

# Trusted Application Programming Reference on Portable TEE

National Institute of Advanced Industrial Science and Technology

2021-02-19

<b>1 Preparation</b>	<b>1</b>
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
<b>2 Building</b>	<b>8</b>
2.1 Install Doxygen-1.9.2	8
2.2 Install Required Packages	8
2.3 Build and Install	8
2.4 ta-ref with Keystone	9
2.4.1 Cloning source and building	9
2.4.2 Check ta-ref by running test_gp, test_hello, on QEMU	9
2.5 ta-ref with OPTEE	11
2.5.1 Cloning source and building	11
2.5.2 Check ta-ref by running test_gp, test_hello, on QEMU	11
2.6 ta-ref with SGX	12
2.6.1 Cloning source and building	12
2.6.2 Check ta-ref by running test_gp, test_hello, simulation mode on any pc	12
<b>3 Running on Dev Boards</b>	<b>14</b>
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	16
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

<b>4 Overview of ta-ref</b>	<b>21</b>
4.1 Features	21
4.1.1 What we did on RISC-V	21
4.1.2 Separate GP TEE Internal API	21
4.2 Diagram	22
4.2.1 Dependency of category	22
<b>5 How to Program on ta-ref</b>	<b>22</b>
5.1 Time Functions	22
5.2 Random Functions	22
5.3 Hash Functions	23
5.4 Symmetric Crypto Functions	23
5.5 Asymmetric Crypto Functions	24
5.6 Asymmetric Crypto Gcm Functions	25
5.7 Open, Read, Write, Close On Secure Storage	26
<b>6 API Compare With Full-Set of GP API</b>	<b>28</b>
6.1 GP API	28
<b>7 Class Index</b>	<b>30</b>
7.1 Class List	30
<b>8 File Index</b>	<b>31</b>
8.1 File List	31
<b>9 Class Documentation</b>	<b>35</b>
9.1 __profiler_data Struct Reference	35
9.1.1 Member Data Documentation	35
9.2 __profiler_header Struct Reference	36
9.2.1 Member Data Documentation	36
9.3 __TEE_ObjectHandle Struct Reference	36
9.3.1 Member Data Documentation	36
9.4 __TEE_OperationHandle Struct Reference	37
9.4.1 Member Data Documentation	37
9.5 _sgx_errlist_t Struct Reference	39
9.5.1 Member Data Documentation	39
9.6 addrinfo Struct Reference	39
9.6.1 Member Data Documentation	40
9.7 enclave_report Struct Reference	41
9.7.1 Member Data Documentation	41
9.8 invoke_command_param_t Struct Reference	41
9.8.1 Member Data Documentation	42
9.9 invoke_command_t Struct Reference	42
9.9.1 Member Data Documentation	43

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

9.31 TEE_Time Struct Reference . . . . .	60
9.31.1 Member Data Documentation . . . . .	60
9.32 TEE_UUID Struct Reference . . . . .	60
9.32.1 Member Data Documentation . . . . .	61
9.33 TEEC_Context Struct Reference . . . . .	61
9.33.1 Detailed Description . . . . .	61
9.33.2 Member Data Documentation . . . . .	61
9.34 TEEC_Operation Struct Reference . . . . .	62
9.34.1 Detailed Description . . . . .	62
9.34.2 Member Data Documentation . . . . .	63
9.35 TEEC_Parameter Union Reference . . . . .	63
9.35.1 Detailed Description . . . . .	63
9.35.2 Member Data Documentation . . . . .	64
9.36 TEEC_RegisteredMemoryReference Struct Reference . . . . .	64
9.36.1 Detailed Description . . . . .	65
9.36.2 Member Data Documentation . . . . .	65
9.37 TEEC_Session Struct Reference . . . . .	66
9.37.1 Detailed Description . . . . .	66
9.37.2 Member Data Documentation . . . . .	66
9.38 TEEC_SharedMemory Struct Reference . . . . .	66
9.38.1 Detailed Description . . . . .	67
9.38.2 Member Data Documentation . . . . .	67
9.39 TEEC_TempMemoryReference Struct Reference . . . . .	68
9.39.1 Detailed Description . . . . .	68
9.39.2 Member Data Documentation . . . . .	68
9.40 TEEC_UUID Struct Reference . . . . .	69
9.40.1 Detailed Description . . . . .	69
9.40.2 Member Data Documentation . . . . .	69
9.41 TEEC_Value Struct Reference . . . . .	70
9.41.1 Detailed Description . . . . .	70
9.41.2 Member Data Documentation . . . . .	70
<b>10 File Documentation . . . . .</b>	<b>71</b>
10.1 ta-ref/api/include/compiler.h File Reference . . . . .	71
10.1.1 Macro Definition Documentation . . . . .	72
10.2 ta-ref/api/include/report.h File Reference . . . . .	78
10.2.1 Macro Definition Documentation . . . . .	78
10.3 ta-ref/api/include/tee-common.h File Reference . . . . .	79
10.3.1 Detailed Description . . . . .	80
10.3.2 Macro Definition Documentation . . . . .	80
10.4 ta-ref/api/include/tee-ta-internal.h File Reference . . . . .	80
10.4.1 Detailed Description . . . . .	82

10.4.2 Function Documentation . . . . .	82
10.5 ta-ref/api/include/tee_api.h File Reference . . . . .	104
10.5.1 Function Documentation . . . . .	108
10.6 ta-ref/api/include/tee_api_defines.h File Reference . . . . .	140
10.6.1 Macro Definition Documentation . . . . .	146
10.7 ta-ref/api/include/tee_api_defines_extensions.h File Reference . . . . .	178
10.7.1 Macro Definition Documentation . . . . .	179
10.8 ta-ref/api/include/tee_api_types.h File Reference . . . . .	182
10.8.1 Macro Definition Documentation . . . . .	183
10.8.2 Typedef Documentation . . . . .	184
10.8.3 Enumeration Type Documentation . . . . .	186
10.9 ta-ref/api/include/tee_client_api.h File Reference . . . . .	186
10.9.1 Macro Definition Documentation . . . . .	189
10.9.2 Typedef Documentation . . . . .	195
10.9.3 Function Documentation . . . . .	195
10.10 ta-ref/api/include/tee_internal_api.h File Reference . . . . .	198
10.11 ta-ref/api/include/tee_internal_api_extensions.h File Reference . . . . .	199
10.11.1 Macro Definition Documentation . . . . .	200
10.11.2 Function Documentation . . . . .	200
10.12 ta-ref/api/include/tee_ta_api.h File Reference . . . . .	202
10.12.1 Macro Definition Documentation . . . . .	202
10.12.2 Function Documentation . . . . .	203
10.13 ta-ref/api/include/test_dev_key.h File Reference . . . . .	204
10.13.1 Variable Documentation . . . . .	204
10.14 ta-ref/api/include/trace.h File Reference . . . . .	205
10.14.1 Macro Definition Documentation . . . . .	206
10.14.2 Function Documentation . . . . .	209
10.14.3 Variable Documentation . . . . .	210
10.15 ta-ref/api/include/trace_levels.h File Reference . . . . .	210
10.15.1 Macro Definition Documentation . . . . .	210
10.16 ta-ref/api/keystone/tee-internal-api-machine.c File Reference . . . . .	211
10.16.1 Function Documentation . . . . .	212
10.17 ta-ref/api/keystone/tee-internal-api.c File Reference . . . . .	212
10.17.1 Macro Definition Documentation . . . . .	213
10.17.2 Function Documentation . . . . .	214
10.18 ta-ref/api/sgx/tee-internal-api.c File Reference . . . . .	223
10.18.1 Macro Definition Documentation . . . . .	224
10.18.2 Function Documentation . . . . .	225
10.19 ta-ref/api/keystone/tee_api_tee_types.h File Reference . . . . .	232
10.19.1 Macro Definition Documentation . . . . .	233
10.20 ta-ref/api/optee/tee_api_tee_types.h File Reference . . . . .	234
10.21 ta-ref/api/sgx/tee_api_tee_types.h File Reference . . . . .	234

10.21.1 Macro Definition Documentation . . . . .	235
10.22 ta-ref/api/keystone/teec_stub.c File Reference . . . . .	236
10.22.1 Function Documentation . . . . .	236
10.23 ta-ref/api/keystone/trace.c File Reference . . . . .	239
10.23.1 Function Documentation . . . . .	240
10.24 ta-ref/test_gp/keystone/Enclave/trace.c File Reference . . . . .	241
10.24.1 Function Documentation . . . . .	242
10.25 ta-ref/test_gp/optee/Enclave/trace.c File Reference . . . . .	242
10.25.1 Function Documentation . . . . .	243
10.26 ta-ref/test_gp/sgx/Enclave/trace.c File Reference . . . . .	244
10.26.1 Function Documentation . . . . .	244
10.27 ta-ref/api/keystone/vsnprintf.c File Reference . . . . .	245
10.27.1 Macro Definition Documentation . . . . .	247
10.27.2 Typedef Documentation . . . . .	248
10.27.3 Function Documentation . . . . .	248
10.28 ta-ref/test_gp/vsnprintf.c File Reference . . . . .	251
10.28.1 Macro Definition Documentation . . . . .	252
10.28.2 Typedef Documentation . . . . .	254
10.28.3 Function Documentation . . . . .	254
10.29 ta-ref/api/tee-internal-api-cryptlib.c File Reference . . . . .	264
10.29.1 Macro Definition Documentation . . . . .	266
10.29.2 Function Documentation . . . . .	267
10.30 ta-ref/benchmark/bench.c File Reference . . . . .	278
10.30.1 Function Documentation . . . . .	279
10.30.2 Variable Documentation . . . . .	281
10.31 ta-ref/benchmark/bench.h File Reference . . . . .	282
10.31.1 Macro Definition Documentation . . . . .	283
10.31.2 Function Documentation . . . . .	283
10.32 ta-ref/benchmark/cpu_bench.c File Reference . . . . .	285
10.32.1 Function Documentation . . . . .	286
10.33 ta-ref/benchmark/include/config_bench.h File Reference . . . . .	286
10.33.1 Macro Definition Documentation . . . . .	287
10.33.2 Enumeration Type Documentation . . . . .	287
10.33.3 Function Documentation . . . . .	288
10.34 ta-ref/benchmark/io_bench.c File Reference . . . . .	288
10.34.1 Macro Definition Documentation . . . . .	289
10.34.2 Function Documentation . . . . .	289
10.35 ta-ref/benchmark/keystone/tee_def.h File Reference . . . . .	290
10.35.1 Function Documentation . . . . .	291
10.35.2 Variable Documentation . . . . .	291
10.36 ta-ref/benchmark/optee/tee_def.h File Reference . . . . .	291
10.36.1 Macro Definition Documentation . . . . .	292

10.36.2 Function Documentation . . . . .	292
10.36.3 Variable Documentation . . . . .	292
10.37 ta-ref/benchmark/sgx/tee_def.h File Reference . . . . .	292
10.37.1 Function Documentation . . . . .	293
10.37.2 Variable Documentation . . . . .	293
10.38 ta-ref/benchmark/memory_bench.c File Reference . . . . .	294
10.38.1 Macro Definition Documentation . . . . .	294
10.38.2 Function Documentation . . . . .	294
10.39 ta-ref/benchmark/time_test.c File Reference . . . . .	295
10.39.1 Function Documentation . . . . .	296
10.40 ta-ref/docs/building.md File Reference . . . . .	296
10.41 ta-ref/docs/gp_api.md File Reference . . . . .	296
10.42 ta-ref/docs/how_to_program_on_ta-ref.md File Reference . . . . .	296
10.43 ta-ref/docs/overview_of_ta-ref.md File Reference . . . . .	296
10.44 ta-ref/docs/preparation.md File Reference . . . . .	296
10.45 ta-ref/docs/running_on_dev_boards.md File Reference . . . . .	296
10.46 ta-ref/edger/edger8r/user_types.h File Reference . . . . .	296
10.46.1 Macro Definition Documentation . . . . .	297
10.46.2 Typedef Documentation . . . . .	297
10.47 ta-ref/edger/keyedge/Enclave.t.c File Reference . . . . .	297
10.48 ta-ref/edger/keyedge/Enclave.t.h File Reference . . . . .	298
10.48.1 Function Documentation . . . . .	298
10.49 ta-ref/edger/optee/Enclave.t.h File Reference . . . . .	299
10.50 ta-ref/edger/keyedge/Enclave.u.c File Reference . . . . .	299
10.51 ta-ref/edger/keyedge/Enclave.u.h File Reference . . . . .	300
10.51.1 Macro Definition Documentation . . . . .	300
10.51.2 Function Documentation . . . . .	300
10.52 ta-ref/edger/keyedge/ocalls.h File Reference . . . . .	301
10.52.1 Macro Definition Documentation . . . . .	302
10.52.2 Typedef Documentation . . . . .	303
10.52.3 Function Documentation . . . . .	303
10.53 ta-ref/edger/optee/Enclave.h File Reference . . . . .	306
10.53.1 Macro Definition Documentation . . . . .	306
10.54 ta-ref/test_gp/sgx/Enclave/Enclave.h File Reference . . . . .	307
10.54.1 Function Documentation . . . . .	307
10.55 ta-ref/gp/asymmetric_key.c File Reference . . . . .	309
10.55.1 Macro Definition Documentation . . . . .	310
10.55.2 Function Documentation . . . . .	310
10.56 ta-ref/gp/include/config_ref_ta.h File Reference . . . . .	310
10.56.1 Macro Definition Documentation . . . . .	311
10.56.2 Function Documentation . . . . .	311
10.57 ta-ref/gp/include/gp_test.h File Reference . . . . .	312



10.57.1 Function Documentation . . . . .	313
10.58 ta-ref/gp/invoke_command.c File Reference . . . . .	315
10.58.1 Macro Definition Documentation . . . . .	315
10.59 ta-ref/gp/message_digest.c File Reference . . . . .	316
10.59.1 Macro Definition Documentation . . . . .	316
10.59.2 Function Documentation . . . . .	317
10.60 ta-ref/gp/random.c File Reference . . . . .	317
10.60.1 Function Documentation . . . . .	318
10.61 ta-ref/gp/secure_storage.c File Reference . . . . .	318
10.61.1 Macro Definition Documentation . . . . .	318
10.61.2 Function Documentation . . . . .	319
10.62 ta-ref/gp/symmetric_key.c File Reference . . . . .	319
10.62.1 Macro Definition Documentation . . . . .	320
10.62.2 Function Documentation . . . . .	320
10.63 ta-ref/gp/symmetric_key_gcm.c File Reference . . . . .	320
10.63.1 Macro Definition Documentation . . . . .	321
10.63.2 Function Documentation . . . . .	321
10.64 ta-ref/gp/time.c File Reference . . . . .	321
10.64.1 Function Documentation . . . . .	322
10.65 ta-ref/profiler/analyzer/analyzer.c File Reference . . . . .	322
10.65.1 Macro Definition Documentation . . . . .	323
10.65.2 Function Documentation . . . . .	323
10.66 ta-ref/profiler/analyzer/analyzer.h File Reference . . . . .	324
10.67 ta-ref/profiler/analyzer/nm_parse.c File Reference . . . . .	325
10.67.1 Macro Definition Documentation . . . . .	326
10.67.2 Function Documentation . . . . .	326
10.67.3 Variable Documentation . . . . .	328
10.68 ta-ref/profiler/analyzer/nm_parse.h File Reference . . . . .	328
10.68.1 Macro Definition Documentation . . . . .	329
10.68.2 Function Documentation . . . . .	329
10.69 ta-ref/profiler/analyzer/stack.h File Reference . . . . .	330
10.69.1 Macro Definition Documentation . . . . .	331
10.69.2 Function Documentation . . . . .	331
10.69.3 Variable Documentation . . . . .	332
10.70 ta-ref/profiler/app/tools.c File Reference . . . . .	332
10.70.1 Function Documentation . . . . .	332
10.71 ta-ref/profiler/keystone/Enclave/tools.c File Reference . . . . .	333
10.71.1 Function Documentation . . . . .	333
10.72 ta-ref/profiler/optee/Enclave/tools.c File Reference . . . . .	334
10.72.1 Function Documentation . . . . .	334
10.73 ta-ref/profiler/sgx/Enclave/tools.c File Reference . . . . .	335
10.73.1 Function Documentation . . . . .	335

10.74 ta-ref/test_gp/tools.c File Reference . . . . .	336
10.74.1 Function Documentation . . . . .	336
10.75 ta-ref/profiler/keystone/tee_config.h File Reference . . . . .	338
10.75.1 Function Documentation . . . . .	338
10.75.2 Variable Documentation . . . . .	338
10.76 ta-ref/profiler/optee/tee_config.h File Reference . . . . .	339
10.76.1 Macro Definition Documentation . . . . .	339
10.76.2 Function Documentation . . . . .	339
10.76.3 Variable Documentation . . . . .	340
10.77 ta-ref/profiler/sgx/tee_config.h File Reference . . . . .	340
10.77.1 Function Documentation . . . . .	340
10.77.2 Variable Documentation . . . . .	341
10.78 ta-ref/profiler/keystone/tee_profiler.c File Reference . . . . .	341
10.78.1 Function Documentation . . . . .	341
10.78.2 Variable Documentation . . . . .	343
10.79 ta-ref/profiler/optee/tee_profiler.c File Reference . . . . .	343
10.79.1 Function Documentation . . . . .	343
10.79.2 Variable Documentation . . . . .	344
10.80 ta-ref/profiler/sgx/tee_profiler.c File Reference . . . . .	344
10.80.1 Function Documentation . . . . .	345
10.80.2 Variable Documentation . . . . .	346
10.81 ta-ref/profiler/keystone/tee_profiler.h File Reference . . . . .	347
10.81.1 Function Documentation . . . . .	347
10.82 ta-ref/profiler/optee/tee_profiler.h File Reference . . . . .	348
10.82.1 Function Documentation . . . . .	348
10.83 ta-ref/profiler/sgx/tee_profiler.h File Reference . . . . .	348
10.83.1 Function Documentation . . . . .	349
10.84 ta-ref/profiler/profiler.c File Reference . . . . .	349
10.84.1 Function Documentation . . . . .	350
10.84.2 Variable Documentation . . . . .	352
10.85 ta-ref/profiler/profiler.h File Reference . . . . .	352
10.85.1 Function Documentation . . . . .	352
10.86 ta-ref/profiler/profiler_attrs.h File Reference . . . . .	353
10.86.1 Macro Definition Documentation . . . . .	353
10.87 ta-ref/profiler/profiler_data.h File Reference . . . . .	354
10.87.1 Macro Definition Documentation . . . . .	355
10.87.2 Typedef Documentation . . . . .	355
10.87.3 Enumeration Type Documentation . . . . .	355
10.87.4 Function Documentation . . . . .	356
10.87.5 Variable Documentation . . . . .	356
10.88 ta-ref/test_gp/crt.c File Reference . . . . .	356
10.88.1 Function Documentation . . . . .	357

10.88.2 Variable Documentation . . . . .	357
10.89 ta-ref/test_gp/optee/Enclave/crt.c File Reference . . . . .	358
10.89.1 Macro Definition Documentation . . . . .	359
10.89.2 Function Documentation . . . . .	359
10.89.3 Variable Documentation . . . . .	363
10.90 ta-ref/test_gp/include/crt.h File Reference . . . . .	363
10.90.1 Function Documentation . . . . .	364
10.91 ta-ref/test_gp/include/ocall_wrapper.h File Reference . . . . .	365
10.91.1 Function Documentation . . . . .	365
10.92 ta-ref/test_gp/include/random.h File Reference . . . . .	366
10.93 ta-ref/test_gp/include/tools.h File Reference . . . . .	367
10.93.1 Function Documentation . . . . .	367
10.94 ta-ref/test_gp/keystone/Enclave/ocall_wrapper.c File Reference . . . . .	368
10.94.1 Function Documentation . . . . .	369
10.95 ta-ref/test_gp/sgx/Enclave/ocall_wrapper.c File Reference . . . . .	369
10.95.1 Function Documentation . . . . .	370
10.96 ta-ref/test_gp/keystone/Enclave/startup.c File Reference . . . . .	370
10.96.1 Function Documentation . . . . .	370
10.97 ta-ref/test_gp/sgx/Enclave/startup.c File Reference . . . . .	371
10.97.1 Function Documentation . . . . .	371
10.98 ta-ref/test_hello/keystone/App/App.cpp File Reference . . . . .	372
10.98.1 Function Documentation . . . . .	372
10.98.2 Variable Documentation . . . . .	373
10.99 ta-ref/test_hello/sgx/App/App.cpp File Reference . . . . .	373
10.99.1 Macro Definition Documentation . . . . .	374
10.99.2 Function Documentation . . . . .	374
10.100 ta-ref/test_gp/keystone/App/App.cpp File Reference . . . . .	375
10.100.1 Function Documentation . . . . .	375
10.100.2 Variable Documentation . . . . .	376
10.101 ta-ref/test_gp/sgx/App/App.cpp File Reference . . . . .	376
10.101.1 Macro Definition Documentation . . . . .	377
10.101.2 Function Documentation . . . . .	377
10.102 ta-ref/test_hello/keystone/App/App_ocalls.cpp File Reference . . . . .	378
10.102.1 Function Documentation . . . . .	379
10.103 ta-ref/test_hello/sgx/App/App_ocalls.cpp File Reference . . . . .	382
10.103.1 Function Documentation . . . . .	383
10.104 ta-ref/test_gp/keystone/App/App_ocalls.cpp File Reference . . . . .	385
10.104.1 Macro Definition Documentation . . . . .	386
10.104.2 Function Documentation . . . . .	386
10.105 ta-ref/test_gp/sgx/App/App_ocalls.cpp File Reference . . . . .	390
10.105.1 Macro Definition Documentation . . . . .	390
10.105.2 Function Documentation . . . . .	391

10.106 ta-ref/test_hello/keystone/Enclave/Enclave.c File Reference	393
10.106.1 Macro Definition Documentation	394
10.106.2 Function Documentation	394
10.107 ta-ref/test_hello/optee/Enclave/Enclave.c File Reference	394
10.107.1 Macro Definition Documentation	395
10.107.2 Function Documentation	395
10.107.3 Variable Documentation	398
10.108 ta-ref/test_hello/sgx/Enclave/Enclave.c File Reference	399
10.108.1 Macro Definition Documentation	399
10.108.2 Function Documentation	399
10.109 ta-ref/test_gp/keystone/Enclave/Enclave.c File Reference	400
10.109.1 Function Documentation	400
10.110 ta-ref/test_gp/optee/Enclave/Enclave.c File Reference	401
10.110.1 Function Documentation	401
10.111 ta-ref/test_gp/sgx/Enclave/Enclave.c File Reference	402
10.111.1 Function Documentation	402
10.112 ta-ref/test_hello/optee/App/main.c File Reference	402
10.112.1 Macro Definition Documentation	403
10.112.2 Function Documentation	403
10.112.3 Variable Documentation	404
10.113 ta-ref/test_gp/optee/App/main.c File Reference	404
10.113.1 Macro Definition Documentation	405
10.113.2 Function Documentation	405
10.114 ta-ref/test_hello/optee/Enclave/user_ta_header.c File Reference	406
10.114.1 Macro Definition Documentation	406
10.114.2 Function Documentation	407
10.114.3 Variable Documentation	408
10.115 ta-ref/test_gp/optee/Enclave/user_ta_header.c File Reference	409
10.115.1 Macro Definition Documentation	410
10.115.2 Function Documentation	410
10.115.3 Variable Documentation	411
10.116 ta-ref/test_hello/optee/Enclave/user_ta_header_defines.h File Reference	412
10.116.1 Macro Definition Documentation	413
10.117 ta-ref/test_gp/optee/Enclave/user_ta_header_defines.h File Reference	414
10.117.1 Macro Definition Documentation	415
10.118 ta-ref/test_hello/sgx/App/App.h File Reference	416
10.118.1 Macro Definition Documentation	416
10.118.2 Variable Documentation	417
10.119 ta-ref/test_gp/sgx/App/App.h File Reference	417
10.119.1 Macro Definition Documentation	418
10.119.2 Variable Documentation	418
10.120 ta-ref/test_hello/sgx/App/App_ocalls.h File Reference	418

10.120.1 Typedef Documentation . . . . .	419
10.120.2 Function Documentation . . . . .	419
10.121 ta-ref/test_gp/sgx/App/App_ocalls.h File Reference . . . . .	425
10.121.1 Typedef Documentation . . . . .	427
10.121.2 Function Documentation . . . . .	427
10.122 ta-ref/test_hello/sgx/App/types.h File Reference . . . . .	434
10.122.1 Typedef Documentation . . . . .	435
10.122.2 Variable Documentation . . . . .	435
10.123 ta-ref/test_gp/sgx/App/types.h File Reference . . . . .	436
10.123.1 Typedef Documentation . . . . .	437
10.123.2 Variable Documentation . . . . .	437
<b>Index</b>	<b>439</b>

# 1 Preparation

## 1.1 Keystone(RISC-V Unleashed)

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

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

### 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 lzip 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` and it adds to your PATH.

### 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 check,

- <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 v3.9.0 for QEMU

Clone optee version 3.9.0 for QEMU

```
mkdir optee_3.9.0_qemu
cd optee_3.9.0_qemu
~/bin/repo init -u https://github.com/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 v3.9.0 for RPI3

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

```
#!/bin/bash -u
export OPTEE_VER=$1
export OPTEE_DIR=${PWD}/optee_${OPTEE_VER}-rpi3
mkdir ${OPTEE_DIR} || true
cd ${OPTEE_DIR}
~/bin/repo init -u https://github.com/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. For more details check,

- <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

Intall following packages on Ubuntu 18.04

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

### 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](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 Install Doxygen-1.9.2

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

### 2.2 Install Required Packages

Install following packages on Ubuntu 18.04

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

Above packages required to generate PDF using doxygen.

### 2.3 Build and Install

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

## 2.4 ta-ref with Keystone

Make sure Keystone and other dependant sources have been built

### 2.4.1 Cloning source and building

Install required packages

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

Setup Env

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

Clone and Build KEYEDGE

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

Clone and Build KEEDGER8R

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

Clone the source

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

Build

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

### 2.4.2 Check ta-ref by running test\_gp, test.hello, on QEMU

Copy the test.hello and test\_gp programs to QEMU.

#### 2.4.2.1 Launch QEMU Console

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

#### 2.4.2.2 test.hello

Run test.hello

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

### 2.4.2.3 test\_gp

#### Run test\_gp

```

cp test_gp/keystone/Enclave/Enclave.eapp.riscv $KEYSTONE_DIR/buildroot_overlay/root/test_gp/
cp test_gp/keystone/Enclave/App.client $KEYSTONE_DIR/buildroot_overlay/root/test_gp/
cp $KEYSTONE_SDK_DIR/rts/eyrie/eyrie-rt $KEYSTONE_DIR/buildroot_overlay/root/test_gp/
insmod keystone-driver.ko
./App.client Enclave.eapp.riscv eyrie-rt
main start
TEE.GenerateRandom(0x000000003FFFFEE0, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@random: 5ea8741bd8a3b298cf53d214eca693fb
TEE.GetREETime(): start
@[SE] gettimeofday 77 sec 865873 usec -> 0
@GP REE time 77 sec 865 millis
TEE.GetSystemTime(): start
@GP System time 100063195 sec 609 millis
TEE.CreatePersistentObject(): start
@[SE] open file FileOne flags 241 -> 3 (0)
TEE.WriteObjectData(): start
@[SE] write desc 3 buf 480d0 len 256-> 256
TEE.CloseObject(): start
@[SE] close desc 3 -> 0
TEE.OpenPersistentObject(): start
@[SE] open file FileOne flags 0 -> 3 (0)
TEE.ReadObjectData(): start
@[SE] read desc 3 buf fff41664 len 256-> 256
TEE.CloseObject(): start
@[SE] close desc 3 -> 0
256 bytes read:
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(0x000000003FFFFFFE28, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AsymmetricSignDigest(): start
TEE.FreeOperation(): start
@signature:
    d6e6b6e54db8b6a62fc1927886938bead27f4813f19ce77182e3016b5426bcad067ca98cd75f9dfddafe9eb0655c48df992d3ad674db69d831f26a
TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation:
TEE.FreeTransientObject(): start
verify ok
main end

```

## 2.5 ta-ref with OPTEE

Make sure optee\_3.9.0\_rpi3 has been built already.

### 2.5.1 Cloning source and building

#### Clone the source

```

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

```

#### Build

```

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

```

### 2.5.2 Check ta-ref by running test\_gp, test\_hello, on QEMU

#### Copy the test.hello and test\_gp programs to QEMU buildroot directory

```

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

```

#### 2.5.2.1 test.hello

##### Run test\_hello

```

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

```

If executed successfully, you see above messages

### 2.5.2.2 test\_gp

#### Run test\_gp

```
cd /home/gitlab/out/test_gp/
cp a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta /home/gitlab/out/
ln -s /home/gitlab/out/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
    /lib64/optee.armtz/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
./optee.ref.ta
start TEEC.InvokeCommand
--- enclave log start---
ecall.ta_main() start
@random: 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
@@TEE.FreeOperation:
verify ok
ecall.ta_main() end
--- enclave log end---
res = TEEC_SUCCESS; TEEC.InvokeCommand succeeded!
```

If executed successfully, you see above messages

## 2.6 ta-ref with SGX

Build ta-ref for Intel SGX platforms

### 2.6.1 Cloning source and building

#### Clone the source

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

#### Build

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

### 2.6.2 Check ta-ref by running test\_gp, test.hello, simulation mode on any pc

Copy the ta-ref's test.hello & test\_gp executables to test directory



### 2.6.2.1 test\_hello

#### Run test\_hello

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

### 2.6.2.2 test\_gp

#### Run test\_gp

```
cp test_gp/sgx/Enclave/enclave.signed.so <test directory>
cp test_gp/sgx/App/sgx_app <test directory>
<test directory>/sgx_app
main start
TEE.GenerateRandom(): start
@random: f35c1d1e4bbf6641c5511c9dc5aaf638
TEE.GetREETime(): start
request to get unix time 1612257364, 199
@GP REE time 1612257364 sec 199 millis
TEE.GetSystemTime(): start
@GP System time 727941859 sec 984 millis
TEE.CreatePersistentObject(): start
request to open FileOne flags 241 -> 3
TEE.WriteObjectData(): start
request to write 256 bytes to descriptor 3
TEE.CloseObject(): start
request to close descriptor 3
TEE.OpenPersistentObject(): start
request to open FileOne flags 0 -> 3
TEE.ReadObjectData(): start
request to read 256 bytes from descriptor 3
TEE.CloseObject(): start
request to close descriptor 3
256 bytes read:
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:
    100b392ce043e9b8dc703088f505dd3083ec47bfc8d59d968a66b54e80464d684d56dc9c44336f08fd9630979863a2d8fb7cd672a819ef609357e
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{KMGT}:
Last sector (2048-15523806, default = 15523806) or {+-}size{KMGT}: 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{KMGT}:
Last sector (67584-15523806, default = 15523806) or {+-}size{KMGT}: 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{KMGT}: 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-Unleased board

```

$ git clone https://github.com/keystone-enclave/freedom-u540-c000-bootloader.git
$ cd freedom-u540-c000-bootloader
$ git checkout -b dev-unleased 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 Unleased

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

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

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

test\_hello:

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

test\_gp:

```
insmod keystone-driver.ko
./App.client Enclave.eapp.riscv eyrie-rt
main start
TEE.GenerateRandom(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:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
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:
e94431cd22a6029185d0dbb1a17b5d62843bfeef25591583d2d668ec6fed1c692f88ce4754d690c346c8d9f2726630e0386abf4e45699a2ca2b3ba
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
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/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
@@TEE.FreeOperation:
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
```

