORK_STACK_SIZE
 - user_ta_header.c, 406, 409
- ta_heap
 - user_ta_header.c, 407, 410
- ta_heap_size
 - user_ta_header.c, 407, 410
- TA_InvokeCommandEntryPoint
 - crt.c, 359
 - Enclave.c, 396
 - tee_ta_api.h, 202
- TA_MAX_SIZE
 - invoke_command.c, 314
- ta_num_props
 - user_ta_header.c, 407, 410
- TA_OpenSessionEntryPoint
 - crt.c, 360
 - Enclave.c, 396
 - tee_ta_api.h, 202
- ta_props
 - user_ta_header.c, 407, 410
- TA_REF_RUN_ALL
 - Enclave.h, 305
- TA_REF_UUID
 - Enclave.h, 305
- TA_STACK_SIZE
 - user_ta_header_defines.h, 413, 414
- TA_UUID

- user_ta_header_defines.h, 413, 414
- TA.VERSION
 - user_ta_header.c, 406, 409
 - user_ta_header_defines.h, 413, 414
- tahead_get_trace_level
 - user_ta_header.c, 407, 410
- tee-common.h
 - pr_deb, 79
- tee-internal-api-cryptlib.c
 - GCM.ST_AAD, 265
 - GCM.ST_ACTIVE, 265
 - GCM.ST_FINAL, 265
 - GCM.ST_INIT, 265
 - SIG.LENGTH, 265
 - TEE.AEDecryptFinal, 266
 - TEE.AEEncryptFinal, 266
 - TEE.AEInit, 267
 - TEE.AEUpdate, 268
 - TEE.AEUpdateAAD, 268
 - TEE.AllocateOperation, 269
 - TEE.AllocateTransientObject, 269
 - TEE.AsymmetricSignDigest, 270
 - TEE.AsymmetricVerifyDigest, 271
 - TEE.CipherDoFinal, 271
 - TEE.CipherInit, 272
 - TEE.CipherUpdate, 272
 - TEE.DigestDoFinal, 273
 - TEE.DigestUpdate, 273
 - TEE.FreeOperation, 274
 - TEE.FreeTransientObject, 274
 - TEE.GenerateKey, 275
 - TEE.InitRefAttribute, 275
 - TEE.InitValueAttribute, 276
 - TEE.SetOperationKey, 276
 - wolfSSL.Free, 277
 - wolfSSL.Malloc, 277
- tee-internal-api-machine.c
 - __attribute__, 211
- tee-internal-api.c
 - __attribute__, 224
 - flags2flags, 213, 224
 - FPERMS, 212, 223
 - GetRelTimeEnd, 214, 224
 - GetRelTimeStart, 214, 225
 - O_CREAT, 213, 223
 - O_EXCL, 213, 223
 - O_RDONLY, 213, 223
 - O_RDWR, 213, 223
 - O_TRUNC, 213, 223
 - O_WRONLY, 213, 224
 - OpenPersistentObject, 214, 225
 - set_object_key, 215, 226
 - TEE.CloseObject, 215, 226
 - TEE.CreatePersistentObject, 216, 227
 - TEE.Free, 217
 - TEE.GenerateRandom, 217, 227
 - TEE.GetObjectInfo1, 218, 228
 - TEE.GetREETime, 218, 228
 - TEE.GetSystemTime, 219, 229
 - TEE.Malloc, 219
 - TEE.OpenPersistentObject, 219, 229
 - TEE.ReadObjectData, 220, 229
 - TEE.Realloc, 220
 - TEE.WriteObjectData, 221, 230
- tee-ta-internal.h
 - __attribute__, 81
 - GetRelTimeEnd, 82
 - GetRelTimeStart, 83
 - TEE.AEDecryptFinal, 83
 - TEE.AEEncryptFinal, 84
 - TEE.AEInit, 85
 - TEE.AEUpdate, 85
 - TEE.AEUpdateAAD, 86
 - TEE.AllocateOperation, 87
 - TEE.AllocateTransientObject, 87
 - TEE.AsymmetricSignDigest, 88
 - TEE.AsymmetricVerifyDigest, 89
 - TEE.CipherInit, 89
 - TEE.CipherUpdate, 90
 - TEE.CloseObject, 91
 - TEE.CreatePersistentObject, 91
 - TEE.DigestDoFinal, 93
 - TEE.DigestUpdate, 93
 - TEE.FreeOperation, 94
 - TEE.FreeTransientObject, 94
 - TEE.GenerateKey, 95
 - TEE.GenerateRandom, 95
 - TEE.GetObjectInfo1, 96
 - TEE.GetREETime, 97
 - TEE.GetSystemTime, 98
 - TEE.InitRefAttribute, 98
 - TEE.InitValueAttribute, 99
 - TEE.OpenPersistentObject, 99
 - TEE.ReadObjectData, 100
 - TEE.SetOperationKey, 101
 - TEE.WriteObjectData, 102
- TEE.AEDecryptFinal
 - tee-internal-api-cryptlib.c, 266
 - tee-ta-internal.h, 83
 - tee_api.h, 107
- TEE.AEEncryptFinal
 - tee-internal-api-cryptlib.c, 266
 - tee-ta-internal.h, 84
 - tee_api.h, 108
- TEE.AEInit
 - tee-internal-api-cryptlib.c, 267
 - tee-ta-internal.h, 85
 - tee_api.h, 108
- TEE.AEUpdate
 - tee-internal-api-cryptlib.c, 268
 - tee-ta-internal.h, 85
 - tee_api.h, 109
- TEE.AEUpdateAAD
 - tee-internal-api-cryptlib.c, 268
 - tee-ta-internal.h, 86
 - tee_api.h, 110

- TEE_ALG_AES_CBC_MAC_NOPAD
tee_api_defines.h, 145
- TEE_ALG_AES_CBC_MAC_PKCS5
tee_api_defines.h, 145
- TEE_ALG_AES_CBC_NOPAD
tee_api_defines.h, 145
- TEE_ALG_AES_CCM
tee_api_defines.h, 146
- TEE_ALG_AES_CMAC
tee_api_defines.h, 146
- TEE_ALG_AES_CTR
tee_api_defines.h, 146
- TEE_ALG_AES_CTS
tee_api_defines.h, 146
- TEE_ALG_AES_ECB_NOPAD
tee_api_defines.h, 146
- TEE_ALG_AES_GCM
tee_api_defines.h, 146
- TEE_ALG_AES_XTS
tee_api_defines.h, 146
- TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY
tee_api_defines_extensions.h, 178
- TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY
tee_api_defines_extensions.h, 178
- TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY
tee_api_defines_extensions.h, 178
- TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY
tee_api_defines_extensions.h, 178
- TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY
tee_api_defines_extensions.h, 178
- TEE_ALG_DES3_CBC_MAC_NOPAD
tee_api_defines.h, 146
- TEE_ALG_DES3_CBC_MAC_PKCS5
tee_api_defines.h, 146
- TEE_ALG_DES3_CBC_NOPAD
tee_api_defines.h, 146
- TEE_ALG_DES3_ECB_NOPAD
tee_api_defines.h, 146
- TEE_ALG_DES_CBC_MAC_NOPAD
tee_api_defines.h, 147
- TEE_ALG_DES_CBC_MAC_PKCS5
tee_api_defines.h, 147
- TEE_ALG_DES_CBC_NOPAD
tee_api_defines.h, 147
- TEE_ALG_DES_ECB_NOPAD
tee_api_defines.h, 147
- TEE_ALG_DH_DERIVE_SHARED_SECRET
tee_api_defines.h, 147
- TEE_ALG_DSA_SHA1
tee_api_defines.h, 147
- TEE_ALG_DSA_SHA224
tee_api_defines.h, 147
- TEE_ALG_DSA_SHA256
tee_api_defines.h, 147
- TEE_ALG_ECDH_P192
tee_api_defines.h, 147
- TEE_ALG_ECDH_P224
tee_api_defines.h, 147
- TEE_ALG_ECDH_P256
tee_api_defines.h, 147
- TEE_ALG_ECDH_P384
tee_api_defines.h, 148
- TEE_ALG_ECDH_P521
tee_api_defines.h, 148
- TEE_ALG_ECDSA_P192
tee_api_defines.h, 148
- TEE_ALG_ECDSA_P224
tee_api_defines.h, 148
- TEE_ALG_ECDSA_P256
tee_api_defines.h, 148
- TEE_ALG_ECDSA_P384
tee_api_defines.h, 148
- TEE_ALG_ECDSA_P521
tee_api_defines.h, 148
- TEE_ALG_HKDF_MD5_DERIVE_KEY
tee_api_defines_extensions.h, 178
- TEE_ALG_HKDF_SHA1_DERIVE_KEY
tee_api_defines_extensions.h, 178
- TEE_ALG_HKDF_SHA224_DERIVE_KEY
tee_api_defines_extensions.h, 178
- TEE_ALG_HKDF_SHA256_DERIVE_KEY
tee_api_defines_extensions.h, 178
- TEE_ALG_HKDF_SHA384_DERIVE_KEY
tee_api_defines_extensions.h, 178
- TEE_ALG_HKDF_SHA512_DERIVE_KEY
tee_api_defines_extensions.h, 179
- TEE_ALG_HMAC_MD5
tee_api_defines.h, 148
- TEE_ALG_HMAC_SHA1
tee_api_defines.h, 148
- TEE_ALG_HMAC_SHA224
tee_api_defines.h, 148
- TEE_ALG_HMAC_SHA256
tee_api_defines.h, 148
- TEE_ALG_HMAC_SHA384
tee_api_defines.h, 149
- TEE_ALG_HMAC_SHA512
tee_api_defines.h, 149
- TEE_ALG_MD5
tee_api_defines.h, 149
- TEE_ALG_MD5SHA1
tee_api_defines.h, 149
- TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY
tee_api_defines_extensions.h, 179
- TEE_ALG_RSA_NOPAD
tee_api_defines.h, 149
- TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1
tee_api_defines.h, 149
- TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224
tee_api_defines.h, 149
- TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256
tee_api_defines.h, 149
- TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384
tee_api_defines.h, 149
- TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512
tee_api_defines.h, 149

- TEE_ALG_RSAES_PKCS1_V1_5
 - tee_api_defines.h, [149](#)
- TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1
 - tee_api_defines.h, [150](#)
- TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224
 - tee_api_defines.h, [150](#)
- TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256
 - tee_api_defines.h, [150](#)
- TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384
 - tee_api_defines.h, [150](#)
- TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512
 - tee_api_defines.h, [150](#)
- TEE_ALG_RSASSA_PKCS1_V1_5_MD5
 - tee_api_defines.h, [150](#)
- TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1
 - tee_api_defines.h, [150](#)
- TEE_ALG_RSASSA_PKCS1_V1_5_SHA1
 - tee_api_defines.h, [150](#)
- TEE_ALG_RSASSA_PKCS1_V1_5_SHA224
 - tee_api_defines.h, [150](#)
- TEE_ALG_RSASSA_PKCS1_V1_5_SHA256
 - tee_api_defines.h, [150](#)
- TEE_ALG_RSASSA_PKCS1_V1_5_SHA384
 - tee_api_defines.h, [151](#)
- TEE_ALG_RSASSA_PKCS1_V1_5_SHA512
 - tee_api_defines.h, [151](#)
- TEE_ALG_SHA1
 - tee_api_defines.h, [151](#)
- TEE_ALG_SHA224
 - tee_api_defines.h, [151](#)
- TEE_ALG_SHA256
 - tee_api_defines.h, [151](#)
- TEE_ALG_SHA384
 - tee_api_defines.h, [151](#)
- TEE_ALG_SHA512
 - tee_api_defines.h, [151](#)
- TEE_AllocateOperation
 - tee-internal-api-cryptlib.c, [269](#)
 - tee-ta-internal.h, [87](#)
 - tee_api.h, [110](#)
- TEE_AllocatePersistentObjectEnumerator
 - tee_api.h, [111](#)
- TEE_AllocatePropertyEnumerator
 - tee_api.h, [111](#)
- TEE_AllocateTransientObject
 - tee-internal-api-cryptlib.c, [269](#)
 - tee-ta-internal.h, [87](#)
 - tee_api.h, [111](#)
- tee_api.h
 - TEE_AEDecryptFinal, [107](#)
 - TEE_AEEncryptFinal, [108](#)
 - TEE_AEInit, [108](#)
 - TEE_AEUpdate, [109](#)
 - TEE_AEUpdateAAD, [110](#)
 - TEE_AllocateOperation, [110](#)
 - TEE_AllocatePersistentObjectEnumerator, [111](#)
 - TEE_AllocatePropertyEnumerator, [111](#)
 - TEE_AllocateTransientObject, [111](#)
 - TEE_AsymmetricDecrypt, [111](#)
 - TEE_AsymmetricEncrypt, [112](#)
 - TEE_AsymmetricSignDigest, [112](#)
 - TEE_AsymmetricVerifyDigest, [112](#)
 - TEE_BigIntAdd, [113](#)
 - TEE_BigIntAddMod, [113](#)
 - TEE_BigIntCmp, [113](#)
 - TEE_BigIntCmpS32, [113](#)
 - TEE_BigIntComputeExtendedGcd, [114](#)
 - TEE_BigIntComputeFMM, [114](#)
 - TEE_BigIntConvertFromFMM, [114](#)
 - TEE_BigIntConvertFromOctetString, [114](#)
 - TEE_BigIntConvertFromS32, [114](#)
 - TEE_BigIntConvertToFMM, [114](#)
 - TEE_BigIntConvertToOctetString, [114](#)
 - TEE_BigIntConvertToS32, [115](#)
 - TEE_BigIntDiv, [115](#)
 - TEE_BigIntFMMContextSizeInU32, [115](#)
 - TEE_BigIntFMMConvertToBigInt, [115](#)
 - TEE_BigIntFMMSizeInU32, [115](#)
 - TEE_BigIntGetBit, [115](#)
 - TEE_BigIntGetBitCount, [115](#)
 - TEE_BigIntInit, [115](#)
 - TEE_BigIntInitFMM, [116](#)
 - TEE_BigIntInitFMMContext, [116](#)
 - TEE_BigIntInvMod, [116](#)
 - TEE_BigIntIsProbablePrime, [116](#)
 - TEE_BigIntMod, [116](#)
 - TEE_BigIntMul, [116](#)
 - TEE_BigIntMulMod, [116](#)
 - TEE_BigIntNeg, [116](#)
 - TEE_BigIntRelativePrime, [117](#)
 - TEE_BigIntShiftRight, [117](#)
 - TEE_BigIntSquare, [117](#)
 - TEE_BigIntSquareMod, [117](#)
 - TEE_BigIntSub, [117](#)
 - TEE_BigIntSubMod, [117](#)
 - TEE_CheckMemoryAccessRights, [117](#)
 - TEE_CipherDoFinal, [118](#)
 - TEE_CipherInit, [118](#)
 - TEE_CipherUpdate, [119](#)
 - TEE_CloseAndDeletePersistentObject, [119](#)
 - TEE_CloseAndDeletePersistentObject1, [119](#)
 - TEE_CloseObject, [119](#)
 - TEE_CloseTASession, [120](#)
 - TEE_CopyObjectAttributes, [120](#)
 - TEE_CopyObjectAttributes1, [120](#)
 - TEE_CopyOperation, [120](#)
 - TEE_CreatePersistentObject, [121](#)
 - TEE_DeriveKey, [122](#)
 - TEE_DigestDoFinal, [122](#)
 - TEE_DigestUpdate, [122](#)
 - TEE_Free, [123](#)
 - TEE_FreeOperation, [123](#)
 - TEE_FreePersistentObjectEnumerator, [124](#)
 - TEE_FreePropertyEnumerator, [124](#)
 - TEE_FreeTransientObject, [124](#)
 - TEE_GenerateKey, [124](#)