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

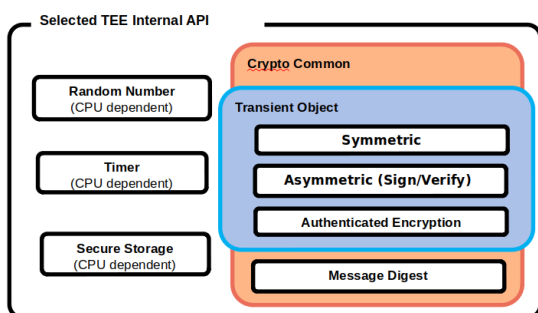
Category	CPU (In)Dependent	Functions
Random Number	Dependent	TEE_GenerateRandom
Time	Dependent	TEE_GetREETime, TEE_GetSystemTime
Secure Storage	Dependent	TEE_CreatePersistentObject, TEE_OpenPersistentObject, TEE_ReadObjectData, TEE_WriteObjectData, TEE_CloseObject
Transient Object	Dependent Independent	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_ECDSA_CURVE,
    TEE_ECDSA_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);
}
--- end ---
```

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

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

### 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 of GP API

### 6.1 GP API

#### API Functions by Category

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

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

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



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:

<a href="#">__profiler_data</a>	35
<a href="#">__profiler_header</a>	36
<a href="#">__TEE_ObjectHandle</a>	36
<a href="#">__TEE_OperationHandle</a>	37
<a href="#">_sgx_errlist_t</a>	39
<a href="#">addrinfo</a>	39
<a href="#">enclave_report</a>	41
<a href="#">invoke_command_param_t</a>	41
<a href="#">invoke_command_t</a>	42
<a href="#">list</a>	43
<a href="#">nm_info</a>	44
<a href="#">ob16_t</a>	44
<a href="#">ob196_t</a>	45
<a href="#">ob256_t</a>	45
<a href="#">out_fct_wrap_type</a>	46
<a href="#">param_buffer_t</a>	46
<a href="#">pollfd</a>	47
<a href="#">ree_time_t</a>	48
<a href="#">report</a>	48
<a href="#">result</a>	49
<a href="#">sm_report</a>	50
<a href="#">TEE_Attribute</a>	51
<a href="#">TEE_Identity</a>	52
<a href="#">TEE_ObjectInfo</a>	53
<a href="#">TEE_OperationInfo</a>	54
<a href="#">TEE_OperationInfoKey</a>	56
<a href="#">TEE_OperationInfoMultiple</a>	56
<a href="#">TEE_Param</a>	58

<a href="#">TEE_SEAID</a>	59
<a href="#">TEE_SEReaderProperties</a>	59
<a href="#">TEE_Time</a>	60
<a href="#">TEE_UUID</a>	60
<a href="#">TEEC_Context</a>	61
<a href="#">TEEC_Operation</a>	62
<a href="#">TEEC_Parameter</a>	63
<a href="#">TEEC_RegisteredMemoryReference</a>	64
<a href="#">TEEC_Session</a>	66
<a href="#">TEEC_SharedMemory</a>	66
<a href="#">TEEC_TempMemoryReference</a>	68
<a href="#">TEEC_UUID</a>	69
<a href="#">TEEC_Value</a>	70

## 8 File Index

### 8.1 File List

Here is a list of all files with brief descriptions:

<a href="#">ta-ref/api/tee-internal-api-cryptlib.c</a>	264
<a href="#">ta-ref/api/include/compiler.h</a>	71
<a href="#">ta-ref/api/include/report.h</a>	78
<a href="#">ta-ref/api/include/tee-common.h</a> Common type and definitions of RISC-V TEE	79
<a href="#">ta-ref/api/include/tee-ta-internal.h</a> Candidate API list for Global Platform like RISC-V TEE	80
<a href="#">ta-ref/api/include/tee_api.h</a>	104
<a href="#">ta-ref/api/include/tee_api_defines.h</a>	140
<a href="#">ta-ref/api/include/tee_api_defines_extensions.h</a>	178
<a href="#">ta-ref/api/include/tee_api_types.h</a>	182
<a href="#">ta-ref/api/include/tee_client_api.h</a>	186
<a href="#">ta-ref/api/include/tee_internal_api.h</a>	198
<a href="#">ta-ref/api/include/tee_internal_api_extensions.h</a>	199
<a href="#">ta-ref/api/include/tee_ta_api.h</a>	202

ta-ref/api/include/test_dev_key.h	204
ta-ref/api/include/trace.h	205
ta-ref/api/include/trace_levels.h	210
ta-ref/api/keystone/tee-internal-api-machine.c	211
ta-ref/api/keystone/tee-internal-api.c	212
ta-ref/api/keystone/tee_api_tee_types.h	232
ta-ref/api/keystone/teec_stub.c	236
ta-ref/api/keystone/trace.c	239
ta-ref/api/keystone/vsnprintf.c	245
ta-ref/api/optee/tee_api_tee_types.h	234
ta-ref/api/sgx/tee-internal-api.c	223
ta-ref/api/sgx/tee_api_tee_types.h	234
ta-ref/benchmark/bench.c	278
ta-ref/benchmark/bench.h	282
ta-ref/benchmark/cpu_bench.c	285
ta-ref/benchmark/io_bench.c	288
ta-ref/benchmark/memory_bench.c	294
ta-ref/benchmark/time_test.c	295
ta-ref/benchmark/include/config_bench.h	286
ta-ref/benchmark/keystone/tee_def.h	290
ta-ref/benchmark/optee/tee_def.h	291
ta-ref/benchmark/sgx/tee_def.h	292
ta-ref/edger/edger8r/user_types.h	296
ta-ref/edger/keyedge/Enclave.t.c	297
ta-ref/edger/keyedge/Enclave.t.h	298
ta-ref/edger/keyedge/Enclave.u.c	299
ta-ref/edger/keyedge/Enclave.u.h	300
ta-ref/edger/keyedge/ocalls.h	301
ta-ref/edger/optee/Enclave.h	306
ta-ref/edger/optee/Enclave.t.h	299
ta-ref/gp/asymmetric_key.c	309
ta-ref/gp/invoke_command.c	315

<a href="#">ta-ref/gp/message_digest.c</a>	316
<a href="#">ta-ref/gp/random.c</a>	317
<a href="#">ta-ref/gp/secure_stoage.c</a>	318
<a href="#">ta-ref/gp/symmetric_key.c</a>	319
<a href="#">ta-ref/gp/symmetric_key_gcm.c</a>	320
<a href="#">ta-ref/gp/time.c</a>	321
<a href="#">ta-ref/gp/include/config_ref_ta.h</a>	310
<a href="#">ta-ref/gp/include/gp_test.h</a>	312
<a href="#">ta-ref/profiler/profiler.c</a>	349
<a href="#">ta-ref/profiler/profiler.h</a>	352
<a href="#">ta-ref/profiler/profiler_attrs.h</a>	353
<a href="#">ta-ref/profiler/profiler_data.h</a>	354
<a href="#">ta-ref/profiler/analyzer/analyzer.c</a>	322
<a href="#">ta-ref/profiler/analyzer/analyzer.h</a>	324
<a href="#">ta-ref/profiler/analyzer/nm_parse.c</a>	325
<a href="#">ta-ref/profiler/analyzer/nm_parse.h</a>	328
<a href="#">ta-ref/profiler/analyzer/stack.h</a>	330
<a href="#">ta-ref/profiler/app/tools.c</a>	332
<a href="#">ta-ref/profiler/keystone/tee_config.h</a>	338
<a href="#">ta-ref/profiler/keystone/tee_profiler.c</a>	341
<a href="#">ta-ref/profiler/keystone/tee_profiler.h</a>	347
<a href="#">ta-ref/profiler/keystone/Enclave/tools.c</a>	333
<a href="#">ta-ref/profiler/optee/tee_config.h</a>	339
<a href="#">ta-ref/profiler/optee/tee_profiler.c</a>	343
<a href="#">ta-ref/profiler/optee/tee_profiler.h</a>	348
<a href="#">ta-ref/profiler/optee/Enclave/tools.c</a>	334
<a href="#">ta-ref/profiler/sgx/tee_config.h</a>	340
<a href="#">ta-ref/profiler/sgx/tee_profiler.c</a>	344
<a href="#">ta-ref/profiler/sgx/tee_profiler.h</a>	348
<a href="#">ta-ref/profiler/sgx/Enclave/tools.c</a>	335
<a href="#">ta-ref/test_gp/crt.c</a>	356
<a href="#">ta-ref/test_gp/tools.c</a>	336

ta-ref/test_gp/vsnprintf.c	251
ta-ref/test_gp/include/crt.h	363
ta-ref/test_gp/include/ocall_wrapper.h	365
ta-ref/test_gp/include/random.h	366
ta-ref/test_gp/include/tools.h	367
ta-ref/test_gp/keystone/App/App.cpp	375
ta-ref/test_gp/keystone/App/App_ocalls.cpp	385
ta-ref/test_gp/keystone/Enclave/Enclave.c	400
ta-ref/test_gp/keystone/Enclave/ocall_wrapper.c	368
ta-ref/test_gp/keystone/Enclave/startup.c	370
ta-ref/test_gp/keystone/Enclave/trace.c	241
ta-ref/test_gp/optee/App/main.c	404
ta-ref/test_gp/optee/Enclave/crt.c	358
ta-ref/test_gp/optee/Enclave/Enclave.c	401
ta-ref/test_gp/optee/Enclave/trace.c	242
ta-ref/test_gp/optee/Enclave/user_ta_header.c	409
ta-ref/test_gp/optee/Enclave/user_ta_header_defines.h	414
ta-ref/test_gp/sgx/App/App.cpp	376
ta-ref/test_gp/sgx/App/App.h	417
ta-ref/test_gp/sgx/App/App_ocalls.cpp	390
ta-ref/test_gp/sgx/App/App_ocalls.h	425
ta-ref/test_gp/sgx/App/types.h	436
ta-ref/test_gp/sgx/Enclave/Enclave.c	402
ta-ref/test_gp/sgx/Enclave/Enclave.h	307
ta-ref/test_gp/sgx/Enclave/ocall_wrapper.c	369
ta-ref/test_gp/sgx/Enclave/startup.c	371
ta-ref/test_gp/sgx/Enclave/trace.c	244
ta-ref/test_hello/keystone/App/App.cpp	372
ta-ref/test_hello/keystone/App/App_ocalls.cpp	378
ta-ref/test_hello/keystone/Enclave/Enclave.c	393
ta-ref/test_hello/optee/App/main.c	402
ta-ref/test_hello/optee/Enclave/Enclave.c	394

<a href="#">ta-ref/test_hello/optee/Enclave/user_ta_header.c</a>	406
<a href="#">ta-ref/test_hello/optee/Enclave/user_ta_header_defines.h</a>	412
<a href="#">ta-ref/test_hello/sgx/App/App.cpp</a>	373
<a href="#">ta-ref/test_hello/sgx/App/App.h</a>	416
<a href="#">ta-ref/test_hello/sgx/App/App_ocalls.cpp</a>	382
<a href="#">ta-ref/test_hello/sgx/App/App_ocalls.h</a>	418
<a href="#">ta-ref/test_hello/sgx/App/types.h</a>	434
<a href="#">ta-ref/test_hello/sgx/Enclave/Enclave.c</a>	399

## 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

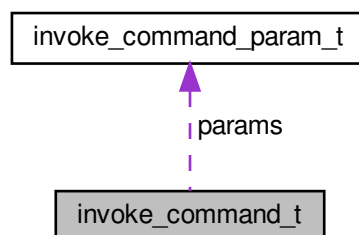
The documentation for this struct was generated from the following file:

- [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`

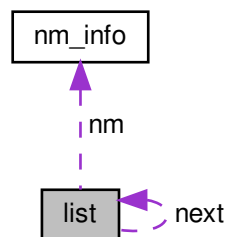
The documentation for this struct was generated from the following file:

- [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

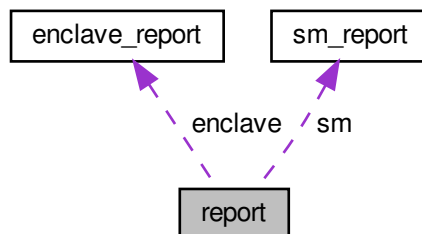
The documentation for this struct was generated from the following files:

- 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* TEE_Attribute::buffer`

**9.22.1.5** `union { ... } TEE_Attribute::content`

**9.22.1.6 length** `uint32_t TEE_Attribute::length`

**9.22.1.7** `struct { ... } TEE_Attribute::ref`

**9.22.1.8** `struct { ... } TEE_Attribute::value`

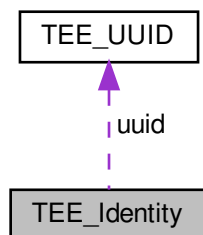
The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee\\_api\\_types.h](#)

## 9.23 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`

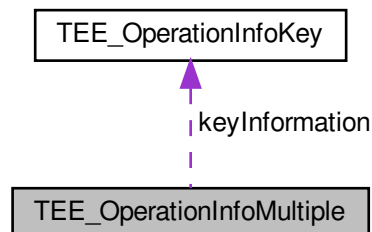
The documentation for this struct was generated from the following file:

- 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`

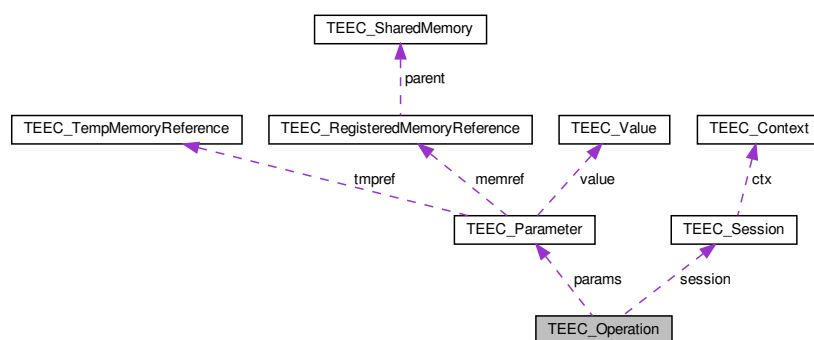
The documentation for this struct was generated from the following file:

- [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 <a href="#">TEEC_Parameter</a> .
<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`

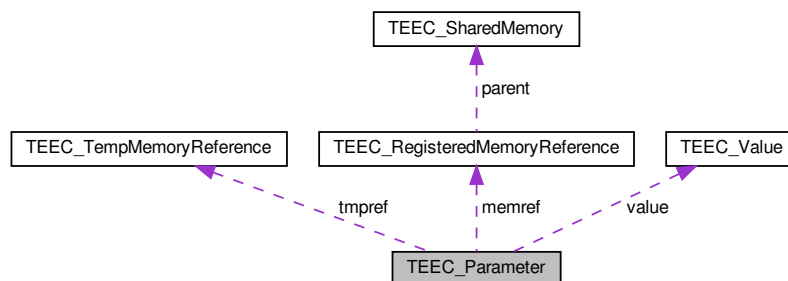
The documentation for this struct was generated from the following file:

- `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`

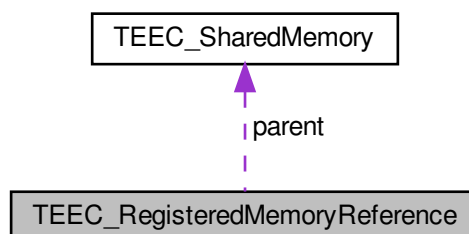
The documentation for this union was generated from the following file:

- [ta-ref/api/include/tee\\_client\\_api.h](#)

**9.36 TEEC\_RegisterMemoryReference Struct Reference**

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

Collaboration diagram for TEEC\_RegisterMemoryReference:



## Public Attributes

- [TEEC\\_SharedMemory](#) \* [parent](#)
- [size\\_t](#) [size](#)
- [size\\_t](#) [offset](#)

### 9.36.1 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`

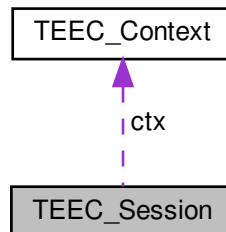
The documentation for this struct was generated from the following file:

- `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 TEEC_MEM_INPUT and TEEC_MEM_OUTPUT flags.

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

**9.38.2 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](#)



- `#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)))`

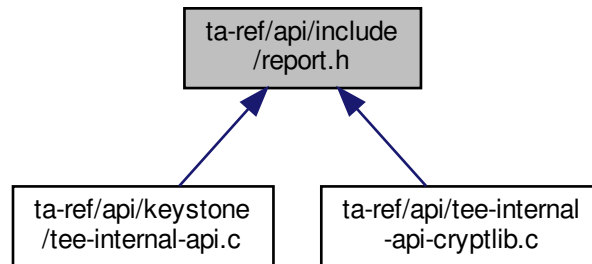
**10.1.1.46** `__unused` `#define __unused __attribute__((unused))`

**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`

**10.2.1.3 PUBLIC\_KEY\_SIZE** `#define PUBLIC_KEY_SIZE 32`

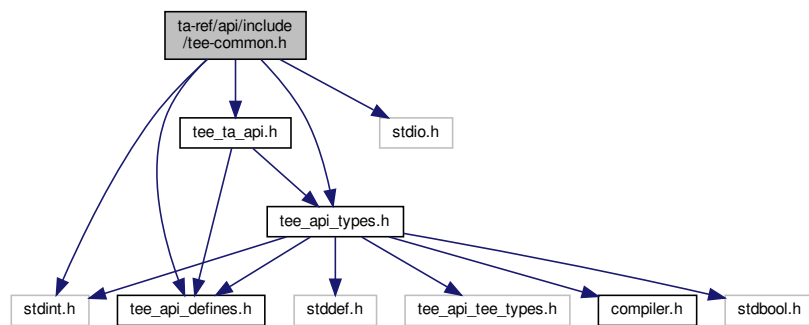
**10.2.1.4 SIGNATURE\_SIZE** `#define SIGNATURE_SIZE 64`

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

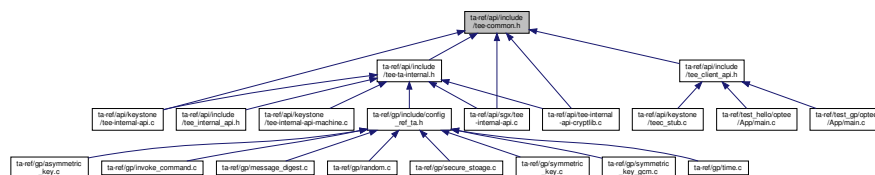
Common type and definitions of RISC-V TEE.

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

Include dependency graph for tee-common.h:



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



## Macros

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



## 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__((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

<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` TEEAllocateOperation (   
`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` TEEAllocateTransientObject (   
`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 success

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\\_getrandom16](#) and [ocall\\_getrandom196](#)

## 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-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

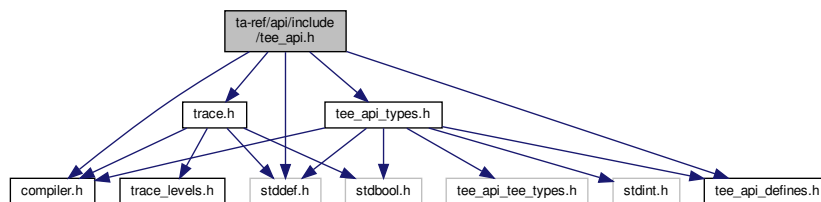
## Returns

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-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.

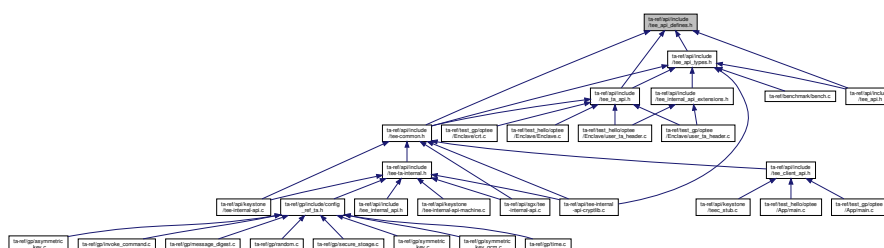
<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write

## 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)0xFFFFFFFF
- #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 0x00001B01

**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`

**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 0x00000000

**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`
- 10.6.1.248 TEE\_PANIC\_ID\_TEE\_GETOPERATIONINFOMULTIPLE** `#define TEE_PANIC_ID_TEE_GETOPERATIONINFOMULTIPLE 0x00000C08`
- 10.6.1.249 TEE\_PANIC\_ID\_TEE\_GETPROPERTYASBINARYBLOCK** `#define TEE_PANIC_ID_TEE_GETPROPERTYASBINARYBLOCK 0x00000C09`
- 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`
- 
- Copyright © National Institute of Advanced Industrial Science and Technology (AIST)

- 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 0x00000↔  
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

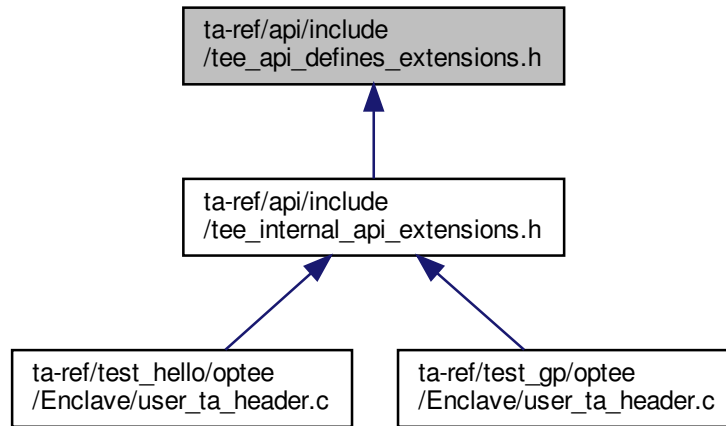
**10.6.1.336 TEE\_USAGE\_MAC** #define TEE\_USAGE\_MAC 0x00000008

**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

```

graph TD
    A["ta-ref/api/include/tee_api_types.h"] --> B["compiler.h"]
    A --> C["stdint.h"]
    A --> D["stdbool.h"]
    A --> E["stddef.h"]
    A --> F["tee_api_defines.h"]
    A --> G["tee_api_tee_types.h"]
  
```

[illegible]

- 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 nfds_t`
- `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](#)

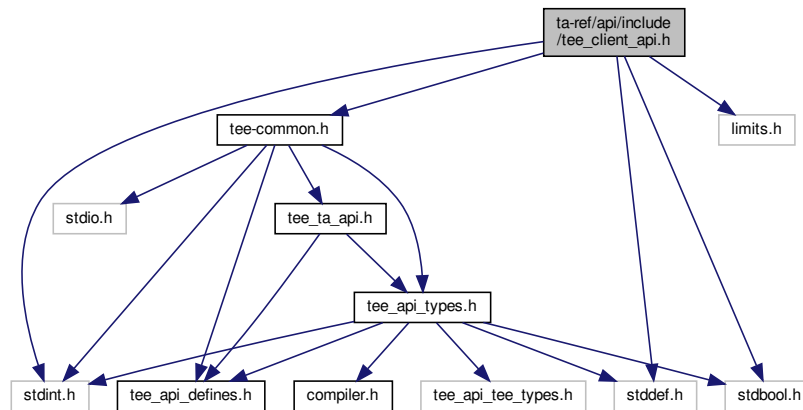
Enumerator

TEE_DATA_SEEK_SET	
TEE_DATA_SEEK_CUR	
TEE_DATA_SEEK_END	

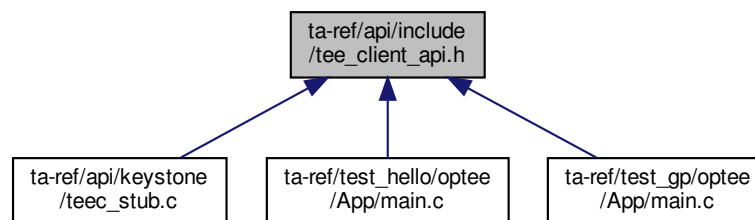
## 10.9 ta-ref/api/include/tee\_client\_api.h File Reference

```
#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.

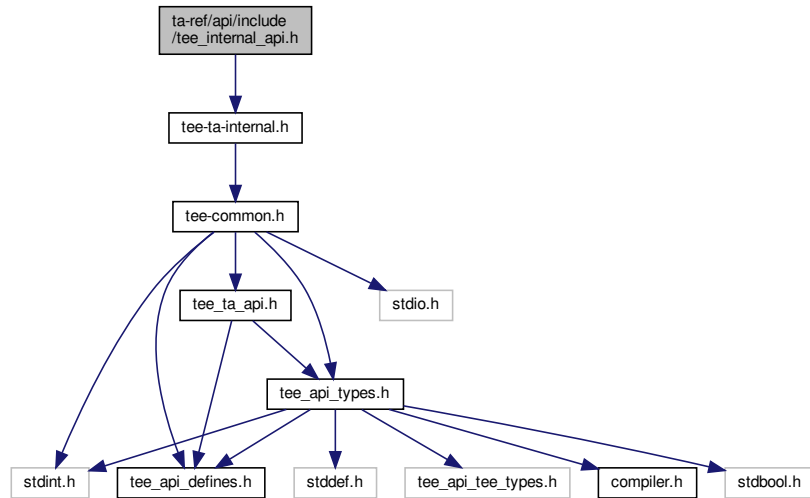
**Parameters**

<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:



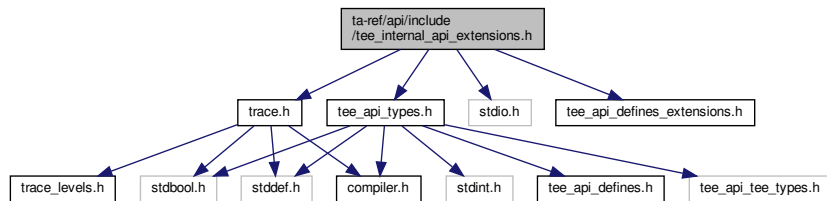
## 10.11 ta-ref/api/include/tee\_internal\_api\_extensions.h File Reference

```

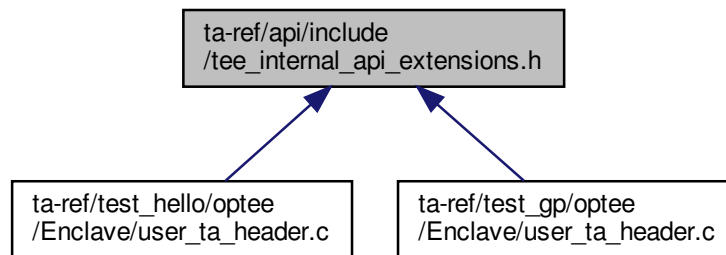
#include <trace.h>
#include <stdio.h>
#include <tee_api_defines_extensions.h>
#include <tee_api_types.h>

```

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 )`

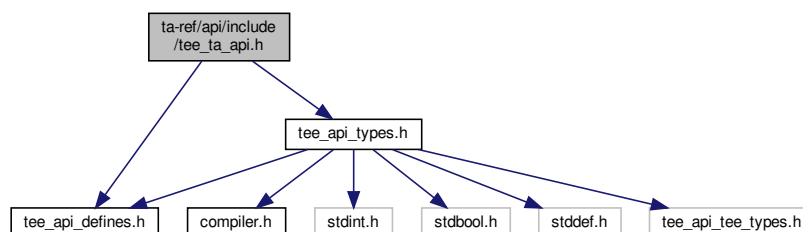
**10.11.2.6 tee\_user\_mem\_check\_heap()** `size_t tee_user_mem_check_heap (`  
    `void )`

**10.11.2.7 tee\_user\_mem\_mark\_heap()** `void tee_user_mem_mark_heap (`  
    `void )`

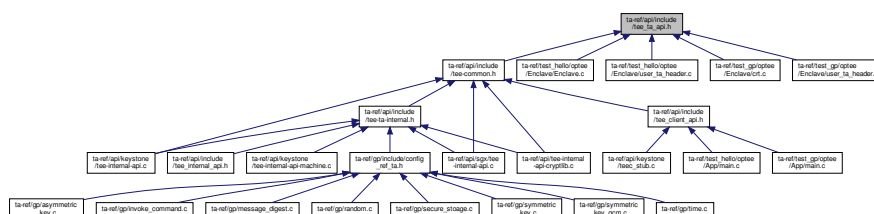
**10.11.2.8 tee\_uuid\_from\_str()** `TEE_Result tee_uuid_from_str (`  
    `TEE_UUID * uuid,`  
    `const char * s )`

## 10.12 ta-ref/api/include/tee\_ta\_api.h File Reference

```
#include <tee_api_defines.h>
#include <tee_api_types.h>
Include dependency graph for tee_ta_api.h:
```



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



### Macros

- `#define TA_EXPORT`

### Functions

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

#### 10.12.1 Macro Definition Documentation

##### 10.12.1.1 TA\_EXPORT #define TA\_EXPORT

## 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
}

```

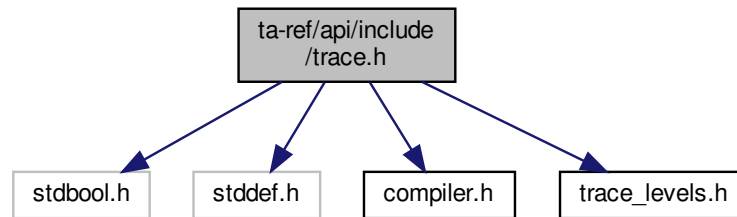
**10.13.1.4 `_sanctum_dev_secret_key_len`** const size\_t `_sanctum_dev_secret_key_len` = 64 [static]



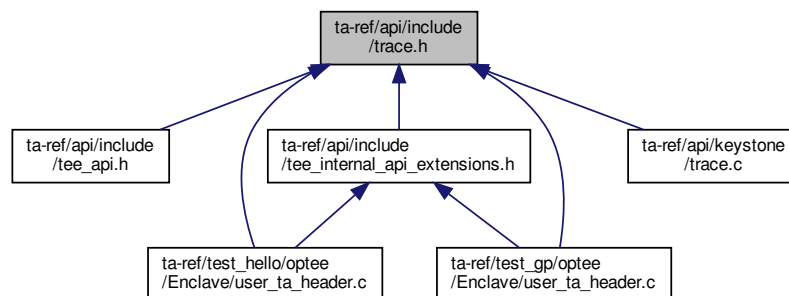
## 10.14 ta-ref/api/include/trace.h File Reference

```
#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 )`

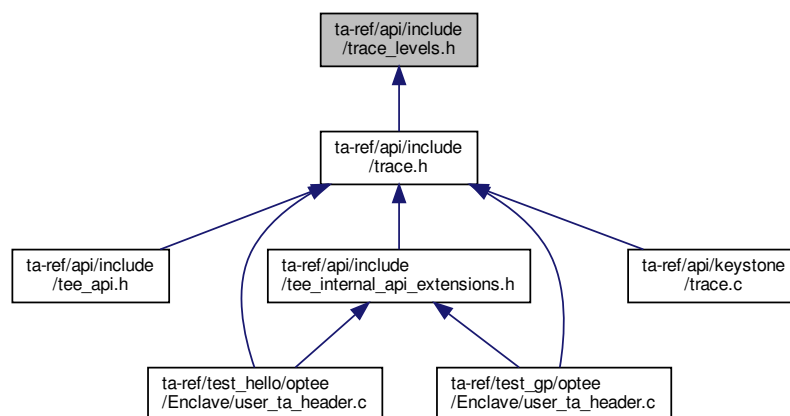
### 10.14.3 Variable Documentation

**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`

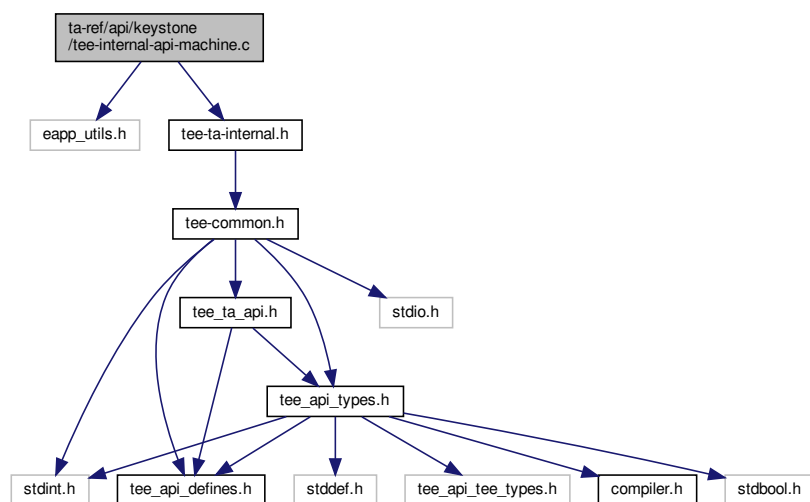
**10.15.1.7 TRACE\_PRINTF\_LEVEL** `#define TRACE_PRINTF_LEVEL TRACE_ERROR`

## 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

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

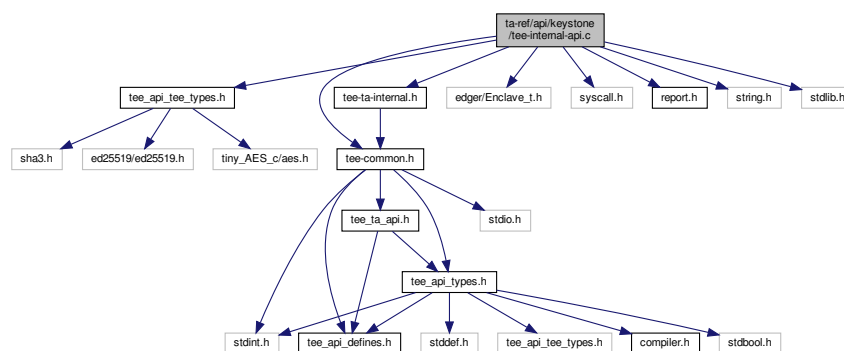
#### Returns

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

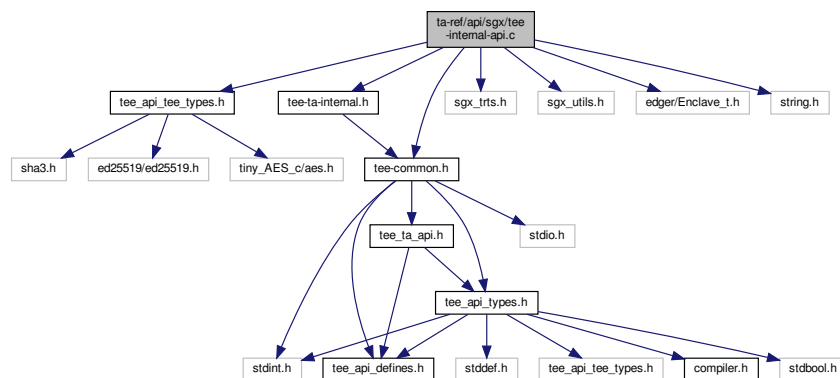
## Returns

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

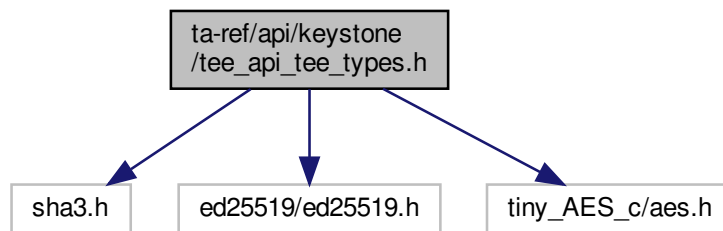
<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write

## Returns

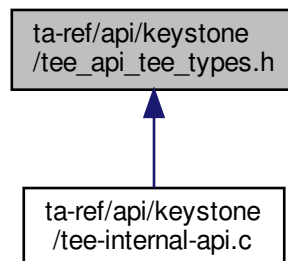
TEE\_SUCCESS if success else error occurred.

## 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

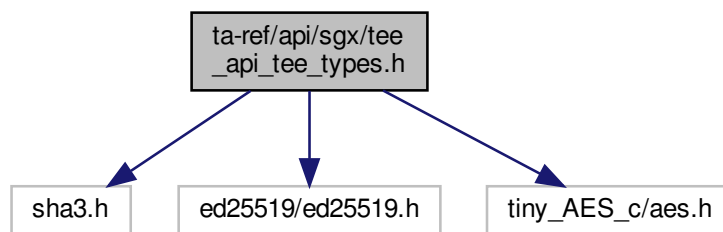
### 10.19.1.9 WOLFCRYPT `#define WOLFCRYPT 2`

## 10.20 ta-ref/api/optee/tee\_api\_tee\_types.h File Reference

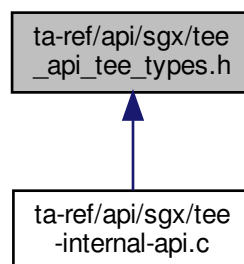
## 10.21 ta-ref/api/sgx/tee\_api\_tee\_types.h File Reference

```
#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`

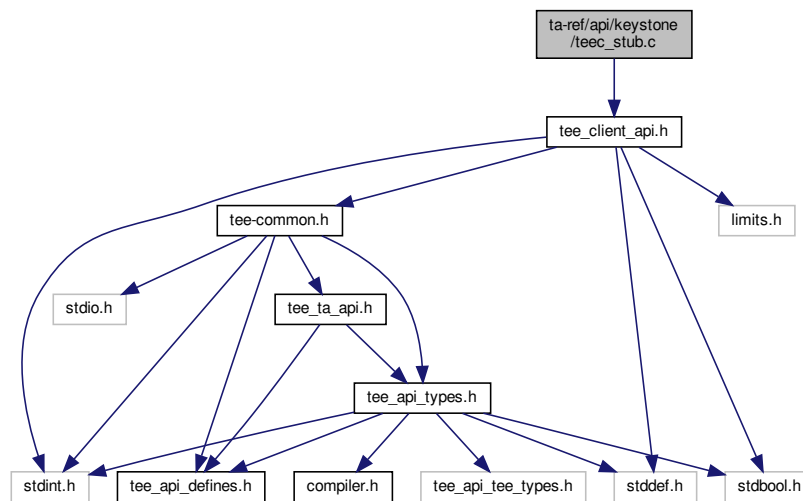
**10.21.1.9 TEE\_OBJECT\_TAG\_SIZE** `#define TEE_OBJECT_TAG_SIZE 16`

**10.21.1.10 WOLFCRYPT** `#define WOLFCRYPT 2`

## 10.22 ta-ref/api/keystone/tee\_client\_api.h File Reference

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

Include dependency graph for tee\_client\_api.h:



### 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

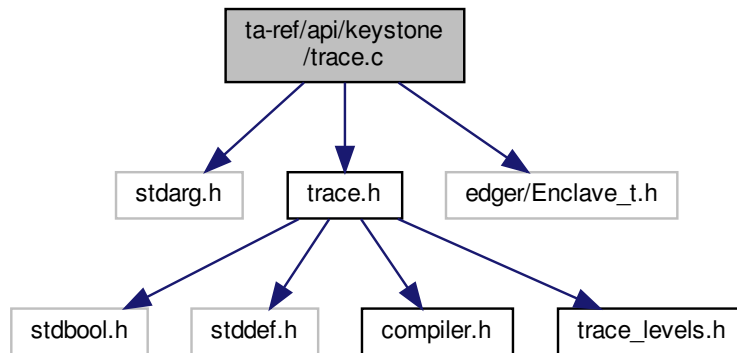
<i>operation</i>	Pointer to an operation previously passed to open session or invoke.
------------------	--

## 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.

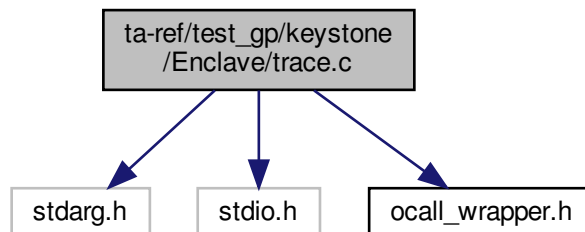
**10.24 ta-ref/test\_gp/keystone/Enclave/trace.c File Reference**

```

#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

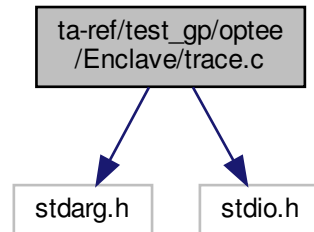
res Based on the condition check it will return string length else returns 0.

## 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**

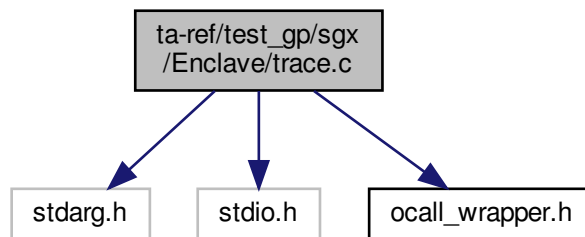
<i>fmt</i>	fmt is constant character argument of type pointer.
------------	---

**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.
------------	---

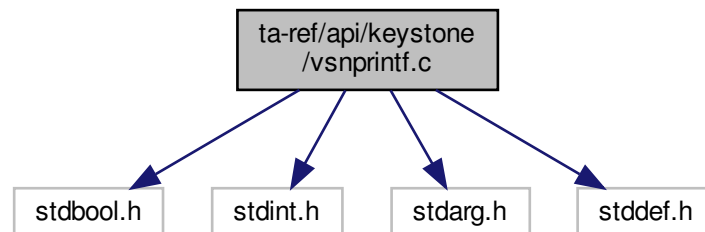
## Returns

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,`  
    `... )`

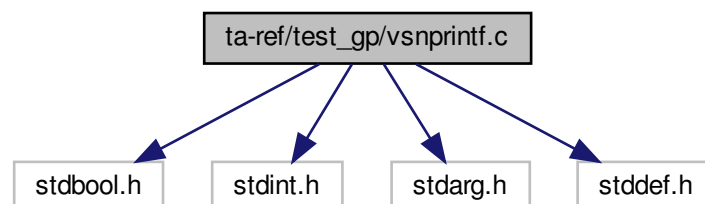
**10.27.3.16 sprintf()** `int sprintf (`  
    `char * buffer,`  
    `const char * format,`  
    `... )`

**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 `vsprintf` (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 precision 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 length. 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>	<i>str</i> 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,`  
`... )`



`fprintf()` - 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 `_vsnprintf` 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.

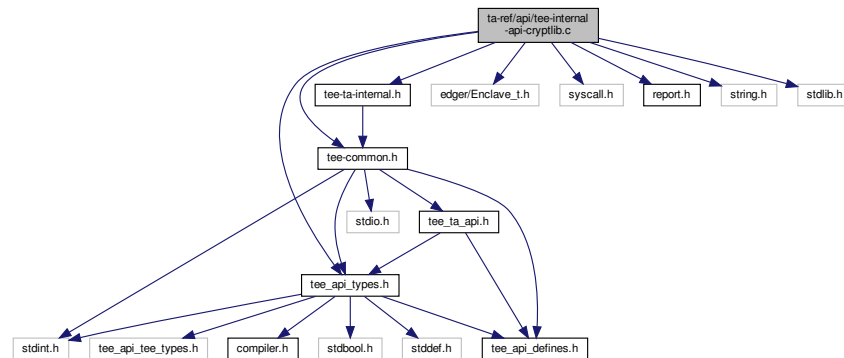
**Returns**

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 <i>params</i> 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.

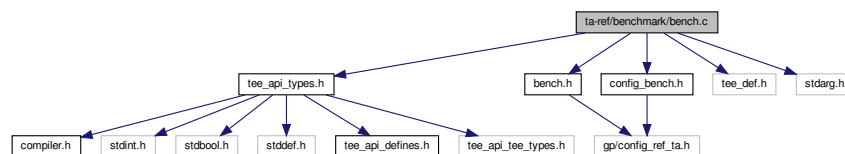
## Parameters

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

**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 statments, 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 <a href="#">TEE_Time</a>
<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 <a href="#">TEE_Time</a> .
-------------	---

**Returns**

It will return time value as a milliseconds.

**10.30.2 Variable Documentation**

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

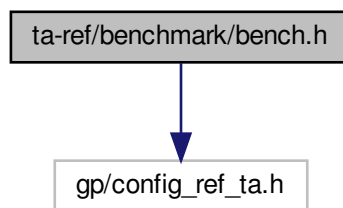
**Initial value:**

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

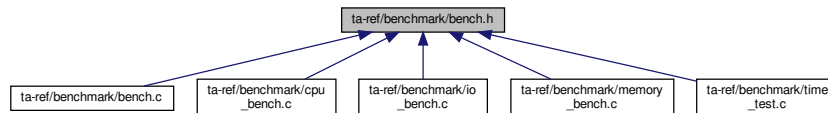
### 10.31 ta-ref/benchmark/bench.h File Reference

```
#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.

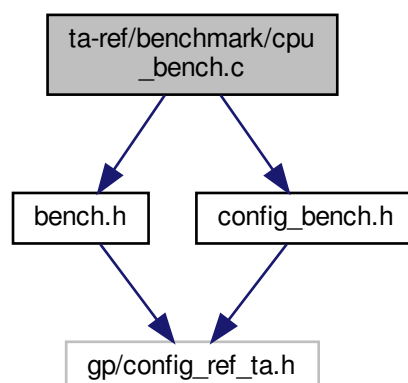
**10.31.2.8 system\_time\_test()** void NO\_PERF system\_time\_test (   
void )

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 )

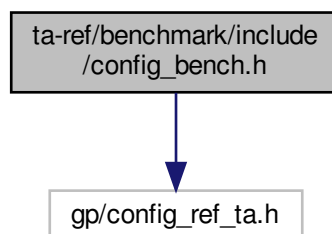
`cpu_int_benchmark()` - TO check the processing of cpu integer benchmark

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.

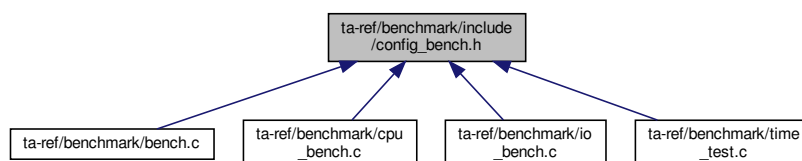
### 10.33 ta-ref/benchmark/include/config\_bench.h File Reference

```
#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 <a href="#">TEE_Time</a> .
<i>end</i>	The pointer type argument of <a href="#">TEE_Time</a> .
<i>size</i>	The maximum size to be recorded.
<i>unit</i>	The integer type argument of memory benchmark.

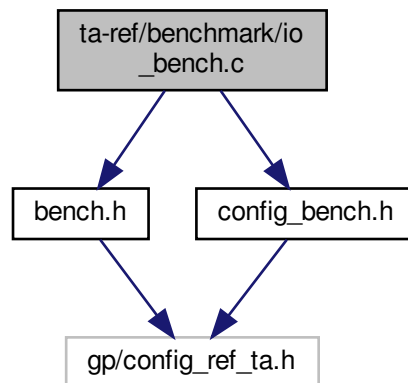
## 10.34 ta-ref/benchmark/io\_bench.c File Reference

```

#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.

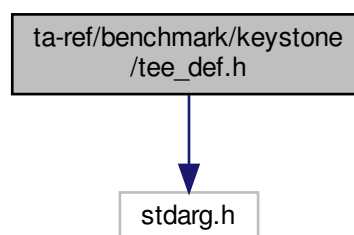
## Parameters

<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

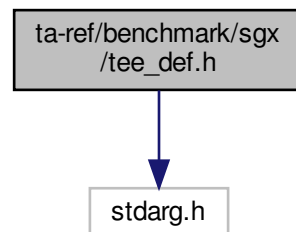
#### 10.36.3.1 `buf` `char buf[BUF_SIZE] [static]`

#### 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

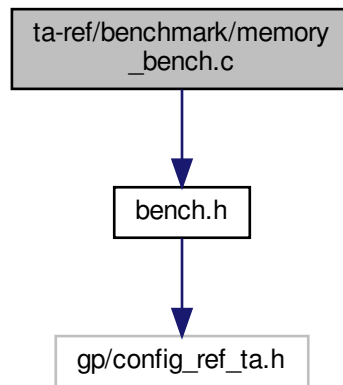
**10.37.2.1 buf** char `buf`[`BUF_SIZE`] [static]

**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.

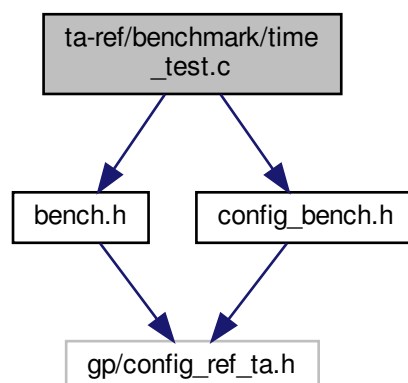
This function invokes a for\_loop for memory write, it iterates upto size -1. Then assigns typecasting character value of "i&255" into "buf[idx]" For read memory another for\_loop is initiated with same condition, Here "sum" is incremented by value of "buf[i]"

## 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

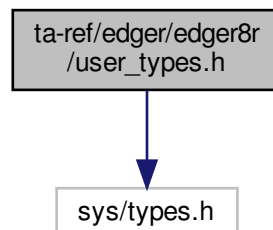
## 10.44 ta-ref/docs/preparation.md File Reference

## 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]

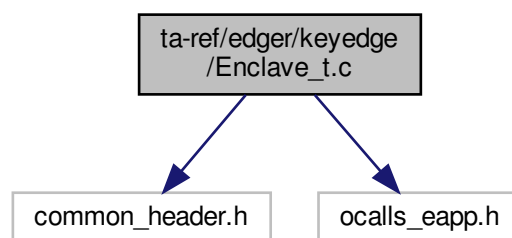
**10.46.2.2 `buffer_t`** typedef void\* `buffer_t`

## 10.47 ta-ref/edger/keyedge/Enclave\_t.c File Reference

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

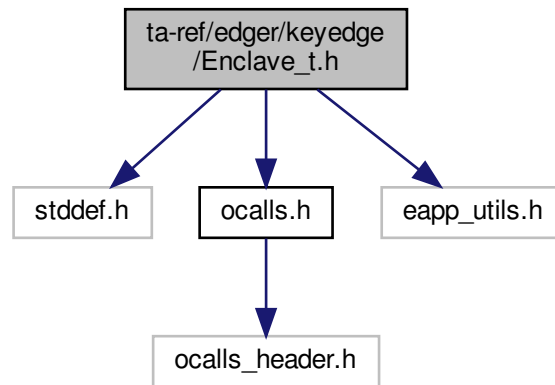
```
#include "ocalls_eapp.h"
```

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 )`

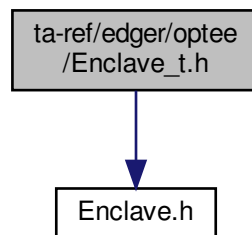
**10.48.1.3 ocall\_file.write()** `int ocall_file_write (`  
    `int fd,`  
    `const void * buf,`  
    `size_t count )`

**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"
```

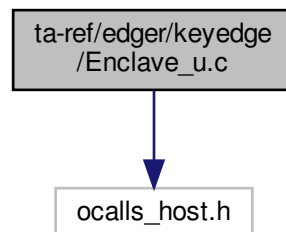
Include dependency graph for Enclave.t.h:



## 10.50 ta-ref/edger/keyedge/Enclave\_u.c File Reference

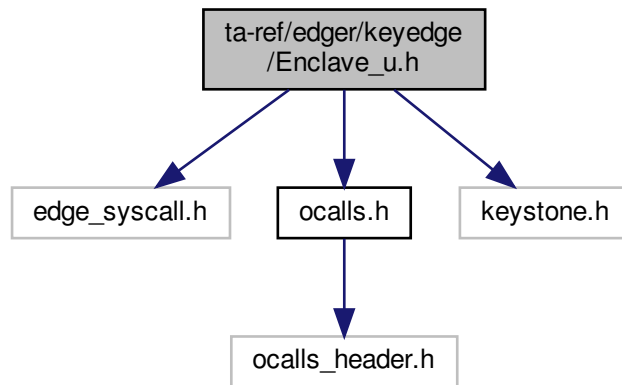
```
#include "ocalls_host.h"
```

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



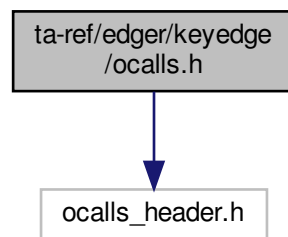
**10.51.2.1** `__wrapper_ocall_close_file()` `void __wrapper_ocall_close_file (`  
`void * buffer )`

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

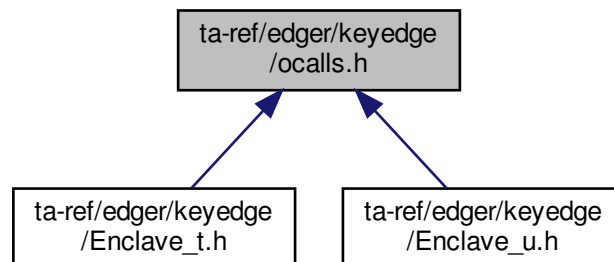
## 10.52 ta-ref/edger/keyedge/ocalls.h File Reference

```
#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 )`

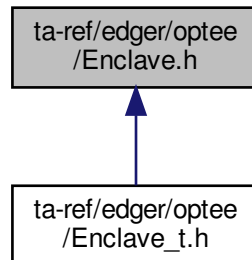
**10.52.3.12 ocall\_unlink()** `int ocall_unlink (`  
    `keyedge_str const char * path )`

**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`

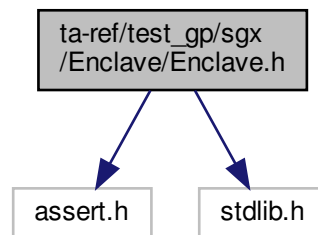
**10.53.1.2 TA\_REF\_UUID** `#define TA_REF_UUID { 0xa6f77c1e, 0x96fe, 0x4a0e, { 0x9e, 0x74, 0x26, 0x25, 0x82, 0xa4, 0xc8, 0xf1 }}`

## 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.

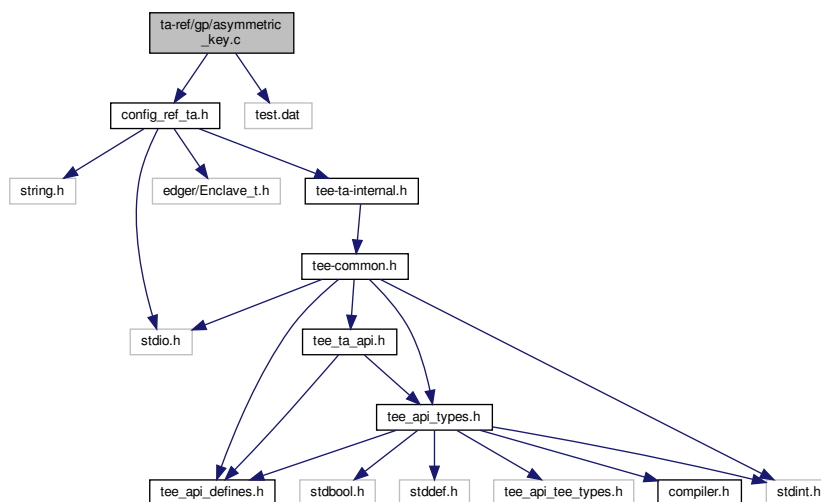
Retrieves the current system time as seen from the point of view of the TA, expressed in the number of seconds and print the "GP System time second and millisecond".

## 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.

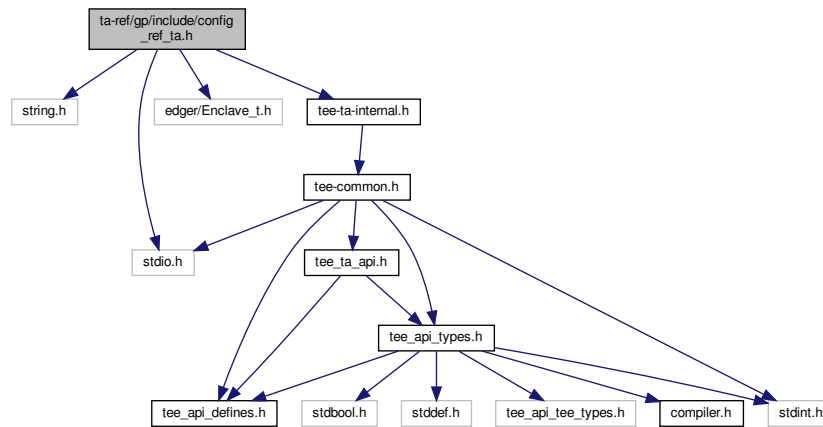
[TEE.AllocateOperation\(\)](#) function allocates a handle for a new cryptographic operation and sets the mode([TEE.MODE\\_DIGEST](#)) and algorithm type ([TEE.ALG\\_SHA256](#)). If this function does not return with [TEE.SUCCESS](#) then there is no valid handle value. [TEE.DigestUpdate\(\)](#) function 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. 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 [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\(\)](#) - 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 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.

**10.57.1.2 [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 )

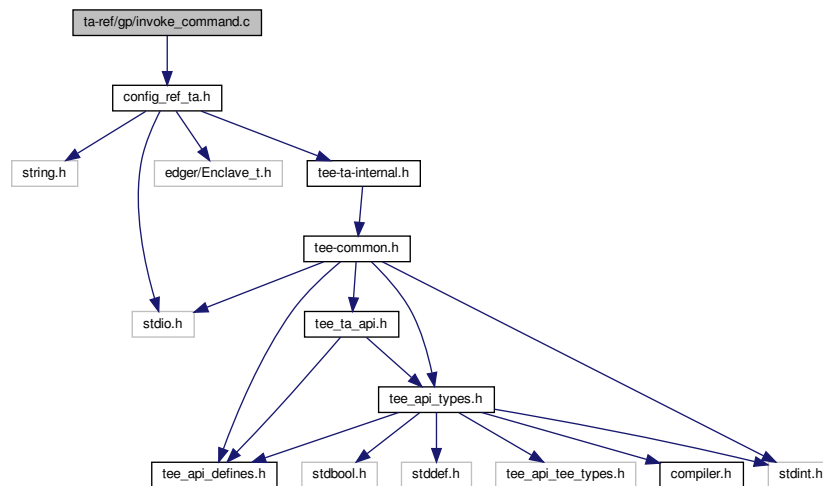
[gp\\_trusted\\_time\\_test\(\)](#) - Retrieves the current system time.

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`

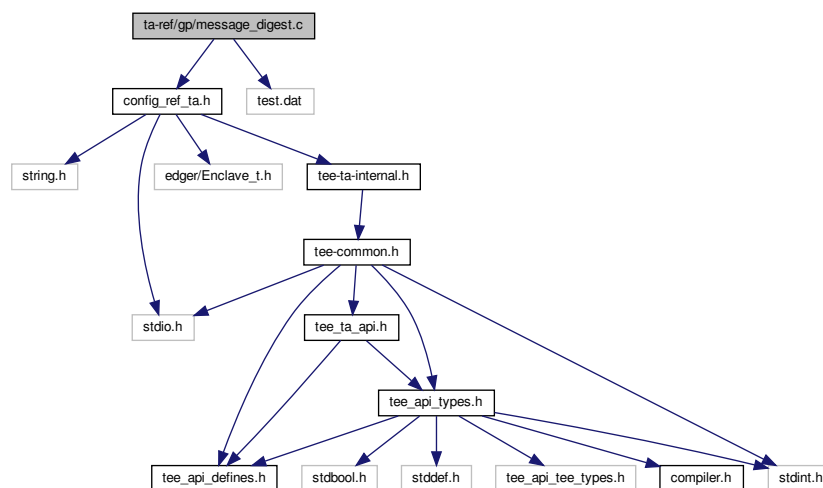
**10.58.1.6 TEEP\_AGENT\_TA\_NONE** `#define TEEP_AGENT_TA_NONE 0`

## 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 )`

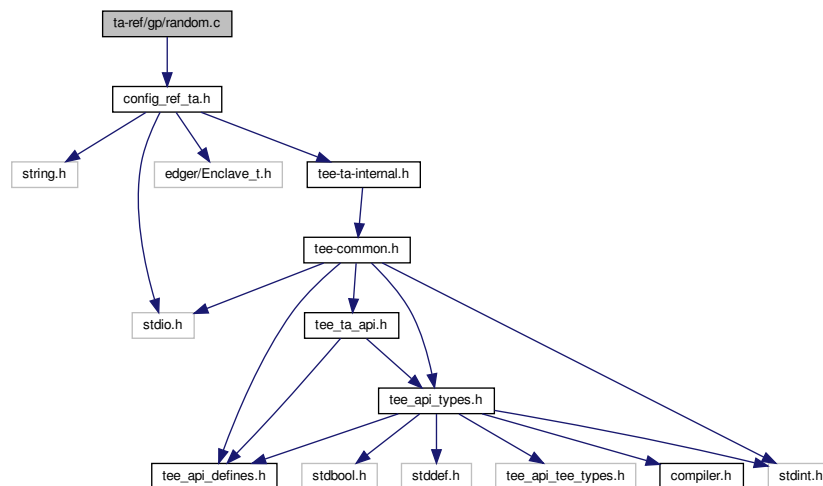
[gp\\_message\\_digest\\_test\(\)](#) - Accumulates message data for hashing.

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.

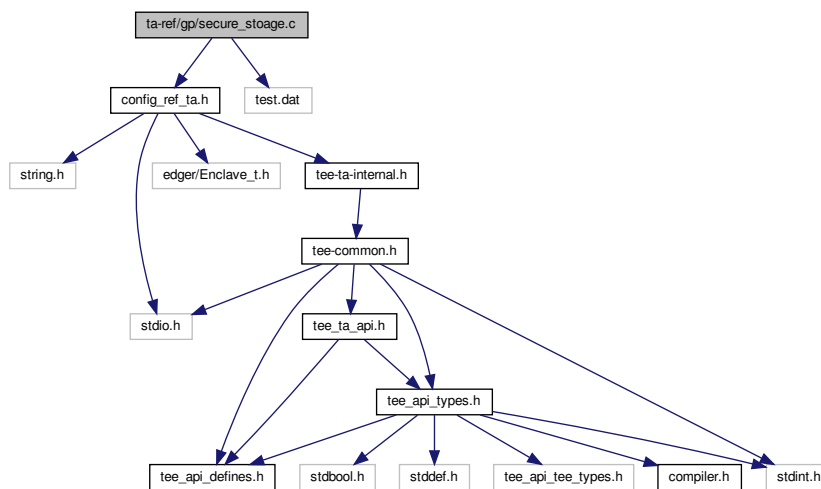
Generates the random data and finally print the generated random data.

### 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.

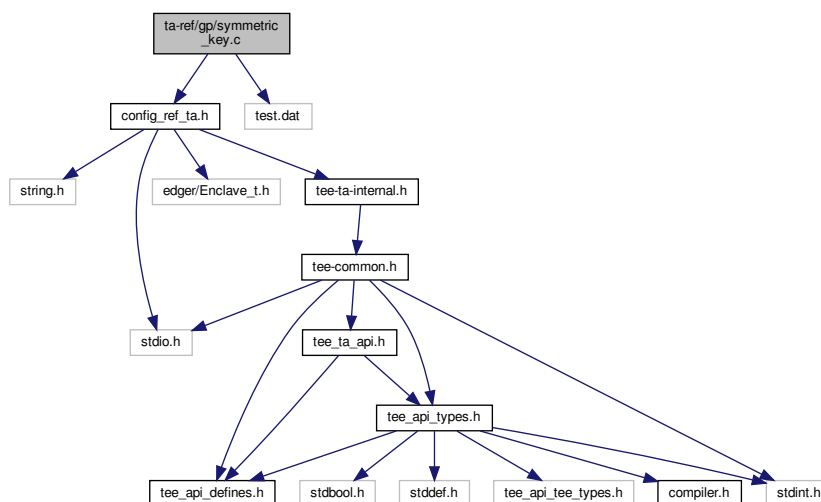
Creates a persistent object with initial attributes and an initial data stream content, and optionally returns either a handle on the created object, or `TEE_HANDLE_NULL` upon failure and `TEE_STORAGE_PRIVATE` parameter indicates which is the Trusted Storage Space to access. `TEE_DATA_FLAG_ACCESS_WRITE` 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.

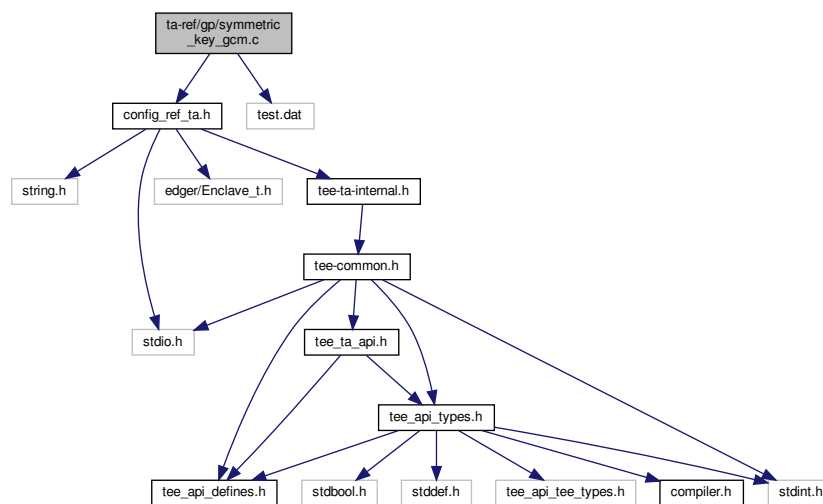
This function allocates an uninitialized transient object, i.e. a container for attributes. Transient objects are used to hold a cryptographic object (key or key-pair). With the generation of a key, a new cryptographic operation for encrypt and decrypt data is initiated with a symmetric cipher operation. 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 )`

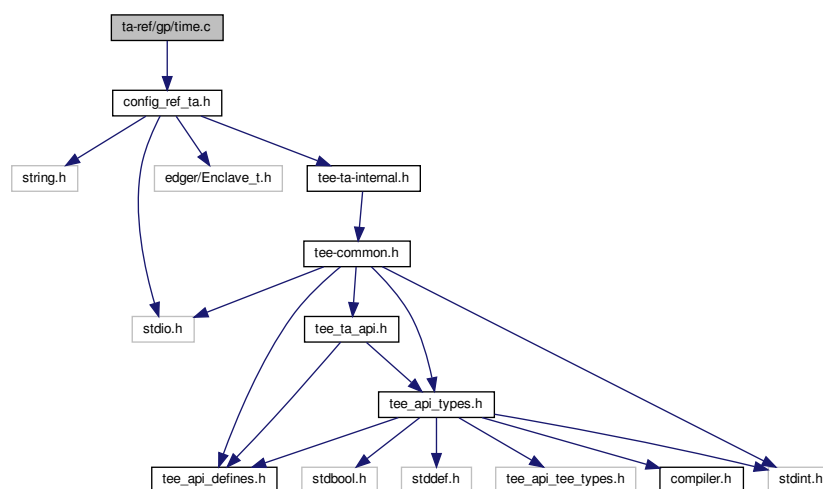
`gp_symmetric_key_gcm_verify_test()` - Encrypt and Decrypt the test data.

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".

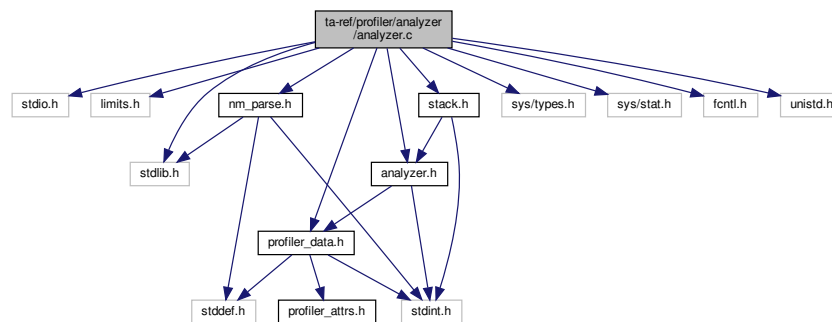
**10.64.1.2 [gp\\_trusted\\_time\\_test\(\)](#)** void gp\_trusted\_time\_test ( void )

[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

<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**

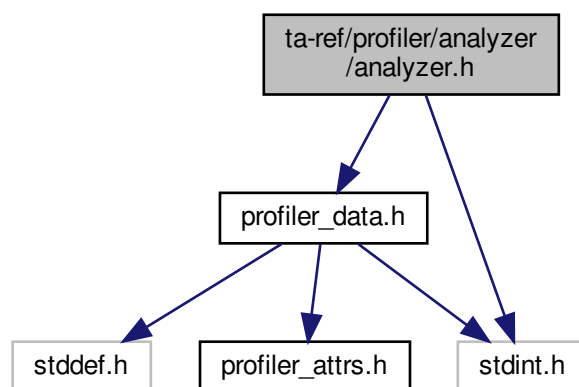
0 If success, else error occurred.

**10.66 ta-ref/profiler/analyzer/analyzer.h File Reference**

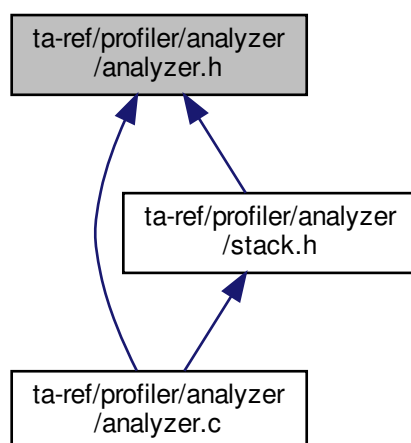
```
#include "profiler_data.h"
```

```
#include <stdint.h>
```

Include dependency graph for analyzer.h:



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





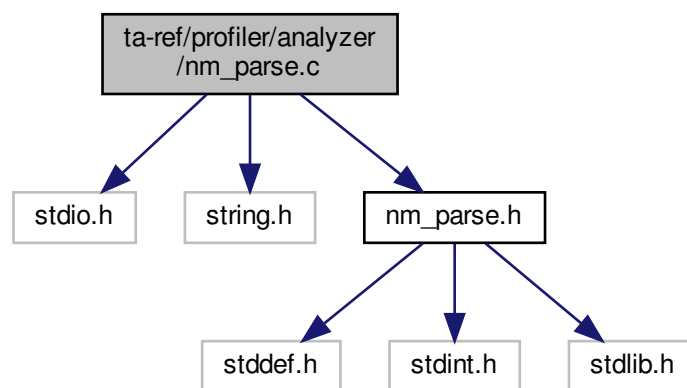
## Classes

- 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 <a href="#">nm_info</a>

**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]`

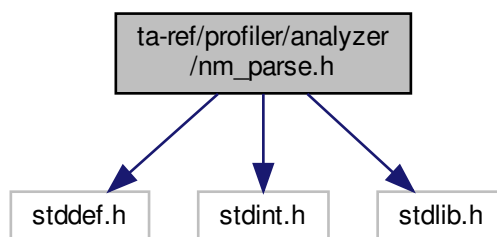
**10.68 ta-ref/profiler/analyzer/nm\_parse.h File Reference**

```
#include <stddef.h>
```

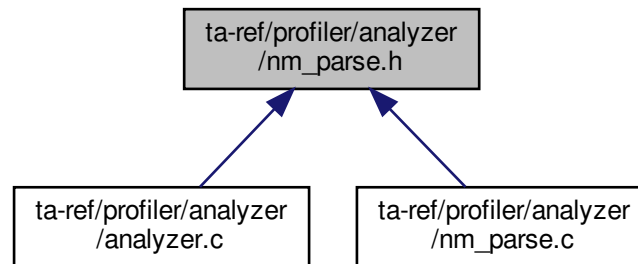
```
#include <stdint.h>
```

```
#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**

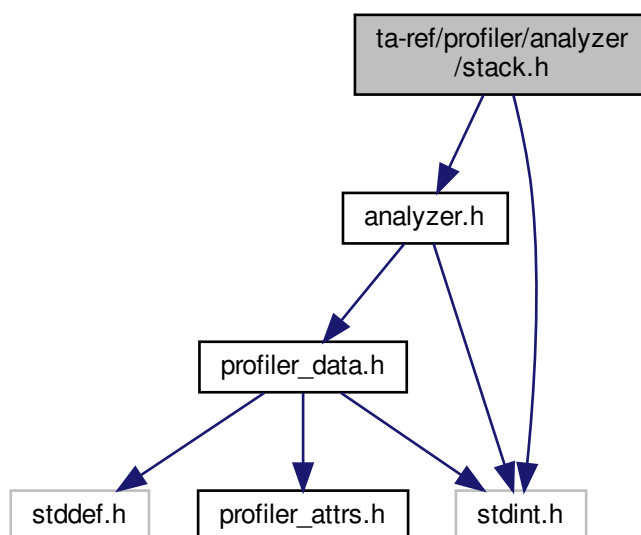
<i>fname</i>	File name.
--------------	------------

**Returns**

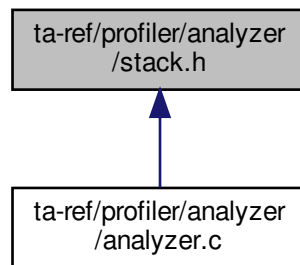
Updated structure list If success, else error occurred.

**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

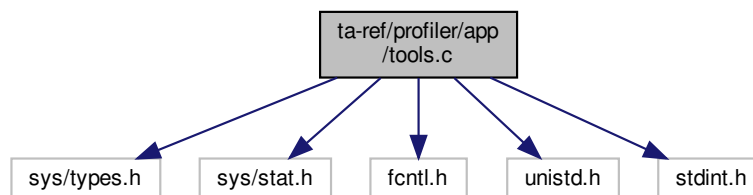
**10.69.3.1 pos** `uint64_t pos = 0 [static]`

**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**

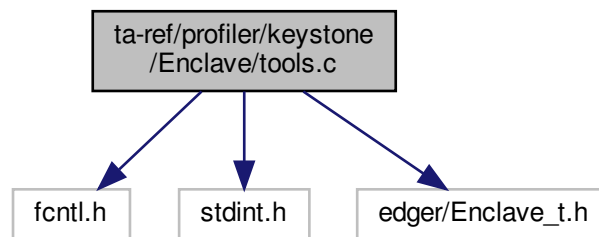
<i>ptr</i>	This is the pointer to the array of elements to be written.
<i>sz</i>	This is the size in bytes of each element to be written.

**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

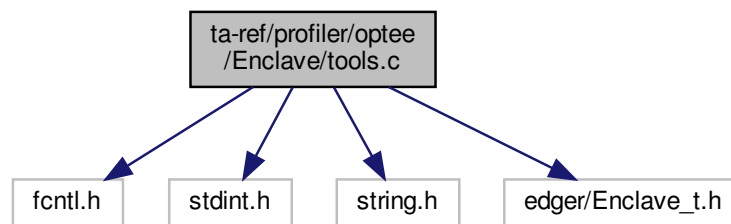
<i>ptr</i>	This is the pointer to the array of elements to be written.
<i>sz</i>	This is the size in bytes of each element to be written.

### 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

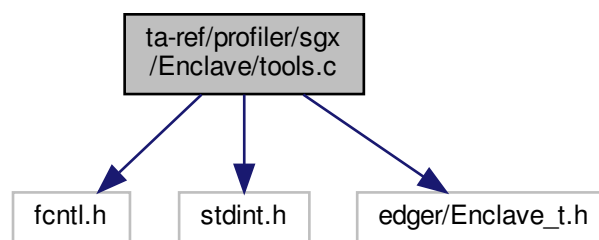
<i>buf</i>	This is a pointer to the destination array where the content is to be copied,
<i>ptr</i>	This is a pointer to the source of data to be copied,
<i>sz</i>	This is the number of bytes to be copied.

## 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

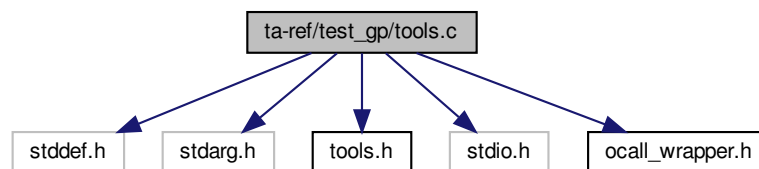
<i>ptr</i>	This is the pointer to the array of elements to be written.
<i>sz</i>	This is the size in bytes of each element to be written.

## 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**

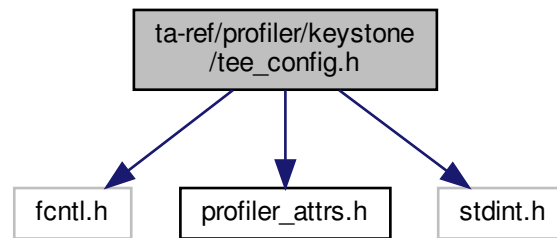
size If success.

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]

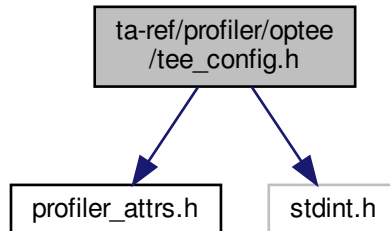
#### 10.75.2 Variable Documentation

**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

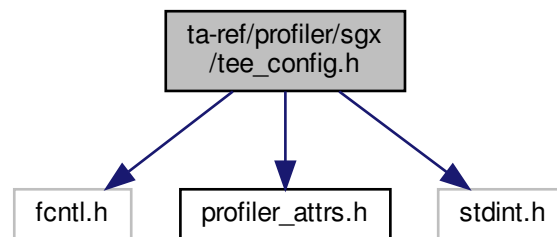
**10.76.3.1** `_ImageBase` `uintptr_t _ImageBase[]` `[extern]`

**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]`



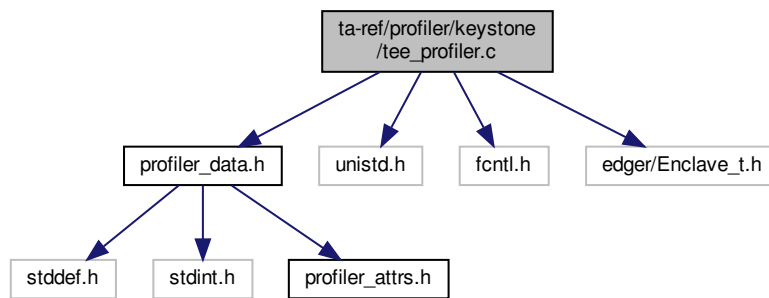
### 10.77.2 Variable Documentation

**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

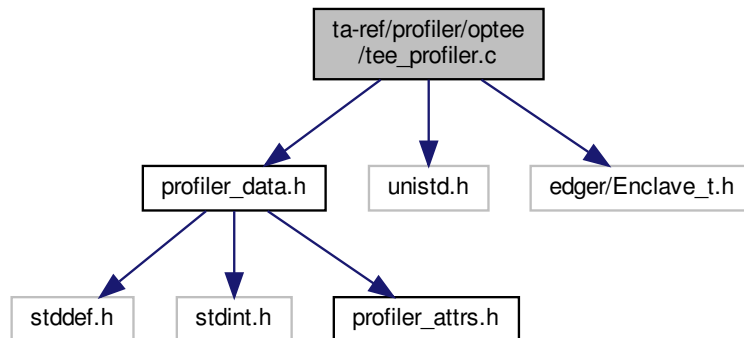
0 If success, else error occurred.

## 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**

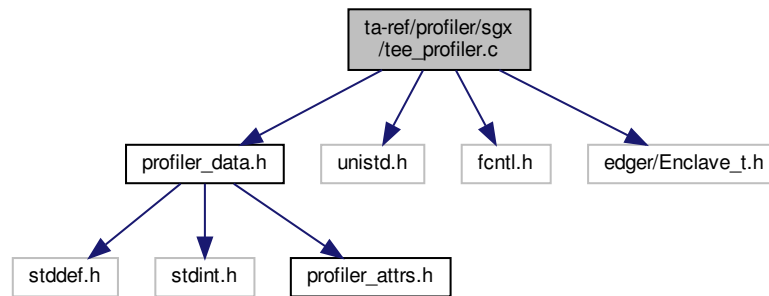
**10.79.2.1 \_\_profiler\_head** `struct __profiler_header* __profiler_head [extern]`

**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.

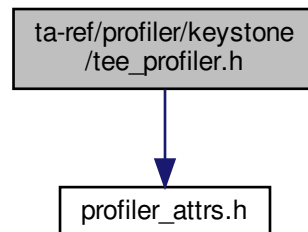
**10.80.2 Variable Documentation**

**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.

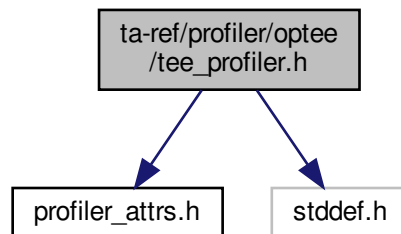
If the `__profiler_head` is not null then it returns the output file.

[\\_\\_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

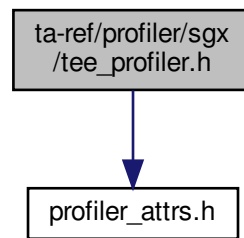
<i>buf</i>	It copies the read string into the buffer buf
<i>size</i>	This is the size in bytes of each element to be written.

## 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.

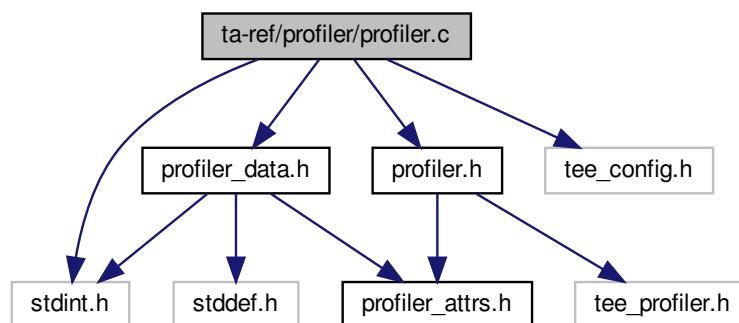
This function used for find the size of file and writing the updated file.

## 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.

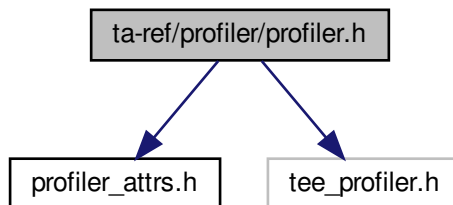
This function creates the new data value in the header of profiler.