- TEE_GenerateRandom, 125
- TEE_GetCancellationFlag, 126
- TEE_GetInstanceData, 126
- TEE_GetNextPersistentObject, 126
- TEE_GetNextProperty, 126
- TEE_GetObjectBufferAttribute, 126
- TEE_GetObjectInfo, 126
- TEE_GetObjectInfo1, 127
- TEE_GetObjectValueAttribute, 127
- TEE_GetOperationInfo, 127
- TEE_GetOperationInfoMultiple, 128
- TEE_GetPropertyAsBinaryBlock, 128
- TEE_GetPropertyAsBool, 128
- TEE_GetPropertyAsIdentity, 128
- TEE_GetPropertyAsString, 128
- TEE_GetPropertyAsU32, 128
- TEE_GetPropertyAsUUID, 128
- TEE_GetPropertyName, 129
- TEE_GetREETime, 129
- TEE_GetSystemTime, 129
- TEE_GetTAPersistentTime, 130
- TEE_InitRefAttribute, 130
- TEE_InitValueAttribute, 130
- TEE_InvokeTACommand, 131
- TEE_IsAlgorithmSupported, 131
- TEE_MACCompareFinal, 131
- TEE_MACComputeFinal, 131
- TEE_MACInit, 131
- TEE_MACUpdate, 132
- TEE_Malloc, 132
- TEE_MaskCancellation, 132
- TEE_MemCompare, 132
- TEE_MemFill, 132
- TEE_MemMove, 133
- TEE_OpenPersistentObject, 133
- TEE_OpenTASession, 134
- TEE_Panic, 134
- TEE_PopulateTransientObject, 134
- TEE_ReadObjectData, 134
- TEE_Realloc, 135
- TEE_RenamePersistentObject, 135
- TEE_ResetOperation, 136
- TEE_ResetPersistentObjectEnumerator, 136
- TEE_ResetPropertyEnumerator, 136
- TEE_ResetTransientObject, 136
- TEE_RestrictObjectUsage, 136
- TEE_RestrictObjectUsage1, 136
- TEE_SeekObjectData, 136
- TEE_SetInstanceData, 136
- TEE_SetOperationKey, 136
- TEE_SetOperationKey2, 137
- TEE_SetTAPersistentTime, 137
- TEE_StartPersistentObjectEnumerator, 137
- TEE_StartPropertyEnumerator, 137
- TEE_TruncateObjectData, 137
- TEE_UnmaskCancellation, 138
- TEE_Wait, 138
- TEE_WriteObjectData, 138
- tee_api_defines.h
 - TEE_ALG_AES_CBC_MAC_NOPAD, 145
 - TEE_ALG_AES_CBC_MAC_PKCS5, 145
 - TEE_ALG_AES_CBC_NOPAD, 145
 - TEE_ALG_AES_CCM, 146
 - TEE_ALG_AES_CMAC, 146
 - TEE_ALG_AES_CTR, 146
 - TEE_ALG_AES_CTS, 146
 - TEE_ALG_AES_ECB_NOPAD, 146
 - TEE_ALG_AES_GCM, 146
 - TEE_ALG_AES_XTS, 146
 - TEE_ALG_DES3_CBC_MAC_NOPAD, 146
 - TEE_ALG_DES3_CBC_MAC_PKCS5, 146
 - TEE_ALG_DES3_CBC_NOPAD, 146
 - TEE_ALG_DES3_ECB_NOPAD, 146
 - TEE_ALG_DES_CBC_MAC_NOPAD, 147
 - TEE_ALG_DES_CBC_MAC_PKCS5, 147
 - TEE_ALG_DES_CBC_NOPAD, 147
 - TEE_ALG_DES_ECB_NOPAD, 147
 - TEE_ALG_DH_DERIVE_SHARED_SECRET, 147
 - TEE_ALG_DSA_SHA1, 147
 - TEE_ALG_DSA_SHA224, 147
 - TEE_ALG_DSA_SHA256, 147
 - TEE_ALG_ECDH_P192, 147
 - TEE_ALG_ECDH_P224, 147
 - TEE_ALG_ECDH_P256, 147
 - TEE_ALG_ECDH_P384, 148
 - TEE_ALG_ECDH_P521, 148
 - TEE_ALG_ECDSA_P192, 148
 - TEE_ALG_ECDSA_P224, 148
 - TEE_ALG_ECDSA_P256, 148
 - TEE_ALG_ECDSA_P384, 148
 - TEE_ALG_ECDSA_P521, 148
 - TEE_ALG_HMAC_MD5, 148
 - TEE_ALG_HMAC_SHA1, 148
 - TEE_ALG_HMAC_SHA224, 148
 - TEE_ALG_HMAC_SHA256, 148
 - TEE_ALG_HMAC_SHA384, 149
 - TEE_ALG_HMAC_SHA512, 149
 - TEE_ALG_MD5, 149
 - TEE_ALG_MD5SHA1, 149
 - TEE_ALG_RSA_NOPAD, 149
 - TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1, 149
 - TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224, 149
 - TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256, 149
 - TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384, 149
 - TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512, 149
 - TEE_ALG_RSAES_PKCS1_V1_5, 149
 - TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1, 150
 - TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224, 150

- TEE.ALG_RSASSA_PKCS1_PSS_MGF1_SHA256, 150
- TEE.ALG_RSASSA_PKCS1_PSS_MGF1_SHA384, 150
- TEE.ALG_RSASSA_PKCS1_PSS_MGF1_SHA512, 150
- TEE.ALG_RSASSA_PKCS1_V1_5_MD5, 150
- TEE.ALG_RSASSA_PKCS1_V1_5_MD5SHA1, 150
- TEE.ALG_RSASSA_PKCS1_V1_5_SHA1, 150
- TEE.ALG_RSASSA_PKCS1_V1_5_SHA224, 150
- TEE.ALG_RSASSA_PKCS1_V1_5_SHA256, 150
- TEE.ALG_RSASSA_PKCS1_V1_5_SHA384, 151
- TEE.ALG_RSASSA_PKCS1_V1_5_SHA512, 151
- TEE.ALG_SHA1, 151
- TEE.ALG_SHA224, 151
- TEE.ALG_SHA256, 151
- TEE.ALG_SHA384, 151
- TEE.ALG_SHA512, 151
- TEE.ATTR_BIT_PROTECTED, 151
- TEE.ATTR_BIT_VALUE, 151
- TEE.ATTR_DH_BASE, 151
- TEE.ATTR_DH_PRIME, 151
- TEE.ATTR_DH_PRIVATE_VALUE, 152
- TEE.ATTR_DH_PUBLIC_VALUE, 152
- TEE.ATTR_DH_SUBPRIME, 152
- TEE.ATTR_DH_X_BITS, 152
- TEE.ATTR_DSA_BASE, 152
- TEE.ATTR_DSA_PRIME, 152
- TEE.ATTR_DSA_PRIVATE_VALUE, 152
- TEE.ATTR_DSA_PUBLIC_VALUE, 152
- TEE.ATTR_DSA_SUBPRIME, 152
- TEE.ATTR_ECC_CURVE, 152
- TEE.ATTR_ECC_PRIVATE_VALUE, 152
- TEE.ATTR_ECC_PUBLIC_VALUE_X, 153
- TEE.ATTR_ECC_PUBLIC_VALUE_Y, 153
- TEE.ATTR_RSA_COEFFICIENT, 153
- TEE.ATTR_RSA_EXPONENT1, 153
- TEE.ATTR_RSA_EXPONENT2, 153
- TEE.ATTR_RSA_MODULUS, 153
- TEE.ATTR_RSA_OAEP_LABEL, 153
- TEE.ATTR_RSA_PRIME1, 153
- TEE.ATTR_RSA_PRIME2, 153
- TEE.ATTR_RSA_PRIVATE_EXPONENT, 153
- TEE.ATTR_RSA_PSS_SALT_LENGTH, 153
- TEE.ATTR_RSA_PUBLIC_EXPONENT, 154
- TEE.ATTR_SECRET_VALUE, 154
- TEE.BigIntSizeInU32, 154
- TEE.DATA_FLAG_ACCESS_READ, 154
- TEE.DATA_FLAG_ACCESS_WRITE, 154
- TEE.DATA_FLAG_ACCESS_WRITE_META, 154
- TEE.DATA_FLAG_OVERWRITE, 154
- TEE.DATA_FLAG_SHARE_READ, 154
- TEE.DATA_FLAG_SHARE_WRITE, 154
- TEE.DATA_MAX_POSITION, 154
- TEE.ECC_CURVE_NIST_P192, 154
- TEE.ECC_CURVE_NIST_P224, 155
- TEE.ECC_CURVE_NIST_P256, 155
- TEE.ECC_CURVE_NIST_P384, 155
- TEE.ECC_CURVE_NIST_P521, 155
- TEE.ERROR_ACCESS_CONFLICT, 155
- TEE.ERROR_ACCESS_DENIED, 155
- TEE.ERROR_BAD_FORMAT, 155
- TEE.ERROR_BAD_PARAMETERS, 155
- TEE.ERROR_BAD_STATE, 155
- TEE.ERROR_BUSY, 155
- TEE.ERROR_CANCEL, 155
- TEE.ERROR_COMMUNICATION, 156
- TEE.ERROR_CORRUPT_OBJECT, 156
- TEE.ERROR_CORRUPT_OBJECT_2, 156
- TEE.ERROR_EXCESS_DATA, 156
- TEE.ERROR_EXTERNAL_CANCEL, 156
- TEE.ERROR_GENERIC, 156
- TEE.ERROR_ITEM_NOT_FOUND, 156
- TEE.ERROR_MAC_INVALID, 156
- TEE.ERROR_NO_DATA, 156
- TEE.ERROR_NOT_IMPLEMENTED, 156
- TEE.ERROR_NOT_SUPPORTED, 156
- TEE.ERROR_OUT_OF_MEMORY, 157
- TEE.ERROR_OVERFLOW, 157
- TEE.ERROR_SECURITY, 157
- TEE.ERROR_SHORT_BUFFER, 157
- TEE.ERROR_SIGNATURE_INVALID, 157
- TEE.ERROR_STORAGE_NO_SPACE, 157
- TEE.ERROR_STORAGE_NOT_AVAILABLE, 157
- TEE.ERROR_STORAGE_NOT_AVAILABLE_2, 157
- TEE.ERROR_TARGET_DEAD, 157
- TEE.ERROR_TIME_NEEDS_RESET, 157
- TEE.ERROR_TIME_NOT_SET, 157
- TEE.HANDLE_FLAG_EXPECT_TWO_KEYS, 158
- TEE.HANDLE_FLAG_INITIALIZED, 158
- TEE.HANDLE_FLAG_KEY_SET, 158
- TEE.HANDLE_FLAG_PERSISTENT, 158
- TEE.HANDLE_NULL, 158
- TEE.INT_CORE_API_SPEC_VERSION, 158
- TEE.LOGIN_APPLICATION, 158
- TEE.LOGIN_APPLICATION_GROUP, 158
- TEE.LOGIN_APPLICATION_USER, 158
- TEE.LOGIN_GROUP, 158
- TEE.LOGIN_PUBLIC, 158
- TEE.LOGIN_TRUSTED_APP, 159
- TEE.LOGIN_USER, 159
- TEE.MALLOC_FILL_ZERO, 159
- TEE.MEMORY_ACCESS_ANY_OWNER, 159
- TEE.MEMORY_ACCESS_READ, 159
- TEE.MEMORY_ACCESS_WRITE, 159
- TEE.NUM_PARAMS, 159
- TEE.OBJECT_ID_MAX_LEN, 159
- TEE.OPERATION_AE, 159
- TEE.OPERATION_ASYMMETRIC_CIPHER, 159
- TEE.OPERATION_ASYMMETRIC_SIGNATURE, 159
- TEE.OPERATION_CIPHER, 160
- TEE.OPERATION_DIGEST, 160
- TEE.OPERATION_KEY_DERIVATION, 160
- TEE.OPERATION_MAC, 160
- TEE.OPERATION_STATE_ACTIVE, 160