## 10.84.2 Variable Documentation

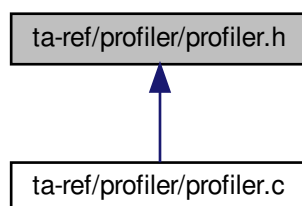
**10.84.2.1** `__profiler_head` `struct __profiler_header* __profiler_head = NULL`

## 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

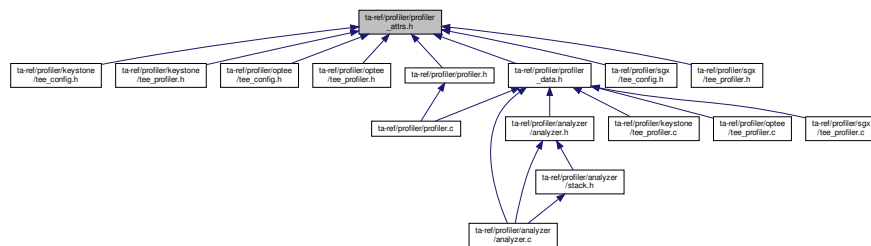
**10.85.1.1** `__profiler_map_info()` void NO\_PERF \_\_profiler\_map\_info ( void )

`__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\_attrs.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

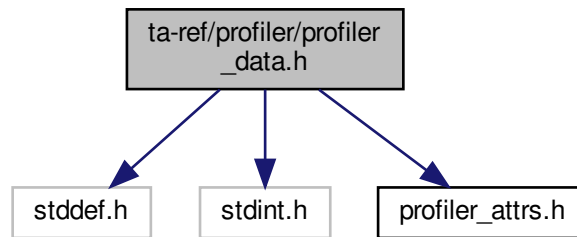
**10.86.1.1 NO\_PERF** `#define NO_PERF __attribute__((no_instrument_function,hot))`

**10.86.1.2 PERF\_SECTION** `#define PERF_SECTION __attribute__((section(".perf.region")))`

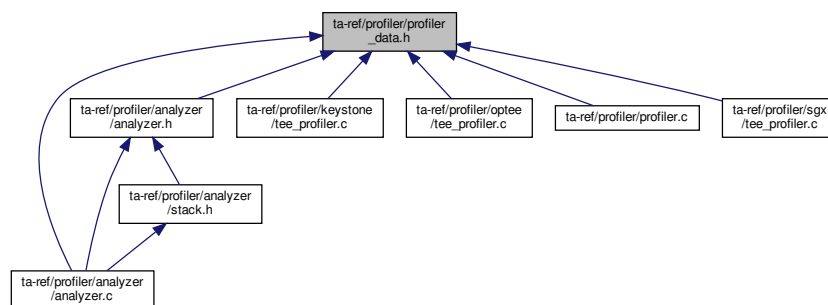
**10.86.1.3 USED** `#define USED __attribute__((used))`

## 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

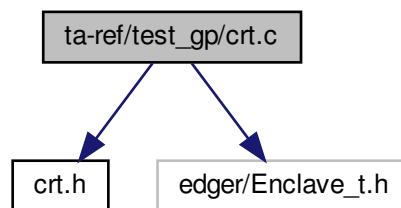
**10.87.5.1** `idx` `uint64_t idx`

**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.

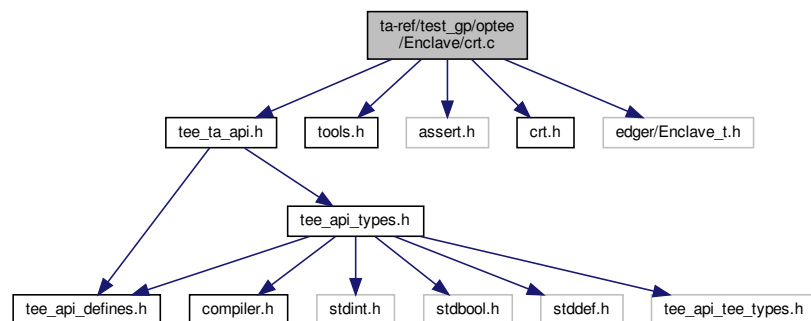
#### Parameters

<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**

sess_ctx	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 [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\(\)](#) - 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>	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

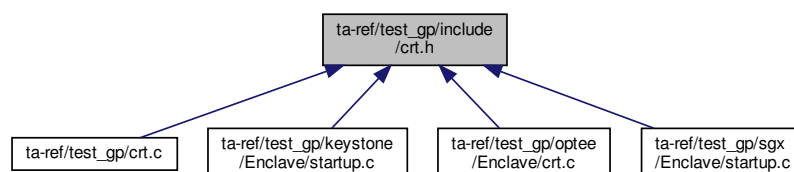
buffer If successfully executed, else error occurred.

## 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.

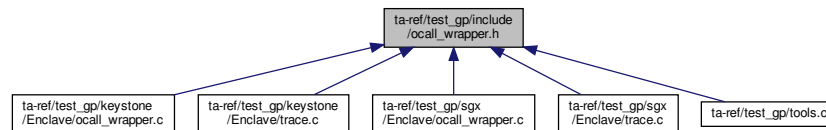
### Returns

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.
------------	-----------------------------

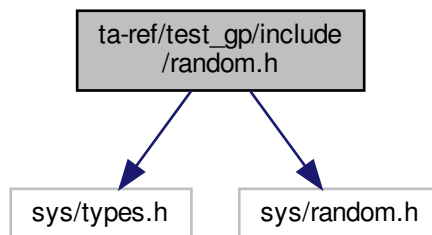
**Returns**

retval Its prints the value of str by calling `ocall_print_string()`.

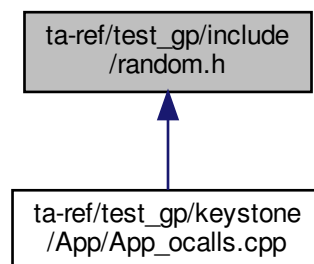
**10.92 ta-ref/test\_gp/include/random.h File Reference**

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

Include dependency graph for random.h:

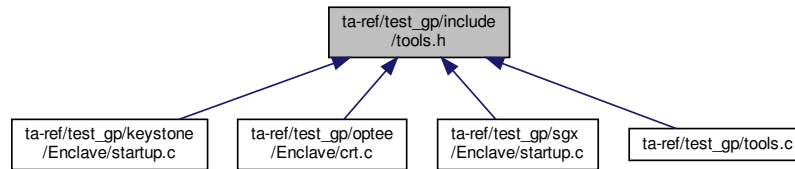


This graph shows which files directly or indirectly include this file:



## 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 occurred.

**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**

<code>c</code>	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**

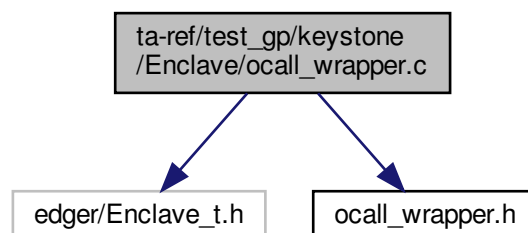
<code>s</code>	This is the C string to be written
----------------	------------------------------------

**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

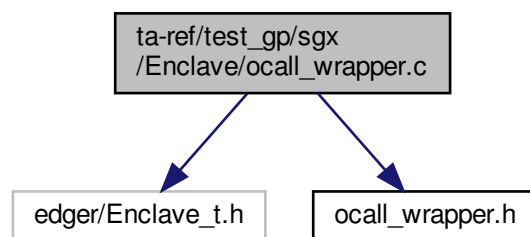
<i>str</i>	The string value for print.
------------	-----------------------------

#### 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

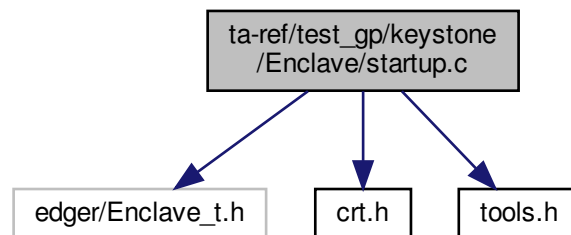
<i>str</i>	The string value for print.
------------	-----------------------------

#### 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.

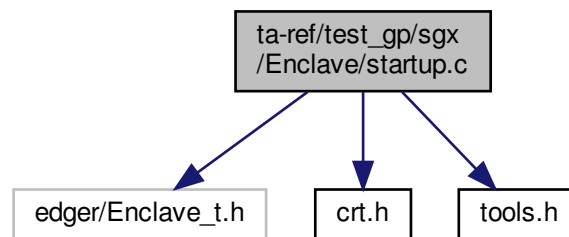
This function invokes [crt\\_begin\(\)](#) if defined macro is ENCLAVE\_VERBOSE then prints the main start and invokes [main\(\)](#). Once [main\(\)](#) is completed prints the main end and invokes the [crt\\_end\(\)](#).

**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\(\)](#).

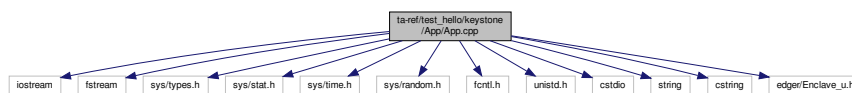
**Returns**

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**

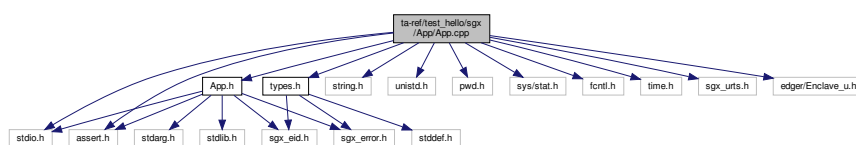
**10.98.2.1 enc\_path** `const char* enc_path = "Enclave.eapp_riscv"`

**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.

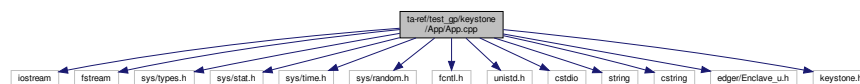
#### Parameters

<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** `const char* enc_path = "Enclave.eapp.riscv"`

**10.100.2.2 runtime\_path** `const char* runtime_path = "eyrie-rt"`

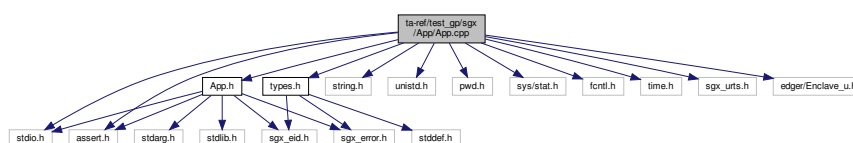
### 10.101 ta-ref/test\_gp/sgx/App/App.cpp File Reference

```

#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.

**Parameters**

<i>argc</i>	Argument Count is an int and it stores number of command-line arguments passed by the user including the name of the program.
<i>argv</i>	Argument Vector is an array of character pointers arguments.

**Returns**

0 If success, else error occurred

**10.101.2.3 print\_error\_message()** `void print_error_message (`  
`sgx_status_t ret )`

[print\\_error\\_message\(\)](#) - Used to print the sgx error list.

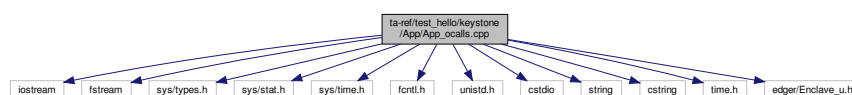
This function is used to print the sgx error list.

**Parameters**

<i>ret</i>	list containing all possible values of this data type.
------------	--

**10.102 ta-ref/test\_hello/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 <cstdio>
#include <string>
#include <cstring>
#include <time.h>
#include "edger/Enclave_u.h"
Include dependency graph for App_ocalls.cpp:
```

**Functions**

- [EDGE\\_EXTERN\\_BEGIN](#) unsigned int [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)
- [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

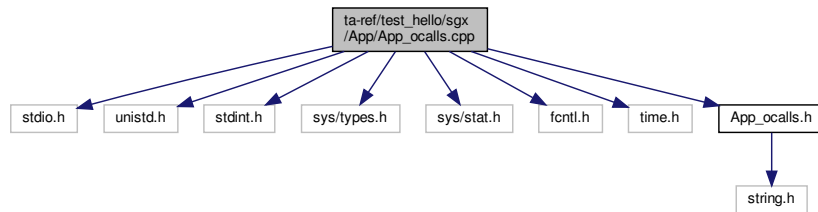
```
10.102.1.11 ocall.write.invoke_param() void ocallwrite.invoke_param (  
    int index,  
    unsigned int offset,  
    unsigned int size,  
    const char * buf )
```

**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 ocured.

**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.

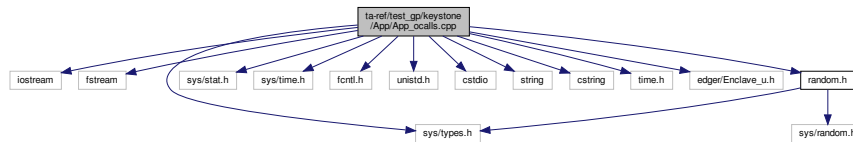
**Returns**

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 <a href="#">ree_time_t</a>
--------------	---

**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.

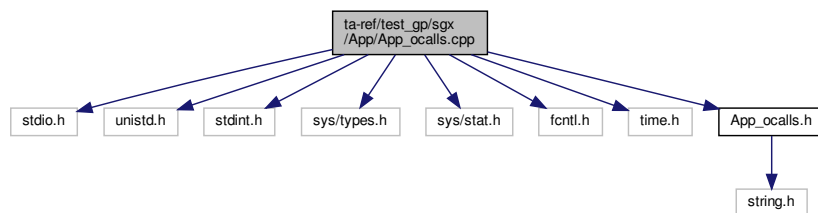
## Returns

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.

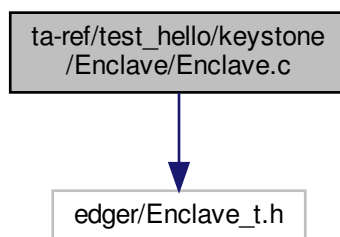
#### Returns

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"`

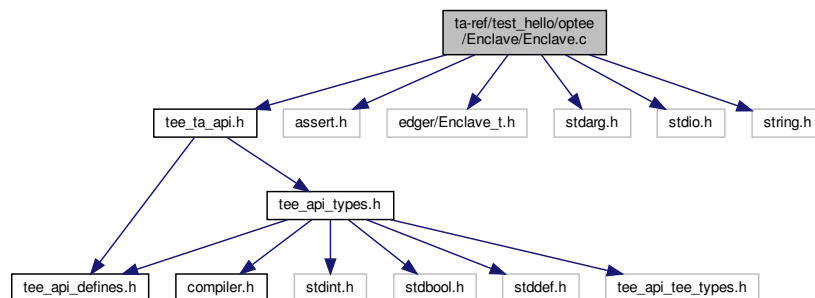
## 10.106.2 Function Documentation

**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**

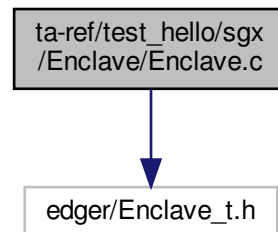
**10.107.3.1 print\_buf** `char print_buf[BUF_SIZE]`

**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

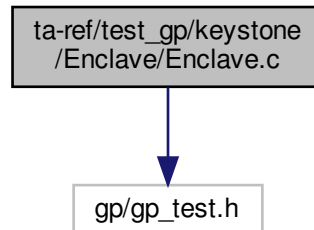
**10.108.2.1 ecall\_ta\_main()** `void ecall_ta_main (void )`

`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.

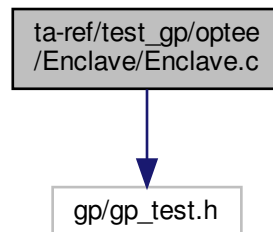
### Returns

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.

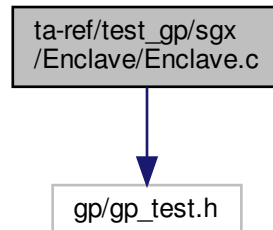
### Returns

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

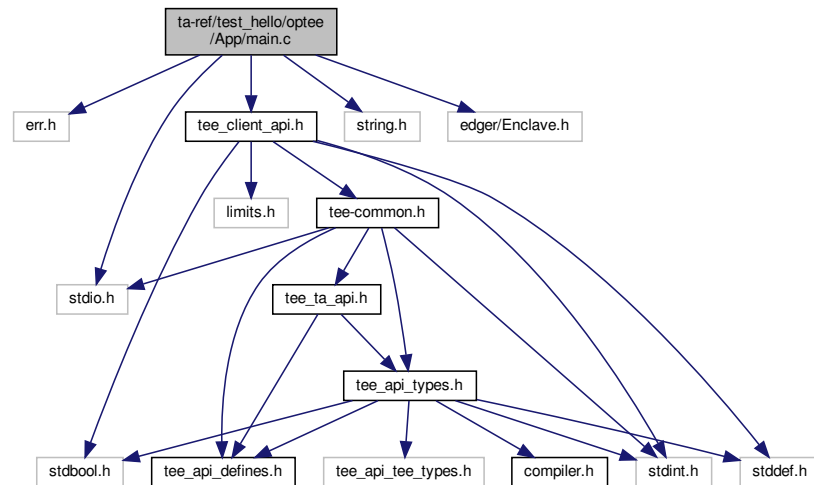
return 0 for success.

### 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

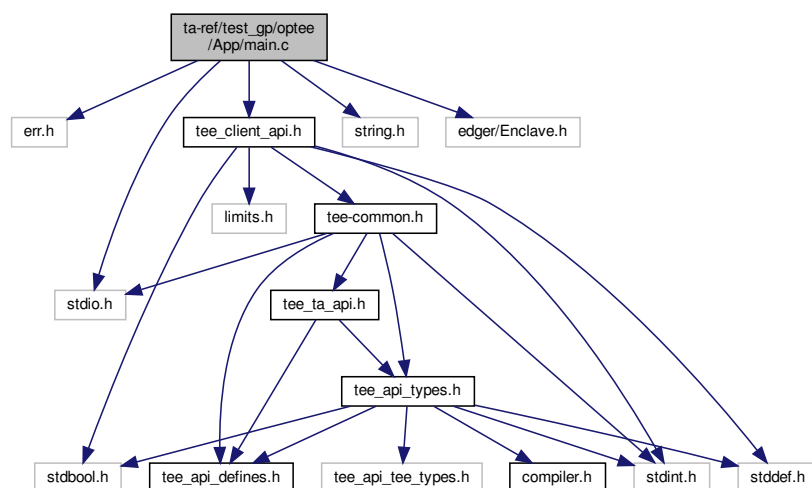
0 If success, else displays error message.

### 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.

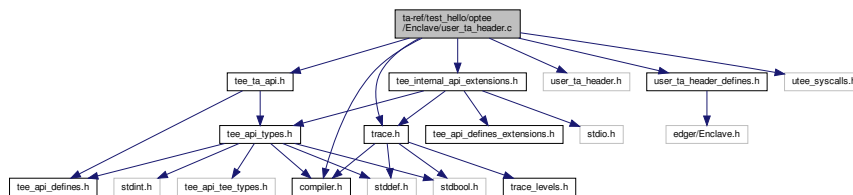
#### Returns

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},  
}
```

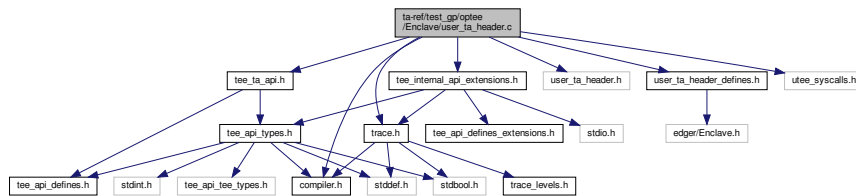
**10.114.3.5 trace\_ext\_prefix** `const char trace_ext_prefix[] = "TA"`

**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 ta_head ta_head __section (  
    ".ta_head" )`

**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},
}
```

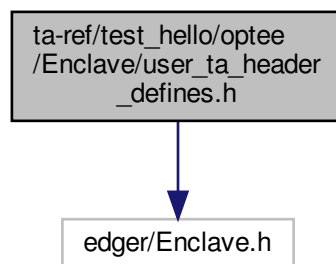
#### 10.115.3.5 `trace_ext_prefix` `const char trace_ext_prefix[] = "TA"`

#### 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

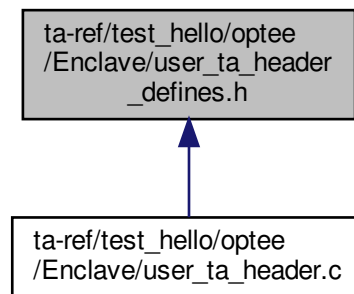
```
#include "edger/Enclave.h"
```

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)`

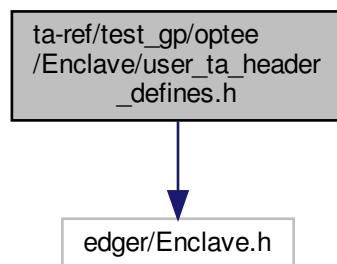
**10.116.1.6 TA\_UUID** `#define TA_UUID TA_REF_UUID`

**10.116.1.7 TA\_VERSION** `#define TA_VERSION "1.0"`

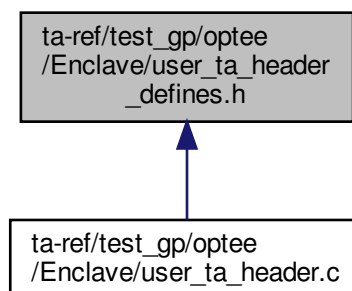
## 10.117 ta-ref/test\_gp/optee/Enclave/user\_ta\_header\_defines.h File Reference

```
#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`

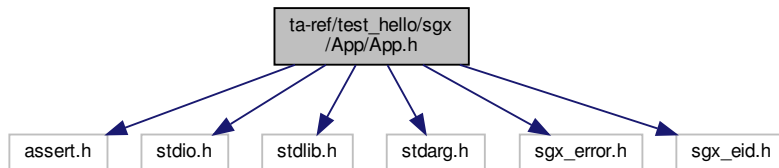
**10.117.1 Macro Definition Documentation****10.117.1.1 TA\_CURRENT\_TA\_EXT\_PROPERTIES** `#define TA_CURRENT_TA_EXT_PROPERTIES`**Value:**

```
{ "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 } }
```

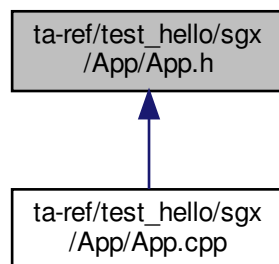
**10.117.1.2 TA\_DATA\_SIZE** `#define TA_DATA_SIZE (32 * 1024)`**10.117.1.3 TA\_DESCRIPTION** `#define TA_DESCRIPTION "Example of OP-TEE TEST Trusted Application"`**10.117.1.4 TA\_FLAGS** `#define TA_FLAGS TA_FLAG_EXEC_DDR`**10.117.1.5 TA\_STACK\_SIZE** `#define TA_STACK_SIZE (2 * 1024)`**10.117.1.6 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`

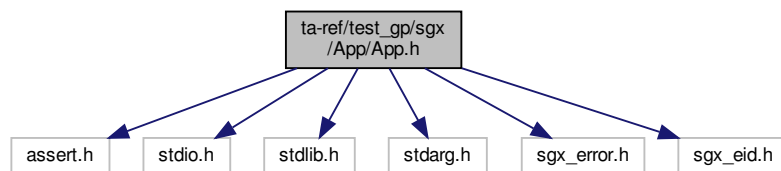
**10.118.1.3 TRUE** `#define TRUE 1`

## 10.118.2 Variable Documentation

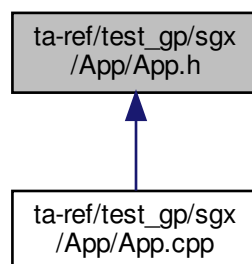
**10.118.2.1 global\_eid** `sgx_enclave_id_t globaleid [extern]`

## 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`

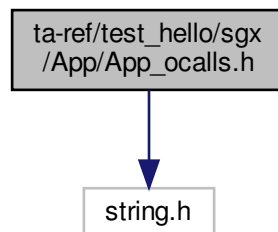
### 10.119.2 Variable Documentation

**10.119.2.1 global\_eid** `sgx_enclave_id_t globaleid [extern]`

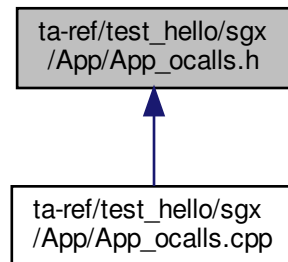
## 10.120 ta-ref/test\_hello/sgx/App/App\_ocalls.h File Reference

```
#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>	<i>fdesc</i> 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

[ocal.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

[ocal.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.

[ocal.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 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 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 <a href="#">ree_time_t</a>
--------------	---

**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.

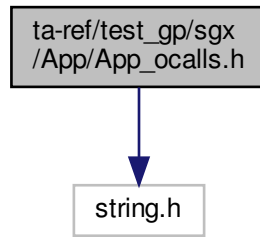
**Returns**

file descriptor If success, else error occurred.

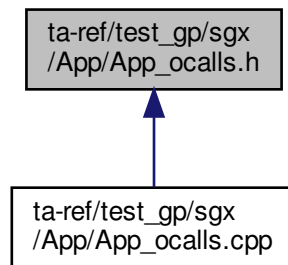
**10.121 ta-ref/test\_gp/sgx/App/App\_ocalls.h File Reference**

```
#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>	<i>fdesc</i> 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 occurred

**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 <a href="#">ree_time_t</a>
--------------	---

**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**

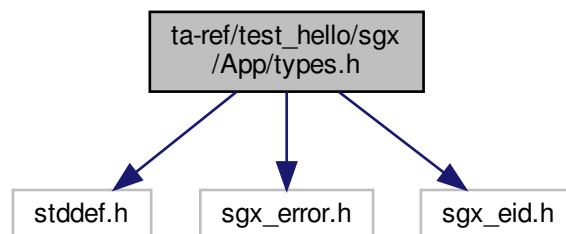
<i>desc</i>	file descriptor.
<i>buf</i>	pointer to a buffer.
<i>len</i>	Size of elements.

**Returns**

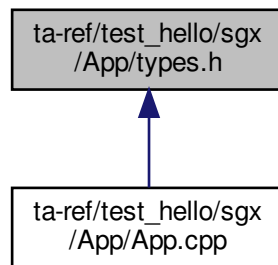
file descriptor If success, else error occurred.

**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](#)

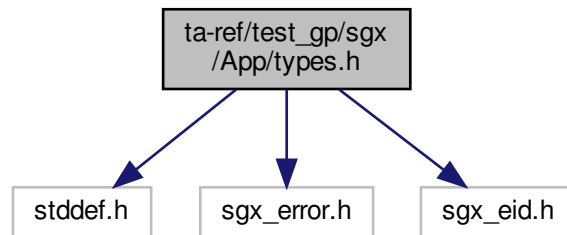
## 10.122.2 Variable Documentation

**10.122.2.1 [global\\_eid](#)** [sgx\\_enclave\\_id\\_t](#) [globaleid](#) = 0

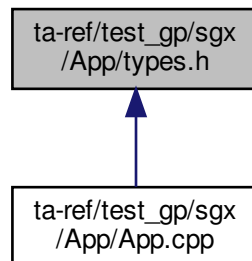
**10.122.2.2 [sgx\\_errlist](#)** [sgx\\_errlist\\_t](#) [sgx\\_errlist](#)[] [static]

### 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 globaleid = 0`

**10.123.2.2** `sgx_errlist` `sgx_errlist_t sgx_errlist[] [static]`



## Index

- `_C_FUNCTION`
  - `user_ta_header.c`, [406](#), [410](#)
- `__GCC_VERSION`
  - `compiler.h`, [74](#)
- `__INTOF_ADD`
  - `compiler.h`, [74](#)
- `__INTOF_ASSIGN`
  - `compiler.h`, [74](#)
- `__INTOF_HALF_MAX_SIGNED`
  - `compiler.h`, [74](#)
- `__INTOF_MAX`
  - `compiler.h`, [74](#)
- `__INTOF_MAX_SIGNED`
  - `compiler.h`, [74](#)
- `__INTOF_MIN`
  - `compiler.h`, [75](#)
- `__INTOF_MIN_SIGNED`
  - `compiler.h`, [75](#)
- `__INTOF_MUL`
  - `compiler.h`, [75](#)
- `__INTOF_SUB`
  - `compiler.h`, [76](#)
- `__ImageBase`
  - `crt.c`, [363](#)
  - `tee_config.h`, [338](#), [340](#), [341](#)
- `__TEE_ObjectHandle`, [36](#)
  - `desc`, [36](#)
  - `flags`, [37](#)
  - `persist_ctx`, [37](#)
  - `private_key`, [37](#)
  - `public_key`, [37](#)
  - `type`, [37](#)
- `__TEE_OperationHandle`, [37](#)
  - `aectx`, [37](#)
  - `aegcm.state`, [38](#)
  - `aeiv`, [38](#)
  - `aekey`, [38](#)
  - `alg`, [38](#)
  - `ctx`, [38](#)
  - `flags`, [38](#)
  - `mode`, [38](#)
  - `prikey`, [38](#)
  - `pubkey`, [38](#)
- `__aligned`
  - `compiler.h`, [72](#)
  - `tee_api_types.h`, [184](#)
- `__attr_const`
  - `compiler.h`, [72](#)
- `__attribute__`
  - `profiler_data.h`, [356](#)
  - `tee-internal-api-machine.c`, [212](#)
  - `tee-internal-api.c`, [225](#)
  - `tee-ta-internal.h`, [82](#)
- `__bss`
  - `compiler.h`, [72](#)
- `__cold`
  - `compiler.h`, [72](#)
- `__compiler_add_overflow`
  - `compiler.h`, [72](#)
- `__compiler_atomic_load`
  - `compiler.h`, [72](#)
- `__compiler_atomic_store`
  - `compiler.h`, [72](#)
- `__compiler_bswap16`
  - `compiler.h`, [73](#)
- `__compiler_bswap32`
  - `compiler.h`, [73](#)
- `__compiler_bswap64`
  - `compiler.h`, [73](#)
- `__compiler_compare_and_swap`
  - `compiler.h`, [73](#)
- `__compiler_mul_overflow`
  - `compiler.h`, [73](#)
- `__compiler_sub_overflow`
  - `compiler.h`, [73](#)
- `__cyg_profile_func`
  - `profiler.c`, [350](#)
- `__cyg_profile_func_enter`
  - `profiler.c`, [350](#)
- `__cyg_profile_func_exit`
  - `profiler.c`, [351](#)
- `__data`
  - `compiler.h`, [73](#)
- `__deprecated`
  - `compiler.h`, [73](#)
- `__early_ta`
  - `compiler.h`, [74](#)
- `__init_array_start`
  - `crt.c`, [357](#)
- `__intof_mul_a0`
  - `compiler.h`, [75](#)
- `__intof_mul_a1`
  - `compiler.h`, [75](#)
- `__intof_mul_b0`
  - `compiler.h`, [75](#)
- `__intof_mul_b1`
  - `compiler.h`, [75](#)
- `__intof_mul_hmask`
  - `compiler.h`, [76](#)
- `__intof_mul_hshift`
  - `compiler.h`, [76](#)
- `__intof_mul_negate`
  - `compiler.h`, [76](#)
- `__intof_mul_t`
  - `compiler.h`, [76](#)
- `__maybe_unused`
  - `compiler.h`, [76](#)
- `__must_check`
  - `compiler.h`, [76](#)
- `__noinline`

- compiler.h, 76
- \_\_noprof
  - compiler.h, 77
- \_\_noreturn
  - compiler.h, 77
- \_\_packed
  - compiler.h, 77
- \_\_printf
  - compiler.h, 77
- \_\_profiler\_data, 35
  - callee, 35
  - direction, 35
  - hartid, 35
  - nsec, 35
- \_\_profiler\_get\_data\_ptr
  - profiler.c, 351
- \_\_profiler\_head
  - profiler.c, 352
  - tee\_profiler.c, 343, 344, 346
- \_\_profiler\_header, 36
  - idx, 36
  - size, 36
  - start, 36
- \_\_profiler\_map\_info
  - profiler.c, 351
  - profiler.h, 352
- \_\_profiler\_nsec\_t
  - profiler\_data.h, 355
- \_\_profiler\_unmap\_info
  - tee\_profiler.c, 341, 343, 345
  - tee\_profiler.h, 347–349
- \_\_pure
  - compiler.h, 77
- \_\_rodata
  - compiler.h, 77
- \_\_rodata\_unpaged
  - compiler.h, 77
- \_\_section
  - compiler.h, 77
  - user\_ta\_header.c, 407, 410
- \_\_ta\_entry
  - user\_ta\_header.c, 407, 410
- \_\_unused
  - compiler.h, 77
- \_\_used
  - compiler.h, 77
- \_\_utee\_entry
  - user\_ta\_header.c, 407, 410
- \_\_weak
  - compiler.h, 77
- \_\_wrapper\_ocall\_close\_file
  - Enclave\_u.h, 300
- .atoi
  - vsprintf.c, 248, 254
- .ftoa
  - vsprintf.c, 248, 255
- .is\_digit
  - vsprintf.c, 249, 255
- .ntoa\_format
  - vsprintf.c, 249, 256
- .ntoa\_long
  - vsprintf.c, 249, 256
- .ntoa\_long\_long
  - vsprintf.c, 249, 257
- .out\_buffer
  - vsprintf.c, 249, 258
- .out\_char
  - vsprintf.c, 250, 258
- .out\_fct
  - vsprintf.c, 250, 259
- .out\_null
  - vsprintf.c, 250, 259
- .putchar
  - vsprintf.c, 247, 252
- .sanctum\_dev\_public\_key
  - test\_dev\_key.h, 204
- .sanctum\_dev\_public\_key\_len
  - test\_dev\_key.h, 204
- .sanctum\_dev\_secret\_key
  - test\_dev\_key.h, 204
- .sanctum\_dev\_secret\_key\_len
  - test\_dev\_key.h, 204
- .sgx\_errlist\_t, 39
  - err, 39
  - msg, 39
  - sug, 39
- .strlen
  - Enclave.c, 395
  - tools.c, 336
  - trace.c, 242, 244
  - vsprintf.c, 250, 259
- .vsprintf
  - vsprintf.c, 250, 260
- a
  - invoke\_command\_param\_t, 42
  - TEE\_Attribute, 51
  - TEE\_Param, 58
  - TEEC\_Value, 70
- addr
  - list, 43
- addrinfo, 39
  - ai\_addr, 40
  - ai\_addrlen, 40
  - ai\_canonname, 40
  - ai\_family, 40
  - ai\_flags, 40
  - ai\_next, 40
  - ai\_protocol, 40
  - ai\_socktype, 40
- aectx
  - \_\_TEE\_OperationHandle, 37
- aegcm\_state
  - \_\_TEE\_OperationHandle, 38
- aeiv
  - \_\_TEE\_OperationHandle, 38
- aekey

- \_\_TEE\_OperationHandle, 38
- AES256
  - tee\_api\_tee\_types.h, 233, 235
- ai\_addr
  - addrinfo, 40
- ai\_addrlen
  - addrinfo, 40
- ai\_canonname
  - addrinfo, 40
- ai\_family
  - addrinfo, 40
- ai\_flags
  - addrinfo, 40
- ai\_next
  - addrinfo, 40
- ai\_protocol
  - addrinfo, 40
- ai\_socktype
  - addrinfo, 40
- alg
  - \_\_TEE\_OperationHandle, 38
- algorithm
  - TEE\_OperationInfo, 55
  - TEE\_OperationInfoMultiple, 57
- aligned
  - crt.c, 357
- allocated\_size
  - TEEC\_SharedMemory, 67
- analyzer.c
  - BUF\_MAX, 323
  - COLS, 323
  - FORMAT, 323
  - main, 323
- App.cpp
  - enc\_path, 373, 376
  - initialize\_enclave, 374, 377
  - main, 372, 374, 375, 377
  - MAX\_PATH, 374, 377
  - print\_error\_message, 374, 378
  - runtime\_path, 373, 376
- App.h
  - ENCLAVE\_FILENAME, 416, 418
  - FALSE, 417, 418
  - global\_eid, 417, 418
  - TRUE, 417, 418
- App\_ocalls.cpp
  - MAX\_PATH, 390
  - NO\_PERF, 386, 391
  - ocall\_close\_file, 379, 383, 386, 391
  - ocall\_getrandom, 379, 387
  - ocall\_invoke\_command\_callback\_write, 379, 387
  - ocall\_open\_file, 380, 383, 388, 391
  - ocall\_print\_string, 380, 384, 388, 391
  - ocall\_put\_invoke\_command\_result, 381
  - ocall\_read\_file, 381, 384, 389, 392
  - ocall\_read\_invoke\_param, 381
  - ocall\_ree\_time, 381, 385, 389, 392
  - ocall\_write\_file, 382, 385, 389, 393
  - ocall\_write\_invoke\_param, 382
- App\_ocalls.h
  - ocall\_close\_file, 419, 427
  - ocall\_open\_file, 420, 428
  - ocall\_print\_string, 422, 429
  - ocall\_read\_file, 422, 430
  - ocall\_ree\_time, 424, 431
  - ocall\_write\_file, 425, 432
  - ree\_time\_t, 419, 427
- arg
  - out\_fct\_wrap\_type, 46
- array\_t
  - user\_types.h, 297
- asymmetric\_key.c
  - gp\_asymmetric\_key\_sign\_test, 310
  - SHA\_LENGTH, 310
  - SIG\_LENGTH, 310
- ATTEST\_DATA\_MAXLEN
  - report.h, 78
- attributeID
  - TEE\_Attribute, 51
- b
  - invoke\_command\_param\_t, 42
  - ob16\_t, 45
  - ob196\_t, 45
  - ob256\_t, 46
  - TEE\_Attribute, 51
  - TEE\_Param, 58
  - TEEC\_Value, 70
- bench.c
  - benchmark, 279
  - init, 279
  - labels, 281
  - record, 279
  - tee\_time\_tests, 280
  - time\_diff, 280
  - time\_test, 281
  - time\_to\_millis, 281
- bench.h
  - cpu\_double\_benchmark, 283
  - cpu\_int\_benchmark, 283
  - io\_read\_benchmark, 283
  - io\_write\_benchmark, 283
  - NO\_PERF, 283
  - random\_memory\_benchmark, 284
  - ree\_time\_test, 284
  - sequential\_memory\_benchmark, 284
  - system\_time\_test, 285
- BENCH\_TYPE
  - config\_bench.h, 287
- benchmark
  - bench.c, 279
- buf
  - param\_buffer\_t, 47
  - tee\_def.h, 291–293
- buf\_flag
  - tee\_def.h, 291–293
- BUF\_MAX

- analyzer.c, 323
- BUF\_SIZE
  - config\_bench.h, 287
  - Enclave.c, 395
  - main.c, 405
  - nm\_parse.c, 326
- buffer
  - TEE\_Attribute, 51
  - TEE\_Param, 58
  - TEE\_SEAID, 59
  - TEEC\_SharedMemory, 67
  - TEEC\_TempMemoryReference, 68
- buffer\_allocated
  - TEEC\_SharedMemory, 67
- buffer\_t
  - user\_types.h, 297
- bufferLen
  - TEE\_SEAID, 59
- CALL
  - profiler\_data.h, 355
- callee
  - \_\_profiler\_data, 35
  - result, 49
- CIPHER\_LENGTH
  - symmetric\_key.c, 320
  - symmetric\_key\_gcm.c, 321
- clockSeqAndNode
  - TEE\_UUID, 61
  - TEEC\_UUID, 69
- COLS
  - analyzer.c, 323
- COMMAND
  - tee\_config.h, 339
- commandID
  - invoke\_command\_t, 43
- compiler.h
  - \_\_GCC\_VERSION, 74
  - \_\_INTOF\_ADD, 74
  - \_\_INTOF\_ASSIGN, 74
  - \_\_INTOF\_HALF\_MAX\_SIGNED, 74
  - \_\_INTOF\_MAX, 74
  - \_\_INTOF\_MAX\_SIGNED, 74
  - \_\_INTOF\_MIN, 75
  - \_\_INTOF\_MIN\_SIGNED, 75
  - \_\_INTOF\_MUL, 75
  - \_\_INTOF\_SUB, 76
  - \_\_aligned, 72
  - \_\_attr\_const, 72
  - \_\_bss, 72
  - \_\_cold, 72
  - \_\_compiler\_add\_overflow, 72
  - \_\_compiler\_atomic\_load, 72
  - \_\_compiler\_atomic\_store, 72
  - \_\_compiler\_bswap16, 73
  - \_\_compiler\_bswap32, 73
  - \_\_compiler\_bswap64, 73
  - \_\_compiler\_compare\_and\_swap, 73
  - \_\_compiler\_mul\_overflow, 73
  - \_\_compiler\_sub\_overflow, 73
  - \_\_data, 73
  - \_\_deprecated, 73
  - \_\_early\_ta, 74
  - \_\_intof\_mul\_a0, 75
  - \_\_intof\_mul\_a1, 75
  - \_\_intof\_mul\_b0, 75
  - \_\_intof\_mul\_b1, 75
  - \_\_intof\_mul\_hmask, 76
  - \_\_intof\_mul\_hshift, 76
  - \_\_intof\_mul\_negate, 76
  - \_\_intof\_mul\_t, 76
  - \_\_maybe\_unused, 76
  - \_\_must\_check, 76
  - \_\_noinline, 76
  - \_\_noprof, 77
  - \_\_noreturn, 77
  - \_\_packed, 77
  - \_\_printf, 77
  - \_\_pure, 77
  - \_\_rodata, 77
  - \_\_rodata\_unpaged, 77
  - \_\_section, 77
  - \_\_unused, 77
  - \_\_used, 77
  - \_\_weak, 77
- config\_bench.h
  - BENCH\_TYPE, 287
  - BUF\_SIZE, 287
  - CPU\_DOUBLE\_SENSITIVE, 288
  - CPU\_INT\_SENSITIVE, 288
  - DOUBLE\_OFFSET, 287
  - IO\_READ\_SENSITIVE, 288
  - IO\_WRITE\_SENSITIVE, 288
  - MULT\_SIZE, 287
  - OFFSET, 287
  - RANDOM\_MEMORY\_SENSITIVE, 288
  - record, 288
  - REE\_TIME\_TEST, 288
  - SEQUENTIAL\_MEMORY\_SENSITIVE, 288
  - SYSTEM\_TIME\_TEST, 288
- config\_ref\_ta.h
  - GP\_ASSERT, 311
  - tee\_printf, 311
- content
  - TEE\_Attribute, 52
- cpu\_bench.c
  - cpu\_double\_benchmark, 286
  - cpu\_int\_benchmark, 286
- cpu\_double\_benchmark
  - bench.h, 283
  - cpu\_bench.c, 286
- CPU\_DOUBLE\_SENSITIVE
  - config\_bench.h, 288
- cpu\_int\_benchmark
  - bench.h, 283
  - cpu\_bench.c, 286
- CPU\_INT\_SENSITIVE

- config\_bench.h, 288
- create\_htable
  - nm\_parse.c, 326
- crt.c
  - \_\_ImageBase, 363
  - \_\_init\_array\_start, 357
  - aligned, 357
  - crt\_end, 357
  - fini\_array, 357
  - init\_array, 358
  - run\_all\_test, 359
  - TA\_CloseSessionEntryPoint, 360
  - TA\_CreateEntryPoint, 360
  - TA\_DestroyEntryPoint, 360
  - TA\_InvokeCommandEntryPoint, 360
  - TA\_OpenSessionEntryPoint, 361
  - TEE\_PARAM\_TYPE0, 359
  - TEE\_PARAM\_TYPE1, 359
  - tee\_printf, 361
- crt.h
  - crt\_begin, 364
  - crt\_end, 364
  - main, 364
- crt\_begin
  - crt.h, 364
- crt\_end
  - crt.c, 357
  - crt.h, 364
- ctx
  - \_\_TEE\_OperationHandle, 38
  - TEEC\_Session, 66
- data
  - enclave\_report, 41
- data\_len
  - enclave\_report, 41
- DATA\_LENGTH
  - secure\_storage.c, 318
- dataPosition
  - TEE\_ObjectInfo, 53
- dataSize
  - TEE\_ObjectInfo, 54
- depth
  - result, 49
- desc
  - \_\_TEE\_ObjectHandle, 36
- dev\_public\_key
  - report, 49
- dhex\_dump
  - trace.h, 209
- DHEXDUMP
  - trace.h, 206
- digestLength
  - TEE\_OperationInfo, 55
  - TEE\_OperationInfoMultiple, 57
- direction
  - \_\_profiler\_data, 35
- direction\_t
  - profiler\_data.h, 355
- DMREQ\_FINISH
  - tee\_api\_types.h, 183
- DMREQ\_WRITE
  - tee\_api\_types.h, 183
- DMSG
  - trace.h, 206
- DMSG\_RAW
  - trace.h, 206
- DOUBLE\_OFFSET
  - config\_bench.h, 287
- DPRINT\_STACK
  - trace.h, 206
- eapp\_entry
  - Enclave.c, 394
  - startup.c, 370
- ecall\_ta\_main
  - Enclave.c, 399
  - startup.c, 371
- EDGE\_EXTERN\_C\_BEGIN
  - Enclave\_u.h, 300
- EDGE\_EXTERN\_C\_END
  - Enclave\_u.h, 300
- EDGE\_OUT\_WITH\_STRUCTURE
  - ocalls.h, 302
- EMSG
  - trace.h, 207
- EMSG\_RAW
  - trace.h, 207
- enc\_path
  - App.cpp, 373, 376
- enclave
  - report, 49
- Enclave.c
  - \_strlen, 395
  - BUF\_SIZE, 395
  - eapp\_entry, 394
  - ecall\_ta\_main, 399
  - main, 400–402
  - MESSAGE, 394, 395, 399
  - print\_buf, 398
  - print\_pos, 398
  - run\_all\_test, 396
  - TA\_CloseSessionEntryPoint, 396
  - TA\_CreateEntryPoint, 396
  - TA\_DestroyEntryPoint, 397
  - TA\_InvokeCommandEntryPoint, 397
  - TA\_OpenSessionEntryPoint, 397
  - TEE\_PARAM\_TYPE1, 395
  - tee\_printf, 398
- Enclave.h
  - gp\_asymmetric\_key\_sign\_test, 307
  - gp\_message\_digest\_test, 307
  - gp\_random\_test, 308
  - gp\_ree\_time\_test, 308
  - gp\_secure\_storage\_test, 308
  - gp\_symmetric\_key\_enc\_verify\_test, 308
  - gp\_symmetric\_key\_gcm\_verify\_test, 308
  - gp\_trusted\_time\_test, 309

- TA\_REF\_RUN\_ALL, 306
- TA\_REF\_UUID, 306
- ENCLAVE\_FILENAME
  - App.h, 416, 418
- enclave\_report, 41
  - data, 41
  - data\_len, 41
  - hash, 41
  - signature, 41
- Enclave\_t.h
  - ocall\_file\_read, 298
  - ocall\_file\_read\_full, 298
  - ocall\_file\_write, 298
  - ocall\_file\_write\_full, 299
- Enclave\_u.h
  - \_\_wrapper\_ocall\_close\_file, 300
  - EDGE\_EXTERN\_BEGIN, 300
  - EDGE\_EXTERN\_END, 300
  - register\_functions, 301
- end
  - result, 50
- end\_hartid
  - result, 50
- EPRINT\_STACK
  - trace.h, 207
- err
  - \_sgx\_errlist\_t, 39
- events
  - pollfd, 47
- FALSE
  - App.h, 417, 418
- fct
  - out\_fct\_wrap\_type, 46
- fctprintf
  - vsnprintf.c, 250, 260
- fd
  - pollfd, 47
  - TEEC\_Context, 61
- fini\_array
  - crt.c, 357
- flags
  - \_\_TEE\_ObjectHandle, 37
  - \_\_TEE\_OperationHandle, 38
  - TEEC\_SharedMemory, 67
- flags2flags
  - tee-internal-api.c, 214, 225
- FLAGS\_CHAR
  - vsnprintf.c, 247, 253
- FLAGS\_HASH
  - vsnprintf.c, 247, 253
- FLAGS\_LEFT
  - vsnprintf.c, 247, 253
- FLAGS\_LONG
  - vsnprintf.c, 247, 253
- FLAGS\_LONG\_LONG
  - vsnprintf.c, 247, 253
- FLAGS\_PLUS
  - vsnprintf.c, 247, 253
- FLAGS\_PRECISION
  - vsnprintf.c, 247, 253
- FLAGS\_SHORT
  - vsnprintf.c, 247, 253
- FLAGS\_SPACE
  - vsnprintf.c, 247, 253
- FLAGS\_UPPERCASE
  - vsnprintf.c, 247, 253
- FLAGS\_ZEROPAD
  - vsnprintf.c, 248, 253
- FMSG
  - trace.h, 207
- FMSG\_RAW
  - trace.h, 207
- FORMAT
  - analyzer.c, 323
- FPERMS
  - tee-internal-api.c, 213, 224
- FPRINT\_STACK
  - trace.h, 207
- func\_name
  - nm\_info, 44
- GCM\_ST\_AAD
  - tee-internal-api-cryptlib.c, 266
- GCM\_ST\_ACTIVE
  - tee-internal-api-cryptlib.c, 266
- GCM\_ST\_FINAL
  - tee-internal-api-cryptlib.c, 266
- GCM\_ST\_INIT
  - tee-internal-api-cryptlib.c, 266
- get\_func\_name
  - nm\_parse.c, 326
  - nm\_parse.h, 329
- get\_key
  - nm\_parse.c, 327
- GetRelTimeEnd
  - tee-internal-api.c, 215, 225
  - tee-ta-internal.h, 83
- GetRelTimeStart
  - tee-internal-api.c, 215, 226
  - tee-ta-internal.h, 84
- global\_eid
  - App.h, 417, 418
  - types.h, 435, 437
- GP\_ASSERT
  - config\_ref\_ta.h, 311
- gp\_asymmetric\_key\_sign\_test
  - asymmetric\_key.c, 310
  - Enclave.h, 307
  - gp\_test.h, 313
- gp\_invokecommand\_test
  - gp\_test.h, 313
- gp\_message\_digest\_test
  - Enclave.h, 307
  - gp\_test.h, 313
  - message\_digest.c, 317
- gp\_random\_test
  - Enclave.h, 308



- gp\_test.h, 313
- random.c, 318
- gp\_ree\_time\_test
  - Enclave.h, 308
  - gp\_test.h, 313
  - time.c, 322
- gp\_secure\_storage\_test
  - Enclave.h, 308
  - gp\_test.h, 314
  - secure\_storage.c, 319
- gp\_symmetric\_key\_enc\_verify\_test
  - Enclave.h, 308
  - gp\_test.h, 314
  - symmetric\_key.c, 320
- gp\_symmetric\_key\_gcm\_verify\_test
  - Enclave.h, 308
  - gp\_test.h, 314
  - symmetric\_key\_gcm.c, 321
- gp\_test.h
  - gp\_asymmetric\_key\_sign\_test, 313
  - gp\_invokecommand\_test, 313
  - gp\_message\_digest\_test, 313
  - gp\_random\_test, 313
  - gp\_ree\_time\_test, 313
  - gp\_secure\_storage\_test, 314
  - gp\_symmetric\_key\_enc\_verify\_test, 314
  - gp\_symmetric\_key\_gcm\_verify\_test, 314
  - gp\_trusted\_time\_test, 314
- gp\_trusted\_time\_test
  - Enclave.h, 309
  - gp\_test.h, 314
  - time.c, 322
- handleFlags
  - TEE\_ObjectInfo, 54
- handleState
  - TEE\_OperationInfo, 55
  - TEE\_OperationInfoMultiple, 57
- hartid
  - \_profiler\_data, 35
- hash
  - enclave\_report, 41
  - sm\_report, 50
- HASH\_SIZE
  - nm\_parse.h, 329
- id
  - TEEC\_SharedMemory, 67
- idx
  - \_profiler\_header, 36
  - nm\_parse.c, 328
  - profiler\_data.h, 356
  - result, 50
- IMSG
  - trace.h, 207
- IMSG\_RAW
  - trace.h, 207
- INC
  - memory\_bench.c, 294
- init
  - bench.c, 279
- init\_array
  - crt.c, 358
- initialize\_enclave
  - App.cpp, 374, 377
- INMSG
  - trace.h, 207
- insert\_nm
  - nm\_parse.c, 327
- invoke\_command.c
  - TA\_MAX\_SIZE, 315
  - TEEP\_AGENT\_TA\_DELETE, 315
  - TEEP\_AGENT\_TA\_EXIT, 315
  - TEEP\_AGENT\_TA\_INSTALL, 315
  - TEEP\_AGENT\_TA\_LOAD, 316
  - TEEP\_AGENT\_TA\_NONE, 316
- invoke\_command\_param\_t, 41
  - a, 42
  - b, 42
  - ocalls.h, 303
  - size, 42
- invoke\_command\_t, 42
  - commandID, 43
  - ocalls.h, 303
  - params, 43
  - paramTypes, 43
- io\_bench.c
  - io\_read\_benchmark, 289
  - io\_write\_benchmark, 290
  - SPLITS, 289
- io\_read\_benchmark
  - bench.h, 283
  - io\_bench.c, 289
- IO\_READ\_SENSITIVE
  - config\_bench.h, 288
- io\_write\_benchmark
  - bench.h, 283
  - io\_bench.c, 290
- IO\_WRITE\_SENSITIVE
  - config\_bench.h, 288
- IPRINT\_STACK
  - trace.h, 207
- is\_empty
  - stack.h, 331
- keyInformation
  - TEE\_OperationInfoMultiple, 57
- keySize
  - TEE\_ObjectInfo, 54
  - TEE\_OperationInfo, 55
  - TEE\_OperationInfoKey, 56
- labels
  - bench.c, 281
- length
  - TEE\_Attribute, 52
- list, 43
  - addr, 43

- next, 44
- nm, 44
- LOG\_FILE
  - profiler\_data.h, 355
- login
  - TEE\_Identity, 53
- LOOPS\_PER\_THREAD
  - user\_types.h, 297
- main
  - analyzer.c, 323
  - App.cpp, 372, 374, 375, 377
  - crt.h, 364
  - Enclave.c, 400–402
  - main.c, 403, 405
- main.c
  - BUF\_SIZE, 405
  - main, 403, 405
  - print\_buf, 404
  - PRINT\_BUF\_SIZE, 403, 405
  - TEEC\_PARAM\_TYPE0, 405
  - TEEC\_PARAM\_TYPE1, 403, 405
- MAX\_ADDR
  - nm\_parse.c, 326
- MAX\_FUNC\_PRINT\_SIZE
  - trace.h, 208
- MAX\_PATH
  - App.cpp, 374, 377
  - App\_ocalls.cpp, 390
- MAX\_PRINT\_SIZE
  - trace.h, 208
- maxKeySize
  - TEE\_ObjectInfo, 54
  - TEE\_OperationInfo, 55
  - TEE\_OperationInfoMultiple, 57
- maxObjectSize
  - TEE\_ObjectInfo, 54
- MBEDCRYPT
  - tee\_api\_tee\_types.h, 233, 235
- MDSIZE
  - report.h, 78
- memory\_bench.c
  - INC, 294
  - random\_memory\_benchmark, 294
  - sequential\_memory\_benchmark, 295
- memref
  - TEE\_Param, 58
  - TEEC\_Parameter, 64
- MESSAGE
  - Enclave.c, 394, 395, 399
- message\_digest.c
  - gp\_message\_digest\_test, 317
  - SHA\_LENGTH, 316
  - SIG\_LENGTH, 317
- millis
  - ree\_time\_t, 48
  - TEE\_Time, 60
- mode
  - \_\_TEE\_OperationHandle, 38
  - TEE\_OperationInfo, 55
  - TEE\_OperationInfoMultiple, 57
- MSG
  - trace.h, 208
- msg
  - \_sgx\_errlist\_t, 39
- MSG\_RAW
  - trace.h, 208
- MULT\_SIZE
  - config\_bench.h, 287
- next
  - list, 44
- nfds\_t
  - tee\_api\_types.h, 184
- nm
  - list, 44
- nm.info, 44
  - func\_name, 44
  - type, 44
- nm\_parse.c
  - BUF\_SIZE, 326
  - create\_htable, 326
  - get\_func\_name, 326
  - get\_key, 327
  - idx, 328
  - insert\_nm, 327
  - MAX\_ADDR, 326
  - nm\_pool, 328
  - parse\_nm, 327
  - POOL\_SIZE, 326
- nm\_parse.h
  - get\_func\_name, 329
  - HASH\_SIZE, 329
  - parse\_nm, 329
- nm\_pool
  - nm\_parse.c, 328
- NO\_PERF
  - App\_ocalls.cpp, 386, 391
  - bench.h, 283
  - profiler\_attrs.h, 353
- nsec
  - \_\_profiler\_data, 35
- numberOfKeys
  - TEE\_OperationInfoMultiple, 57
- O\_CREAT
  - ocalls.h, 302
  - tee-internal-api.c, 214, 224
- O\_EXCL
  - ocalls.h, 302
  - tee-internal-api.c, 214, 224
- O\_RDONLY
  - ocalls.h, 303
  - tee-internal-api.c, 214, 224
- O\_RDWR
  - ocalls.h, 303
  - tee-internal-api.c, 214, 224
- O\_TRUNC

- ocalls.h, 303
  - tee-internal-api.c, 214, 224
- O\_WRONLY
  - ocalls.h, 303
  - tee-internal-api.c, 214, 225
- ob16\_t, 44
  - b, 45
  - ocalls.h, 303
  - ret, 45
- ob196\_t, 45
  - b, 45
  - ocalls.h, 303
  - ret, 45
- ob256\_t, 45
  - b, 46
  - ocalls.h, 303
  - ret, 46
- objectSize
  - TEE\_ObjectInfo, 54
- objectType
  - TEE\_ObjectInfo, 54
- objectUsage
  - TEE\_ObjectInfo, 54
- ocall\_close\_file
  - App\_ocalls.cpp, 379, 383, 386, 391
  - App\_ocalls.h, 419, 427
  - ocalls.h, 303
- ocall\_file\_read
  - Enclave.t.h, 298
- ocall\_file\_read\_full
  - Enclave.t.h, 298
- ocall\_file\_write
  - Enclave.t.h, 298
- ocall\_file\_write\_full
  - Enclave.t.h, 299
- ocall\_fstat\_size
  - ocalls.h, 304
- ocall\_getrandom
  - App\_ocalls.cpp, 379, 387
- ocall\_getrandom16
  - ocalls.h, 304
- ocall\_getrandom196
  - ocalls.h, 305
- ocall\_invoke\_command\_callback\_write
  - App\_ocalls.cpp, 379, 387
- ocall\_open\_file
  - App\_ocalls.cpp, 380, 383, 388, 391
  - App\_ocalls.h, 420, 428
  - ocalls.h, 305
- ocall\_print\_string
  - App\_ocalls.cpp, 380, 384, 388, 391
  - App\_ocalls.h, 422, 429
  - ocalls.h, 305
- ocall\_print\_string\_wrapper
  - ocall\_wrapper.c, 369, 370
  - ocall\_wrapper.h, 365
- ocall\_pull\_invoke\_command
  - ocalls.h, 305
- ocall\_put\_invoke\_command\_result
  - App\_ocalls.cpp, 381
  - ocalls.h, 305
- ocall\_read\_file
  - App\_ocalls.cpp, 381, 384, 389, 392
  - App\_ocalls.h, 422, 430
- ocall\_read\_file256
  - ocalls.h, 305
- ocall\_read\_invoke\_param
  - App\_ocalls.cpp, 381
  - ocalls.h, 305
- ocall\_ree\_time
  - App\_ocalls.cpp, 381, 385, 389, 392
  - App\_ocalls.h, 424, 431
  - ocalls.h, 305
- ocall\_unlink
  - ocalls.h, 305
- ocall\_wrapper.c
  - ocall\_print\_string\_wrapper, 369, 370
- ocall\_wrapper.h
  - ocall\_print\_string\_wrapper, 365
- ocall\_write\_file
  - App\_ocalls.cpp, 382, 385, 389, 393
  - App\_ocalls.h, 425, 432
- ocall\_write\_file256
  - ocalls.h, 306
- ocall\_write\_invoke\_param
  - App\_ocalls.cpp, 382
  - ocalls.h, 306
- ocalls.h
  - EDGE\_OUT\_WITH\_STRUCTURE, 302
  - invoke\_command\_param\_t, 303
  - invoke\_command\_t, 303
  - O\_CREAT, 302
  - O\_EXCL, 302
  - O\_RDONLY, 303
  - O\_RDWR, 303
  - O\_TRUNC, 303
  - O\_WRONLY, 303
  - ob16\_t, 303
  - ob196\_t, 303
  - ob256\_t, 303
  - ocall\_close\_file, 303
  - ocall\_fstat\_size, 304
  - ocall\_getrandom16, 304
  - ocall\_getrandom196, 305
  - ocall\_open\_file, 305
  - ocall\_print\_string, 305
  - ocall\_pull\_invoke\_command, 305
  - ocall\_put\_invoke\_command\_result, 305
  - ocall\_read\_file256, 305
  - ocall\_read\_invoke\_param, 305
  - ocall\_ree\_time, 305
  - ocall\_unlink, 305
  - ocall\_write\_file256, 306
  - ocall\_write\_invoke\_param, 306
  - param\_buffer\_t, 303
  - ree\_time\_t, 303

- OFFSET
  - config\_bench.h, 287
- offset
  - TEEC\_RegisteredMemoryReference, 65
- OpenPersistentObject
  - tee-internal-api.c, 215, 226
- operationClass
  - TEE\_OperationInfo, 55
  - TEE\_OperationInfoMultiple, 57
- operationState
  - TEE\_OperationInfoMultiple, 57
- out\_fct\_type
  - vsnprintf.c, 248, 254
- out\_fct\_wrap\_type, 46
  - arg, 46
  - fct, 46
- OUTMSG
  - trace.h, 208
- OUTRMSG
  - trace.h, 208
- param\_buffer\_t, 46
  - buf, 47
  - ocalls.h, 303
  - size, 47
- params
  - invoke\_command\_t, 43
  - TEEC\_Operation, 63
- paramTypes
  - invoke\_command\_t, 43
  - TEEC\_Operation, 63
- parent
  - TEEC\_RegisteredMemoryReference, 65
- parse\_nm
  - nm\_parse.c, 327
  - nm\_parse.h, 329
- perf\_buffer
  - tee\_config.h, 338, 340, 341
- PERF\_SECTION
  - profiler\_attrs.h, 353
- PERF\_SIZE
  - profiler\_data.h, 355
- persist\_ctx
  - \_\_TEE\_ObjectHandle, 37
- pollfd, 47
  - events, 47
  - fd, 47
  - revents, 47
- POOL\_SIZE
  - nm\_parse.c, 326
- pop
  - stack.h, 331
- pos
  - stack.h, 332
- pr\_deb
  - tee-common.h, 80
- prikey
  - \_\_TEE\_OperationHandle, 38
- print\_buf
  - Enclave.c, 398
  - main.c, 404
- PRINT\_BUF\_SIZE
  - main.c, 403, 405
- print\_error\_message
  - App.cpp, 374, 378
- print\_pos
  - Enclave.c, 398
- printf
  - tools.c, 336
  - tools.h, 367
- PRINTF\_FTOA\_BUFFER\_SIZE
  - vsnprintf.c, 248, 254
- PRINTF\_NTOA\_BUFFER\_SIZE
  - vsnprintf.c, 248, 254
- PRINTF\_SUPPORT\_FLOAT
  - vsnprintf.c, 248, 254
- PRINTF\_SUPPORT\_LONG\_LONG
  - vsnprintf.c, 248, 254
- PRINTF\_SUPPORT\_PTRDIFF\_T
  - vsnprintf.c, 248, 254
- private\_key
  - \_\_TEE\_ObjectHandle, 37
- profiler.c
  - \_\_cyg\_profile\_func, 350
  - \_\_cyg\_profile\_func\_enter, 350
  - \_\_cyg\_profile\_func\_exit, 351
  - \_\_profiler\_get\_data\_ptr, 351
  - \_\_profiler\_head, 352
  - \_\_profiler\_map\_info, 351
- profiler.h
  - \_\_profiler\_map\_info, 352
- profiler\_attrs.h
  - NO\_PERF, 353
  - PERF\_SECTION, 353
  - USED, 353
- profiler\_data.h
  - \_\_attribute\_\_, 356
  - \_\_profiler\_nsec\_t, 355
  - CALL, 355