- TEE_OPERATION_STATE_INITIAL, 160
- TEE_ORIGIN_API, 160
- TEE_ORIGIN_COMMS, 160
- TEE_ORIGIN_TEE, 160
- TEE_ORIGIN_TRUSTED_APP, 160
- TEE_PANIC.ID_TA_CLOSESESSIONENTRYPOINT, 160
- TEE_PANIC.ID_TA_CREATEENTRYPOINT, 161
- TEE_PANIC.ID_TA_DESTROYENTRYPOINT, 161
- TEE_PANIC.ID_TA_INVOKECOMMANDENTRYPOINT, 161
- TEE_PANIC.ID_TA_OPENSESSIONENTRYPOINT, 161
- TEE_PANIC.ID_TEE_AEDECRIPTFINAL, 161
- TEE_PANIC.ID_TEE_AEENCRYPTFINAL, 161
- TEE_PANIC.ID_TEE_AEINIT, 161
- TEE_PANIC.ID_TEE_AEUPDATE, 161
- TEE_PANIC.ID_TEE_AEUPDATEAAD, 161
- TEE_PANIC.ID_TEE_ALLOCATEOPERATION, 161
- TEE_PANIC.ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR, 161
- TEE_PANIC.ID_TEE_ALLOCATEPROPERTYENUMERATOR, 162
- TEE_PANIC.ID_TEE_ALLOCATETRANSIENTOBJECT, 162
- TEE_PANIC.ID_TEE_ASYMMETRICDECRYPT, 162
- TEE_PANIC.ID_TEE_ASYMMETRICENCRYPT, 162
- TEE_PANIC.ID_TEE_ASYMMETRICSIGNDIGEST, 162
- TEE_PANIC.ID_TEE_ASYMMETRICVERIFYDIGEST, 162
- TEE_PANIC.ID_TEE_BIGINTADD, 162
- TEE_PANIC.ID_TEE_BIGINTADDMOD, 162
- TEE_PANIC.ID_TEE_BIGINTCMP, 162
- TEE_PANIC.ID_TEE_BIGINTCMPS32, 162
- TEE_PANIC.ID_TEE_BIGINTCOMPUTEEXTENDEDGCD, 162
- TEE_PANIC.ID_TEE_BIGINTCOMPUTEFM, 163
- TEE_PANIC.ID_TEE_BIGINTCONVERTFROMFM, 163
- TEE_PANIC.ID_TEE_BIGINTCONVERTFROMOCTETSTRING, 163
- TEE_PANIC.ID_TEE_BIGINTCONVERTFROMS32, 163
- TEE_PANIC.ID_TEE_BIGINTCONVERTTOFM, 163
- TEE_PANIC.ID_TEE_BIGINTCONVERTTOOCTETSTRING, 163
- TEE_PANIC.ID_TEE_BIGINTCONVERTTOS32, 163
- TEE_PANIC.ID_TEE_BIGINTDIV, 163
- TEE_PANIC.ID_TEE_BIGINTFMCONTEXTSIZEINU32, 163
- TEE_PANIC.ID_TEE_BIGINTFMMSIZEINU32, 163
- TEE_PANIC.ID_TEE_BIGINTGETBIT, 164
- TEE_PANIC.ID_TEE_BIGINTGETBITCOUNT, 164
- TEE_PANIC.ID_TEE_BIGINTINIT, 164
- TEE_PANIC.ID_TEE_BIGINTINITFM, 164
- TEE_PANIC.ID_TEE_BIGINTINITFMCONTEXT, 164
- TEE_PANIC.ID_TEE_BIGINTINVMOD, 164
- TEE_PANIC.ID_TEE_BIGINTISPROBABLEPRIME, 164
- TEE_PANIC.ID_TEE_BIGINTMOD, 164
- TEE_PANIC.ID_TEE_BIGINTMUL, 164
- TEE_PANIC.ID_TEE_BIGINTMULMOD, 164
- TEE_PANIC.ID_TEE_BIGINTNEG, 164
- TEE_PANIC.ID_TEE_BIGINTRELATIVEPRIME, 165
- TEE_PANIC.ID_TEE_BIGINTSHIFTRIGHT, 165
- TEE_PANIC.ID_TEE_BIGINTSQUARE, 165
- TEE_PANIC.ID_TEE_BIGINTSQUAREMOD, 165
- TEE_PANIC.ID_TEE_BIGINTSUB, 165
- TEE_PANIC.ID_TEE_BIGINTSUBMOD, 165
- TEE_PANIC.ID_TEE_CHECKMEMORYACCESSRIGHTS, 165
- TEE_PANIC.ID_TEE_CIPHERDOFINAL, 165
- TEE_PANIC.ID_TEE_CIPHERINIT, 165
- TEE_PANIC.ID_TEE_CIPHERUPDATE, 165
- TEE_PANIC.ID_TEE_CLOSEANDDELETERPERSISTENTOBJECT, 165
- TEE_PANIC.ID_TEE_CLOSEANDDELETERPERSISTENTOBJECT1, 166
- TEE_PANIC.ID_TEE_CLOSEOBJECT, 166
- TEE_PANIC.ID_TEE_CLOSETASESSION, 166
- TEE_PANIC.ID_TEE_COPYOBJECTATTRIBUTES, 166
- TEE_PANIC.ID_TEE_COPYOBJECTATTRIBUTES1, 166
- TEE_PANIC.ID_TEE_COPYOPERATION, 166
- TEE_PANIC.ID_TEE_CREATEPERSISTENTOBJECT, 166
- TEE_PANIC.ID_TEE_DERIVEKEY, 166
- TEE_PANIC.ID_TEE_DIGESTDOFINAL, 166
- TEE_PANIC.ID_TEE_DIGESTUPDATE, 166
- TEE_PANIC.ID_TEE_FREE, 166
- TEE_PANIC.ID_TEE_FREEOPERATION, 167
- TEE_PANIC.ID_TEE_FREEPERSISTENTOBJECTENUMERATOR, 167
- TEE_PANIC.ID_TEE_FREEPROPERTYENUMERATOR, 167
- TEE_PANIC.ID_TEE_FREETRANSIENTOBJECT, 167
- TEE_PANIC.ID_TEE_GENERATEKEY, 167
- TEE_PANIC.ID_TEE_GENERATERANDOM, 167
- TEE_PANIC.ID_TEE_GETCANCELLATIONFLAG, 167
- TEE_PANIC.ID_TEE_GETINSTANCEDATA, 167
- TEE_PANIC.ID_TEE_GETNEXTPERSISTENTOBJECT, 167
- TEE_PANIC.ID_TEE_GETNEXTPROPERTY, 167
- TEE_PANIC.ID_TEE_GETOBJECTBUFFERATTRIBUTE, 167
- TEE_PANIC.ID_TEE_GETOBJECTINFO, 168

- TEE.PANIC.ID.TEE.GETOBJECTINFO1, 168
- TEE.PANIC.ID.TEE.GETOBJECTVALUEATTRIBUTE, 168
- TEE.PANIC.ID.TEE.GETOPERATIONINFO, 168
- TEE.PANIC.ID.TEE.GETOPERATIONINFOMULTIPLE, 168
- TEE.PANIC.ID.TEE.GETPROPERTYASBINARYBLOCK, 168
- TEE.PANIC.ID.TEE.GETPROPERTYASBOOL, 168
- TEE.PANIC.ID.TEE.GETPROPERTYASIDENTITY, 168
- TEE.PANIC.ID.TEE.GETPROPERTYASSTRING, 168
- TEE.PANIC.ID.TEE.GETPROPERTYASU32, 168
- TEE.PANIC.ID.TEE.GETPROPERTYASUUID, 168
- TEE.PANIC.ID.TEE.GETPROPERTYNAME, 169
- TEE.PANIC.ID.TEE.GETREETIME, 169
- TEE.PANIC.ID.TEE.GETSYSTEMTIME, 169
- TEE.PANIC.ID.TEE.GETTAPERSISTENTTIME, 169
- TEE.PANIC.ID.TEE.INITREFATTRIBUTE, 169
- TEE.PANIC.ID.TEE.INITVALUEATTRIBUTE, 169
- TEE.PANIC.ID.TEE.INVOKETACOMMAND, 169
- TEE.PANIC.ID.TEE.MACCOMPAREFINAL, 169
- TEE.PANIC.ID.TEE.MACCOMPUTEFINAL, 169
- TEE.PANIC.ID.TEE.MACINIT, 169
- TEE.PANIC.ID.TEE.MACUPDATE, 169
- TEE.PANIC.ID.TEE.MALLOC, 170
- TEE.PANIC.ID.TEE.MASKCANCELLATION, 170
- TEE.PANIC.ID.TEE.MEMCOMPARE, 170
- TEE.PANIC.ID.TEE.MEMFILL, 170
- TEE.PANIC.ID.TEE.MEMMOVE, 170
- TEE.PANIC.ID.TEE.OPENPERSISTENTOBJECT, 170
- TEE.PANIC.ID.TEE.OPENTASESSION, 170
- TEE.PANIC.ID.TEE.PANIC, 170
- TEE.PANIC.ID.TEE.POPULATETRANSIENTOBJECT, 170
- TEE.PANIC.ID.TEE.READOBJECTDATA, 170
- TEE.PANIC.ID.TEE.REALLOC, 170
- TEE.PANIC.ID.TEE.RENAMEPERSISTENTOBJECT, 171
- TEE.PANIC.ID.TEE.RESETOPERATION, 171
- TEE.PANIC.ID.TEE.RESETPERSISTENTOBJECTENUMERATOR, 171
- TEE.PANIC.ID.TEE.RESETPROPERTYENUMERATOR, 171
- TEE.PANIC.ID.TEE.RESETTRANSIENTOBJECT, 171
- TEE.PANIC.ID.TEE.RESTRICTOBJECTUSAGE, 171
- TEE.PANIC.ID.TEE.RESTRICTOBJECTUSAGE1, 171
- TEE.PANIC.ID.TEE.SEEKOBJECTDATA, 171
- TEE.PANIC.ID.TEE.SETINSTANCEDATA, 171
- TEE.PANIC.ID.TEE.SETOPERATIONKEY, 171
- TEE.PANIC.ID.TEE.SETOPERATIONKEY2, 171
- TEE.PANIC.ID.TEE.SETTAPERSISTENTTIME, 172
- TEE.PANIC.ID.TEE.STARTPERSISTENTOBJECTENUMERATOR, 172
- TEE.PANIC.ID.TEE.STARTPROPERTYENUMERATOR, 172
- TEE.PANIC.ID.TEE.TRUNCATEOBJECTDATA, 172
- TEE.PANIC.ID.TEE.UNMASKCANCELLATION, 172
- TEE.PANIC.ID.TEE.WAIT, 172
- TEE.PANIC.ID.TEE.WRITEOBJECTDATA, 172
- TEE.PARAM.TYPE.GET, 172
- TEE.PARAM.TYPE.MEMREF_INOUT, 172
- TEE.PARAM.TYPE.MEMREF_INPUT, 172
- TEE.PARAM.TYPE.MEMREF_OUTPUT, 173
- TEE.PARAM.TYPE.NONE, 173
- TEE.PARAM.TYPE.SET, 173
- TEE.PARAM.TYPE.VALUE_INOUT, 173
- TEE.PARAM.TYPE.VALUE_INPUT, 173
- TEE.PARAM.TYPE.VALUE_OUTPUT, 173
- TEE.PARAM.TYPES, 173
- TEE.PROPSET.CURRENT_CLIENT, 173
- TEE.PROPSET.CURRENT_TA, 173
- TEE.PROPSET.TEE_IMPLEMENTATION, 173
- TEE.STORAGE.PRIVATE, 174
- TEE.SUCCESS, 174
- TEE.TIMEOUT.INFINITE, 174
- TEE.TYPE.AES, 174
- TEE.TYPE.CORRUPTED_OBJECT, 174
- TEE.TYPE.DATA, 174
- TEE.TYPE.DES, 174
- TEE.TYPE.DES3, 174
- TEE.TYPE.DH.KEYPAIR, 174
- TEE.TYPE.DSA.KEYPAIR, 174
- TEE.TYPE.DSA.PUBLIC_KEY, 174
- TEE.TYPE.ECDH.KEYPAIR, 175
- TEE.TYPE.ECDH.PUBLIC_KEY, 175
- TEE.TYPE.ECDSA.KEYPAIR, 175
- TEE.TYPE.ECDSA.PUBLIC_KEY, 175
- TEE.TYPE.GENERIC.SECRET, 175
- TEE.TYPE.HMAC.MD5, 175
- TEE.TYPE.HMAC.SHA1, 175
- TEE.TYPE.HMAC.SHA224, 175
- TEE.TYPE.HMAC.SHA256, 175
- TEE.TYPE.HMAC.SHA384, 175
- TEE.TYPE.HMAC.SHA512, 175
- TEE.TYPE.RSA.KEYPAIR, 176
- TEE.TYPE.RSA.PUBLIC_KEY, 176
- TEE.USAGE.DECRYPT, 176
- TEE.USAGE.DERIVE, 176
- TEE.USAGE.ENCRYPT, 176
- TEE.USAGE.EXTRACTABLE, 176
- TEE.USAGE.MAC, 176
- TEE.USAGE.SIGN, 176
- TEE.USAGE.VERIFY, 176
- tee.api.defines.extensions.h

- TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY, [178](#)
- TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY, [178](#)
- TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY, [178](#)
- TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY, [178](#)
- TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY, [178](#)
- TEE_ALG_HKDF_MD5_DERIVE_KEY, [178](#)
- TEE_ALG_HKDF_SHA1_DERIVE_KEY, [178](#)
- TEE_ALG_HKDF_SHA224_DERIVE_KEY, [178](#)
- TEE_ALG_HKDF_SHA256_DERIVE_KEY, [178](#)
- TEE_ALG_HKDF_SHA384_DERIVE_KEY, [178](#)
- TEE_ALG_HKDF_SHA512_DERIVE_KEY, [179](#)
- TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY, [179](#)
- TEE_ATTR_CONCAT_KDF_DKM_LENGTH, [179](#)
- TEE_ATTR_CONCAT_KDF_OTHER_INFO, [179](#)
- TEE_ATTR_CONCAT_KDF_Z, [179](#)
- TEE_ATTR_HKDF_IKM, [179](#)
- TEE_ATTR_HKDF_INFO, [179](#)
- TEE_ATTR_HKDF_OKM_LENGTH, [179](#)
- TEE_ATTR_HKDF_SALT, [179](#)
- TEE_ATTR_PBKDF2_DKM_LENGTH, [179](#)
- TEE_ATTR_PBKDF2_ITERATION_COUNT, [179](#)
- TEE_ATTR_PBKDF2_PASSWORD, [180](#)
- TEE_ATTR_PBKDF2_SALT, [180](#)
- TEE_MEMORY_ACCESS_NONSECURE, [180](#)
- TEE_MEMORY_ACCESS_SECURE, [180](#)
- TEE_STORAGE_PRIVATE_REE, [180](#)
- TEE_STORAGE_PRIVATE_RPMB, [180](#)
- TEE_STORAGE_PRIVATE_SQL_RESERVED, [180](#)
- TEE_TYPE_CONCAT_KDF_Z, [180](#)
- TEE_TYPE_HKDF_IKM, [180](#)
- TEE_TYPE_PBKDF2_PASSWORD, [180](#)
- tee_api.tee.types.h
 - AES256, [232](#), [234](#)
 - MBEDCRYPT, [232](#), [234](#)
 - SHA_LENGTH, [232](#), [234](#)
 - TEE_HANDLE_NULL, [234](#)
 - TEE_OBJECT_AAD_SIZE, [232](#), [234](#)
 - TEE_OBJECT_KEY_SIZE, [232](#), [234](#)
 - TEE_OBJECT_NONCE_SIZE, [232](#), [234](#)
 - TEE_OBJECT_SKEY_SIZE, [232](#), [234](#)
 - TEE_OBJECT_TAG_SIZE, [232](#), [235](#)
 - WOLFCRYPT, [232](#), [235](#)
- tee_api.types.h
 - _aligned, [183](#)
 - DMREQ_FINISH, [182](#)
 - DMREQ_WRITE, [182](#)
 - nfds_t, [183](#)
 - socklen_t, [183](#)
 - TEE_BigInt, [183](#)
 - TEE_BigIntFMM, [183](#)
 - TEE_DATA_SEEK_CUR, [185](#)
 - TEE_DATA_SEEK_END, [185](#)
 - TEE_DATA_SEEK_SET, [185](#)
 - TEE_ErrorOrigin, [184](#)
 - TEE_MEM_INPUT, [182](#)
 - TEE_MEM_OUTPUT, [183](#)
 - TEE_MEMREF_0_USED, [183](#)
 - TEE_MEMREF_1_USED, [183](#)
 - TEE_MEMREF_2_USED, [183](#)
 - TEE_MEMREF_3_USED, [183](#)
 - TEE_MODE_DECRYPT, [185](#)
 - TEE_MODE_DERIVE, [185](#)
 - TEE_MODE_DIGEST, [185](#)
 - TEE_MODE_ENCRYPT, [185](#)
 - TEE_MODE_MAC, [185](#)
 - TEE_MODE_SIGN, [185](#)
 - TEE_MODE_VERIFY, [185](#)
 - TEE_ObjectEnumHandle, [184](#)
 - TEE_ObjectHandle, [184](#)
 - TEE_ObjectType, [184](#)
 - TEE_OperationHandle, [184](#)
 - TEE_OperationMode, [185](#)
 - TEE_PropSetHandle, [184](#)
 - TEE_Result, [184](#)
 - TEE_SE_READER_NAME_MAX, [183](#)
 - TEE_SEChannelHandle, [184](#)
 - TEE_SEReaderHandle, [184](#)
 - TEESessionHandle, [184](#)
 - TEESessionHandle, [184](#)
 - TEE_Session, [185](#)
 - TEETASessionHandle, [185](#)
 - TEE_Whence, [185](#)
 - TEE_AsymmetricDecrypt
 - tee_api.h, [111](#)
 - TEE_AsymmetricEncrypt
 - tee_api.h, [112](#)
 - TEE_AsymmetricSignDigest
 - tee-internal-api-cryptlib.c, [270](#)
 - tee-ta-internal.h, [88](#)
 - tee_api.h, [112](#)
 - TEE_AsymmetricVerifyDigest
 - tee-internal-api-cryptlib.c, [271](#)
 - tee-ta-internal.h, [89](#)
 - tee_api.h, [112](#)
 - TEE_ATTR_BIT_PROTECTED
 - tee_api.defines.h, [151](#)
 - TEE_ATTR_BIT_VALUE
 - tee_api.defines.h, [151](#)
 - TEE_ATTR_CONCAT_KDF_DKM_LENGTH
 - tee_api.defines_extensions.h, [179](#)
 - TEE_ATTR_CONCAT_KDF_OTHER_INFO
 - tee_api.defines_extensions.h, [179](#)
 - TEE_ATTR_CONCAT_KDF_Z
 - tee_api.defines_extensions.h, [179](#)
 - TEE_ATTR_DH_BASE
 - tee_api.defines.h, [151](#)
 - TEE_ATTR_DH_PRIME
 - tee_api.defines.h, [151](#)
 - TEE_ATTR_DH_PRIVATE_VALUE
 - tee_api.defines.h, [152](#)

- TEE_ATTR_DH_PUBLIC.VALUE
 - tee_api_defines.h, 152
- TEE_ATTR_DH_SUBPRIME
 - tee_api_defines.h, 152
- TEE_ATTR_DH_X_BITS
 - tee_api_defines.h, 152
- TEE_ATTR_DSA_BASE
 - tee_api_defines.h, 152
- TEE_ATTR_DSA_PRIME
 - tee_api_defines.h, 152
- TEE_ATTR_DSA_PRIVATE.VALUE
 - tee_api_defines.h, 152
- TEE_ATTR_DSA_PUBLIC.VALUE
 - tee_api_defines.h, 152
- TEE_ATTR_DSA_SUBPRIME
 - tee_api_defines.h, 152
- TEE_ATTR_ECC_CURVE
 - tee_api_defines.h, 152
- TEE_ATTR_ECC_PRIVATE.VALUE
 - tee_api_defines.h, 152
- TEE_ATTR_ECC_PUBLIC.VALUE_X
 - tee_api_defines.h, 153
- TEE_ATTR_ECC_PUBLIC.VALUE_Y
 - tee_api_defines.h, 153
- TEE_ATTR_HKDF_IKM
 - tee_api_defines_extensions.h, 179
- TEE_ATTR_HKDF_INFO
 - tee_api_defines_extensions.h, 179
- TEE_ATTR_HKDF_OKM.LENGTH
 - tee_api_defines_extensions.h, 179
- TEE_ATTR_HKDF_SALT
 - tee_api_defines_extensions.h, 179
- TEE_ATTR_PBKDF2_DKM.LENGTH
 - tee_api_defines_extensions.h, 179
- TEE_ATTR_PBKDF2_ITERATION.COUNT
 - tee_api_defines_extensions.h, 179
- TEE_ATTR_PBKDF2_PASSWORD
 - tee_api_defines_extensions.h, 180
- TEE_ATTR_PBKDF2_SALT
 - tee_api_defines_extensions.h, 180
- TEE_ATTR_RSA_COEFFICIENT
 - tee_api_defines.h, 153
- TEE_ATTR_RSA_EXPONENT1
 - tee_api_defines.h, 153
- TEE_ATTR_RSA_EXPONENT2
 - tee_api_defines.h, 153
- TEE_ATTR_RSA_MODULUS
 - tee_api_defines.h, 153
- TEE_ATTR_RSA_OAEP_LABEL
 - tee_api_defines.h, 153
- TEE_ATTR_RSA_PRIME1
 - tee_api_defines.h, 153
- TEE_ATTR_RSA_PRIME2
 - tee_api_defines.h, 153
- TEE_ATTR_RSA_PRIVATE_EXPONENT
 - tee_api_defines.h, 153
- TEE_ATTR_RSA_PSS_SALT_LENGTH
 - tee_api_defines.h, 153
- TEE_ATTR_RSA_PUBLIC_EXPONENT
 - tee_api_defines.h, 154
- TEE_ATTR_SECRET.VALUE
 - tee_api_defines.h, 154
- TEE_Attribute, 50
 - a, 50
 - attributeID, 50
 - b, 50
 - buffer, 50
 - content, 51
 - length, 51
 - ref, 51
 - value, 51
- TEE_BigInt
 - tee_api_types.h, 183
- TEE_BigIntAdd
 - tee_api.h, 113
- TEE_BigIntAddMod
 - tee_api.h, 113
- TEE_BigIntCmp
 - tee_api.h, 113
- TEE_BigIntCmpS32
 - tee_api.h, 113
- TEE_BigIntComputeExtendedGcd
 - tee_api.h, 114
- TEE_BigIntComputeFMM
 - tee_api.h, 114
- TEE_BigIntConvertFromFMM
 - tee_api.h, 114
- TEE_BigIntConvertFromOctetString
 - tee_api.h, 114
- TEE_BigIntConvertFromS32
 - tee_api.h, 114
- TEE_BigIntConvertToFMM
 - tee_api.h, 114
- TEE_BigIntConvertToOctetString
 - tee_api.h, 114
- TEE_BigIntConvertToS32
 - tee_api.h, 115
- TEE_BigIntDiv
 - tee_api.h, 115
- TEE_BigIntFMM
 - tee_api_types.h, 183
- TEE_BigIntFMMContextSizeInU32
 - tee_api.h, 115
- TEE_BigIntFMMConvertToBigInt
 - tee_api.h, 115
- TEE_BigIntFMMSizeInU32
 - tee_api.h, 115
- TEE_BigIntGetBit
 - tee_api.h, 115
- TEE_BigIntGetBitCount
 - tee_api.h, 115
- TEE_BigIntInit
 - tee_api.h, 115
- TEE_BigIntInitFMM
 - tee_api.h, 116
- TEE_BigIntInitFMMContext