  - direction\_t, 355
  - idx, 356
  - LOG\_FILE, 355
  - PERF\_SIZE, 355
  - RET, 355
  - size, 356
  - START, 355
  - start, 356
- profiler\_write
  - tee\_profiler.c, 342, 344, 345
  - tools.c, 332–335
- pubkey
  - \_\_TEE\_OperationHandle, 38
- public\_key
  - \_\_TEE\_ObjectHandle, 37
  - sm\_report, 50
- PUBLIC\_KEY\_SIZE
  - report.h, 78

- push
  - stack.h, 332
- putchar
  - tools.c, 337
  - tools.h, 367
  - vsnprintf.c, 250, 262
- puts
  - tools.c, 337
  - tools.h, 368
- random.c
  - gp\_random\_test, 318
- random\_memory\_benchmark
  - bench.h, 284
  - memory\_bench.c, 294
- RANDOM\_MEMORY\_SENSITIVE
  - config\_bench.h, 288
- record
  - bench.c, 279
  - config\_bench.h, 288
- ree\_time\_t, 48
  - App\_ocalls.h, 419, 427
  - millis, 48
  - ocalls.h, 303
  - seconds, 48
- REE\_TIME\_TEST
  - config\_bench.h, 288
- ree\_time\_test
  - bench.h, 284
  - time\_test.c, 296
- ref
  - TEE\_Attribute, 52
- reg\_mem
  - TEEC\_Context, 62
- register\_functions
  - Enclave\_u.h, 301
- registered\_fd
  - TEEC\_SharedMemory, 68
- report, 48
  - dev\_public\_key, 49
  - enclave, 49
  - sm, 49
- report.h
  - ATTEST\_DATA\_MAXLEN, 78
  - MDSIZE, 78
  - PUBLIC\_KEY\_SIZE, 78
  - SIGNATURE\_SIZE, 79
- requiredKeyUsage
  - TEE\_OperationInfo, 55
  - TEE\_OperationInfoKey, 56
- result, 49
  - callee, 49
  - depth, 49
  - end, 50
  - end\_hartid, 50
  - idx, 50
  - start, 50
  - start\_hartid, 50
- RET
  - profiler\_data.h, 355
- ret
  - ob16\_t, 45
  - ob196\_t, 45
  - ob256\_t, 46
- revents
  - pollfd, 47
- run\_all\_test
  - crt.c, 359
  - Enclave.c, 396
- runtime\_path
  - App.cpp, 373, 376
- seconds
  - ree\_time\_t, 48
  - TEE\_Time, 60
- secure\_storage.c
  - DATA\_LENGTH, 318
  - gp\_secure\_storage\_test, 319
- selectResponseEnable
  - TEE\_SEReaderProperties, 59
- sePresent
  - TEE\_SEReaderProperties, 60
- sequential\_memory\_benchmark
  - bench.h, 284
  - memory\_bench.c, 295
- SEQUENTIAL\_MEMORY\_SENSITIVE
  - config\_bench.h, 288
- session
  - TEEC\_Operation, 63
- session\_id
  - TEEC\_Session, 66
- set\_object\_key
  - tee-internal-api.c, 216, 227
- sgx\_errlist
  - types.h, 435, 437
- sgx\_errlist\_t
  - types.h, 435, 437
- SHA\_LENGTH
  - asymmetric\_key.c, 310
  - message\_digest.c, 316
  - tee\_api\_tee\_types.h, 233, 235
- shadow\_buffer
  - TEEC\_SharedMemory, 68
- SIG\_LENGTH
  - asymmetric\_key.c, 310
  - message\_digest.c, 317
  - tee-internal-api-cryptlib.c, 266
- signature
  - enclave\_report, 41
  - sm\_report, 51
- SIGNATURE\_SIZE
  - report.h, 79
- size
  - \_profiler\_header, 36
  - invoke\_command\_param\_t, 42
  - param\_buffer\_t, 47
  - profiler\_data.h, 356
  - TEE\_Param, 58

- TEEC\_RegisteredMemoryReference, 65
- TEEC\_SharedMemory, 68
- TEEC\_TempMemoryReference, 69
- sm
  - report, 49
- sm\_report, 50
  - hash, 50
  - public\_key, 50
  - signature, 51
- SMSG
  - trace.h, 208
- snprintf
  - vsnprintf.c, 251, 262
- socklen\_t
  - tee\_api\_types.h, 184
- SPLITS
  - io\_bench.c, 289
- sprintf
  - vsnprintf.c, 251, 262
- stack
  - stack.h, 332
- stack.h
  - is\_empty, 331
  - pop, 331
  - pos, 332
  - push, 332
  - stack, 332
  - STACK\_SIZE, 331
- STACK\_SIZE
  - stack.h, 331
- START
  - profiler\_data.h, 355
- start
  - \_profiler\_header, 36
  - profiler\_data.h, 356
  - result, 50
- start\_hartid
  - result, 50
- started
  - TEEC\_Operation, 63
- startup.c
  - eapp\_entry, 370
  - ecall\_ta\_main, 371
- sug
  - \_sgx\_errlist\_t, 39
- symmetric.key.c
  - CIPHER\_LENGTH, 320
  - gp\_symmetric\_key\_enc\_verify\_test, 320
- symmetric.key\_gcm.c
  - CIPHER\_LENGTH, 321
  - gp\_symmetric\_key\_gcm\_verify\_test, 321
- SYSTEM\_TIME\_TEST
  - config\_bench.h, 288
- system\_time\_test
  - bench.h, 285
  - time\_test.c, 296
- ta-ref/api/include/compiler.h, 71
- ta-ref/api/include/report.h, 78
- ta-ref/api/include/tee-common.h, 79
- ta-ref/api/include/tee-ta-internal.h, 80
- ta-ref/api/include/tee\_api.h, 104
- ta-ref/api/include/tee\_api\_defines.h, 140
- ta-ref/api/include/tee\_api\_defines\_extensions.h, 178
- ta-ref/api/include/tee\_api\_types.h, 182
- ta-ref/api/include/tee\_client\_api.h, 186
- ta-ref/api/include/tee\_internal\_api.h, 198
- ta-ref/api/include/tee\_internal\_api\_extensions.h, 199
- ta-ref/api/include/tee\_ta\_api.h, 202
- ta-ref/api/include/test\_dev\_key.h, 204
- ta-ref/api/include/trace.h, 205
- ta-ref/api/include/trace\_levels.h, 210
- ta-ref/api/keystone/tee-internal-api-machine.c, 211
- ta-ref/api/keystone/tee-internal-api.c, 212
- ta-ref/api/keystone/tee\_api\_tee\_types.h, 232
- ta-ref/api/keystone/teec\_stub.c, 236
- ta-ref/api/keystone/trace.c, 239
- ta-ref/api/keystone/vsnprintf.c, 245
- ta-ref/api/optee/tee\_api\_tee\_types.h, 234
- ta-ref/api/sgx/tee-internal-api.c, 223
- ta-ref/api/sgx/tee\_api\_tee\_types.h, 234
- ta-ref/api/tee-internal-api-cryptlib.c, 264
- ta-ref/benchmark/bench.c, 278
- ta-ref/benchmark/bench.h, 282
- ta-ref/benchmark/cpu\_bench.c, 285
- ta-ref/benchmark/include/config\_bench.h, 286
- ta-ref/benchmark/io\_bench.c, 288
- ta-ref/benchmark/keystone/tee\_def.h, 290
- ta-ref/benchmark/memory\_bench.c, 294
- ta-ref/benchmark/optee/tee\_def.h, 291
- ta-ref/benchmark/sgx/tee\_def.h, 292
- ta-ref/benchmark/time\_test.c, 295
- ta-ref/docs/building.md, 296
- ta-ref/docs/gp\_api.md, 296
- ta-ref/docs/how\_to\_program\_on\_ta-ref.md, 296
- ta-ref/docs/overview\_of\_ta-ref.md, 296
- ta-ref/docs/preparation.md, 296
- ta-ref/docs/running\_on\_dev\_boards.md, 296
- ta-ref/edger/edger8r/user\_types.h, 296
- ta-ref/edger/keyedge/Enclave\_t.c, 297
- ta-ref/edger/keyedge/Enclave\_t.h, 298
- ta-ref/edger/keyedge/Enclave\_u.c, 299
- ta-ref/edger/keyedge/Enclave\_u.h, 300
- ta-ref/edger/keyedge/ocalls.h, 301
- ta-ref/edger/optee/Enclave.h, 306
- ta-ref/edger/optee/Enclave\_t.h, 299
- ta-ref/gp/asymmetric.key.c, 309
- ta-ref/gp/include/config\_ref\_ta.h, 310
- ta-ref/gp/include/gp\_test.h, 312
- ta-ref/gp/invoke\_command.c, 315
- ta-ref/gp/message\_digest.c, 316
- ta-ref/gp/random.c, 317
- ta-ref/gp/secure\_storage.c, 318
- ta-ref/gp/symmetric.key.c, 319
- ta-ref/gp/symmetric.key\_gcm.c, 320
- ta-ref/gp/time.c, 321
- ta-ref/profiler/analyzer/analyzer.c, 322

- ta-ref/profiler/analyzer/analyzer.h, 324
- ta-ref/profiler/analyzer/nm\_parse.c, 325
- ta-ref/profiler/analyzer/nm\_parse.h, 328
- ta-ref/profiler/analyzer/stack.h, 330
- ta-ref/profiler/app/tools.c, 332
- ta-ref/profiler/keystone/Enclave/tools.c, 333
- ta-ref/profiler/keystone/tee\_config.h, 338
- ta-ref/profiler/keystone/tee\_profiler.c, 341
- ta-ref/profiler/keystone/tee\_profiler.h, 347
- ta-ref/profiler/optee/Enclave/tools.c, 334
- ta-ref/profiler/optee/tee\_config.h, 339
- ta-ref/profiler/optee/tee\_profiler.c, 343
- ta-ref/profiler/optee/tee\_profiler.h, 348
- ta-ref/profiler/profiler.c, 349
- ta-ref/profiler/profiler.h, 352
- ta-ref/profiler/profiler\_attr.h, 353
- ta-ref/profiler/profiler\_data.h, 354
- ta-ref/profiler/sgx/Enclave/tools.c, 335
- ta-ref/profiler/sgx/tee\_config.h, 340
- ta-ref/profiler/sgx/tee\_profiler.c, 344
- ta-ref/profiler/sgx/tee\_profiler.h, 348
- ta-ref/test\_gp/crt.c, 356
- ta-ref/test\_gp/include/crt.h, 363
- ta-ref/test\_gp/include/ocall\_wrapper.h, 365
- ta-ref/test\_gp/include/random.h, 366
- ta-ref/test\_gp/include/tools.h, 367
- ta-ref/test\_gp/keystone/App/App.cpp, 375
- ta-ref/test\_gp/keystone/App/App\_ocalls.cpp, 385
- ta-ref/test\_gp/keystone/Enclave/Enclave.c, 400
- ta-ref/test\_gp/keystone/Enclave/ocall\_wrapper.c, 368
- ta-ref/test\_gp/keystone/Enclave/startup.c, 370
- ta-ref/test\_gp/keystone/Enclave/trace.c, 241
- ta-ref/test\_gp/optee/App/main.c, 404
- ta-ref/test\_gp/optee/Enclave/crt.c, 358
- ta-ref/test\_gp/optee/Enclave/Enclave.c, 401
- ta-ref/test\_gp/optee/Enclave/trace.c, 242
- ta-ref/test\_gp/optee/Enclave/user\_ta\_header.c, 409
- ta-ref/test\_gp/optee/Enclave/user\_ta\_header\_defines.h, 414
- ta-ref/test\_gp/sgx/App/App.cpp, 376
- ta-ref/test\_gp/sgx/App/App.h, 417
- ta-ref/test\_gp/sgx/App/App\_ocalls.cpp, 390
- ta-ref/test\_gp/sgx/App/App\_ocalls.h, 425
- ta-ref/test\_gp/sgx/App/types.h, 436
- ta-ref/test\_gp/sgx/Enclave/Enclave.c, 402
- ta-ref/test\_gp/sgx/Enclave/Enclave.h, 307
- ta-ref/test\_gp/sgx/Enclave/ocall\_wrapper.c, 369
- ta-ref/test\_gp/sgx/Enclave/startup.c, 371
- ta-ref/test\_gp/sgx/Enclave/trace.c, 244
- ta-ref/test\_gp/tools.c, 336
- ta-ref/test\_gp/vsnprintf.c, 251
- ta-ref/test\_hello/keystone/App/App.cpp, 372
- ta-ref/test\_hello/keystone/App/App\_ocalls.cpp, 378
- ta-ref/test\_hello/keystone/Enclave/Enclave.c, 393
- ta-ref/test\_hello/optee/App/main.c, 402
- ta-ref/test\_hello/optee/Enclave/Enclave.c, 394
- ta-ref/test\_hello/optee/Enclave/user\_ta\_header.c, 406
- ta-ref/test\_hello/optee/Enclave/user\_ta\_header\_defines.h, 412
- ta-ref/test\_hello/sgx/App/App.cpp, 373
- ta-ref/test\_hello/sgx/App/App.h, 416
- ta-ref/test\_hello/sgx/App/App\_ocalls.cpp, 382
- ta-ref/test\_hello/sgx/App/App\_ocalls.h, 418
- ta-ref/test\_hello/sgx/App/types.h, 434
- ta-ref/test\_hello/sgx/Enclave/Enclave.c, 399
- TA\_CloseSessionEntryPoint
  - crt.c, 360
  - Enclave.c, 396
  - tee\_ta\_api.h, 203
- TA\_CreateEntryPoint
  - crt.c, 360
  - Enclave.c, 396
  - tee\_ta\_api.h, 203
- TA\_CURRENT\_TA\_EXT\_PROPERTIES
  - user\_ta\_header\_defines.h, 413, 415
- TA\_DATA\_SIZE
  - user\_ta\_header\_defines.h, 413, 415
- TA\_DESCRIPTION
  - user\_ta\_header.c, 407, 410
  - user\_ta\_header\_defines.h, 413, 415
- TA\_DestroyEntryPoint
  - crt.c, 360
  - Enclave.c, 397
  - tee\_ta\_api.h, 203
- TA\_EXPORT
  - tee\_ta\_api.h, 202
- TA\_FLAGS
  - user\_ta\_header\_defines.h, 413, 415
- TA\_FRAMEWORK\_STACK\_SIZE
  - user\_ta\_header.c, 407, 410
- ta\_heap
  - user\_ta\_header.c, 408, 411
- ta\_heap\_size
  - user\_ta\_header.c, 408, 411
- TA\_InvokeCommandEntryPoint
  - crt.c, 360
  - Enclave.c, 397
  - tee\_ta\_api.h, 203
- TA\_MAX\_SIZE
  - invoke\_command.c, 315
- ta\_num\_props
  - user\_ta\_header.c, 408, 411
- TA\_OpenSessionEntryPoint
  - crt.c, 361
  - Enclave.c, 397
  - tee\_ta\_api.h, 203
- ta\_props
  - user\_ta\_header.c, 408, 411
- TA\_REF\_RUN\_ALL
  - Enclave.h, 306
- TA\_REF\_UUID
  - Enclave.h, 306
- TA\_STACK\_SIZE
  - user\_ta\_header\_defines.h, 414, 415
- TA\_UUID



- user\_ta\_header\_defines.h, 414, 415
- TA.VERSION
  - user\_ta\_header.c, 407, 410
  - user\_ta\_header\_defines.h, 414, 415
- tahead\_get\_trace\_level
  - user\_ta\_header.c, 408, 411
- tee-common.h
  - pr\_deb, 80
- tee-internal-api-cryptlib.c
  - GCM.ST\_AAD, 266
  - GCM.ST\_ACTIVE, 266
  - GCM.ST\_FINAL, 266
  - GCM.ST\_INIT, 266
  - SIG.LENGTH, 266
  - TEE.AEDecryptFinal, 267
  - TEE.AEEncryptFinal, 267
  - TEE.AEInit, 268
  - TEE.AEUpdate, 269
  - TEE.AEUpdateAAD, 269
  - TEE.AllocateOperation, 270
  - TEE.AllocateTransientObject, 270
  - TEE.AsymmetricSignDigest, 271
  - TEE.AsymmetricVerifyDigest, 272
  - TEE.CipherDoFinal, 272
  - TEE.CipherInit, 273
  - TEE.CipherUpdate, 273
  - TEE.DigestDoFinal, 274
  - TEE.DigestUpdate, 274
  - TEE.FreeOperation, 275
  - TEE.FreeTransientObject, 275
  - TEE.GenerateKey, 276
  - TEE.InitRefAttribute, 276
  - TEE.InitValueAttribute, 277
  - TEE.SetOperationKey, 277
  - wolfSSL.Free, 278
  - wolfSSL.Malloc, 278
- tee-internal-api-machine.c
  - \_\_attribute\_\_, 212
- tee-internal-api.c
  - \_\_attribute\_\_, 225
  - flags2flags, 214, 225
  - FPERMS, 213, 224
  - GetRelTimeEnd, 215, 225
  - GetRelTimeStart, 215, 226
  - O\_CREAT, 214, 224
  - O\_EXCL, 214, 224
  - O\_RDONLY, 214, 224
  - O\_RDWR, 214, 224
  - O\_TRUNC, 214, 224
  - O\_WRONLY, 214, 225
  - OpenPersistentObject, 215, 226
  - set\_object\_key, 216, 227
  - TEE.CloseObject, 216, 227
  - TEE.CreatePersistentObject, 217, 228
  - TEE.Free, 218
  - TEE.GenerateRandom, 218, 228
  - TEE.GetObjectInfo1, 219, 229
  - TEE.GetREETime, 219, 229
  - TEE.GetSystemTime, 220, 230
  - TEE.Malloc, 220
  - TEE.OpenPersistentObject, 220, 230
  - TEE.ReadObjectData, 221, 230
  - TEE.Realloc, 221
  - TEE.WriteObjectData, 222, 231
- tee-ta-internal.h
  - \_\_attribute\_\_, 82
  - GetRelTimeEnd, 83
  - GetRelTimeStart, 84
  - TEE.AEDecryptFinal, 84
  - TEE.AEEncryptFinal, 85
  - TEE.AEInit, 86
  - TEE.AEUpdate, 86
  - TEE.AEUpdateAAD, 87
  - TEE.AllocateOperation, 88
  - TEE.AllocateTransientObject, 88
  - TEE.AsymmetricSignDigest, 89
  - TEE.AsymmetricVerifyDigest, 90
  - TEE.CipherInit, 90
  - TEE.CipherUpdate, 91
  - TEE.CloseObject, 92
  - TEE.CreatePersistentObject, 92
  - TEE.DigestDoFinal, 94
  - TEE.DigestUpdate, 94
  - TEE.FreeOperation, 95
  - TEE.FreeTransientObject, 95
  - TEE.GenerateKey, 96
  - TEE.GenerateRandom, 96
  - TEE.GetObjectInfo1, 97
  - TEE.GetREETime, 98
  - TEE.GetSystemTime, 99
  - TEE.InitRefAttribute, 99
  - TEE.InitValueAttribute, 100
  - TEE.OpenPersistentObject, 100
  - TEE.ReadObjectData, 101
  - TEE.SetOperationKey, 102
  - TEE.WriteObjectData, 103
- TEE.AEDecryptFinal
  - tee-internal-api-cryptlib.c, 267
  - tee-ta-internal.h, 84
  - tee\_api.h, 108
- TEE.AEEncryptFinal
  - tee-internal-api-cryptlib.c, 267
  - tee-ta-internal.h, 85
  - tee\_api.h, 109
- TEE.AEInit
  - tee-internal-api-cryptlib.c, 268
  - tee-ta-internal.h, 86
  - tee\_api.h, 109
- TEE.AEUpdate
  - tee-internal-api-cryptlib.c, 269
  - tee-ta-internal.h, 86
  - tee\_api.h, 110
- TEE.AEUpdateAAD
  - tee-internal-api-cryptlib.c, 269
  - tee-ta-internal.h, 87
  - tee\_api.h, 111



- TEE\_ALG\_AES\_CBC\_MAC\_NOPAD  
tee\_api\_defines.h, [146](#)
- TEE\_ALG\_AES\_CBC\_MAC\_PKCS5  
tee\_api\_defines.h, [146](#)
- TEE\_ALG\_AES\_CBC\_NOPAD  
tee\_api\_defines.h, [146](#)
- TEE\_ALG\_AES\_CCM  
tee\_api\_defines.h, [147](#)
- TEE\_ALG\_AES\_CMAC  
tee\_api\_defines.h, [147](#)
- TEE\_ALG\_AES\_CTR  
tee\_api\_defines.h, [147](#)
- TEE\_ALG\_AES\_CTS  
tee\_api\_defines.h, [147](#)
- TEE\_ALG\_AES\_ECB\_NOPAD  
tee\_api\_defines.h, [147](#)
- TEE\_ALG\_AES\_GCM  
tee\_api\_defines.h, [147](#)
- TEE\_ALG\_AES\_XTS  
tee\_api\_defines.h, [147](#)
- TEE\_ALG\_CONCAT\_KDF\_SHA1\_DERIVE\_KEY  
tee\_api\_defines\_extensions.h, [179](#)
- TEE\_ALG\_CONCAT\_KDF\_SHA224\_DERIVE\_KEY  
tee\_api\_defines\_extensions.h, [179](#)
- TEE\_ALG\_CONCAT\_KDF\_SHA256\_DERIVE\_KEY  
tee\_api\_defines\_extensions.h, [179](#)
- TEE\_ALG\_CONCAT\_KDF\_SHA384\_DERIVE\_KEY  
tee\_api\_defines\_extensions.h, [179](#)
- TEE\_ALG\_CONCAT\_KDF\_SHA512\_DERIVE\_KEY  
tee\_api\_defines\_extensions.h, [179](#)
- TEE\_ALG\_DES3\_CBC\_MAC\_NOPAD  
tee\_api\_defines.h, [147](#)
- TEE\_ALG\_DES3\_CBC\_MAC\_PKCS5  
tee\_api\_defines.h, [147](#)
- TEE\_ALG\_DES3\_CBC\_NOPAD  
tee\_api\_defines.h, [147](#)
- TEE\_ALG\_DES3\_ECB\_NOPAD  
tee\_api\_defines.h, [147](#)
- TEE\_ALG\_DES\_CBC\_MAC\_NOPAD  
tee\_api\_defines.h, [148](#)
- TEE\_ALG\_DES\_CBC\_MAC\_PKCS5  
tee\_api\_defines.h, [148](#)
- TEE\_ALG\_DES\_CBC\_NOPAD  
tee\_api\_defines.h, [148](#)
- TEE\_ALG\_DES\_ECB\_NOPAD  
tee\_api\_defines.h, [148](#)
- TEE\_ALG\_DH\_DERIVE\_SHARED\_SECRET  
tee\_api\_defines.h, [148](#)
- TEE\_ALG\_DSA\_SHA1  
tee\_api\_defines.h, [148](#)
- TEE\_ALG\_DSA\_SHA224  
tee\_api\_defines.h, [148](#)
- TEE\_ALG\_DSA\_SHA256  
tee\_api\_defines.h, [148](#)
- TEE\_ALG\_ECDH\_P192  
tee\_api\_defines.h, [148](#)
- TEE\_ALG\_ECDH\_P224  
tee\_api\_defines.h, [148](#)
- TEE\_ALG\_ECDH\_P256  
tee\_api\_defines.h, [148](#)
- TEE\_ALG\_ECDH\_P384  
tee\_api\_defines.h, [149](#)
- TEE\_ALG\_ECDH\_P521  
tee\_api\_defines.h, [149](#)
- TEE\_ALG\_ECDSA\_P192  
tee\_api\_defines.h, [149](#)
- TEE\_ALG\_ECDSA\_P224  
tee\_api\_defines.h, [149](#)
- TEE\_ALG\_ECDSA\_P256  
tee\_api\_defines.h, [149](#)
- TEE\_ALG\_ECDSA\_P384  
tee\_api\_defines.h, [149](#)
- TEE\_ALG\_ECDSA\_P521  
tee\_api\_defines.h, [149](#)
- TEE\_ALG\_HKDF\_MD5\_DERIVE\_KEY  
tee\_api\_defines\_extensions.h, [179](#)
- TEE\_ALG\_HKDF\_SHA1\_DERIVE\_KEY  
tee\_api\_defines\_extensions.h, [179](#)
- TEE\_ALG\_HKDF\_SHA224\_DERIVE\_KEY  
tee\_api\_defines\_extensions.h, [179](#)
- TEE\_ALG\_HKDF\_SHA256\_DERIVE\_KEY  
tee\_api\_defines\_extensions.h, [179](#)
- TEE\_ALG\_HKDF\_SHA384\_DERIVE\_KEY  
tee\_api\_defines\_extensions.h, [179](#)
- TEE\_ALG\_HKDF\_SHA512\_DERIVE\_KEY  
tee\_api\_defines\_extensions.h, [180](#)
- TEE\_ALG\_HMAC\_MD5  
tee\_api\_defines.h, [149](#)
- TEE\_ALG\_HMAC\_SHA1  
tee\_api\_defines.h, [149](#)
- TEE\_ALG\_HMAC\_SHA224  
tee\_api\_defines.h, [149](#)
- TEE\_ALG\_HMAC\_SHA256  
tee\_api\_defines.h, [149](#)
- TEE\_ALG\_HMAC\_SHA384  
tee\_api\_defines.h, [150](#)
- TEE\_ALG\_HMAC\_SHA512  
tee\_api\_defines.h, [150](#)
- TEE\_ALG\_MD5  
tee\_api\_defines.h, [150](#)
- TEE\_ALG\_MD5SHA1  
tee\_api\_defines.h, [150](#)
- TEE\_ALG\_PBKDF2\_HMAC\_SHA1\_DERIVE\_KEY  
tee\_api\_defines\_extensions.h, [180](#)
- TEE\_ALG\_RSA\_NOPAD  
tee\_api\_defines.h, [150](#)
- TEE\_ALG\_RSAES\_PKCS1\_OAEP\_MGF1\_SHA1  
tee\_api\_defines.h, [150](#)
- TEE\_ALG\_RSAES\_PKCS1\_OAEP\_MGF1\_SHA224  
tee\_api\_defines.h, [150](#)
- TEE\_ALG\_RSAES\_PKCS1\_OAEP\_MGF1\_SHA256  
tee\_api\_defines.h, [150](#)
- TEE\_ALG\_RSAES\_PKCS1\_OAEP\_MGF1\_SHA384  
tee\_api\_defines.h, [150](#)
- TEE\_ALG\_RSAES\_PKCS1\_OAEP\_MGF1\_SHA512  
tee\_api\_defines.h, [150](#)

- TEE\_ALG\_RSAES\_PKCS1\_V1\_5
  - tee\_api\_defines.h, [150](#)
- TEE\_ALG\_RSASSA\_PKCS1\_PSS\_MGF1\_SHA1
  - tee\_api\_defines.h, [151](#)
- TEE\_ALG\_RSASSA\_PKCS1\_PSS\_MGF1\_SHA224
  - tee\_api\_defines.h, [151](#)
- TEE\_ALG\_RSASSA\_PKCS1\_PSS\_MGF1\_SHA256
  - tee\_api\_defines.h, [151](#)
- TEE\_ALG\_RSASSA\_PKCS1\_PSS\_MGF1\_SHA384
  - tee\_api\_defines.h, [151](#)
- TEE\_ALG\_RSASSA\_PKCS1\_PSS\_MGF1\_SHA512
  - tee\_api\_defines.h, [151](#)
- TEE\_ALG\_RSASSA\_PKCS1\_V1\_5\_MD5
  - tee\_api\_defines.h, [151](#)
- TEE\_ALG\_RSASSA\_PKCS1\_V1\_5\_MD5SHA1
  - tee\_api\_defines.h, [151](#)
- TEE\_ALG\_RSASSA\_PKCS1\_V1\_5\_SHA1
  - tee\_api\_defines.h, [151](#)
- TEE\_ALG\_RSASSA\_PKCS1\_V1\_5\_SHA224
  - tee\_api\_defines.h, [151](#)
- TEE\_ALG\_RSASSA\_PKCS1\_V1\_5\_SHA256
  - tee\_api\_defines.h, [151](#)
- TEE\_ALG\_RSASSA\_PKCS1\_V1\_5\_SHA384
  - tee\_api\_defines.h, [152](#)
- TEE\_ALG\_RSASSA\_PKCS1\_V1\_5\_SHA512
  - tee\_api\_defines.h, [152](#)
- TEE\_ALG\_SHA1
  - tee\_api\_defines.h, [152](#)
- TEE\_ALG\_SHA224
  - tee\_api\_defines.h, [152](#)
- TEE\_ALG\_SHA256
  - tee\_api\_defines.h, [152](#)
- TEE\_ALG\_SHA384
  - tee\_api\_defines.h, [152](#)
- TEE\_ALG\_SHA512
  - tee\_api\_defines.h, [152](#)
- TEE\_AllocateOperation
  - tee-internal-api-cryptlib.c, [270](#)
  - tee-ta-internal.h, [88](#)
  - tee\_api.h, [111](#)
- TEE\_AllocatePersistentObjectEnumerator
  - tee\_api.h, [112](#)
- TEE\_AllocatePropertyEnumerator
  - tee\_api.h, [112](#)
- TEE\_AllocateTransientObject
  - tee-internal-api-cryptlib.c, [270](#)
  - tee-ta-internal.h, [88](#)
  - tee\_api.h, [112](#)
- tee\_api.h
  - TEE\_AEDecryptFinal, [108](#)
  - TEE\_AEEncryptFinal, [109](#)
  - TEE\_AEInit, [109](#)
  - TEE\_AEUpdate, [110](#)
  - TEE\_AEUpdateAAD, [111](#)
  - TEE\_AllocateOperation, [111](#)
  - TEE\_AllocatePersistentObjectEnumerator, [112](#)
  - TEE\_AllocatePropertyEnumerator, [112](#)
  - TEE\_AllocateTransientObject, [112](#)
  - TEE\_AsymmetricDecrypt, [112](#)
  - TEE\_AsymmetricEncrypt, [113](#)
  - TEE\_AsymmetricSignDigest, [113](#)
  - TEE\_AsymmetricVerifyDigest, [113](#)
  - TEE\_BigIntAdd, [114](#)
  - TEE\_BigIntAddMod, [114](#)
  - TEE\_BigIntCmp, [114](#)
  - TEE\_BigIntCmpS32, [114](#)
  - TEE\_BigIntComputeExtendedGcd, [115](#)
  - TEE\_BigIntComputeFMM, [115](#)
  - TEE\_BigIntConvertFromFMM, [115](#)
  - TEE\_BigIntConvertFromOctetString, [115](#)
  - TEE\_BigIntConvertFromS32, [115](#)
  - TEE\_BigIntConvertToFMM, [115](#)
  - TEE\_BigIntConvertToOctetString, [115](#)
  - TEE\_BigIntConvertToS32, [116](#)
  - TEE\_BigIntDiv, [116](#)
  - TEE\_BigIntFMMContextSizeInU32, [116](#)
  - TEE\_BigIntFMMConvertToBigInt, [116](#)
  - TEE\_BigIntFMMSizeInU32, [116](#)
  - TEE\_BigIntGetBit, [116](#)
  - TEE\_BigIntGetBitCount, [116](#)
  - TEE\_BigIntInit, [116](#)
  - TEE\_BigIntInitFMM, [117](#)
  - TEE\_BigIntInitFMMContext, [117](#)
  - TEE\_BigIntInvMod, [117](#)
  - TEE\_BigIntIsProbablePrime, [117](#)
  - TEE\_BigIntMod, [117](#)
  - TEE\_BigIntMul, [117](#)
  - TEE\_BigIntMulMod, [117](#)
  - TEE\_BigIntNeg, [117](#)
  - TEE\_BigIntRelativePrime, [118](#)
  - TEE\_BigIntShiftRight, [118](#)
  - TEE\_BigIntSquare, [118](#)
  - TEE\_BigIntSquareMod, [118](#)
  - TEE\_BigIntSub, [118](#)
  - TEE\_BigIntSubMod, [118](#)
  - TEE\_CheckMemoryAccessRights, [118](#)
  - TEE\_CipherDoFinal, [119](#)
  - TEE\_CipherInit, [119](#)
  - TEE\_CipherUpdate, [120](#)
  - TEE\_CloseAndDeletePersistentObject, [120](#)
  - TEE\_CloseAndDeletePersistentObject1, [120](#)
  - TEE\_CloseObject, [120](#)
  - TEE\_CloseTASession, [121](#)
  - TEE\_CopyObjectAttributes, [121](#)
  - TEE\_CopyObjectAttributes1, [121](#)
  - TEE\_CopyOperation, [121](#)
  - TEE\_CreatePersistentObject, [122](#)
  - TEE\_DeriveKey, [123](#)
  - TEE\_DigestDoFinal, [123](#)
  - TEE\_DigestUpdate, [123](#)
  - TEE\_Free, [124](#)
  - TEE\_FreeOperation, [124](#)
  - TEE\_FreePersistentObjectEnumerator, [125](#)
  - TEE\_FreePropertyEnumerator, [125](#)
  - TEE\_FreeTransientObject, [125](#)
  - TEE\_GenerateKey, [125](#)

TEE\_GenerateRandom, 126  
 TEE\_GetCancellationFlag, 127  
 TEE\_GetInstanceData, 127  
 TEE\_GetNextPersistentObject, 127  
 TEE\_GetNextProperty, 127  
 TEE\_GetObjectBufferAttribute, 127  
 TEE\_GetObjectInfo, 127  
 TEE\_GetObjectInfo1, 128  
 TEE\_GetObjectValueAttribute, 128  
 TEE\_GetOperationInfo, 128  
 TEE\_GetOperationInfoMultiple, 129  
 TEE\_GetPropertyAsBinaryBlock, 129  
 TEE\_GetPropertyAsBool, 129  
 TEE\_GetPropertyAsIdentity, 129  
 TEE\_GetPropertyAsString, 129  
 TEE\_GetPropertyAsU32, 129  
 TEE\_GetPropertyAsUUID, 129  
 TEE\_GetPropertyName, 130  
 TEE\_GetREETime, 130  
 TEE\_GetSystemTime, 130  
 TEE\_GetTAPersistentTime, 131  
 TEE\_InitRefAttribute, 131  
 TEE\_InitValueAttribute, 131  
 TEE\_InvokeTACommand, 132  
 TEE\_IsAlgorithmSupported, 132  
 TEE\_MACCompareFinal, 132  
 TEE\_MACComputeFinal, 132  
 TEE\_MACInit, 132  
 TEE\_MACUpdate, 133  
 TEE\_Malloc, 133  
 TEE\_MaskCancellation, 133  
 TEE\_MemCompare, 133  
 TEE\_MemFill, 133  
 TEE\_MemMove, 134  
 TEE\_OpenPersistentObject, 134  
 TEE\_OpenTASession, 135  
 TEE\_Panic, 135  
 TEE\_PopulateTransientObject, 135  
 TEE\_ReadObjectData, 135  
 TEE\_Realloc, 136  
 TEE\_RenamePersistentObject, 136  
 TEE\_ResetOperation, 137  
 TEE\_ResetPersistentObjectEnumerator, 137  
 TEE\_ResetPropertyEnumerator, 137  
 TEE\_ResetTransientObject, 137  
 TEE\_RestrictObjectUsage, 137  
 TEE\_RestrictObjectUsage1, 137  
 TEE\_SeekObjectData, 137  
 TEE\_SetInstanceData, 137  
 TEE\_SetOperationKey, 137  
 TEE\_SetOperationKey2, 138  
 TEE\_SetTAPersistentTime, 138  
 TEE\_StartPersistentObjectEnumerator, 138  
 TEE\_StartPropertyEnumerator, 138  
 TEE\_TruncateObjectData, 138  
 TEE\_UnmaskCancellation, 139  
 TEE\_Wait, 139  
 TEE\_WriteObjectData, 139

## tee\_api\_defines.h

TEE\_ALG\_AES\_CBC\_MAC\_NOPAD, 146  
 TEE\_ALG\_AES\_CBC\_MAC\_PKCS5, 146  
 TEE\_ALG\_AES\_CBC\_NOPAD, 146  
 TEE\_ALG\_AES\_CCM, 147  
 TEE\_ALG\_AES\_CMAC, 147  
 TEE\_ALG\_AES\_CTR, 147  
 TEE\_ALG\_AES\_CTS, 147  
 TEE\_ALG\_AES\_ECB\_NOPAD, 147  
 TEE\_ALG\_AES\_GCM, 147  
 TEE\_ALG\_AES\_XTS, 147  
 TEE\_ALG\_DES3\_CBC\_MAC\_NOPAD, 147  
 TEE\_ALG\_DES3\_CBC\_MAC\_PKCS5, 147  
 TEE\_ALG\_DES3\_CBC\_NOPAD, 147  
 TEE\_ALG\_DES3\_ECB\_NOPAD, 147  
 TEE\_ALG\_DES\_CBC\_MAC\_NOPAD, 148  
 TEE\_ALG\_DES\_CBC\_MAC\_PKCS5, 148  
 TEE\_ALG\_DES\_CBC\_NOPAD, 148  
 TEE\_ALG\_DES\_ECB\_NOPAD, 148  
 TEE\_ALG\_DH\_DERIVE\_SHARED\_SECRET, 148  
 TEE\_ALG\_DSA\_SHA1, 148  
 TEE\_ALG\_DSA\_SHA224, 148  
 TEE\_ALG\_DSA\_SHA256, 148  
 TEE\_ALG\_ECDH\_P192, 148  
 TEE\_ALG\_ECDH\_P224, 148  
 TEE\_ALG\_ECDH\_P256, 148  
 TEE\_ALG\_ECDH\_P384, 149  
 TEE\_ALG\_ECDH\_P521, 149  
 TEE\_ALG\_ECDSA\_P192, 149  
 TEE\_ALG\_ECDSA\_P224, 149  
 TEE\_ALG\_ECDSA\_P256, 149  
 TEE\_ALG\_ECDSA\_P384, 149  
 TEE\_ALG\_ECDSA\_P521, 149  
 TEE\_ALG\_HMAC\_MD5, 149  
 TEE\_ALG\_HMAC\_SHA1, 149  
 TEE\_ALG\_HMAC\_SHA224, 149  
 TEE\_ALG\_HMAC\_SHA256, 149  
 TEE\_ALG\_HMAC\_SHA384, 150  
 TEE\_ALG\_HMAC\_SHA512, 150  
 TEE\_ALG\_MD5, 150  
 TEE\_ALG\_MD5SHA1, 150  
 TEE\_ALG\_RSA\_NOPAD, 150  
 TEE\_ALG\_RSAES\_PKCS1\_OAEP\_MGF1\_SHA1, 150  
 TEE\_ALG\_RSAES\_PKCS1\_OAEP\_MGF1\_SHA224, 150  
 TEE\_ALG\_RSAES\_PKCS1\_OAEP\_MGF1\_SHA256, 150  
 TEE\_ALG\_RSAES\_PKCS1\_OAEP\_MGF1\_SHA384, 150  
 TEE\_ALG\_RSAES\_PKCS1\_OAEP\_MGF1\_SHA512, 150  
 TEE\_ALG\_RSAES\_PKCS1\_V1\_5, 150  
 TEE\_ALG\_RSASSA\_PKCS1\_PSS\_MGF1\_SHA1, 151  
 TEE\_ALG\_RSASSA\_PKCS1\_PSS\_MGF1\_SHA224, 151

- TEE.ALG\_RSASSA\_PKCS1\_PSS\_MGF1\_SHA256, 151
- TEE.ALG\_RSASSA\_PKCS1\_PSS\_MGF1\_SHA384, 151
- TEE.ALG\_RSASSA\_PKCS1\_PSS\_MGF1\_SHA512, 151
- TEE.ALG\_RSASSA\_PKCS1\_V1\_5\_MD5, 151
- TEE.ALG\_RSASSA\_PKCS1\_V1\_5\_MD5\_SHA1, 151
- TEE.ALG\_RSASSA\_PKCS1\_V1\_5\_SHA1, 151
- TEE.ALG\_RSASSA\_PKCS1\_V1\_5\_SHA224, 151
- TEE.ALG\_RSASSA\_PKCS1\_V1\_5\_SHA256, 151
- TEE.ALG\_RSASSA\_PKCS1\_V1\_5\_SHA384, 152
- TEE.ALG\_RSASSA\_PKCS1\_V1\_5\_SHA512, 152
- TEE.ALG\_SHA1, 152
- TEE.ALG\_SHA224, 152
- TEE.ALG\_SHA256, 152
- TEE.ALG\_SHA384, 152
- TEE.ALG\_SHA512, 152
- TEE.ATTR\_BIT\_PROTECTED, 152
- TEE.ATTR\_BIT\_VALUE, 152
- TEE.ATTR\_DH\_BASE, 152
- TEE.ATTR\_DH\_PRIME, 152
- TEE.ATTR\_DH\_PRIVATE\_VALUE, 153
- TEE.ATTR\_DH\_PUBLIC\_VALUE, 153
- TEE.ATTR\_DH\_SUBPRIME, 153
- TEE.ATTR\_DH\_X\_BITS, 153
- TEE.ATTR\_DSA\_BASE, 153
- TEE.ATTR\_DSA\_PRIME, 153
- TEE.ATTR\_DSA\_PRIVATE\_VALUE, 153
- TEE.ATTR\_DSA\_PUBLIC\_VALUE, 153
- TEE.ATTR\_DSA\_SUBPRIME, 153
- TEE.ATTR\_ECC\_CURVE, 153
- TEE.ATTR\_ECC\_PRIVATE\_VALUE, 153
- TEE.ATTR\_ECC\_PUBLIC\_VALUE\_X, 154
- TEE.ATTR\_ECC\_PUBLIC\_VALUE\_Y, 154
- TEE.ATTR\_RSA\_COEFFICIENT, 154