- tee_api.h, 116
- TEE.BigIntInvMod
 - tee_api.h, 116
- TEE.BigIntIsProbablePrime
 - tee_api.h, 116
- TEE.BigIntMod
 - tee_api.h, 116
- TEE.BigIntMul
 - tee_api.h, 116
- TEE.BigIntMulMod
 - tee_api.h, 116
- TEE.BigIntNeg
 - tee_api.h, 116
- TEE.BigIntRelativePrime
 - tee_api.h, 117
- TEE.BigIntShiftRight
 - tee_api.h, 117
- TEE.BigIntSizeInU32
 - tee_api_defines.h, 154
- TEE.BigIntSquare
 - tee_api.h, 117
- TEE.BigIntSquareMod
 - tee_api.h, 117
- TEE.BigIntSub
 - tee_api.h, 117
- TEE.BigIntSubMod
 - tee_api.h, 117
- TEE.CacheClean
 - tee_internal_api_extensions.h, 199
- TEE.CacheFlush
 - tee_internal_api_extensions.h, 199
- TEE.CacheInvalidate
 - tee_internal_api_extensions.h, 200
- TEE.CheckMemoryAccessRights
 - tee_api.h, 117
- TEE.CipherDoFinal
 - tee-internal-api-cryptlib.c, 271
 - tee_api.h, 118
- TEE.CipherInit
 - tee-internal-api-cryptlib.c, 272
 - tee-ta-internal.h, 89
 - tee_api.h, 118
- TEE.CipherUpdate
 - tee-internal-api-cryptlib.c, 272
 - tee-ta-internal.h, 90
 - tee_api.h, 119
- tee_client_api.h
 - TEEC.AllocateSharedMemory, 194
 - TEEC.CloseSession, 194
 - TEEC.CONFIG.PAYLOAD.REF_COUNT, 188
 - TEEC.CONFIG.SHAREDMEM.MAX.SIZE, 188
 - TEEC.ERROR.ACCESS.CONFLICT, 188
 - TEEC.ERROR.ACCESS.DENIED, 188
 - TEEC.ERROR.BAD.FORMAT, 188
 - TEEC.ERROR.BAD.PARAMETERS, 188
 - TEEC.ERROR.BAD.STATE, 188
 - TEEC.ERROR.BUSY, 188
 - TEEC.ERROR.CANCEL, 189
 - TEEC.ERROR.COMMUNICATION, 189
 - TEEC.ERROR.EXCESS.DATA, 189
 - TEEC.ERROR.EXTERNAL.CANCEL, 189
 - TEEC.ERROR.GENERIC, 189
 - TEEC.ERROR.ITEM.NOT.FOUND, 189
 - TEEC.ERROR.NO.DATA, 189
 - TEEC.ERROR.NOT.IMPLEMENTED, 189
 - TEEC.ERROR.NOT.SUPPORTED, 189
 - TEEC.ERROR.OUT.OF.MEMORY, 189
 - TEEC.ERROR.SECURITY, 189
 - TEEC.ERROR.SHORT.BUFFER, 190
 - TEEC.ERROR.TARGET.DEAD, 190
 - TEEC.FinalizeContext, 194
 - TEEC.InitializeContext, 195
 - TEEC.InvokeCommand, 195
 - TEEC.LOGIN.APPLICATION, 190
 - TEEC.LOGIN.GROUP, 190
 - TEEC.LOGIN.GROUP.APPLICATION, 190
 - TEEC.LOGIN.PUBLIC, 190
 - TEEC.LOGIN.USER, 190
 - TEEC.LOGIN.USER.APPLICATION, 190
 - TEEC.MEM.INPUT, 190
 - TEEC.MEM.OUTPUT, 191
 - TEEC.MEMREF.PARTIAL.INOUT, 191
 - TEEC.MEMREF.PARTIAL.INPUT, 191
 - TEEC.MEMREF.PARTIAL.OUTPUT, 191
 - TEEC.MEMREF.TEMP.INOUT, 191
 - TEEC.MEMREF.TEMP.INPUT, 191
 - TEEC.MEMREF.TEMP.OUTPUT, 191
 - TEEC.MEMREF.WHOLE, 191
 - TEEC.NONE, 191
 - TEEC.OpenSession, 196
 - TEEC.ORIGIN.API, 192
 - TEEC.ORIGIN.COMMS, 192
 - TEEC.ORIGIN.TEE, 192
 - TEEC.ORIGIN.TRUSTED.APP, 192
 - TEEC.PARAM.TYPE.GET, 192
 - TEEC.PARAM.TYPES, 193
 - TEEC.RegisterSharedMemory, 196
 - TEEC.ReleaseSharedMemory, 197
 - TEEC.RequestCancellation, 197
 - TEEC.Result, 194
 - TEEC.SUCCESS, 193
 - TEEC.VALUE.INOUT, 193
 - TEEC.VALUE.INPUT, 193
 - TEEC.VALUE.OUTPUT, 193
- TEE.CloseAndDeletePersistentObject
 - tee_api.h, 119
- TEE.CloseAndDeletePersistentObject1
 - tee_api.h, 119
- TEE.CloseObject
 - tee-internal-api.c, 215, 226
 - tee-ta-internal.h, 91
 - tee_api.h, 119
- TEE.CloseTASession
 - tee_api.h, 120
- tee_config.h
 - ..ImageBase, 337, 339, 340

- COMMAND, 338
- perf_buffer, 337, 339, 340
- tee_rdtscp, 337–339
- TEE_CopyObjectAttributes
 - tee_api.h, 120
- TEE_CopyObjectAttributes1
 - tee_api.h, 120
- TEE_CopyOperation
 - tee_api.h, 120
- TEE_CreatePersistentObject
 - tee-internal-api.c, 216, 227
 - tee-ta-internal.h, 91
 - tee_api.h, 121
- TEE_DATA_FLAG_ACCESS_READ
 - tee_api_defines.h, 154
- TEE_DATA_FLAG_ACCESS_WRITE
 - tee_api_defines.h, 154
- TEE_DATA_FLAG_ACCESS_WRITE_META
 - tee_api_defines.h, 154
- TEE_DATA_FLAG_OVERWRITE
 - tee_api_defines.h, 154
- TEE_DATA_FLAG_SHARE_READ
 - tee_api_defines.h, 154
- TEE_DATA_FLAG_SHARE_WRITE
 - tee_api_defines.h, 154
- TEE_DATA_MAX_POSITION
 - tee_api_defines.h, 154
- TEE_DATA_SEEK_CUR
 - tee_api_types.h, 185
- TEE_DATA_SEEK_END
 - tee_api_types.h, 185
- TEE_DATA_SEEK_SET
 - tee_api_types.h, 185
- tee_def.h
 - buf, 290–292
 - buf_flag, 290–292
 - tee_init, 290–292
 - test_printf, 290–292
- TEE_DeriveKey
 - tee_api.h, 122
- TEE_DigestDoFinal
 - tee-internal-api-cryptlib.c, 273
 - tee-ta-internal.h, 93
 - tee_api.h, 122
- TEE_DigestUpdate
 - tee-internal-api-cryptlib.c, 273
 - tee-ta-internal.h, 93
 - tee_api.h, 122
- TEE_ECC_CURVE_NIST_P192
 - tee_api_defines.h, 154
- TEE_ECC_CURVE_NIST_P224
 - tee_api_defines.h, 155
- TEE_ECC_CURVE_NIST_P256
 - tee_api_defines.h, 155
- TEE_ECC_CURVE_NIST_P384
 - tee_api_defines.h, 155
- TEE_ECC_CURVE_NIST_P521
 - tee_api_defines.h, 155
- TEE_ERROR_ACCESS_CONFLICT
 - tee_api_defines.h, 155
- TEE_ERROR_ACCESS_DENIED
 - tee_api_defines.h, 155
- TEE_ERROR_BAD_FORMAT
 - tee_api_defines.h, 155
- TEE_ERROR_BAD_PARAMETERS
 - tee_api_defines.h, 155
- TEE_ERROR_BAD_STATE
 - tee_api_defines.h, 155
- TEE_ERROR_BUSY
 - tee_api_defines.h, 155
- TEE_ERROR_CANCEL
 - tee_api_defines.h, 155
- TEE_ERROR_COMMUNICATION
 - tee_api_defines.h, 156
- TEE_ERROR_CORRUPT_OBJECT
 - tee_api_defines.h, 156
- TEE_ERROR_CORRUPT_OBJECT_2
 - tee_api_defines.h, 156
- TEE_ERROR_EXCESS_DATA
 - tee_api_defines.h, 156
- TEE_ERROR_EXTERNAL_CANCEL
 - tee_api_defines.h, 156
- TEE_ERROR_GENERIC
 - tee_api_defines.h, 156
- TEE_ERROR_ITEM_NOT_FOUND
 - tee_api_defines.h, 156
- TEE_ERROR_MAC_INVALID
 - tee_api_defines.h, 156
- TEE_ERROR_NO_DATA
 - tee_api_defines.h, 156
- TEE_ERROR_NOT_IMPLEMENTED
 - tee_api_defines.h, 156
- TEE_ERROR_NOT_SUPPORTED
 - tee_api_defines.h, 156
- TEE_ERROR_OUT_OF_MEMORY
 - tee_api_defines.h, 157
- TEE_ERROR_OVERFLOW
 - tee_api_defines.h, 157
- TEE_ERROR_SECURITY
 - tee_api_defines.h, 157
- TEE_ERROR_SHORT_BUFFER
 - tee_api_defines.h, 157
- TEE_ERROR_SIGNATURE_INVALID
 - tee_api_defines.h, 157
- TEE_ERROR_STORAGE_NO_SPACE
 - tee_api_defines.h, 157
- TEE_ERROR_STORAGE_NOT_AVAILABLE
 - tee_api_defines.h, 157
- TEE_ERROR_STORAGE_NOT_AVAILABLE_2
 - tee_api_defines.h, 157
- TEE_ERROR_TARGET_DEAD
 - tee_api_defines.h, 157
- TEE_ERROR_TIME_NEEDS_RESET
 - tee_api_defines.h, 157
- TEE_ERROR_TIME_NOT_SET
 - tee_api_defines.h, 157

- TEE.ErrorOrigin
 - tee_api.types.h, 184
- TEE.Free
 - tee-internal-api.c, 217
 - tee_api.h, 123
- TEE.FreeOperation
 - tee-internal-api-cryptlib.c, 274
 - tee-ta-internal.h, 94
 - tee_api.h, 123
- TEE.FreePersistentObjectEnumerator
 - tee_api.h, 124
- TEE.FreePropertyEnumerator
 - tee_api.h, 124
- TEE.FreeTransientObject
 - tee-internal-api-cryptlib.c, 274
 - tee-ta-internal.h, 94
 - tee_api.h, 124
- TEE.GenerateKey
 - tee-internal-api-cryptlib.c, 275
 - tee-ta-internal.h, 95
 - tee_api.h, 124
- TEE.GenerateRandom
 - tee-internal-api.c, 217, 227
 - tee-ta-internal.h, 95
 - tee_api.h, 125
- TEE.GetCancellationFlag
 - tee_api.h, 126
- TEE.GetInstanceData
 - tee_api.h, 126
- TEE.GetNextPersistentObject
 - tee_api.h, 126
- TEE.GetNextProperty
 - tee_api.h, 126
- TEE.GetObjectBufferAttribute
 - tee_api.h, 126
- TEE.GetObjectInfo
 - tee_api.h, 126
- TEE.GetObjectInfo1
 - tee-internal-api.c, 218, 228
 - tee-ta-internal.h, 96
 - tee_api.h, 127
- TEE.GetObjectValueAttribute
 - tee_api.h, 127
- TEE.GetOperationInfo
 - tee_api.h, 127
- TEE.GetOperationInfoMultiple
 - tee_api.h, 128
- TEE.GetPropertyAsBinaryBlock
 - tee_api.h, 128
- TEE.GetPropertyAsBool
 - tee_api.h, 128
- TEE.GetPropertyAsIdentity
 - tee_api.h, 128
- TEE.GetPropertyAsString
 - tee_api.h, 128
- TEE.GetPropertyAsU32
 - tee_api.h, 128
- TEE.GetPropertyAsUUID
 - tee_api.h, 128
- TEE.GetPropertyName
 - tee_api.h, 129
- TEE.GetREETime
 - tee-internal-api.c, 218, 228
 - tee-ta-internal.h, 97
 - tee_api.h, 129
- TEE.GetSystemTime
 - tee-internal-api.c, 219, 229
 - tee-ta-internal.h, 98
 - tee_api.h, 129
- TEE.GetTAPersistentTime
 - tee_api.h, 130
- TEE.HANDLE_FLAG.EXPECT_TWO_KEYS
 - tee_api.defines.h, 158
- TEE.HANDLE_FLAG.INITIALIZED
 - tee_api.defines.h, 158
- TEE.HANDLE_FLAG.KEY_SET
 - tee_api.defines.h, 158
- TEE.HANDLE_FLAG.PERSISTENT
 - tee_api.defines.h, 158
- TEE.HANDLE_NULL
 - tee_api.defines.h, 158
 - tee_api.tee_types.h, 234
- TEE.Identity, 51
 - login, 52
 - uuid, 52
- tee.init
 - tee_def.h, 290–292
- TEE.InitRefAttribute
 - tee-internal-api-cryptlib.c, 275
 - tee-ta-internal.h, 98
 - tee_api.h, 130
- TEE.InitValueAttribute
 - tee-internal-api-cryptlib.c, 276
 - tee-ta-internal.h, 99
 - tee_api.h, 130
- TEE.INT_CORE_API.SPEC_VERSION
 - tee_api.defines.h, 158
- tee.internal-api.extensions.h
 - TEE.CacheClean, 199
 - TEE.CacheFlush, 199
 - TEE.CacheInvalidate, 200
 - tee_map_zi, 200
 - tee_unmap, 200
 - tee_user_mem.check_heap, 200
 - TEE.USER.MEM.HINT.NO.FILL.ZERO, 199
 - tee_user_mem.mark_heap, 200
 - tee_uuid.from_str, 200
- TEE.InvokeTACommand
 - tee_api.h, 131
- TEE.IsAlgorithmSupported
 - tee_api.h, 131
- TEE.LOGIN.APPLICATION
 - tee_api.defines.h, 158
- TEE.LOGIN.APPLICATION.GROUP
 - tee_api.defines.h, 158
- TEE.LOGIN.APPLICATION.USER

- tee_api_defines.h, 158
- TEE.LOGIN_GROUP
 - tee_api_defines.h, 158
- TEE.LOGIN_PUBLIC
 - tee_api_defines.h, 158
- TEE.LOGIN_TRUSTED_APP
 - tee_api_defines.h, 159
- TEE.LOGIN_USER
 - tee_api_defines.h, 159
- TEE.MACCompareFinal
 - tee_api.h, 131
- TEE.MACComputeFinal
 - tee_api.h, 131
- TEE.MACInit
 - tee_api.h, 131
- TEE.MACUpdate
 - tee_api.h, 132
- TEE.Malloc
 - tee-internal-api.c, 219
 - tee_api.h, 132
- TEE.MALLOC_FILL_ZERO
 - tee_api_defines.h, 159
- tee_map_zi
 - tee_internal_api_extensions.h, 200
- TEE.MaskCancellation
 - tee_api.h, 132
- TEE.MEM.INPUT
 - tee_api_types.h, 182
- TEE.MEM.OUTPUT
 - tee_api_types.h, 183
- TEE.MemCompare
 - tee_api.h, 132
- TEE.MemFill
 - tee_api.h, 132
- TEE.MemMove
 - tee_api.h, 133
- TEE.MEMORY_ACCESS.ANY_OWNER
 - tee_api_defines.h, 159
- TEE.MEMORY_ACCESS.NONSECURE
 - tee_api_defines_extensions.h, 180
- TEE.MEMORY_ACCESS.READ
 - tee_api_defines.h, 159
- TEE.MEMORY_ACCESS.SECURE
 - tee_api_defines_extensions.h, 180
- TEE.MEMORY_ACCESS.WRITE
 - tee_api_defines.h, 159
- TEE.MEMREF_0_USED
 - tee_api_types.h, 183
- TEE.MEMREF_1_USED
 - tee_api_types.h, 183
- TEE.MEMREF_2_USED
 - tee_api_types.h, 183
- TEE.MEMREF_3_USED
 - tee_api_types.h, 183
- TEE.MODE.DECRYPT
 - tee_api_types.h, 185
- TEE.MODE.DERIVE
 - tee_api_types.h, 185
- TEE.MODE.DIGEST
 - tee_api_types.h, 185
- TEE.MODE.ENCRYPT
 - tee_api_types.h, 185
- TEE.MODE.MAC
 - tee_api_types.h, 185
- TEE.MODE.SIGN
 - tee_api_types.h, 185
- TEE.MODE.VERIFY
 - tee_api_types.h, 185
- TEE.NUM.PARAMS
 - tee_api_defines.h, 159
- TEE.OBJECT.AAD.SIZE
 - tee_api_tee_types.h, 232, 234
- TEE.OBJECT.ID.MAX.LEN
 - tee_api_defines.h, 159
- TEE.OBJECT.KEY.SIZE
 - tee_api_tee_types.h, 232, 234
- TEE.OBJECT.NONCE.SIZE
 - tee_api_tee_types.h, 232, 234
- TEE.OBJECT.SKEY.SIZE
 - tee_api_tee_types.h, 232, 234
- TEE.OBJECT.TAG.SIZE
 - tee_api_tee_types.h, 232, 235
- TEE.ObjectEnumHandle
 - tee_api_types.h, 184
- TEE.ObjectHandle
 - tee_api_types.h, 184
- TEE.ObjectInfo, 52
 - dataPosition, 52
 - dataSize, 53
 - handleFlags, 53
 - keySize, 53
 - maxKeySize, 53
 - maxObjectSize, 53
 - objectSize, 53
 - objectType, 53
 - objectUsage, 53
- TEE.ObjectType
 - tee_api_types.h, 184
- TEE.OpenPersistentObject
 - tee-internal-api.c, 219, 229
 - tee-ta-internal.h, 99
 - tee_api.h, 133
- TEE.OpenTASession
 - tee_api.h, 134
- TEE.OPERATION.AE
 - tee_api_defines.h, 159
- TEE.OPERATION.ASYMMETRIC.CIPHER
 - tee_api_defines.h, 159
- TEE.OPERATION.ASYMMETRIC.SIGNATURE
 - tee_api_defines.h, 159
- TEE.OPERATION.CIPHER
 - tee_api_defines.h, 160
- TEE.OPERATION.DIGEST
 - tee_api_defines.h, 160
- TEE.OPERATION.KEY.DERIVATION
 - tee_api_defines.h, 160

- TEE_OPERATION_MAC
 - tee_api_defines.h, 160
- TEE_OPERATION_STATE_ACTIVE
 - tee_api_defines.h, 160
- TEE_OPERATION_STATE_INITIAL
 - tee_api_defines.h, 160
- TEE_OperationHandle
 - tee_api_types.h, 184
- TEE_OperationInfo, 53
 - algorithm, 54
 - digestLength, 54
 - handleState, 54
 - keySize, 54
 - maxKeySize, 54
 - mode, 54
 - operationClass, 54
 - requiredKeyUsage, 54
- TEE_OperationInfoKey, 55
 - keySize, 55
 - requiredKeyUsage, 55
- TEE_OperationInfoMultiple, 55
 - algorithm, 56
 - digestLength, 56
 - handleState, 56
 - keyInformation, 56
 - maxKeySize, 56
 - mode, 56
 - numberOfKeys, 56
 - operationClass, 56
 - operationState, 56
- TEE_OperationMode
 - tee_api_types.h, 185
- TEE_ORIGIN_API
 - tee_api_defines.h, 160
- TEE_ORIGIN_COMMS
 - tee_api_defines.h, 160
- TEE_ORIGIN_TEE
 - tee_api_defines.h, 160
- TEE_ORIGIN_TRUSTED_APP
 - tee_api_defines.h, 160
- TEE_Panic
 - tee_api.h, 134
- TEE_PANIC_ID_TA_CLOSESESSIONENTRYPOINT
 - tee_api_defines.h, 160
- TEE_PANIC_ID_TA_CREATEENTRYPOINT
 - tee_api_defines.h, 161
- TEE_PANIC_ID_TA_DESTROYENTRYPOINT
 - tee_api_defines.h, 161
- TEE_PANIC_ID_TA_INVOKECOMMANDENTRYPOINT
 - tee_api_defines.h, 161
- TEE_PANIC_ID_TA_OPENSESSIONENTRYPOINT
 - tee_api_defines.h, 161
- TEE_PANIC_ID_TEE_AEDECRIPTFINAL
 - tee_api_defines.h, 161
- TEE_PANIC_ID_TEE_AEENCRYPTFINAL
 - tee_api_defines.h, 161
- TEE_PANIC_ID_TEE_AEINIT
 - tee_api_defines.h, 161
- TEE_PANIC_ID_TEE_AEUPDATE
 - tee_api_defines.h, 161
- TEE_PANIC_ID_TEE_AEUPDATEAAD
 - tee_api_defines.h, 161
- TEE_PANIC_ID_TEE_ALLOCATEOPERATION
 - tee_api_defines.h, 161
- TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR
 - tee_api_defines.h, 161
- TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERATOR
 - tee_api_defines.h, 162
- TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT
 - tee_api_defines.h, 162
- TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT
 - tee_api_defines.h, 162
- TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT
 - tee_api_defines.h, 162
- TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST
 - tee_api_defines.h, 162
- TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST
 - tee_api_defines.h, 162
- TEE_PANIC_ID_TEE_BIGINTADD
 - tee_api_defines.h, 162
- TEE_PANIC_ID_TEE_BIGINTADDMOD
 - tee_api_defines.h, 162
- TEE_PANIC_ID_TEE_BIGINTCMP
 - tee_api_defines.h, 162
- TEE_PANIC_ID_TEE_BIGINTCMPS32
 - tee_api_defines.h, 162
- TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD
 - tee_api_defines.h, 162
- TEE_PANIC_ID_TEE_BIGINTCOMPUTEFMM
 - tee_api_defines.h, 163
- TEE_PANIC_ID_TEE_BIGINTCONVERTFROMFMM
 - tee_api_defines.h, 163
- TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING
 - tee_api_defines.h, 163
- TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32
 - tee_api_defines.h, 163
- TEE_PANIC_ID_TEE_BIGINTCONVERTTOFMM
 - tee_api_defines.h, 163
- TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING
 - tee_api_defines.h, 163
- TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32
 - tee_api_defines.h, 163
- TEE_PANIC_ID_TEE_BIGINTDIV
 - tee_api_defines.h, 163
- TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32
 - tee_api_defines.h, 163
- TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32
 - tee_api_defines.h, 163
- TEE_PANIC_ID_TEE_BIGINTGETBIT
 - tee_api_defines.h, 164
- TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT
 - tee_api_defines.h, 164
- TEE_PANIC_ID_TEE_BIGINTINIT
 - tee_api_defines.h, 164
- TEE_PANIC_ID_TEE_BIGINTINITFMM
 - tee_api_defines.h, 164

- TEE.PANIC.ID.TEE.BIGINTINITFMMCONTEXT
tee_api_defines.h, 164
- TEE.PANIC.ID.TEE.BIGINTINVMOD
tee_api_defines.h, 164
- TEE.PANIC.ID.TEE.BIGINTISPROBABLEPRIME
tee_api_defines.h, 164
- TEE.PANIC.ID.TEE.BIGINTMOD
tee_api_defines.h, 164
- TEE.PANIC.ID.TEE.BIGINTMUL
tee_api_defines.h, 164
- TEE.PANIC.ID.TEE.BIGINTMULMOD
tee_api_defines.h, 164
- TEE.PANIC.ID.TEE.BIGINTNEG
tee_api_defines.h, 164
- TEE.PANIC.ID.TEE.BIGINTRELATIVEPRIME
tee_api_defines.h, 165
- TEE.PANIC.ID.TEE.BIGINTSHIFTRIGHT
tee_api_defines.h, 165
- TEE.PANIC.ID.TEE.BIGINTSQUARE
tee_api_defines.h, 165
- TEE.PANIC.ID.TEE.BIGINTSQUAREMOD
tee_api_defines.h, 165
- TEE.PANIC.ID.TEE.BIGINTSUB
tee_api_defines.h, 165
- TEE.PANIC.ID.TEE.BIGINTSUBMOD
tee_api_defines.h, 165
- TEE.PANIC.ID.TEE.CHECKMEMORYACCESSRIGHTS
tee_api_defines.h, 165
- TEE.PANIC.ID.TEE.CIPHERDOFINAL
tee_api_defines.h, 165
- TEE.PANIC.ID.TEE.CIPHERINIT
tee_api_defines.h, 165
- TEE.PANIC.ID.TEE.CIPHERUPDATE
tee_api_defines.h, 165
- TEE.PANIC.ID.TEE.CLOSEANDDELETERPERSISTENTOBJECT
tee_api_defines.h, 165
- TEE.PANIC.ID.TEE.CLOSEANDDELETERPERSISTENTOBJECT
tee_api_defines.h, 166
- TEE.PANIC.ID.TEE.CLOSEOBJECT
tee_api_defines.h, 166
- TEE.PANIC.ID.TEE.CLOSETASESSION
tee_api_defines.h, 166
- TEE.PANIC.ID.TEE.COPYOBJECTATTRIBUTES
tee_api_defines.h, 166
- TEE.PANIC.ID.TEE.COPYOBJECTATTRIBUTES1
tee_api_defines.h, 166
- TEE.PANIC.ID.TEE.COPYOPERATION
tee_api_defines.h, 166
- TEE.PANIC.ID.TEE.CREATEPERSISTENTOBJECT
tee_api_defines.h, 166
- TEE.PANIC.ID.TEE.DERIVEKEY
tee_api_defines.h, 166
- TEE.PANIC.ID.TEE.DIGESTDOFINAL
tee_api_defines.h, 166
- TEE.PANIC.ID.TEE.DIGESTUPDATE
tee_api_defines.h, 166
- TEE.PANIC.ID.TEE.FREE
tee_api_defines.h, 166
- TEE.PANIC.ID.TEE.FREEOPERATION
tee_api_defines.h, 167
- TEE.PANIC.ID.TEE.FREEPERSISTENTOBJECTENUMERATOR
tee_api_defines.h, 167
- TEE.PANIC.ID.TEE.FREEPROPERTYENUMERATOR
tee_api_defines.h, 167
- TEE.PANIC.ID.TEE.FREETRANSIENTOBJECT
tee_api_defines.h, 167
- TEE.PANIC.ID.TEE.GENERATEKEY
tee_api_defines.h, 167
- TEE.PANIC.ID.TEE.GENERATERANDOM
tee_api_defines.h, 167
- TEE.PANIC.ID.TEE.GETCANCELLATIONFLAG
tee_api_defines.h, 167
- TEE.PANIC.ID.TEE.GETINSTANCEDATA
tee_api_defines.h, 167
- TEE.PANIC.ID.TEE.GETNEXTPERSISTENTOBJECT
tee_api_defines.h, 167
- TEE.PANIC.ID.TEE.GETNEXTPROPERTY
tee_api_defines.h, 167
- TEE.PANIC.ID.TEE.GETOBJECTBUFFERATTRIBUTE
tee_api_defines.h, 167
- TEE.PANIC.ID.TEE.GETOBJECTINFO
tee_api_defines.h, 168
- TEE.PANIC.ID.TEE.GETOBJECTINFO1
tee_api_defines.h, 168
- TEE.PANIC.ID.TEE.GETOBJECTVALUEATTRIBUTE
tee_api_defines.h, 168
- TEE.PANIC.ID.TEE.GETOPERATIONINFO
tee_api_defines.h, 168
- TEE.PANIC.ID.TEE.GETOPERATIONINFOMULTIPLE
tee_api_defines.h, 168
- TEE.PANIC.ID.TEE.GETPROPERTYASBINARYBLOCK
tee_api_defines.h, 168
- TEE.PANIC.ID.TEE.GETPROPERTYASBOOL
tee_api_defines.h, 168
- TEE.PANIC.ID.TEE.GETPROPERTYASIDENTITY
tee_api_defines.h, 168
- TEE.PANIC.ID.TEE.GETPROPERTYASSTRING
tee_api_defines.h, 168
- TEE.PANIC.ID.TEE.GETPROPERTYASU32
tee_api_defines.h, 168
- TEE.PANIC.ID.TEE.GETPROPERTYASUUID
tee_api_defines.h, 168
- TEE.PANIC.ID.TEE.GETPROPERTYNAME
tee_api_defines.h, 169
- TEE.PANIC.ID.TEE.GETREETIME
tee_api_defines.h, 169
- TEE.PANIC.ID.TEE.GETSYSTEMTIME
tee_api_defines.h, 169
- TEE.PANIC.ID.TEE.GETTAPERSTENTTIME
tee_api_defines.h, 169
- TEE.PANIC.ID.TEE.INITREFATTRIBUTE
tee_api_defines.h, 169
- TEE.PANIC.ID.TEE.INITVALUEATTRIBUTE
tee_api_defines.h, 169
- TEE.PANIC.ID.TEE.INVOKETACOMMAND
tee_api_defines.h, 169