- TEE.ATTR\_RSA\_EXPONENT1, 154
- TEE.ATTR\_RSA\_EXPONENT2, 154
- TEE.ATTR\_RSA\_MODULUS, 154
- TEE.ATTR\_RSA\_OAEP\_LABEL, 154
- TEE.ATTR\_RSA\_PRIME1, 154
- TEE.ATTR\_RSA\_PRIME2, 154
- TEE.ATTR\_RSA\_PRIVATE\_EXPONENT, 154
- TEE.ATTR\_RSA\_PSS\_SALT\_LENGTH, 154
- TEE.ATTR\_RSA\_PUBLIC\_EXPONENT, 155
- TEE.ATTR\_SECRET\_VALUE, 155
- TEE.BigIntSizeInU32, 155
- TEE.DATA\_FLAG\_ACCESS\_READ, 155
- TEE.DATA\_FLAG\_ACCESS\_WRITE, 155
- TEE.DATA\_FLAG\_ACCESS\_WRITE\_META, 155
- TEE.DATA\_FLAG\_OVERWRITE, 155
- TEE.DATA\_FLAG\_SHARE\_READ, 155
- TEE.DATA\_FLAG\_SHARE\_WRITE, 155
- TEE.DATA\_MAX\_POSITION, 155
- TEE.ECC\_CURVE\_NIST\_P192, 155
- TEE.ECC\_CURVE\_NIST\_P224, 156
- TEE.ECC\_CURVE\_NIST\_P256, 156
- TEE.ECC\_CURVE\_NIST\_P384, 156
- TEE.ECC\_CURVE\_NIST\_P521, 156
- TEE.ERROR\_ACCESS\_CONFLICT, 156
- TEE.ERROR\_ACCESS\_DENIED, 156
- TEE.ERROR\_BAD\_FORMAT, 156
- TEE.ERROR\_BAD\_PARAMETERS, 156
- TEE.ERROR\_BAD\_STATE, 156
- TEE.ERROR\_BUSY, 156
- TEE.ERROR\_CANCEL, 156
- TEE.ERROR\_COMMUNICATION, 157
- TEE.ERROR\_CORRUPT\_OBJECT, 157
- TEE.ERROR\_CORRUPT\_OBJECT\_2, 157
- TEE.ERROR\_EXCESS\_DATA, 157
- TEE.ERROR\_EXTERNAL\_CANCEL, 157
- TEE.ERROR\_GENERIC, 157
- TEE.ERROR\_ITEM\_NOT\_FOUND, 157
- TEE.ERROR\_MAC\_INVALID, 157
- TEE.ERROR\_NO\_DATA, 157
- TEE.ERROR\_NOT\_IMPLEMENTED, 157
- TEE.ERROR\_NOT\_SUPPORTED, 157
- TEE.ERROR\_OUT\_OF\_MEMORY, 158
- TEE.ERROR\_OVERFLOW, 158
- TEE.ERROR\_SECURITY, 158
- TEE.ERROR\_SHORT\_BUFFER, 158
- TEE.ERROR\_SIGNATURE\_INVALID, 158
- TEE.ERROR\_STORAGE\_NO\_SPACE, 158
- TEE.ERROR\_STORAGE\_NOT\_AVAILABLE, 158
- TEE.ERROR\_STORAGE\_NOT\_AVAILABLE\_2, 158
- TEE.ERROR\_TARGET\_DEAD, 158
- TEE.ERROR\_TIME\_NEEDS\_RESET, 158
- TEE.ERROR\_TIME\_NOT\_SET, 158
- TEE.HANDLE\_FLAG\_EXPECT\_TWO\_KEYS, 159
- TEE.HANDLE\_FLAG\_INITIALIZED, 159
- TEE.HANDLE\_FLAG\_KEY\_SET, 159
- TEE.HANDLE\_FLAG\_PERSISTENT, 159
- TEE.HANDLE\_NULL, 159
- TEE.INT\_CORE\_API\_SPEC\_VERSION, 159
- TEE.LOGIN\_APPLICATION, 159
- TEE.LOGIN\_APPLICATION\_GROUP, 159
- TEE.LOGIN\_APPLICATION\_USER, 159
- TEE.LOGIN\_GROUP, 159
- TEE.LOGIN\_PUBLIC, 159
- TEE.LOGIN\_TRUSTED\_APP, 160
- TEE.LOGIN\_USER, 160
- TEE.MALLOC\_FILL\_ZERO, 160
- TEE.MEMORY\_ACCESS\_ANY\_OWNER, 160
- TEE.MEMORY\_ACCESS\_READ, 160
- TEE.MEMORY\_ACCESS\_WRITE, 160
- TEE.NUM\_PARAMS, 160
- TEE.OBJECT\_ID\_MAX\_LEN, 160
- TEE.OPERATION\_AE, 160
- TEE.OPERATION\_ASYMMETRIC\_CIPHER, 160
- TEE.OPERATION\_ASYMMETRIC\_SIGNATURE, 160
- TEE.OPERATION\_CIPHER, 161
- TEE.OPERATION\_DIGEST, 161
- TEE.OPERATION\_KEY\_DERIVATION, 161
- TEE.OPERATION\_MAC, 161
- TEE.OPERATION\_STATE\_ACTIVE, 161

- TEE\_OPERATION\_STATE\_INITIAL, [161](#)
- TEE\_ORIGIN\_API, [161](#)
- TEE\_ORIGIN\_COMMS, [161](#)
- TEE\_ORIGIN\_TEE, [161](#)
- TEE\_ORIGIN\_TRUSTED\_APP, [161](#)
- TEE\_PANIC.ID\_TA\_CLOSESESSIONENTRYPOINT, [161](#)
- TEE\_PANIC.ID\_TA\_CREATEENTRYPOINT, [162](#)
- TEE\_PANIC.ID\_TA\_DESTROYENTRYPOINT, [162](#)
- TEE\_PANIC.ID\_TA\_INVOKECOMMANDENTRYPOINT, [162](#)
- TEE\_PANIC.ID\_TA\_OPENSESSIONENTRYPOINT, [162](#)
- TEE\_PANIC.ID\_TEE\_AEDECRIPTFINAL, [162](#)
- TEE\_PANIC.ID\_TEE\_AEENCRYPTFINAL, [162](#)
- TEE\_PANIC.ID\_TEE\_AEINIT, [162](#)
- TEE\_PANIC.ID\_TEE\_AEUPDATE, [162](#)
- TEE\_PANIC.ID\_TEE\_AEUPDATEAAD, [162](#)
- TEE\_PANIC.ID\_TEE\_ALLOCATEOPERATION, [162](#)
- TEE\_PANIC.ID\_TEE\_ALLOCATEPERSISTENTOBJECTENUMERATOR, [162](#)
- TEE\_PANIC.ID\_TEE\_ALLOCATEPROPERTYENUMERATOR, [163](#)
- TEE\_PANIC.ID\_TEE\_ALLOCATETRANSIENTOBJECT, [163](#)
- TEE\_PANIC.ID\_TEE\_ASYMMETRICDECRYPT, [163](#)
- TEE\_PANIC.ID\_TEE\_ASYMMETRICENCRYPT, [163](#)
- TEE\_PANIC.ID\_TEE\_ASYMMETRICSIGNDIGEST, [163](#)
- TEE\_PANIC.ID\_TEE\_ASYMMETRICVERIFYDIGEST, [163](#)
- TEE\_PANIC.ID\_TEE\_BIGINTADD, [163](#)
- TEE\_PANIC.ID\_TEE\_BIGINTADDMOD, [163](#)
- TEE\_PANIC.ID\_TEE\_BIGINTCMP, [163](#)
- TEE\_PANIC.ID\_TEE\_BIGINTCMPS32, [163](#)
- TEE\_PANIC.ID\_TEE\_BIGINTCOMPUTEEXTENDEDGCD, [163](#)
- TEE\_PANIC.ID\_TEE\_BIGINTCOMPUTEFM, [164](#)
- TEE\_PANIC.ID\_TEE\_BIGINTCONVERTFROMFM, [164](#)
- TEE\_PANIC.ID\_TEE\_BIGINTCONVERTFROMOCTETSTRING, [164](#)
- TEE\_PANIC.ID\_TEE\_BIGINTCONVERTFROMS32, [164](#)
- TEE\_PANIC.ID\_TEE\_BIGINTCONVERTTOFM, [164](#)
- TEE\_PANIC.ID\_TEE\_BIGINTCONVERTTOOCTETSTRING, [164](#)
- TEE\_PANIC.ID\_TEE\_BIGINTCONVERTTOS32, [164](#)
- TEE\_PANIC.ID\_TEE\_BIGINTDIV, [164](#)
- TEE\_PANIC.ID\_TEE\_BIGINTFMCONTEXTSIZEINU32, [164](#)
- TEE\_PANIC.ID\_TEE\_BIGINTFMMSIZEINU32, [164](#)
- TEE\_PANIC.ID\_TEE\_BIGINTGETBIT, [165](#)
- TEE\_PANIC.ID\_TEE\_BIGINTGETBITCOUNT, [165](#)
- TEE\_PANIC.ID\_TEE\_BIGINTINIT, [165](#)
- TEE\_PANIC.ID\_TEE\_BIGINTINITFM, [165](#)
- TEE\_PANIC.ID\_TEE\_BIGINTINITFMCONTEXT, [165](#)
- TEE\_PANIC.ID\_TEE\_BIGINTINVMOD, [165](#)
- TEE\_PANIC.ID\_TEE\_BIGINTISPROBABLEPRIME, [165](#)
- TEE\_PANIC.ID\_TEE\_BIGINTMOD, [165](#)
- TEE\_PANIC.ID\_TEE\_BIGINTMUL, [165](#)
- TEE\_PANIC.ID\_TEE\_BIGINTMULMOD, [165](#)
- TEE\_PANIC.ID\_TEE\_BIGINTNEG, [165](#)
- TEE\_PANIC.ID\_TEE\_BIGINTRELATIVEPRIME, [166](#)
- TEE\_PANIC.ID\_TEE\_BIGINTSHIFTRIGHT, [166](#)
- TEE\_PANIC.ID\_TEE\_BIGINTSQUARE, [166](#)
- TEE\_PANIC.ID\_TEE\_BIGINTSQUAREMOD, [166](#)
- TEE\_PANIC.ID\_TEE\_BIGINTSUB, [166](#)
- TEE\_PANIC.ID\_TEE\_BIGINTSUBMOD, [166](#)
- TEE\_PANIC.ID\_TEE\_CHECKMEMORYACCESSRIGHTS, [166](#)
- TEE\_PANIC.ID\_TEE\_CIPHERDOFINAL, [166](#)
- TEE\_PANIC.ID\_TEE\_CIPHERINIT, [166](#)
- TEE\_PANIC.ID\_TEE\_CIPHERUPDATE, [166](#)
- TEE\_PANIC.ID\_TEE\_CLOSEANDDELETERPERSISTENTOBJECT, [166](#)
- TEE\_PANIC.ID\_TEE\_CLOSEANDDELETERPERSISTENTOBJECT1, [167](#)
- TEE\_PANIC.ID\_TEE\_CLOSEOBJECT, [167](#)
- TEE\_PANIC.ID\_TEE\_CLOSETASESSION, [167](#)
- TEE\_PANIC.ID\_TEE\_COPYOBJECTATTRIBUTES, [167](#)
- TEE\_PANIC.ID\_TEE\_COPYOBJECTATTRIBUTES1, [167](#)
- TEE\_PANIC.ID\_TEE\_COPYOPERATION, [167](#)
- TEE\_PANIC.ID\_TEE\_CREATEPERSISTENTOBJECT, [167](#)
- TEE\_PANIC.ID\_TEE\_DERIVEKEY, [167](#)
- TEE\_PANIC.ID\_TEE\_DIGESTDOFINAL, [167](#)
- TEE\_PANIC.ID\_TEE\_DIGESTUPDATE, [167](#)
- TEE\_PANIC.ID\_TEE\_FREE, [167](#)
- TEE\_PANIC.ID\_TEE\_FREEOPERATION, [168](#)
- TEE\_PANIC.ID\_TEE\_FREEPERSISTENTOBJECTENUMERATOR, [168](#)
- TEE\_PANIC.ID\_TEE\_FREEPROPERTYENUMERATOR, [168](#)
- TEE\_PANIC.ID\_TEE\_FREETRANSIENTOBJECT, [168](#)
- TEE\_PANIC.ID\_TEE\_GENERATEKEY, [168](#)
- TEE\_PANIC.ID\_TEE\_GENERATERANDOM, [168](#)
- TEE\_PANIC.ID\_TEE\_GETCANCELLATIONFLAG, [168](#)
- TEE\_PANIC.ID\_TEE\_GETINSTANCEDATA, [168](#)
- TEE\_PANIC.ID\_TEE\_GETNEXTPERSISTENTOBJECT, [168](#)
- TEE\_PANIC.ID\_TEE\_GETNEXTPROPERTY, [168](#)
- TEE\_PANIC.ID\_TEE\_GETOBJECTBUFFERATTRIBUTE, [168](#)
- TEE\_PANIC.ID\_TEE\_GETOBJECTINFO, [169](#)

- TEE.PANIC.ID.TEE.GETOBJECTINFO1, 169
- TEE.PANIC.ID.TEE.GETOBJECTVALUEATTRIBUTE, 169
- TEE.PANIC.ID.TEE.GETOPERATIONINFO, 169
- TEE.PANIC.ID.TEE.GETOPERATIONINFOMULTIPLE, 169
- TEE.PANIC.ID.TEE.GETPROPERTYASBINARYBLOCK, 169
- TEE.PANIC.ID.TEE.GETPROPERTYASBOOL, 169
- TEE.PANIC.ID.TEE.GETPROPERTYASIDENTITY, 169
- TEE.PANIC.ID.TEE.GETPROPERTYASSTRING, 169
- TEE.PANIC.ID.TEE.GETPROPERTYASU32, 169
- TEE.PANIC.ID.TEE.GETPROPERTYASUUID, 169
- TEE.PANIC.ID.TEE.GETPROPERTYNAME, 170
- TEE.PANIC.ID.TEE.GETREETIME, 170
- TEE.PANIC.ID.TEE.GETSYSTEMTIME, 170
- TEE.PANIC.ID.TEE.GETTAPERSISTENTTIME, 170
- TEE.PANIC.ID.TEE.INITREFATTRIBUTE, 170
- TEE.PANIC.ID.TEE.INITVALUEATTRIBUTE, 170
- TEE.PANIC.ID.TEE.INVOKETACOMMAND, 170
- TEE.PANIC.ID.TEE.MACCOMPAREFINAL, 170
- TEE.PANIC.ID.TEE.MACCOMPUTEFINAL, 170
- TEE.PANIC.ID.TEE.MACINIT, 170
- TEE.PANIC.ID.TEE.MACUPDATE, 170
- TEE.PANIC.ID.TEE.MALLOC, 171
- TEE.PANIC.ID.TEE.MASKCANCELLATION, 171
- TEE.PANIC.ID.TEE.MEMCOMPARE, 171
- TEE.PANIC.ID.TEE.MEMFILL, 171
- TEE.PANIC.ID.TEE.MEMMOVE, 171
- TEE.PANIC.ID.TEE.OPENPERSISTENTOBJECT, 171
- TEE.PANIC.ID.TEE.OPENTASESSION, 171
- TEE.PANIC.ID.TEE.PANIC, 171
- TEE.PANIC.ID.TEE.POPULATETRANSIENTOBJECT, 171
- TEE.PANIC.ID.TEE.READOBJECTDATA, 171
- TEE.PANIC.ID.TEE.REALLOC, 171
- TEE.PANIC.ID.TEE.RENAMEPERSISTENTOBJECT, 172
- TEE.PANIC.ID.TEE.RESETOPERATION, 172
- TEE.PANIC.ID.TEE.RESETPERSISTENTOBJECTENUMERATOR, 172
- TEE.PANIC.ID.TEE.RESETPROPERTYENUMERATOR, 172
- TEE.PANIC.ID.TEE.RESETTRANSIENTOBJECT, 172
- TEE.PANIC.ID.TEE.RESTRICTOBJECTUSAGE, 172
- TEE.PANIC.ID.TEE.RESTRICTOBJECTUSAGE1, 172
- TEE.PANIC.ID.TEE.SEEKOBJECTDATA, 172
- TEE.PANIC.ID.TEE.SETINSTANCEDATA, 172
- TEE.PANIC.ID.TEE.SETOPERATIONKEY, 172
- TEE.PANIC.ID.TEE.SETOPERATIONKEY2, 172
- TEE.PANIC.ID.TEE.SETTAPERSISTENTTIME, 173
- TEE.PANIC.ID.TEE.STARTPERSISTENTOBJECTENUMERATOR, 173
- TEE.PANIC.ID.TEE.STARTPROPERTYENUMERATOR, 173
- TEE.PANIC.ID.TEE.TRUNCATEOBJECTDATA, 173
- TEE.PANIC.ID.TEE.UNMASKCANCELLATION, 173
- TEE.PANIC.ID.TEE.WAIT, 173
- TEE.PANIC.ID.TEE.WRITEOBJECTDATA, 173
- TEE.PARAM.TYPE.GET, 173
- TEE.PARAM.TYPE.MEMREF\_INOUT, 173
- TEE.PARAM.TYPE.MEMREF\_INPUT, 173
- TEE.PARAM.TYPE.MEMREF\_OUTPUT, 174
- TEE.PARAM.TYPE.NONE, 174
- TEE.PARAM.TYPE.SET, 174
- TEE.PARAM.TYPE.VALUE\_INOUT, 174
- TEE.PARAM.TYPE.VALUE\_INPUT, 174
- TEE.PARAM.TYPE.VALUE\_OUTPUT, 174
- TEE.PARAM.TYPES, 174
- TEE.PROPSET.CURRENT\_CLIENT, 174
- TEE.PROPSET.CURRENT\_TA, 174
- TEE.PROPSET.TEE\_IMPLEMENTATION, 174
- TEE.STORAGE.PRIVATE, 175
- TEE.SUCCESS, 175
- TEE.TIMEOUT.INFINITE, 175
- TEE.TYPE.AES, 175
- TEE.TYPE.CORRUPTED\_OBJECT, 175
- TEE.TYPE.DATA, 175
- TEE.TYPE.DES, 175
- TEE.TYPE.DES3, 175
- TEE.TYPE.DH.KEYPAIR, 175
- TEE.TYPE.DSA.KEYPAIR, 175
- TEE.TYPE.DSA.PUBLIC\_KEY, 175
- TEE.TYPE.ECDH.KEYPAIR, 176
- TEE.TYPE.ECDH.PUBLIC\_KEY, 176
- TEE.TYPE.ECDSA.KEYPAIR, 176
- TEE.TYPE.ECDSA.PUBLIC\_KEY, 176
- TEE.TYPE.GENERIC.SECRET, 176
- TEE.TYPE.HMAC.MD5, 176
- TEE.TYPE.HMAC.SHA1, 176
- TEE.TYPE.HMAC.SHA224, 176
- TEE.TYPE.HMAC.SHA256, 176
- TEE.TYPE.HMAC.SHA384, 176
- TEE.TYPE.HMAC.SHA512, 176
- TEE.TYPE.RSA.KEYPAIR, 177
- TEE.TYPE.RSA.PUBLIC\_KEY, 177
- TEE.USAGE.DECRYPT, 177
- TEE.USAGE.DERIVE, 177
- TEE.USAGE.ENCRYPT, 177
- TEE.USAGE.EXTRACTABLE, 177
- TEE.USAGE.MAC, 177
- TEE.USAGE.SIGN, 177
- TEE.USAGE.VERIFY, 177
- tee\_api.defines.extensions.h



- TEE\_ALG\_CONCAT\_KDF\_SHA1\_DERIVE\_KEY, 179
- TEE\_ALG\_CONCAT\_KDF\_SHA224\_DERIVE\_KEY, 179
- TEE\_ALG\_CONCAT\_KDF\_SHA256\_DERIVE\_KEY, 179
- TEE\_ALG\_CONCAT\_KDF\_SHA384\_DERIVE\_KEY, 179
- TEE\_ALG\_CONCAT\_KDF\_SHA512\_DERIVE\_KEY, 179
- TEE\_ALG\_HKDF\_MD5\_DERIVE\_KEY, 179
- TEE\_ALG\_HKDF\_SHA1\_DERIVE\_KEY, 179
- TEE\_ALG\_HKDF\_SHA224\_DERIVE\_KEY, 179
- TEE\_ALG\_HKDF\_SHA256\_DERIVE\_KEY, 179
- TEE\_ALG\_HKDF\_SHA384\_DERIVE\_KEY, 179
- TEE\_ALG\_HKDF\_SHA512\_DERIVE\_KEY, 180
- TEE\_ALG\_PBKDF2\_HMAC\_SHA1\_DERIVE\_KEY, 180
- TEE\_ATTR\_CONCAT\_KDF\_DKM\_LENGTH, 180
- TEE\_ATTR\_CONCAT\_KDF\_OTHER\_INFO, 180
- TEE\_ATTR\_CONCAT\_KDF\_Z, 180
- TEE\_ATTR\_HKDF\_IKM, 180
- TEE\_ATTR\_HKDF\_INFO, 180
- TEE\_ATTR\_HKDF\_OKM\_LENGTH, 180
- TEE\_ATTR\_HKDF\_SALT, 180
- TEE\_ATTR\_PBKDF2\_DKM\_LENGTH, 180
- TEE\_ATTR\_PBKDF2\_ITERATION\_COUNT, 180
- TEE\_ATTR\_PBKDF2\_PASSWORD, 181
- TEE\_ATTR\_PBKDF2\_SALT, 181
- TEE\_MEMORY\_ACCESS\_NONSECURE, 181
- TEE\_MEMORY\_ACCESS\_SECURE, 181
- TEE\_STORAGE\_PRIVATE\_REE, 181
- TEE\_STORAGE\_PRIVATE\_RPMB, 181
- TEE\_STORAGE\_PRIVATE\_SQL\_RESERVED, 181
- TEE\_TYPE\_CONCAT\_KDF\_Z, 181
- TEE\_TYPE\_HKDF\_IKM, 181
- TEE\_TYPE\_PBKDF2\_PASSWORD, 181
- tee\_api.tee.types.h
  - AES256, 233, 235
  - MBEDCRYPT, 233, 235
  - SHA\_LENGTH, 233, 235
  - TEE\_HANDLE\_NULL, 235
  - TEE\_OBJECT\_AAD\_SIZE, 233, 235
  - TEE\_OBJECT\_KEY\_SIZE, 233, 235
  - TEE\_OBJECT\_NONCE\_SIZE, 233, 235
  - TEE\_OBJECT\_SKEY\_SIZE, 233, 235
  - TEE\_OBJECT\_TAG\_SIZE, 233, 236
  - WOLFCRYPT, 233, 236
- tee\_api.types.h
  - \_aligned, 184
  - DMREQ\_FINISH, 183
  - DMREQ\_WRITE, 183
  - nfds\_t, 184
  - socklen\_t, 184
  - TEE\_BigInt, 184
  - TEE\_BigIntFMM, 184
  - TEE\_DATA\_SEEK\_CUR, 186
  - TEE\_DATA\_SEEK\_END, 186
  - TEE\_DATA\_SEEK\_SET, 186
  - TEE\_ErrorOrigin, 185
  - TEE\_MEM\_INPUT, 183
  - TEE\_MEM\_OUTPUT, 184
  - TEE\_MEMREF\_0\_USED, 184
  - TEE\_MEMREF\_1\_USED, 184
  - TEE\_MEMREF\_2\_USED, 184
  - TEE\_MEMREF\_3\_USED, 184
  - TEE\_MODE\_DECRYPT, 186
  - TEE\_MODE\_DERIVE, 186
  - TEE\_MODE\_DIGEST, 186
  - TEE\_MODE\_ENCRYPT, 186
  - TEE\_MODE\_MAC, 186
  - TEE\_MODE\_SIGN, 186
  - TEE\_MODE\_VERIFY, 186
  - TEE\_ObjectEnumHandle, 185
  - TEE\_ObjectHandle, 185
  - TEE\_ObjectType, 185
  - TEE\_OperationHandle, 185
  - TEE\_OperationMode, 186
  - TEE\_PropSetHandle, 185
  - TEE\_Result, 185
  - TEE\_SE\_READER\_NAME\_MAX, 184
  - TEE\_SEChannelHandle, 185
  - TEE\_SEReaderHandle, 185
  - TEESessionHandle, 185
  - TEESessionHandle, 185
  - TEE\_Session, 186
  - TEETASessionHandle, 186
  - TEE\_Whence, 186
- TEE\_AsymmetricDecrypt
  - tee\_api.h, 112
- TEE\_AsymmetricEncrypt
  - tee\_api.h, 113
- TEE\_AsymmetricSignDigest
  - tee-internal-api-cryptlib.c, 271
  - tee-ta-internal.h, 89
  - tee\_api.h, 113
- TEE\_AsymmetricVerifyDigest
  - tee-internal-api-cryptlib.c, 272
  - tee-ta-internal.h, 90
  - tee\_api.h, 113
- TEE\_ATTR\_BIT\_PROTECTED
  - tee\_api.defines.h, 152
- TEE\_ATTR\_BIT\_VALUE
  - tee\_api.defines.h, 152
- TEE\_ATTR\_CONCAT\_KDF\_DKM\_LENGTH
  - tee\_api.defines\_extensions.h, 180
- TEE\_ATTR\_CONCAT\_KDF\_OTHER\_INFO
  - tee\_api.defines\_extensions.h, 180
- TEE\_ATTR\_CONCAT\_KDF\_Z
  - tee\_api.defines\_extensions.h, 180
- TEE\_ATTR\_DH\_BASE
  - tee\_api.defines.h, 152
- TEE\_ATTR\_DH\_PRIME
  - tee\_api.defines.h, 152
- TEE\_ATTR\_DH\_PRIVATE\_VALUE
  - tee\_api.defines.h, 153

- TEE\_ATTR\_DH\_PUBLIC.VALUE  
tee\_api\_defines.h, 153
- TEE\_ATTR\_DH\_SUBPRIME  
tee\_api\_defines.h, 153
- TEE\_ATTR\_DH\_X\_BITS  
tee\_api\_defines.h, 153
- TEE\_ATTR\_DSA\_BASE  
tee\_api\_defines.h, 153
- TEE\_ATTR\_DSA\_PRIME  
tee\_api\_defines.h, 153
- TEE\_ATTR\_DSA\_PRIVATE.VALUE  
tee\_api\_defines.h, 153
- TEE\_ATTR\_DSA\_PUBLIC.VALUE  
tee\_api\_defines.h, 153
- TEE\_ATTR\_DSA\_SUBPRIME  
tee\_api\_defines.h, 153
- TEE\_ATTR\_ECC\_CURVE  
tee\_api\_defines.h, 153
- TEE\_ATTR\_ECC\_PRIVATE.VALUE  
tee\_api\_defines.h, 153
- TEE\_ATTR\_ECC\_PUBLIC.VALUE\_X  
tee\_api\_defines.h, 154
- TEE\_ATTR\_ECC\_PUBLIC.VALUE\_Y  
tee\_api\_defines.h, 154
- TEE\_ATTR\_HKDF\_IKM  
tee\_api\_defines\_extensions.h, 180
- TEE\_ATTR\_HKDF\_INFO  
tee\_api\_defines\_extensions.h, 180
- TEE\_ATTR\_HKDF\_OKM.LENGTH  
tee\_api\_defines\_extensions.h, 180
- TEE\_ATTR\_HKDF\_SALT  
tee\_api\_defines\_extensions.h, 180
- TEE\_ATTR\_PBKDF2\_DKM.LENGTH  
tee\_api\_defines\_extensions.h, 180
- TEE\_ATTR\_PBKDF2\_ITERATION.COUNT  
tee\_api\_defines\_extensions.h, 180
- TEE\_ATTR\_PBKDF2\_PASSWORD  
tee\_api\_defines\_extensions.h, 181
- TEE\_ATTR\_PBKDF2\_SALT  
tee\_api\_defines\_extensions.h, 181
- TEE\_ATTR\_RSA\_COEFFICIENT  
tee\_api\_defines.h, 154
- TEE\_ATTR\_RSA\_EXPONENT1  
tee\_api\_defines.h, 154
- TEE\_ATTR\_RSA\_EXPONENT2  
tee\_api\_defines.h, 154
- TEE\_ATTR\_RSA\_MODULUS  
tee\_api\_defines.h, 154
- TEE\_ATTR\_RSA\_OAEP\_LABEL  
tee\_api\_defines.h, 154
- TEE\_ATTR\_RSA\_PRIME1  
tee\_api\_defines.h, 154
- TEE\_ATTR\_RSA\_PRIME2  
tee\_api\_defines.h, 154
- TEE\_ATTR\_RSA\_PRIVATE\_EXPONENT  
tee\_api\_defines.h, 154
- TEE\_ATTR\_RSA\_PSS\_SALT\_LENGTH  
tee\_api\_defines.h, 154
- TEE\_ATTR\_RSA\_PUBLIC\_EXPONENT  
tee\_api\_defines.h, 155
- TEE\_ATTR\_SECRET.VALUE  
tee\_api\_defines.h, 155
- TEE\_Attribute, 51
  - a, 51
  - attributeID, 51
  - b, 51
  - buffer, 51
  - content, 52
  - length, 52
  - ref, 52
  - value, 52
- TEE\_BigInt  
tee\_api\_types.h, 184
- TEE\_BigIntAdd  
tee\_api.h, 114
- TEE\_BigIntAddMod  
tee\_api.h, 114
- TEE\_BigIntCmp  
tee\_api.h, 114
- TEE\_BigIntCmpS32  
tee\_api.h, 114
- TEE\_BigIntComputeExtendedGcd  
tee\_api.h, 115
- TEE\_BigIntComputeFMM  
tee\_api.h, 115
- TEE\_BigIntConvertFromFMM  
tee\_api.h, 115
- TEE\_BigIntConvertFromOctetString  
tee\_api.h, 115
- TEE\_BigIntConvertFromS32  
tee\_api.h, 115
- TEE\_BigIntConvertToFMM  
tee\_api.h, 115
- TEE\_BigIntConvertToOctetString  
tee\_api.h, 115
- TEE\_BigIntConvertToS32  
tee\_api.h, 116
- TEE\_BigIntDiv  
tee\_api.h, 116
- TEE\_BigIntFMM  
tee\_api\_types.h, 184
- TEE\_BigIntFMMContextSizeInU32  
tee\_api.h, 116
- TEE\_BigIntFMMConvertToBigInt  
tee\_api.h, 116
- TEE\_BigIntFMMSizeInU32  
tee\_api.h, 116
- TEE\_BigIntGetBit  
tee\_api.h, 116
- TEE\_BigIntGetBitCount  
tee\_api.h, 116
- TEE\_BigIntInit  
tee\_api.h, 116
- TEE\_BigIntInitFMM  
tee\_api.h, 117
- TEE\_BigIntInitFMMContext



- tee\_api.h, 117
- TEE.BigIntInvMod
  - tee\_api.h, 117
- TEE.BigIntIsProbablePrime
  - tee\_api.h, 117
- TEE.BigIntMod
  - tee\_api.h, 117
- TEE.BigIntMul
  - tee\_api.h, 117
- TEE.BigIntMulMod
  - tee\_api.h, 117
- TEE.BigIntNeg
  - tee\_api.h, 117
- TEE.BigIntRelativePrime
  - tee\_api.h, 118
- TEE.BigIntShiftRight
  - tee\_api.h, 118
- TEE.BigIntSizeInU32
  - tee\_api\_defines.h, 155
- TEE.BigIntSquare
  - tee\_api.h, 118
- TEE.BigIntSquareMod
  - tee\_api.h, 118
- TEE.BigIntSub
  - tee\_api.h, 118
- TEE.BigIntSubMod
  - tee\_api.h, 118
- TEE.CacheClean
  - tee\_internal\_api\_extensions.h, 200
- TEE.CacheFlush
  - tee\_internal\_api\_extensions.h, 200
- TEE.CacheInvalidate
  - tee\_internal\_api\_extensions.h, 201
- TEE.CheckMemoryAccessRights
  - tee\_api.h, 118
- TEE.CipherDoFinal
  - tee-internal-api-cryptlib.c, 272
  - tee\_api.h, 119
- TEE.CipherInit
  - tee-internal-api-cryptlib.c, 273
  - tee-ta-internal.h, 90
  - tee\_api.h, 119
- TEE.CipherUpdate
  - tee-internal-api-cryptlib.c, 273
  - tee-ta-internal.h, 91
  - tee\_api.h, 120
- tee\_client\_api.h
  - TEEC.AllocateSharedMemory, 195
  - TEEC.CloseSession, 195
  - TEEC.CONFIG.PAYLOAD.REF\_COUNT, 189
  - TEEC.CONFIG.SHAREDMEM.MAX.SIZE, 189
  - TEEC.ERROR.ACCESS.CONFLICT, 189
  - TEEC.ERROR.ACCESS.DENIED, 189
  - TEEC.ERROR.BAD.FORMAT, 189
  - TEEC.ERROR.BAD.PARAMETERS, 189
  - TEEC.ERROR.BAD.STATE, 189
  - TEEC.ERROR.BUSY, 189
  - TEEC.ERROR.CANCEL, 190
  - TEEC.ERROR.COMMUNICATION, 190
  - TEEC.ERROR.EXCESS.DATA, 190
  - TEEC.ERROR.EXTERNAL.CANCEL, 190
  - TEEC.ERROR.GENERIC, 190
  - TEEC.ERROR.ITEM.NOT.FOUND, 190
  - TEEC.ERROR.NO.DATA, 190
  - TEEC.ERROR.NOT.IMPLEMENTED, 190
  - TEEC.ERROR.NOT.SUPPORTED, 190
  - TEEC.ERROR.OUT.OF.MEMORY, 190
  - TEEC.ERROR.SECURITY, 190
  - TEEC.ERROR.SHORT.BUFFER, 191
  - TEEC.ERROR.TARGET.DEAD, 191
  - TEEC.FinalizeContext, 195
  - TEEC.InitializeContext, 196
  - TEEC.InvokeCommand, 196
  - TEEC.LOGIN.APPLICATION, 191
  - TEEC.LOGIN.GROUP, 191
  - TEEC.LOGIN.GROUP.APPLICATION, 191
  - TEEC.LOGIN.PUBLIC, 191
  - TEEC.LOGIN.USER, 191
  - TEEC.LOGIN.USER.APPLICATION, 191
  - TEEC.MEM.INPUT, 191
  - TEEC.MEM.OUTPUT, 192
  - TEEC.MEMREF.PARTIAL.INOUT, 192
  - TEEC.MEMREF.PARTIAL.INPUT, 192
  - TEEC.MEMREF.PARTIAL.OUTPUT, 192
  - TEEC.MEMREF.TEMP.INOUT, 192
  - TEEC.MEMREF.TEMP.INPUT, 192
  - TEEC.MEMREF.TEMP.OUTPUT, 192
  - TEEC.MEMREF.WHOLE, 192
  - TEEC.NONE, 192
  - TEEC.OpenSession, 197
  - TEEC.ORIGIN.API, 193
  - TEEC.ORIGIN.COMMS, 193
  - TEEC.ORIGIN.TEE, 193
  - TEEC.ORIGIN.TRUSTED.APP, 193
  - TEEC.PARAM.TYPE.GET, 193
  - TEEC.PARAM.TYPES, 194
  - TEEC.RegisterSharedMemory, 197
  - TEEC.ReleaseSharedMemory, 198
  - TEEC.RequestCancellation, 198
  - TEEC.Result, 195
  - TEEC.SUCCESS, 194
  - TEEC.VALUE.INOUT, 194
  - TEEC.VALUE.INPUT, 194
  - TEEC.VALUE.OUTPUT, 194
- TEE.CloseAndDeletePersistentObject
  - tee\_api.h, 120
- TEE.CloseAndDeletePersistentObject1
  - tee\_api.h, 120
- TEE.CloseObject
  - tee-internal-api.c, 216, 227
  - tee-ta-internal.h, 92
  - tee\_api.h, 120
- TEE.CloseTASession
  - tee\_api.h, 121
- tee\_config.h
  - \_ImageBase, 338, 340, 341

- COMMAND, 339
- perf\_buffer, 338, 340, 341
- tee\_rdtscp, 338–340
- TEE\_CopyObjectAttributes
  - tee\_api.h, 121
- TEE\_CopyObjectAttributes1
  - tee\_api.h, 121
- TEE\_CopyOperation
  - tee\_api.h, 121
- TEE\_CreatePersistentObject
  - tee-internal-api.c, 217, 228
  - tee-ta-internal.h, 92
  - tee\_api.h, 122
- TEE\_DATA\_FLAG\_ACCESS\_READ
  - tee\_api\_defines.h, 155
- TEE\_DATA\_FLAG\_ACCESS\_WRITE
  - tee\_api\_defines.h, 155
- TEE\_DATA\_FLAG\_ACCESS\_WRITE\_META
  - tee\_api\_defines.h, 155
- TEE\_DATA\_FLAG\_OVERWRITE
  - tee\_api\_defines.h, 155
- TEE\_DATA\_FLAG\_SHARE\_READ
  - tee\_api\_defines.h, 155
- TEE\_DATA\_FLAG\_SHARE\_WRITE
  - tee\_api\_defines.h, 155
- TEE\_DATA\_MAX\_POSITION
  - tee\_api\_defines.h, 155
- TEE\_DATA\_SEEK\_CUR
  - tee\_api\_types.h, 186
- TEE\_DATA\_SEEK\_END
  - tee\_api\_types.h, 186
- TEE\_DATA\_SEEK\_SET
  - tee\_api\_types.h, 186
- tee\_def.h
  - buf, 291–293
  - buf\_flag, 291–293
  - tee\_init, 291–293
  - test\_printf, 291–293
- TEE\_DeriveKey
  - tee\_api.h, 123
- TEE\_DigestDoFinal
  - tee-internal-api-cryptlib.c, 274
  - tee-ta-internal.h, 94
  - tee\_api.h, 123
- TEE\_DigestUpdate
  - tee-internal-api-cryptlib.c, 274
  - tee-ta-internal.h, 94
  - tee\_api.h, 123
- TEE\_ECC\_CURVE\_NIST\_P192
  - tee\_api\_defines.h, 155
- TEE\_ECC\_CURVE\_NIST\_P224
  - tee\_api\_defines.h, 156
- TEE\_ECC\_CURVE\_NIST\_P256
  - tee\_api\_defines.h, 156
- TEE\_ECC\_CURVE\_NIST\_P384
  - tee\_api\_defines.h, 156
- TEE\_ECC\_CURVE\_NIST\_P521
  - tee\_api\_defines.h, 156
- TEE\_ERROR\_ACCESS\_CONFLICT
  - tee\_api\_defines.h, 156
- TEE\_ERROR\_ACCESS\_DENIED
  - tee\_api\_defines.h, 156
- TEE\_ERROR\_BAD\_FORMAT
  - tee\_api\_defines.h, 156
- TEE\_ERROR\_BAD\_PARAMETERS
  - tee\_api\_defines.h, 156
- TEE\_ERROR\_BAD\_STATE
  - tee\_api\_defines.h, 156
- TEE\_ERROR\_BUSY
  - tee\_api\_defines.h, 156
- TEE\_ERROR\_CANCEL
  - tee\_api\_defines.h, 156
- TEE\_ERROR\_COMMUNICATION
  - tee\_api\_defines.h, 157
- TEE\_ERROR\_CORRUPT\_OBJECT
  - tee\_api\_defines.h, 157
- TEE\_ERROR\_CORRUPT\_OBJECT\_2
  - tee\_api\_defines.h, 157
- TEE\_ERROR\_EXCESS\_DATA
  - tee\_api\_defines.h, 157
- TEE\_ERROR\_EXTERNAL\_CANCEL
  - tee\_api\_defines.h, 157
- TEE\_ERROR\_GENERIC
  - tee\_api\_defines.h, 157
- TEE\_ERROR\_ITEM\_NOT\_FOUND
  - tee\_api\_defines.h, 157
- TEE\_ERROR\_MAC\_INVALID
  - tee\_api\_defines.h, 157
- TEE\_ERROR\_NO\_DATA
  - tee\_api\_defines.h, 157
- TEE\_ERROR\_NOT\_IMPLEMENTED
  - tee\_api\_defines.h, 157
- TEE\_ERROR\_NOT\_SUPPORTED
  - tee\_api\_defines.h, 157
- TEE\_ERROR\_OUT\_OF\_MEMORY
  - tee\_api\_defines.h, 158
- TEE\_ERROR\_OVERFLOW
  - tee\_api\_defines.h, 158
- TEE\_ERROR\_SECURITY
  - tee\_api\_defines.h, 158
- TEE\_ERROR\_SHORT\_BUFFER
  - tee\_api\_defines.h, 158
- TEE\_ERROR\_SIGNATURE\_INVALID
  - tee\_api\_defines.h, 158
- TEE\_ERROR\_STORAGE\_NO\_SPACE
  - tee\_api\_defines.h, 158
- TEE\_ERROR\_STORAGE\_NOT\_AVAILABLE
  - tee\_api\_defines.h, 158
- TEE\_ERROR\_STORAGE\_NOT\_AVAILABLE\_2
  - tee\_api\_defines.h, 158
- TEE\_ERROR\_TARGET\_DEAD
  - tee\_api\_defines.h, 158
- TEE\_ERROR\_TIME\_NEEDS\_RESET
  - tee\_api\_defines.h, 158
- TEE\_ERROR\_TIME\_NOT\_SET
  - tee\_api\_defines.h, 158

- TEE\_ErrorOrigin
  - tee\_api\_types.h, 185
- TEE\_Free
  - tee-internal-api.c, 218
  - tee\_api.h, 124
- TEE\_FreeOperation
  - tee-internal-api-cryptlib.c, 275
  - tee-ta-internal.h, 95
  - tee\_api.h, 124
- TEE\_FreePersistentObjectEnumerator
  - tee\_api.h, 125
- TEE\_FreePropertyEnumerator
  - tee\_api.h, 125
- TEE\_FreeTransientObject
  - tee-internal-api-cryptlib.c, 275
  - tee-ta-internal.h, 95
  - tee\_api.h, 125
- TEE\_GenerateKey
  - tee-internal-api-cryptlib.c, 276
  - tee-ta-internal.h, 96
  - tee\_api.h, 125
- TEE\_GenerateRandom
  - tee-internal-api.c, 218, 228
  - tee-ta-internal.h, 96
  - tee\_api.h, 126
- TEE\_GetCancellationFlag
  - tee\_api.h, 127
- TEE\_GetInstanceData
  - tee\_api.h, 127
- TEE\_GetNextPersistentObject
  - tee\_api.h, 127
- TEE\_GetNextProperty
  - tee\_api.h, 127
- TEE\_GetObjectBufferAttribute
  - tee\_api.h, 127
- TEE\_GetObjectInfo
  - tee\_api.h, 127
- TEE\_GetObjectInfo1
  - tee-internal-api.c, 219, 229
  - tee-ta-internal.h, 97
  - tee\_api.h, 128
- TEE\_GetObjectValueAttribute
  - tee\_api.h, 128
- TEE\_GetOperationInfo
  - tee\_api.h, 128
- TEE\_GetOperationInfoMultiple
  - tee\_api.h, 129
- TEE\_GetPropertyAsBinaryBlock
  - tee\_api.h, 129
- TEE\_GetPropertyAsBool
  - tee\_api.h, 129
- TEE\_GetPropertyAsIdentity
  - tee\_api.h, 129
- TEE\_GetPropertyAsString
  - tee\_api.h, 129
- TEE\_GetPropertyAsU32
  - tee\_api.h, 129
- TEE\_GetPropertyAsUUID
  - tee\_api.h, 129
- TEE\_GetPropertyName
  - tee\_api.h, 130
- TEE\_GetREETime
  - tee-internal-api.c, 219, 229
  - tee-ta-internal.h, 98
  - tee\_api.h, 130
- TEE\_GetSystemTime
  - tee-internal-api.c, 220, 230
  - tee-ta-internal.h, 99
  - tee\_api.h, 130
- TEE\_GetTAPersistentTime
  - tee\_api.h, 131
- TEE\_HANDLE\_FLAG\_EXPECT\_TWO\_KEYS
  - tee\_api\_defines.h, 159
- TEE\_HANDLE\_FLAG\_INITIALIZED
  - tee\_api\_defines.h, 159
- TEE\_HANDLE\_FLAG\_KEY\_SET
  - tee\_api\_defines.h, 159
- TEE\_HANDLE\_FLAG\_PERSISTENT
  - tee\_api\_defines.h, 159
- TEE\_HANDLE\_NULL
  - tee\_api\_defines.h, 159
  - tee\_api\_tee\_types.h, 235
- TEE\_Identity, 52
  - login, 53
  - uuid, 53
- tee\_init
  - tee\_def.h, 291–293
- TEE\_InitRefAttribute
  - tee-internal-api-cryptlib.c, 276
  - tee-ta-internal.h, 99
  - tee\_api.h, 131
- TEE\_InitValueAttribute
  - tee-internal-api-cryptlib.c, 277
  - tee-ta-internal.h, 100
  - tee\_api.h, 131
- TEE\_INT\_CORE\_API\_SPEC\_VERSION
  - tee\_api\_defines.h, 159
- tee\_internal\_api\_extensions.h
  - TEE\_CacheClean, 200
  - TEE\_CacheFlush, 200
  - TEE\_CacheInvalidate, 201
  - tee\_map\_zi, 201
  - tee\_unmap, 201
  - tee\_user\_mem\_check\_heap, 201
  - TEE\_USER\_MEM\_HINT\_NO\_FILL\_ZERO, 200
  - tee\_user\_mem\_mark\_heap, 201
  - tee\_uuid\_from\_str, 201
- TEE\_InvokeTACCommand
  - tee\_api.h, 132
- TEE\_IsAlgorithmSupported
  - tee\_api.h, 132
- TEE\_LOGIN\_APPLICATION
  - tee\_api\_defines.h, 159
- TEE\_LOGIN\_APPLICATION\_GROUP
  - tee\_api\_defines.h, 159
- TEE\_LOGIN\_APPLICATION\_USER

- tee\_api\_defines.h, 159
- TEE.LOGIN\_GROUP
  - tee\_api\_defines.h, 159