- TEE.PANIC.ID.TEE.MACCOMPAREFINAL
tee_api_defines.h, [169](#)
- TEE.PANIC.ID.TEE.MACCOMPUTEFINAL
tee_api_defines.h, [169](#)
- TEE.PANIC.ID.TEE.MACINIT
tee_api_defines.h, [169](#)
- TEE.PANIC.ID.TEE.MACUPDATE
tee_api_defines.h, [169](#)
- TEE.PANIC.ID.TEE.MALLOC
tee_api_defines.h, [170](#)
- TEE.PANIC.ID.TEE.MASKCANCELLATION
tee_api_defines.h, [170](#)
- TEE.PANIC.ID.TEE.MEMCOMPARE
tee_api_defines.h, [170](#)
- TEE.PANIC.ID.TEE.MEMFILL
tee_api_defines.h, [170](#)
- TEE.PANIC.ID.TEE.MEMMOVE
tee_api_defines.h, [170](#)
- TEE.PANIC.ID.TEE.OPENPERSISTENTOBJECT
tee_api_defines.h, [170](#)
- TEE.PANIC.ID.TEE.OPENTASESSION
tee_api_defines.h, [170](#)
- TEE.PANIC.ID.TEE.PANIC
tee_api_defines.h, [170](#)
- TEE.PANIC.ID.TEE.POPULATETRANSIENTOBJECT
tee_api_defines.h, [170](#)
- TEE.PANIC.ID.TEE.READOBJECTDATA
tee_api_defines.h, [170](#)
- TEE.PANIC.ID.TEE.REALLOC
tee_api_defines.h, [170](#)
- TEE.PANIC.ID.TEE.RENAMEPERSISTENTOBJECT
tee_api_defines.h, [171](#)
- TEE.PANIC.ID.TEE.RESETOPERATION
tee_api_defines.h, [171](#)
- TEE.PANIC.ID.TEE.RESETPERSISTENTOBJECTENUMERATOR
tee_api_defines.h, [171](#)
- TEE.PANIC.ID.TEE.RESETPROPERTYENUMERATOR
tee_api_defines.h, [171](#)
- TEE.PANIC.ID.TEE.RESETTRANSIENTOBJECT
tee_api_defines.h, [171](#)
- TEE.PANIC.ID.TEE.RESTRICTOBJECTUSAGE
tee_api_defines.h, [171](#)
- TEE.PANIC.ID.TEE.RESTRICTOBJECTUSAGE1
tee_api_defines.h, [171](#)
- TEE.PANIC.ID.TEE.SEEKOBJECTDATA
tee_api_defines.h, [171](#)
- TEE.PANIC.ID.TEE.SETINSTANCEDATA
tee_api_defines.h, [171](#)
- TEE.PANIC.ID.TEE.SETOPERATIONKEY
tee_api_defines.h, [171](#)
- TEE.PANIC.ID.TEE.SETOPERATIONKEY2
tee_api_defines.h, [171](#)
- TEE.PANIC.ID.TEE.SETTAPERSISTENTTIME
tee_api_defines.h, [172](#)
- TEE.PANIC.ID.TEE.STARTPERSISTENTOBJECTENUMERATOR
tee_api_defines.h, [172](#)
- TEE.PANIC.ID.TEE.STARTPROPERTYENUMERATOR
tee_api_defines.h, [172](#)
- TEE.PANIC.ID.TEE.TRUNCATEOBJECTDATA
tee_api_defines.h, [172](#)
- TEE.PANIC.ID.TEE.UNMASKCANCELLATION
tee_api_defines.h, [172](#)
- TEE.PANIC.ID.TEE.WAIT
tee_api_defines.h, [172](#)
- TEE.PANIC.ID.TEE.WRITEOBJECTDATA
tee_api_defines.h, [172](#)
- TEE.Param, [57](#)
 - a, [57](#)
 - b, [57](#)
 - buffer, [57](#)
 - memref, [57](#)
 - size, [57](#)
 - value, [57](#)
- TEE.PARAM.TYPE0
crt.c, [358](#)
- TEE.PARAM.TYPE1
crt.c, [358](#)
Enclave.c, [394](#)
- TEE.PARAM.TYPE.GET
tee_api_defines.h, [172](#)
- TEE.PARAM.TYPE.MEMREF.INOUT
tee_api_defines.h, [172](#)
- TEE.PARAM.TYPE.MEMREF.INPUT
tee_api_defines.h, [172](#)
- TEE.PARAM.TYPE.MEMREF.OUTPUT
tee_api_defines.h, [173](#)
- TEE.PARAM.TYPE.NONE
tee_api_defines.h, [173](#)
- TEE.PARAM.TYPE.SET
tee_api_defines.h, [173](#)
- TEE.PARAM.TYPE.VALUE.INOUT
tee_api_defines.h, [173](#)
- TEE.PARAM.TYPE.VALUE.INPUT
tee_api_defines.h, [173](#)
- TEE.PARAM.TYPE.VALUE.OUTPUT
tee_api_defines.h, [173](#)
- TEE.PARAM.TYPES
tee_api_defines.h, [173](#)
- TEE.PopulateTransientObject
tee_api.h, [134](#)
- tee.printf
config_ref.ta.h, [310](#)
crt.c, [360](#)
Enclave.c, [397](#)
trace.c, [241](#), [242](#), [244](#)
- tee.profiler.c
 - ...profiler.head, [342](#), [343](#), [345](#)
 - ...profiler.unmap.info, [340](#), [342](#), [344](#)
 - profiler.write, [341](#), [343](#), [344](#)
- tee.profiler.h
 - ...profiler.unmap.info, [346–348](#)
- TEE.PROPSET.CURRENT.CLIENT
tee_api_defines.h, [173](#)
- TEE.PROPSET.CURRENT.TA
tee_api_defines.h, [173](#)
- TEE.PROPSET.TEE.IMPLEMENTATION

- tee_api_defines.h, 173
- TEE_PropSetHandle
 - tee_api_types.h, 184
- tee_rdtscp
 - tee_config.h, 337–339
- TEE_ReadObjectData
 - tee-internal-api.c, 220, 229
 - tee-ta-internal.h, 100
 - tee_api.h, 134
- TEE_Realloc
 - tee-internal-api.c, 220
 - tee_api.h, 135
- TEE_RenamePersistentObject
 - tee_api.h, 135
- TEE_ResetOperation
 - tee_api.h, 136
- TEE_ResetPersistentObjectEnumerator
 - tee_api.h, 136
- TEE_ResetPropertyEnumerator
 - tee_api.h, 136
- TEE_ResetTransientObject
 - tee_api.h, 136
- TEE_RestrictObjectUsage
 - tee_api.h, 136
- TEE_RestrictObjectUsage1
 - tee_api.h, 136
- TEE_Result
 - tee_api_types.h, 184
- TEE_SE_READER_NAME_MAX
 - tee_api_types.h, 183
- TEE_SEAID, 58
 - buffer, 58
 - bufferLen, 58
- TEE_SEChannelHandle
 - tee_api_types.h, 184
- TEE_SeekObjectData
 - tee_api.h, 136
- TEE_SEReaderHandle
 - tee_api_types.h, 184
- TEE_SEReaderProperties, 58
 - selectResponseEnable, 58
 - sePresent, 59
 - teeOnly, 59
- TEE_SEServiceHandle
 - tee_api_types.h, 184
- TEE_SESessionHandle
 - tee_api_types.h, 184
- TEE_Session
 - tee_api_types.h, 185
- TEE_SetInstanceData
 - tee_api.h, 136
- TEE_SetOperationKey
 - tee-internal-api-cryptlib.c, 276
 - tee-ta-internal.h, 101
 - tee_api.h, 136
- TEE_SetOperationKey2
 - tee_api.h, 137
- TEE_SetTAPersistentTime
 - tee_api.h, 137
- TEE_StartPersistentObjectEnumerator
 - tee_api.h, 137
- TEE_StartPropertyEnumerator
 - tee_api.h, 137
- TEE_STORAGE_PRIVATE
 - tee_api_defines.h, 174
- TEE_STORAGE_PRIVATE_REE
 - tee_api_defines_extensions.h, 180
- TEE_STORAGE_PRIVATE_RPMB
 - tee_api_defines_extensions.h, 180
- TEE_STORAGE_PRIVATE_SQL_RESERVED
 - tee_api_defines_extensions.h, 180
- TEE_SUCCESS
 - tee_api_defines.h, 174
- tee_ta_api.h
 - TA_CloseSessionEntryPoint, 202
 - TA_CreateEntryPoint, 202
 - TA_DestroyEntryPoint, 202
 - TA_EXPORT, 201
 - TA_InvokeCommandEntryPoint, 202
 - TA_OpenSessionEntryPoint, 202
- TEE_TASessionHandle
 - tee_api_types.h, 185
- TEE_Time, 59
 - millis, 59
 - seconds, 59
- tee_time_tests
 - bench.c, 279
- TEE_TIMEOUT_INFINITE
 - tee_api_defines.h, 174
- TEE_TruncateObjectData
 - tee_api.h, 137
- TEE_TYPE_AES
 - tee_api_defines.h, 174
- TEE_TYPE_CONCAT_KDF_Z
 - tee_api_defines_extensions.h, 180
- TEE_TYPE_CORRUPTED_OBJECT
 - tee_api_defines.h, 174
- TEE_TYPE_DATA
 - tee_api_defines.h, 174
- TEE_TYPE_DES
 - tee_api_defines.h, 174
- TEE_TYPE_DES3
 - tee_api_defines.h, 174
- TEE_TYPE_DH_KEYPAIR
 - tee_api_defines.h, 174
- TEE_TYPE_DSA_KEYPAIR
 - tee_api_defines.h, 174
- TEE_TYPE_DSA_PUBLIC_KEY
 - tee_api_defines.h, 174
- TEE_TYPE_ECDH_KEYPAIR
 - tee_api_defines.h, 175
- TEE_TYPE_ECDH_PUBLIC_KEY
 - tee_api_defines.h, 175
- TEE_TYPE_ECDSA_KEYPAIR
 - tee_api_defines.h, 175
- TEE_TYPE_ECDSA_PUBLIC_KEY
 - tee_api_defines.h, 175

- tee_api_defines.h, 175
- TEE_TYPE_GENERIC_SECRET
 - tee_api_defines.h, 175
- TEE_TYPE_HKDF_IKM
 - tee_api_defines_extensions.h, 180
- TEE_TYPE_HMAC_MD5
 - tee_api_defines.h, 175
- TEE_TYPE_HMAC_SHA1
 - tee_api_defines.h, 175
- TEE_TYPE_HMAC_SHA224
 - tee_api_defines.h, 175
- TEE_TYPE_HMAC_SHA256
 - tee_api_defines.h, 175
- TEE_TYPE_HMAC_SHA384
 - tee_api_defines.h, 175
- TEE_TYPE_HMAC_SHA512
 - tee_api_defines.h, 175
- TEE_TYPE_PBKDF2_PASSWORD
 - tee_api_defines_extensions.h, 180
- TEE_TYPE_RSA_KEYPAIR
 - tee_api_defines.h, 176
- TEE_TYPE_RSA_PUBLIC_KEY
 - tee_api_defines.h, 176
- tee_unmap
 - tee_internal_api_extensions.h, 200
- TEE_UnmaskCancellation
 - tee_api.h, 138
- TEE_USAGE_DECRYPT
 - tee_api_defines.h, 176
- TEE_USAGE_DERIVE
 - tee_api_defines.h, 176
- TEE_USAGE_ENCRYPT
 - tee_api_defines.h, 176
- TEE_USAGE_EXTRACTABLE
 - tee_api_defines.h, 176
- TEE_USAGE_MAC
 - tee_api_defines.h, 176
- TEE_USAGE_SIGN
 - tee_api_defines.h, 176
- TEE_USAGE_VERIFY
 - tee_api_defines.h, 176
- tee_user_mem_check_heap
 - tee_internal_api_extensions.h, 200
- TEE_USER_MEM_HINT_NO_FILL_ZERO
 - tee_internal_api_extensions.h, 199
- tee_user_mem_mark_heap
 - tee_internal_api_extensions.h, 200
- TEE_UUID, 59
 - clockSeqAndNode, 60
 - timeHiAndVersion, 60
 - timeLow, 60
 - timeMid, 60
- tee_uuid_from_str
 - tee_internal_api_extensions.h, 200
- TEE.Wait
 - tee_api.h, 138
- TEE.Whence
 - tee_api_types.h, 185
- TEE.WriteObjectData
 - tee-internal-api.c, 221, 230
 - tee-ta-internal.h, 102
 - tee_api.h, 138
- TEEC.AllocateSharedMemory
 - tee_client_api.h, 194
 - teec_stub.c, 235
- TEEC.CloseSession
 - tee_client_api.h, 194
 - teec_stub.c, 236
- TEEC_CONFIG_PAYLOAD_REF_COUNT
 - tee_client_api.h, 188
- TEEC_CONFIG_SHAREDMEM_MAX_SIZE
 - tee_client_api.h, 188
- TEEC.Context, 60
 - fd, 60
 - reg_mem, 61
- TEEC_ERROR_ACCESS_CONFLICT
 - tee_client_api.h, 188
- TEEC_ERROR_ACCESS_DENIED
 - tee_client_api.h, 188
- TEEC_ERROR_BAD_FORMAT
 - tee_client_api.h, 188
- TEEC_ERROR_BAD_PARAMETERS
 - tee_client_api.h, 188
- TEEC_ERROR_BAD_STATE
 - tee_client_api.h, 188
- TEEC_ERROR_BUSY
 - tee_client_api.h, 188
- TEEC_ERROR_CANCEL
 - tee_client_api.h, 189
- TEEC_ERROR_COMMUNICATION
 - tee_client_api.h, 189
- TEEC_ERROR_EXCESS_DATA
 - tee_client_api.h, 189
- TEEC_ERROR_EXTERNAL_CANCEL
 - tee_client_api.h, 189
- TEEC_ERROR_GENERIC
 - tee_client_api.h, 189
- TEEC_ERROR_ITEM_NOT_FOUND
 - tee_client_api.h, 189
- TEEC_ERROR_NO_DATA
 - tee_client_api.h, 189
- TEEC_ERROR_NOT_IMPLEMENTED
 - tee_client_api.h, 189
- TEEC_ERROR_NOT_SUPPORTED
 - tee_client_api.h, 189
- TEEC_ERROR_OUT_OF_MEMORY
 - tee_client_api.h, 189
- TEEC_ERROR_SECURITY
 - tee_client_api.h, 189
- TEEC_ERROR_SHORT_BUFFER
 - tee_client_api.h, 190
- TEEC_ERROR_TARGET_DEAD
 - tee_client_api.h, 190
- TEEC.FinalizeContext
 - tee_client_api.h, 194
 - teec_stub.c, 236

- TEEC.InitializeContext
 - tee_client_api.h, 195
 - teec_stub.c, 236
- TEEC.InvokeCommand
 - tee_client_api.h, 195
- TEEC.LOGIN_APPLICATION
 - tee_client_api.h, 190
- TEEC.LOGIN_GROUP
 - tee_client_api.h, 190
- TEEC.LOGIN_GROUP_APPLICATION
 - tee_client_api.h, 190
- TEEC.LOGIN_PUBLIC
 - tee_client_api.h, 190
- TEEC.LOGIN_USER
 - tee_client_api.h, 190
- TEEC.LOGIN_USER_APPLICATION
 - tee_client_api.h, 190
- TEEC.MEM.INPUT
 - tee_client_api.h, 190
- TEEC.MEM.OUTPUT
 - tee_client_api.h, 191
- TEEC.MEMREF.PARTIAL.INOUT
 - tee_client_api.h, 191
- TEEC.MEMREF.PARTIAL.INPUT
 - tee_client_api.h, 191
- TEEC.MEMREF.PARTIAL.OUTPUT
 - tee_client_api.h, 191
- TEEC.MEMREF.TEMP.INOUT
 - tee_client_api.h, 191
- TEEC.MEMREF.TEMP.INPUT
 - tee_client_api.h, 191
- TEEC.MEMREF.TEMP.OUTPUT
 - tee_client_api.h, 191
- TEEC.MEMREF.WHOLE
 - tee_client_api.h, 191
- TEEC.NONE
 - tee_client_api.h, 191
- TEEC.OpenSession
 - tee_client_api.h, 196
 - teec_stub.c, 237
- TEEC.Operation, 61
 - params, 62
 - paramTypes, 62
 - session, 62
 - started, 62
- TEEC.ORIGIN.API
 - tee_client_api.h, 192
- TEEC.ORIGIN.COMMS
 - tee_client_api.h, 192
- TEEC.ORIGIN.TEE
 - tee_client_api.h, 192
- TEEC.ORIGIN.TRUSTED_APP
 - tee_client_api.h, 192
- TEEC.PARAM.TYPE0
 - main.c, 404
- TEEC.PARAM.TYPE1
 - main.c, 402, 404
- TEEC.PARAM.TYPE.GET
 - tee_client_api.h, 192
- TEEC.PARAM.TYPES
 - tee_client_api.h, 193
- TEEC.Parameter, 62
 - memref, 63
 - tmpref, 63
 - value, 63
- TEEC.RegisteredMemoryReference, 63
 - offset, 64
 - parent, 64
 - size, 64
- TEEC.RegisterSharedMemory
 - tee_client_api.h, 196
 - teec_stub.c, 237
- TEEC.ReleaseSharedMemory
 - tee_client_api.h, 197
 - teec_stub.c, 238
- TEEC.RequestCancellation
 - tee_client_api.h, 197
 - teec_stub.c, 238
- TEEC.Result
 - tee_client_api.h, 194
- TEEC.Session, 65
 - ctx, 65
 - session_id, 65
- TEEC.SharedMemory, 65
 - allocated_size, 66
 - buffer, 66
 - buffer_allocated, 66
 - flags, 66
 - id, 66
 - registered_fd, 67
 - shadow_buffer, 67
 - size, 67
- teec_stub.c
 - TEEC.AllocateSharedMemory, 235
 - TEEC.CloseSession, 236
 - TEEC.FinalizeContext, 236
 - TEEC.InitializeContext, 236
 - TEEC.OpenSession, 237
 - TEEC.RegisterSharedMemory, 237
 - TEEC.ReleaseSharedMemory, 238
 - TEEC.RequestCancellation, 238
- TEEC.SUCCESS
 - tee_client_api.h, 193
- TEEC.TempMemoryReference, 67
 - buffer, 67
 - size, 68
- TEEC.UUID, 68
 - clockSeqAndNode, 68
 - timeHiAndVersion, 68
 - timeLow, 68
 - timeMid, 68
- TEEC.Value, 69
 - a, 69
 - b, 69
- TEEC.VALUE.INOUT
 - tee_client_api.h, 193

- TEEC_VALUE_INPUT
 - tee_client_api.h, 193
- TEEC_VALUE_OUTPUT
 - tee_client_api.h, 193
- teeOnly
 - TEE_SEReaderProperties, 59
- TEEP_AGENT_TA_DELETE
 - invoke_command.c, 314
- TEEP_AGENT_TA_EXIT
 - invoke_command.c, 314
- TEEP_AGENT_TA_INSTALL
 - invoke_command.c, 314
- TEEP_AGENT_TA_LOAD
 - invoke_command.c, 315
- TEEP_AGENT_TA_NONE
 - invoke_command.c, 315
- test_dev_key.h
 - _sanctum_dev_public_key, 203
 - _sanctum_dev_public_key_len, 203
 - _sanctum_dev_secret_key, 203
 - _sanctum_dev_secret_key_len, 203
- test_printf
 - tee_def.h, 290–292
- time.c
 - gp_ree_time_test, 321
 - gp_trusted_time_test, 321
- time_diff
 - bench.c, 279
- time_test
 - bench.c, 280
- time_test.c
 - ree_time_test, 295
 - system_time_test, 295
- time_to_millis
 - bench.c, 280
- timeHiAndVersion
 - TEE_UUID, 60
 - TEEC_UUID, 68
- timeLow
 - TEE_UUID, 60
 - TEEC_UUID, 68
- timeMid
 - TEE_UUID, 60
 - TEEC_UUID, 68
- tmpref
 - TEEC_Parameter, 63
- tools.c
 - _strlen, 335
 - printf, 335
 - profiler_write, 331–334
 - putchar, 336
 - puts, 336
- tools.h
 - printf, 366
 - putchar, 366
 - puts, 367
- trace.c
 - _strlen, 241, 243
 - tee_printf, 241, 242, 244
 - trace_printf, 239
 - trace_vprintf, 240
- trace.h
 - dhex_dump, 208
 - DHEXDUMP, 205
 - DMSG, 205
 - DMSG_RAW, 205
 - DPRINT_STACK, 205
 - EMSG, 206
 - EMSG_RAW, 206
 - EPRINT_STACK, 206
 - FMSG, 206
 - FMSG_RAW, 206
 - FPRINT_STACK, 206
 - IMSG, 206
 - IMSG_RAW, 206
 - INMSG, 206
 - IPRINT_STACK, 206
 - MAX_FUNC_PRINT_SIZE, 207
 - MAX_PRINT_SIZE, 207
 - MSG, 207
 - MSG_RAW, 207
 - OUTMSG, 207
 - OUTRMSG, 207
 - SMSG, 207
 - trace_ext_get_thread_id, 208
 - trace_ext_prefix, 209
 - trace_ext_puts, 208
 - trace_get_level, 208
 - TRACE_LEVEL, 207
 - trace_level, 209
 - trace_printf, 208
 - trace_printf_helper, 207
 - trace_printf_helper_raw, 208
 - trace_set_level, 208
- TRACE_DEBUG
 - trace_levels.h, 209
- TRACE_ERROR
 - trace_levels.h, 210
- trace_ext_get_thread_id
 - trace.h, 208
- trace_ext_prefix
 - trace.h, 209
- user_ta_header.c, 407, 411
- trace_ext_puts
 - trace.h, 208
- TRACE_FLOW
 - trace_levels.h, 210
- trace_get_level
 - trace.h, 208
- TRACE_INFO
 - trace_levels.h, 210
- TRACE_LEVEL
 - trace.h, 207
- trace_level
 - trace.h, 209
- user_ta_header.c, 408, 411

- trace_levels.h
 - TRACE_DEBUG, 209
 - TRACE_ERROR, 210
 - TRACE_FLOW, 210
 - TRACE_INFO, 210
 - TRACE_MAX, 210
 - TRACE_MIN, 210
 - TRACE_PRINTF_LEVEL, 210
- TRACE_MAX
 - trace_levels.h, 210
- TRACE_MIN
 - trace_levels.h, 210
- trace_printf
 - trace.c, 239
 - trace.h, 208
- trace_printf_helper
 - trace.h, 207
- trace_printf_helper_raw
 - trace.h, 208
- TRACE_PRINTF_LEVEL
 - trace_levels.h, 210
- trace_set_level
 - trace.h, 208
- trace_vprintf
 - trace.c, 240
- TRUE
 - App.h, 416, 417
- type
 - __TEE_ObjectHandle, 36
 - nm_info, 43
- types.h
 - global_eid, 434, 436
 - sgx_errlist, 434, 436
 - sgx_errlist_t, 434, 436
- USED
 - profiler_attrs.h, 352
- user_ta_header.c
 - _C_FUNCTION, 405, 409
 - __section, 406, 409
 - __ta_entry, 406, 409
 - __utee_entry, 406, 409
 - TA_DESCRIPTION, 406, 409
 - TA_FRAMEWORK_STACK_SIZE, 406, 409
 - ta_heap, 407, 410
 - ta_heap_size, 407, 410
 - ta_num_props, 407, 410
 - ta_props, 407, 410
 - TA_VERSION, 406, 409
 - tahead_get_trace_level, 407, 410
 - trace_ext_prefix, 407, 411
 - trace_level, 408, 411
- user_ta_header_defines.h
 - TA_CURRENT_TA_EXT_PROPERTIES, 412, 414
 - TA_DATA_SIZE, 412, 414
 - TA_DESCRIPTION, 412, 414
 - TA_FLAGS, 412, 414
 - TA_STACK_SIZE, 413, 414
 - TA_UUID, 413, 414
 - TA_VERSION, 413, 414
- user_types.h
 - array_t, 296
 - buffer_t, 296
 - LOOPS_PER_THREAD, 296
- uuid
 - TEE.Identity, 52
- value
 - TEE_Attribute, 51
 - TEE_Param, 57
 - TEEC_Parameter, 63
- vsnprintf
 - vsnprintf.c, 250, 263
- vsnprintf.c
 - .atoi, 247, 253
 - .ftoa, 247, 254
 - .is_digit, 248, 254
 - .ntoa_format, 248, 255
 - .ntoa_long, 248, 255
 - .ntoa_long_long, 248, 256
 - .out_buffer, 248, 257
 - .out_char, 249, 257
 - .out_fct, 249, 258
 - .out_null, 249, 258
 - .putchar, 246, 251
 - .strlen, 249, 258
 - .vsnprintf, 249, 259
 - fctprintf, 249, 259
 - FLAGS_CHAR, 246, 252
 - FLAGS_HASH, 246, 252
 - FLAGS_LEFT, 246, 252
 - FLAGS_LONG, 246, 252
 - FLAGS_LONG_LONG, 246, 252
 - FLAGS_PLUS, 246, 252
 - FLAGS_PRECISION, 246, 252
 - FLAGS_SHORT, 246, 252
 - FLAGS_SPACE, 246, 252
 - FLAGS_UPPERCASE, 246, 252
 - FLAGS_ZEROPAD, 247, 252
 - out_fct_type, 247, 253
 - PRINTF_FTOA_BUFFER_SIZE, 247, 253
 - PRINTF_NTOA_BUFFER_SIZE, 247, 253
 - PRINTF_SUPPORT_FLOAT, 247, 253
 - PRINTF_SUPPORT_LONG_LONG, 247, 253
 - PRINTF_SUPPORT_PTRDIFF_T, 247, 253
 - putchar, 249, 261
 - snprintf, 250, 261
 - sprintf, 250, 261
 - vsnprintf, 250, 263
- WOLFCRYPT
 - tee_api_tee_types.h, 232, 235
- wolfSSL_Free
 - tee-internal-api-cryptlib.c, 277
- wolfSSL_Malloc
 - tee-internal-api-cryptlib.c, 277