- TEE.LOGIN\_PUBLIC
  - tee\_api\_defines.h, 159
- TEE.LOGIN\_TRUSTED\_APP
  - tee\_api\_defines.h, 160
- TEE.LOGIN\_USER
  - tee\_api\_defines.h, 160
- TEE.MACCompareFinal
  - tee\_api.h, 132
- TEE.MACComputeFinal
  - tee\_api.h, 132
- TEE.MACInit
  - tee\_api.h, 132
- TEE.MACUpdate
  - tee\_api.h, 133
- TEE.Malloc
  - tee-internal-api.c, 220
  - tee\_api.h, 133
- TEE.MALLOC\_FILL\_ZERO
  - tee\_api\_defines.h, 160
- tee\_map\_zi
  - tee\_internal\_api\_extensions.h, 201
- TEE.MaskCancellation
  - tee\_api.h, 133
- TEE.MEM.INPUT
  - tee\_api\_types.h, 183
- TEE.MEM.OUTPUT
  - tee\_api\_types.h, 184
- TEE.MemCompare
  - tee\_api.h, 133
- TEE.MemFill
  - tee\_api.h, 133
- TEE.MemMove
  - tee\_api.h, 134
- TEE.MEMORY\_ACCESS.ANY\_OWNER
  - tee\_api\_defines.h, 160
- TEE.MEMORY\_ACCESS.NONSECURE
  - tee\_api\_defines\_extensions.h, 181
- TEE.MEMORY\_ACCESS.READ
  - tee\_api\_defines.h, 160
- TEE.MEMORY\_ACCESS.SECURE
  - tee\_api\_defines\_extensions.h, 181
- TEE.MEMORY\_ACCESS.WRITE
  - tee\_api\_defines.h, 160
- TEE.MEMREF\_0\_USED
  - tee\_api\_types.h, 184
- TEE.MEMREF\_1\_USED
  - tee\_api\_types.h, 184
- TEE.MEMREF\_2\_USED
  - tee\_api\_types.h, 184
- TEE.MEMREF\_3\_USED
  - tee\_api\_types.h, 184
- TEE.MODE.DECRYPT
  - tee\_api\_types.h, 186
- TEE.MODE.DERIVE
  - tee\_api\_types.h, 186
- TEE.MODE.DIGEST
  - tee\_api\_types.h, 186
- TEE.MODE.ENCRYPT
  - tee\_api\_types.h, 186
- TEE.MODE.MAC
  - tee\_api\_types.h, 186
- TEE.MODE.SIGN
  - tee\_api\_types.h, 186
- TEE.MODE.VERIFY
  - tee\_api\_types.h, 186
- TEE.NUM.PARAMS
  - tee\_api\_defines.h, 160
- TEE.OBJECT.AAD.SIZE
  - tee\_api\_tee\_types.h, 233, 235
- TEE.OBJECT.ID.MAX.LEN
  - tee\_api\_defines.h, 160
- TEE.OBJECT.KEY.SIZE
  - tee\_api\_tee\_types.h, 233, 235
- TEE.OBJECT.NONCE.SIZE
  - tee\_api\_tee\_types.h, 233, 235
- TEE.OBJECT.SKEY.SIZE
  - tee\_api\_tee\_types.h, 233, 235
- TEE.OBJECT.TAG.SIZE
  - tee\_api\_tee\_types.h, 233, 236
- TEE.ObjectEnumHandle
  - tee\_api\_types.h, 185
- TEE.ObjectHandle
  - tee\_api\_types.h, 185
- TEE.ObjectInfo, 53
  - dataPosition, 53
  - dataSize, 54
  - handleFlags, 54
  - keySize, 54
  - maxKeySize, 54
  - maxObjectSize, 54
  - objectSize, 54
  - objectType, 54
  - objectUsage, 54
- TEE.ObjectType
  - tee\_api\_types.h, 185
- TEE.OpenPersistentObject
  - tee-internal-api.c, 220, 230
  - tee-ta-internal.h, 100
  - tee\_api.h, 134
- TEE.OpenTASession
  - tee\_api.h, 135
- TEE.OPERATION.AE
  - tee\_api\_defines.h, 160
- TEE.OPERATION.ASYMMETRIC.CIPHER
  - tee\_api\_defines.h, 160
- TEE.OPERATION.ASYMMETRIC.SIGNATURE
  - tee\_api\_defines.h, 160
- TEE.OPERATION.CIPHER
  - tee\_api\_defines.h, 161
- TEE.OPERATION.DIGEST
  - tee\_api\_defines.h, 161
- TEE.OPERATION.KEY.DERIVATION
  - tee\_api\_defines.h, 161

- TEE\_OPERATION\_MAC
  - tee\_api\_defines.h, 161
- TEE\_OPERATION\_STATE\_ACTIVE
  - tee\_api\_defines.h, 161
- TEE\_OPERATION\_STATE\_INITIAL
  - tee\_api\_defines.h, 161
- TEE\_OperationHandle
  - tee\_api\_types.h, 185
- TEE\_OperationInfo, 54
  - algorithm, 55
  - digestLength, 55
  - handleState, 55
  - keySize, 55
  - maxKeySize, 55
  - mode, 55
  - operationClass, 55
  - requiredKeyUsage, 55
- TEE\_OperationInfoKey, 56
  - keySize, 56
  - requiredKeyUsage, 56
- TEE\_OperationInfoMultiple, 56
  - algorithm, 57
  - digestLength, 57
  - handleState, 57
  - keyInformation, 57
  - maxKeySize, 57
  - mode, 57
  - numberOfKeys, 57
  - operationClass, 57
  - operationState, 57
- TEE\_OperationMode
  - tee\_api\_types.h, 186
- TEE\_ORIGIN\_API
  - tee\_api\_defines.h, 161
- TEE\_ORIGIN\_COMMS
  - tee\_api\_defines.h, 161
- TEE\_ORIGIN\_TEE
  - tee\_api\_defines.h, 161
- TEE\_ORIGIN\_TRUSTED\_APP
  - tee\_api\_defines.h, 161
- TEE\_Panic
  - tee\_api.h, 135
- TEE\_PANIC\_ID\_TA\_CLOSESESSIONENTRYPOINT
  - tee\_api\_defines.h, 161
- TEE\_PANIC\_ID\_TA\_CREATEENTRYPOINT
  - tee\_api\_defines.h, 162
- TEE\_PANIC\_ID\_TA\_DESTROYENTRYPOINT
  - tee\_api\_defines.h, 162
- TEE\_PANIC\_ID\_TA\_INVOKECOMMANDENTRYPOINT
  - tee\_api\_defines.h, 162
- TEE\_PANIC\_ID\_TA\_OPENSESSIONENTRYPOINT
  - tee\_api\_defines.h, 162
- TEE\_PANIC\_ID\_TEE\_AEDECRIPTFINAL
  - tee\_api\_defines.h, 162
- TEE\_PANIC\_ID\_TEE\_AEENCRYPTFINAL
  - tee\_api\_defines.h, 162
- TEE\_PANIC\_ID\_TEE\_AEINIT
  - tee\_api\_defines.h, 162
- TEE\_PANIC\_ID\_TEE\_AEUPDATE
  - tee\_api\_defines.h, 162
- TEE\_PANIC\_ID\_TEE\_AEUPDATEAAD
  - tee\_api\_defines.h, 162
- TEE\_PANIC\_ID\_TEE\_ALLOCATEOPERATION
  - tee\_api\_defines.h, 162
- TEE\_PANIC\_ID\_TEE\_ALLOCATEPERSISTENTOBJECTENUMERATOR
  - tee\_api\_defines.h, 162
- TEE\_PANIC\_ID\_TEE\_ALLOCATEPROPERTYENUMERATOR
  - tee\_api\_defines.h, 163
- TEE\_PANIC\_ID\_TEE\_ALLOCATETRANSIENTOBJECT
  - tee\_api\_defines.h, 163
- TEE\_PANIC\_ID\_TEE\_ASYMMETRICDECRYPT
  - tee\_api\_defines.h, 163
- TEE\_PANIC\_ID\_TEE\_ASYMMETRICENCRYPT
  - tee\_api\_defines.h, 163
- TEE\_PANIC\_ID\_TEE\_ASYMMETRICSIGNDIGEST
  - tee\_api\_defines.h, 163
- TEE\_PANIC\_ID\_TEE\_ASYMMETRICVERIFYDIGEST
  - tee\_api\_defines.h, 163
- TEE\_PANIC\_ID\_TEE\_BIGINTADD
  - tee\_api\_defines.h, 163
- TEE\_PANIC\_ID\_TEE\_BIGINTADDMOD
  - tee\_api\_defines.h, 163
- TEE\_PANIC\_ID\_TEE\_BIGINTCMP
  - tee\_api\_defines.h, 163
- TEE\_PANIC\_ID\_TEE\_BIGINTCMPS32
  - tee\_api\_defines.h, 163
- TEE\_PANIC\_ID\_TEE\_BIGINTCOMPUTEEXTENDEDGCD
  - tee\_api\_defines.h, 163
- TEE\_PANIC\_ID\_TEE\_BIGINTCOMPUTEFMM
  - tee\_api\_defines.h, 164
- TEE\_PANIC\_ID\_TEE\_BIGINTCONVERTFROMFMM
  - tee\_api\_defines.h, 164
- TEE\_PANIC\_ID\_TEE\_BIGINTCONVERTFROMOCTETSTRING
  - tee\_api\_defines.h, 164
- TEE\_PANIC\_ID\_TEE\_BIGINTCONVERTFROMS32
  - tee\_api\_defines.h, 164
- TEE\_PANIC\_ID\_TEE\_BIGINTCONVERTTOFMM
  - tee\_api\_defines.h, 164
- TEE\_PANIC\_ID\_TEE\_BIGINTCONVERTTOOCTETSTRING
  - tee\_api\_defines.h, 164
- TEE\_PANIC\_ID\_TEE\_BIGINTCONVERTTOS32
  - tee\_api\_defines.h, 164
- TEE\_PANIC\_ID\_TEE\_BIGINTDIV
  - tee\_api\_defines.h, 164
- TEE\_PANIC\_ID\_TEE\_BIGINTFMMCONTEXTSIZEINU32
  - tee\_api\_defines.h, 164
- TEE\_PANIC\_ID\_TEE\_BIGINTFMMSIZEINU32
  - tee\_api\_defines.h, 164
- TEE\_PANIC\_ID\_TEE\_BIGINTGETBIT
  - tee\_api\_defines.h, 165
- TEE\_PANIC\_ID\_TEE\_BIGINTGETBITCOUNT
  - tee\_api\_defines.h, 165
- TEE\_PANIC\_ID\_TEE\_BIGINTINIT
  - tee\_api\_defines.h, 165
- TEE\_PANIC\_ID\_TEE\_BIGINTINITFMM
  - tee\_api\_defines.h, 165

- TEE.PANIC.ID.TEE.BIGINTINITFMMCONTEXT  
tee\_api.defines.h, [165](#)
- TEE.PANIC.ID.TEE.BIGINTINVMOD  
tee\_api.defines.h, [165](#)
- TEE.PANIC.ID.TEE.BIGINTISPROBABLEPRIME  
tee\_api.defines.h, [165](#)
- TEE.PANIC.ID.TEE.BIGINTMOD  
tee\_api.defines.h, [165](#)
- TEE.PANIC.ID.TEE.BIGINTMUL  
tee\_api.defines.h, [165](#)
- TEE.PANIC.ID.TEE.BIGINTMULMOD  
tee\_api.defines.h, [165](#)
- TEE.PANIC.ID.TEE.BIGINTNEG  
tee\_api.defines.h, [165](#)
- TEE.PANIC.ID.TEE.BIGINTRELATIVEPRIME  
tee\_api.defines.h, [166](#)
- TEE.PANIC.ID.TEE.BIGINTSHIFTRIGHT  
tee\_api.defines.h, [166](#)
- TEE.PANIC.ID.TEE.BIGINTSQUARE  
tee\_api.defines.h, [166](#)
- TEE.PANIC.ID.TEE.BIGINTSQUAREMOD  
tee\_api.defines.h, [166](#)
- TEE.PANIC.ID.TEE.BIGINTSUB  
tee\_api.defines.h, [166](#)
- TEE.PANIC.ID.TEE.BIGINTSUBMOD  
tee\_api.defines.h, [166](#)
- TEE.PANIC.ID.TEE.CHECKMEMORYACCESSRIGHTS  
tee\_api.defines.h, [166](#)
- TEE.PANIC.ID.TEE.CIPHERDOFINAL  
tee\_api.defines.h, [166](#)
- TEE.PANIC.ID.TEE.CIPHERINIT  
tee\_api.defines.h, [166](#)
- TEE.PANIC.ID.TEE.CIPHERUPDATE  
tee\_api.defines.h, [166](#)
- TEE.PANIC.ID.TEE.CLOSEANDDELETERPERSISTENTOBJECT  
tee\_api.defines.h, [166](#)
- TEE.PANIC.ID.TEE.CLOSEANDDELETERPERSISTENTOBJECT  
tee\_api.defines.h, [167](#)
- TEE.PANIC.ID.TEE.CLOSEOBJECT  
tee\_api.defines.h, [167](#)
- TEE.PANIC.ID.TEE.CLOSETASESSION  
tee\_api.defines.h, [167](#)
- TEE.PANIC.ID.TEE.COPYOBJECTATTRIBUTES  
tee\_api.defines.h, [167](#)
- TEE.PANIC.ID.TEE.COPYOBJECTATTRIBUTES1  
tee\_api.defines.h, [167](#)
- TEE.PANIC.ID.TEE.COPYOPERATION  
tee\_api.defines.h, [167](#)
- TEE.PANIC.ID.TEE.CREATEPERSISTENTOBJECT  
tee\_api.defines.h, [167](#)
- TEE.PANIC.ID.TEE.DERIVEKEY  
tee\_api.defines.h, [167](#)
- TEE.PANIC.ID.TEE.DIGESTDOFINAL  
tee\_api.defines.h, [167](#)
- TEE.PANIC.ID.TEE.DIGESTUPDATE  
tee\_api.defines.h, [167](#)
- TEE.PANIC.ID.TEE.FREE  
tee\_api.defines.h, [167](#)
- TEE.PANIC.ID.TEE.FREEOPERATION  
tee\_api.defines.h, [168](#)
- TEE.PANIC.ID.TEE.FREEPERSISTENTOBJECTENUMERATOR  
tee\_api.defines.h, [168](#)
- TEE.PANIC.ID.TEE.FREEPROPERTYENUMERATOR  
tee\_api.defines.h, [168](#)
- TEE.PANIC.ID.TEE.FREETRANSIENTOBJECT  
tee\_api.defines.h, [168](#)
- TEE.PANIC.ID.TEE.GENERATEKEY  
tee\_api.defines.h, [168](#)
- TEE.PANIC.ID.TEE.GENERATERANDOM  
tee\_api.defines.h, [168](#)
- TEE.PANIC.ID.TEE.GETCANCELLATIONFLAG  
tee\_api.defines.h, [168](#)
- TEE.PANIC.ID.TEE.GETINSTANCEDATA  
tee\_api.defines.h, [168](#)
- TEE.PANIC.ID.TEE.GETNEXTPERSISTENTOBJECT  
tee\_api.defines.h, [168](#)
- TEE.PANIC.ID.TEE.GETNEXTPROPERTY  
tee\_api.defines.h, [168](#)
- TEE.PANIC.ID.TEE.GETOBJECTBUFFERATTRIBUTE  
tee\_api.defines.h, [168](#)
- TEE.PANIC.ID.TEE.GETOBJECTINFO  
tee\_api.defines.h, [169](#)
- TEE.PANIC.ID.TEE.GETOBJECTINFO1  
tee\_api.defines.h, [169](#)
- TEE.PANIC.ID.TEE.GETOBJECTVALUEATTRIBUTE  
tee\_api.defines.h, [169](#)
- TEE.PANIC.ID.TEE.GETOPERATIONINFO  
tee\_api.defines.h, [169](#)
- TEE.PANIC.ID.TEE.GETOPERATIONINFOMULTIPLE  
tee\_api.defines.h, [169](#)
- TEE.PANIC.ID.TEE.GETPROPERTYASBINARYBLOCK  
tee\_api.defines.h, [169](#)
- TEE.PANIC.ID.TEE.GETPROPERTYASBOOL  
tee\_api.defines.h, [169](#)
- TEE.PANIC.ID.TEE.GETPROPERTYASIDENTITY  
tee\_api.defines.h, [169](#)
- TEE.PANIC.ID.TEE.GETPROPERTYASSTRING  
tee\_api.defines.h, [169](#)
- TEE.PANIC.ID.TEE.GETPROPERTYASU32  
tee\_api.defines.h, [169](#)
- TEE.PANIC.ID.TEE.GETPROPERTYASUUID  
tee\_api.defines.h, [169](#)
- TEE.PANIC.ID.TEE.GETPROPERTYNAME  
tee\_api.defines.h, [170](#)
- TEE.PANIC.ID.TEE.GETREETIME  
tee\_api.defines.h, [170](#)
- TEE.PANIC.ID.TEE.GETSYSTEMTIME  
tee\_api.defines.h, [170](#)
- TEE.PANIC.ID.TEE.GETTAPERSTENTTIME  
tee\_api.defines.h, [170](#)
- TEE.PANIC.ID.TEE.INITREFATTRIBUTE  
tee\_api.defines.h, [170](#)
- TEE.PANIC.ID.TEE.INITVALUEATTRIBUTE  
tee\_api.defines.h, [170](#)
- TEE.PANIC.ID.TEE.INVOKETACOMMAND  
tee\_api.defines.h, [170](#)

- TEE.PANIC.ID.TEE.MACCOMPAREFINAL  
tee\_api\_defines.h, 170
- TEE.PANIC.ID.TEE.MACCOMPUTEFINAL  
tee\_api\_defines.h, 170
- TEE.PANIC.ID.TEE.MACINIT  
tee\_api\_defines.h, 170
- TEE.PANIC.ID.TEE.MACUPDATE  
tee\_api\_defines.h, 170
- TEE.PANIC.ID.TEE.MALLOC  
tee\_api\_defines.h, 171
- TEE.PANIC.ID.TEE.MASKCANCELLATION  
tee\_api\_defines.h, 171
- TEE.PANIC.ID.TEE.MEMCOMPARE  
tee\_api\_defines.h, 171
- TEE.PANIC.ID.TEE.MEMFILL  
tee\_api\_defines.h, 171
- TEE.PANIC.ID.TEE.MEMMOVE  
tee\_api\_defines.h, 171
- TEE.PANIC.ID.TEE.OPENPERSISTENTOBJECT  
tee\_api\_defines.h, 171
- TEE.PANIC.ID.TEE.OPENTASESSION  
tee\_api\_defines.h, 171
- TEE.PANIC.ID.TEE.PANIC  
tee\_api\_defines.h, 171
- TEE.PANIC.ID.TEE.POPULATETRANSIENTOBJECT  
tee\_api\_defines.h, 171
- TEE.PANIC.ID.TEE.READOBJECTDATA  
tee\_api\_defines.h, 171
- TEE.PANIC.ID.TEE.REALLOC  
tee\_api\_defines.h, 171
- TEE.PANIC.ID.TEE.RENAMEPERSISTENTOBJECT  
tee\_api\_defines.h, 172
- TEE.PANIC.ID.TEE.RESETOPERATION  
tee\_api\_defines.h, 172
- TEE.PANIC.ID.TEE.RESETPERSISTENTOBJECTENUMERATOR  
tee\_api\_defines.h, 172
- TEE.PANIC.ID.TEE.RESETPROPERTYENUMERATOR  
tee\_api\_defines.h, 172
- TEE.PANIC.ID.TEE.RESETTRANSIENTOBJECT  
tee\_api\_defines.h, 172
- TEE.PANIC.ID.TEE.RESTRICTOBJECTUSAGE  
tee\_api\_defines.h, 172
- TEE.PANIC.ID.TEE.RESTRICTOBJECTUSAGE1  
tee\_api\_defines.h, 172
- TEE.PANIC.ID.TEE.SEEKOBJECTDATA  
tee\_api\_defines.h, 172
- TEE.PANIC.ID.TEE.SETINSTANCEDATA  
tee\_api\_defines.h, 172
- TEE.PANIC.ID.TEE.SETOPERATIONKEY  
tee\_api\_defines.h, 172
- TEE.PANIC.ID.TEE.SETOPERATIONKEY2  
tee\_api\_defines.h, 172
- TEE.PANIC.ID.TEE.SETTAPERSISTENTTIME  
tee\_api\_defines.h, 173
- TEE.PANIC.ID.TEE.STARTPERSISTENTOBJECTENUMERATOR  
tee\_api\_defines.h, 173
- TEE.PANIC.ID.TEE.STARTPROPERTYENUMERATOR  
tee\_api\_defines.h, 173
- TEE.PANIC.ID.TEE.TRUNCATEOBJECTDATA  
tee\_api\_defines.h, 173
- TEE.PANIC.ID.TEE.UNMASKCANCELLATION  
tee\_api\_defines.h, 173
- TEE.PANIC.ID.TEE.WAIT  
tee\_api\_defines.h, 173
- TEE.PANIC.ID.TEE.WRITEOBJECTDATA  
tee\_api\_defines.h, 173
- TEE.Param, 58
  - a, 58
  - b, 58
  - buffer, 58
  - memref, 58
  - size, 58
  - value, 58
- TEE.PARAM.TYPE0  
crt.c, 359
- TEE.PARAM.TYPE1  
crt.c, 359  
Enclave.c, 395
- TEE.PARAM.TYPE.GET  
tee\_api\_defines.h, 173
- TEE.PARAM.TYPE.MEMREF.INOUT  
tee\_api\_defines.h, 173
- TEE.PARAM.TYPE.MEMREF.INPUT  
tee\_api\_defines.h, 173
- TEE.PARAM.TYPE.MEMREF.OUTPUT  
tee\_api\_defines.h, 174
- TEE.PARAM.TYPE.NONE  
tee\_api\_defines.h, 174
- TEE.PARAM.TYPE.SET  
tee\_api\_defines.h, 174
- TEE.PARAM.TYPE.VALUE.INOUT  
tee\_api\_defines.h, 174
- TEE.PARAM.TYPE.VALUE.INPUT  
tee\_api\_defines.h, 174
- TEE.PARAM.TYPE.VALUE.OUTPUT  
tee\_api\_defines.h, 174
- TEE.PARAM.TYPES  
tee\_api\_defines.h, 174
- TEE.PopulateTransientObject  
tee\_api.h, 135
- tee.printf  
config\_ref.ta.h, 311  
crt.c, 361  
Enclave.c, 398  
trace.c, 242, 243, 245
- tee.profiler.c
  - \_\_profiler.head, 343, 344, 346
  - \_\_profiler.unmap.info, 341, 343, 345
  - profiler.write, 342, 344, 345
- tee.profiler.h
  - \_\_profiler.unmap.info, 347–349
- TEE.PROPSET.CURRENT.CLIENT  
tee\_api\_defines.h, 174
- TEE.PROPSET.CURRENT.TA  
tee\_api\_defines.h, 174
- TEE.PROPSET.TEE.IMPLEMENTATION



- tee\_api\_defines.h, 174
- TEE\_PropSetHandle
  - tee\_api\_types.h, 185
- tee\_rdtscp
  - tee\_config.h, 338–340
- TEE\_ReadObjectData
  - tee-internal-api.c, 221, 230
  - tee-ta-internal.h, 101
  - tee\_api.h, 135
- TEE\_Realloc
  - tee-internal-api.c, 221
  - tee\_api.h, 136
- TEE\_RenamePersistentObject
  - tee\_api.h, 136
- TEE\_ResetOperation
  - tee\_api.h, 137
- TEE\_ResetPersistentObjectEnumerator
  - tee\_api.h, 137
- TEE\_ResetPropertyEnumerator
  - tee\_api.h, 137
- TEE\_ResetTransientObject
  - tee\_api.h, 137
- TEE\_RestrictObjectUsage
  - tee\_api.h, 137
- TEE\_RestrictObjectUsage1
  - tee\_api.h, 137
- TEE\_Result
  - tee\_api\_types.h, 185
- TEE\_SE\_READER\_NAME\_MAX
  - tee\_api\_types.h, 184
- TEE\_SEAID, 59
  - buffer, 59
  - bufferLen, 59
- TEE\_SEChannelHandle
  - tee\_api\_types.h, 185
- TEE\_SeekObjectData
  - tee\_api.h, 137
- TEE\_SEReaderHandle
  - tee\_api\_types.h, 185
- TEE\_SEReaderProperties, 59
  - selectResponseEnable, 59
  - sePresent, 60
  - teeOnly, 60
- TEE\_SEServiceHandle
  - tee\_api\_types.h, 185
- TEE\_SESessionHandle
  - tee\_api\_types.h, 185
- TEE\_Session
  - tee\_api\_types.h, 186
- TEE\_SetInstanceData
  - tee\_api.h, 137
- TEE\_SetOperationKey
  - tee-internal-api-cryptlib.c, 277
  - tee-ta-internal.h, 102
  - tee\_api.h, 137
- TEE\_SetOperationKey2
  - tee\_api.h, 138
- TEE\_SetTAPersistentTime
  - tee\_api.h, 138
- TEE\_StartPersistentObjectEnumerator
  - tee\_api.h, 138
- TEE\_StartPropertyEnumerator
  - tee\_api.h, 138
- TEE\_STORAGE\_PRIVATE
  - tee\_api\_defines.h, 175
- TEE\_STORAGE\_PRIVATE\_REE
  - tee\_api\_defines\_extensions.h, 181
- TEE\_STORAGE\_PRIVATE\_RPMB
  - tee\_api\_defines\_extensions.h, 181
- TEE\_STORAGE\_PRIVATE\_SQL\_RESERVED
  - tee\_api\_defines\_extensions.h, 181
- TEE\_SUCCESS
  - tee\_api\_defines.h, 175
- tee\_ta\_api.h
  - TA\_CloseSessionEntryPoint, 203
  - TA\_CreateEntryPoint, 203
  - TA\_DestroyEntryPoint, 203
  - TA\_EXPORT, 202
  - TA\_InvokeCommandEntryPoint, 203
  - TA\_OpenSessionEntryPoint, 203
- TEE\_TASessionHandle
  - tee\_api\_types.h, 186
- TEE\_Time, 60
  - millis, 60
  - seconds, 60
- tee\_time\_tests
  - bench.c, 280
- TEE\_TIMEOUT\_INFINITE
  - tee\_api\_defines.h, 175
- TEE\_TruncateObjectData
  - tee\_api.h, 138
- TEE\_TYPE\_AES
  - tee\_api\_defines.h, 175
- TEE\_TYPE\_CONCAT\_KDF\_Z
  - tee\_api\_defines\_extensions.h, 181
- TEE\_TYPE\_CORRUPTED\_OBJECT
  - tee\_api\_defines.h, 175
- TEE\_TYPE\_DATA
  - tee\_api\_defines.h, 175
- TEE\_TYPE\_DES
  - tee\_api\_defines.h, 175
- TEE\_TYPE\_DES3
  - tee\_api\_defines.h, 175
- TEE\_TYPE\_DH\_KEYPAIR
  - tee\_api\_defines.h, 175
- TEE\_TYPE\_DSA\_KEYPAIR
  - tee\_api\_defines.h, 175
- TEE\_TYPE\_DSA\_PUBLIC\_KEY
  - tee\_api\_defines.h, 175
- TEE\_TYPE\_ECDH\_KEYPAIR
  - tee\_api\_defines.h, 176
- TEE\_TYPE\_ECDH\_PUBLIC\_KEY
  - tee\_api\_defines.h, 176
- TEE\_TYPE\_ECDSA\_KEYPAIR
  - tee\_api\_defines.h, 176
- TEE\_TYPE\_ECDSA\_PUBLIC\_KEY
  - tee\_api\_defines.h, 176



- tee\_api\_defines.h, 176
- TEE\_TYPE\_GENERIC\_SECRET
  - tee\_api\_defines.h, 176
- TEE\_TYPE\_HKDF\_IKM
  - tee\_api\_defines\_extensions.h, 181
- TEE\_TYPE\_HMAC\_MD5
  - tee\_api\_defines.h, 176
- TEE\_TYPE\_HMAC\_SHA1
  - tee\_api\_defines.h, 176
- TEE\_TYPE\_HMAC\_SHA224
  - tee\_api\_defines.h, 176
- TEE\_TYPE\_HMAC\_SHA256
  - tee\_api\_defines.h, 176
- TEE\_TYPE\_HMAC\_SHA384
  - tee\_api\_defines.h, 176
- TEE\_TYPE\_HMAC\_SHA512
  - tee\_api\_defines.h, 176
- TEE\_TYPE\_PBKDF2\_PASSWORD
  - tee\_api\_defines\_extensions.h, 181
- TEE\_TYPE\_RSA\_KEYPAIR
  - tee\_api\_defines.h, 177
- TEE\_TYPE\_RSA\_PUBLIC\_KEY
  - tee\_api\_defines.h, 177
- tee\_unmap
  - tee\_internal\_api\_extensions.h, 201
- TEE\_UnmaskCancellation
  - tee\_api.h, 139
- TEE\_USAGE\_DECRYPT
  - tee\_api\_defines.h, 177
- TEE\_USAGE\_DERIVE
  - tee\_api\_defines.h, 177
- TEE\_USAGE\_ENCRYPT
  - tee\_api\_defines.h, 177
- TEE\_USAGE\_EXTRACTABLE
  - tee\_api\_defines.h, 177
- TEE\_USAGE\_MAC
  - tee\_api\_defines.h, 177
- TEE\_USAGE\_SIGN
  - tee\_api\_defines.h, 177
- TEE\_USAGE\_VERIFY
  - tee\_api\_defines.h, 177
- tee\_user\_mem\_check\_heap
  - tee\_internal\_api\_extensions.h, 201
- TEE\_USER\_MEM\_HINT\_NO\_FILL\_ZERO
  - tee\_internal\_api\_extensions.h, 200
- tee\_user\_mem\_mark\_heap
  - tee\_internal\_api\_extensions.h, 201
- TEE\_UUID, 60
  - clockSeqAndNode, 61
  - timeHiAndVersion, 61
  - timeLow, 61
  - timeMid, 61
- tee\_uuid\_from\_str
  - tee\_internal\_api\_extensions.h, 201
- TEE.Wait
  - tee\_api.h, 139
- TEE.Whence
  - tee\_api\_types.h, 186
- TEE.WriteObjectData
  - tee-internal-api.c, 222, 231
  - tee-ta-internal.h, 103
  - tee\_api.h, 139
- TEEC.AllocateSharedMemory
  - tee\_client\_api.h, 195
  - teec\_stub.c, 236
- TEEC.CloseSession
  - tee\_client\_api.h, 195
  - teec\_stub.c, 237
- TEEC.CONFIG\_PAYLOAD\_REF\_COUNT
  - tee\_client\_api.h, 189
- TEEC.CONFIG\_SHAREDMEM\_MAX\_SIZE
  - tee\_client\_api.h, 189
- TEEC.Context, 61
  - fd, 61
  - reg\_mem, 62
- TEEC.ERROR.ACCESS\_CONFLICT
  - tee\_client\_api.h, 189
- TEEC.ERROR.ACCESS\_DENIED
  - tee\_client\_api.h, 189
- TEEC.ERROR.BAD\_FORMAT
  - tee\_client\_api.h, 189
- TEEC.ERROR.BAD\_PARAMETERS
  - tee\_client\_api.h, 189
- TEEC.ERROR.BAD\_STATE
  - tee\_client\_api.h, 189
- TEEC.ERROR.BUSY
  - tee\_client\_api.h, 189
- TEEC.ERROR.CANCEL
  - tee\_client\_api.h, 190
- TEEC.ERROR.COMMUNICATION
  - tee\_client\_api.h, 190
- TEEC.ERROR.EXCESS\_DATA
  - tee\_client\_api.h, 190
- TEEC.ERROR.EXTERNAL\_CANCEL
  - tee\_client\_api.h, 190
- TEEC.ERROR.GENERIC
  - tee\_client\_api.h, 190
- TEEC.ERROR.ITEM\_NOT\_FOUND
  - tee\_client\_api.h, 190
- TEEC.ERROR.NO\_DATA
  - tee\_client\_api.h, 190
- TEEC.ERROR.NOT\_IMPLEMENTED
  - tee\_client\_api.h, 190
- TEEC.ERROR.NOT\_SUPPORTED
  - tee\_client\_api.h, 190
- TEEC.ERROR.OUT\_OF\_MEMORY
  - tee\_client\_api.h, 190
- TEEC.ERROR.SECURITY
  - tee\_client\_api.h, 190
- TEEC.ERROR.SHORT\_BUFFER
  - tee\_client\_api.h, 191
- TEEC.ERROR.TARGET\_DEAD
  - tee\_client\_api.h, 191
- TEEC.FinalizeContext
  - tee\_client\_api.h, 195
  - teec\_stub.c, 237

- TEEC.InitializeContext
  - tee\_client\_api.h, 196
  - teec\_stub.c, 237
- TEEC.InvokeCommand
  - tee\_client\_api.h, 196
- TEEC.LOGIN\_APPLICATION
  - tee\_client\_api.h, 191
- TEEC.LOGIN\_GROUP
  - tee\_client\_api.h, 191
- TEEC.LOGIN\_GROUP\_APPLICATION
  - tee\_client\_api.h, 191
- TEEC.LOGIN\_PUBLIC
  - tee\_client\_api.h, 191
- TEEC.LOGIN\_USER
  - tee\_client\_api.h, 191
- TEEC.LOGIN\_USER\_APPLICATION
  - tee\_client\_api.h, 191
- TEEC.MEM.INPUT
  - tee\_client\_api.h, 191
- TEEC.MEM.OUTPUT
  - tee\_client\_api.h, 192
- TEEC.MEMREF.PARTIAL.INOUT
  - tee\_client\_api.h, 192
- TEEC.MEMREF.PARTIAL.INPUT
  - tee\_client\_api.h, 192
- TEEC.MEMREF.PARTIAL.OUTPUT
  - tee\_client\_api.h, 192
- TEEC.MEMREF.TEMP.INOUT
  - tee\_client\_api.h, 192
- TEEC.MEMREF.TEMP.INPUT
  - tee\_client\_api.h, 192
- TEEC.MEMREF.TEMP.OUTPUT
  - tee\_client\_api.h, 192
- TEEC.MEMREF.WHOLE
  - tee\_client\_api.h, 192
- TEEC.NONE
  - tee\_client\_api.h, 192
- TEEC.OpenSession
  - tee\_client\_api.h, 197
  - teec\_stub.c, 238
- TEEC.Operation, 62
  - params, 63
  - paramTypes, 63
  - session, 63
  - started, 63
- TEEC.ORIGIN.API
  - tee\_client\_api.h, 193
- TEEC.ORIGIN.COMMS
  - tee\_client\_api.h, 193
- TEEC.ORIGIN.TEE
  - tee\_client\_api.h, 193
- TEEC.ORIGIN.TRUSTED\_APP
  - tee\_client\_api.h, 193
- TEEC.PARAM.TYPE0
  - main.c, 405
- TEEC.PARAM.TYPE1
  - main.c, 403, 405
- TEEC.PARAM.TYPE.GET
  - tee\_client\_api.h, 193
- TEEC.PARAM.TYPES
  - tee\_client\_api.h, 194
- TEEC.Parameter, 63
  - memref, 64
  - tmpref, 64
  - value, 64
- TEEC.RegisteredMemoryReference, 64
  - offset, 65
  - parent, 65
  - size, 65
- TEEC.RegisterSharedMemory
  - tee\_client\_api.h, 197
  - teec\_stub.c, 238
- TEEC.ReleaseSharedMemory
  - tee\_client\_api.h, 198
  - teec\_stub.c, 239
- TEEC.RequestCancellation
  - tee\_client\_api.h, 198
  - teec\_stub.c, 239
- TEEC.Result
  - tee\_client\_api.h, 195
- TEEC.Session, 66
  - ctx, 66
  - session\_id, 66
- TEEC.SharedMemory, 66
  - allocated\_size, 67
  - buffer, 67
  - buffer\_allocated, 67
  - flags, 67
  - id, 67
  - registered\_fd, 68
  - shadow\_buffer, 68
  - size, 68
- teec\_stub.c
  - TEEC.AllocateSharedMemory, 236
  - TEEC.CloseSession, 237
  - TEEC.FinalizeContext, 237
  - TEEC.InitializeContext, 237
  - TEEC.OpenSession, 238
  - TEEC.RegisterSharedMemory, 238
  - TEEC.ReleaseSharedMemory, 239
  - TEEC.RequestCancellation, 239
- TEEC.SUCCESS
  - tee\_client\_api.h, 194
- TEEC.TempMemoryReference, 68
  - buffer, 68
  - size, 69
- TEEC.UUID, 69
  - clockSeqAndNode, 69
  - timeHiAndVersion, 69
  - timeLow, 69
  - timeMid, 69
- TEEC.Value, 70
  - a, 70
  - b, 70
- TEEC.VALUE.INOUT
  - tee\_client\_api.h, 194

- TEEC\_VALUE\_INPUT
  - tee\_client\_api.h, 194
- TEEC\_VALUE\_OUTPUT
  - tee\_client\_api.h, 194
- teeOnly
  - TEE\_SEReaderProperties, 60
- TEEP\_AGENT\_TA\_DELETE
  - invoke\_command.c, 315
- TEEP\_AGENT\_TA\_EXIT
  - invoke\_command.c, 315
- TEEP\_AGENT\_TA\_INSTALL
  - invoke\_command.c, 315
- TEEP\_AGENT\_TA\_LOAD
  - invoke\_command.c, 316
- TEEP\_AGENT\_TA\_NONE
  - invoke\_command.c, 316
- test\_dev\_key.h
  - \_sanctum\_dev\_public\_key, 204
  - \_sanctum\_dev\_public\_key\_len, 204
  - \_sanctum\_dev\_secret\_key, 204
  - \_sanctum\_dev\_secret\_key\_len, 204
- test\_printf
  - tee\_def.h, 291–293
- time.c
  - gp\_ree\_time\_test, 322
  - gp\_trusted\_time\_test, 322
- time\_diff
  - bench.c, 280
- time\_test
  - bench.c, 281
- time\_test.c
  - ree\_time\_test, 296
  - system\_time\_test, 296
- time\_to\_millis
  - bench.c, 281
- timeHiAndVersion
  - TEE\_UUID, 61
  - TEEC\_UUID, 69
- timeLow
  - TEE\_UUID, 61
  - TEEC\_UUID, 69
- timeMid
  - TEE\_UUID, 61
  - TEEC\_UUID, 69
- tmpref
  - TEEC\_Parameter, 64
- tools.c
  - \_strlen, 336
  - printf, 336
  - profiler\_write, 332–335
  - putchar, 337
  - puts, 337
- tools.h
  - printf, 367
  - putchar, 367
  - puts, 368
- trace.c
  - \_strlen, 242, 244
  - tee\_printf, 242, 243, 245
  - trace\_printf, 240
  - trace\_vprintf, 241
- trace.h
  - dhex\_dump, 209
  - DHEXDUMP, 206
  - DMSG, 206
  - DMSG\_RAW, 206
  - DPRINT\_STACK, 206
  - EMSG, 207
  - EMSG\_RAW, 207
  - EPRINT\_STACK, 207
  - FMSG, 207
  - FMSG\_RAW, 207
  - FPRINT\_STACK, 207
  - IMSG, 207
  - IMSG\_RAW, 207
  - INMSG, 207
  - IPRINT\_STACK, 207
  - MAX\_FUNC\_PRINT\_SIZE, 208
  - MAX\_PRINT\_SIZE, 208
  - MSG, 208
  - MSG\_RAW, 208
  - OUTMSG, 208
  - OUTRMSG, 208
  - SMSG, 208
  - trace\_ext\_get\_thread\_id, 209
  - trace\_ext\_prefix, 210
  - trace\_ext\_puts, 209
  - trace\_get\_level, 209
  - TRACE\_LEVEL, 208
  - trace\_level, 210
  - trace\_printf, 209
  - trace\_printf\_helper, 208
  - trace\_printf\_helper\_raw, 209
  - trace\_set\_level, 209
- TRACE\_DEBUG
  - trace\_levels.h, 210
- TRACE\_ERROR
  - trace\_levels.h, 211
- trace\_ext\_get\_thread\_id
  - trace.h, 209
- trace\_ext\_prefix
  - trace.h, 210
  - user\_ta\_header.c, 408, 412
- trace\_ext\_puts
  - trace.h, 209
- TRACE\_FLOW
  - trace\_levels.h, 211
- trace\_get\_level
  - trace.h, 209
- TRACE\_INFO
  - trace\_levels.h, 211
- TRACE\_LEVEL
  - trace.h, 208
- trace\_level
  - trace.h, 210
  - user\_ta\_header.c, 409, 412

- trace\_levels.h
  - TRACE\_DEBUG, 210
  - TRACE\_ERROR, 211
  - TRACE\_FLOW, 211
  - TRACE\_INFO, 211
  - TRACE\_MAX, 211
  - TRACE\_MIN, 211
  - TRACE\_PRINTF\_LEVEL, 211
- TRACE\_MAX
  - trace\_levels.h, 211
- TRACE\_MIN
  - trace\_levels.h, 211
- trace\_printf
  - trace.c, 240
  - trace.h, 209
- trace\_printf\_helper
  - trace.h, 208
- trace\_printf\_helper\_raw
  - trace.h, 209
- TRACE\_PRINTF\_LEVEL
  - trace\_levels.h, 211
- trace\_set\_level
  - trace.h, 209
- trace\_vprintf
  - trace.c, 241
- TRUE
  - App.h, 417, 418
- type
  - \_\_TEE\_ObjectHandle, 37
  - nm\_info, 44
- types.h
  - global\_eid, 435, 437
  - sgx\_errlist, 435, 437
  - sgx\_errlist\_t, 435, 437
- USED
  - profiler\_attrs.h, 353
- user\_ta\_header.c
  - \_C\_FUNCTION, 406, 410
  - \_section, 407, 410
  - \_ta\_entry, 407, 410
  - \_utee\_entry, 407, 410
  - TA\_DESCRIPTION, 407, 410
  - TA\_FRAMEWORK\_STACK\_SIZE, 407, 410
  - ta\_heap, 408, 411
  - ta\_heap\_size, 408, 411
  - ta\_num\_props, 408, 411
  - ta\_props, 408, 411
  - TA\_VERSION, 407, 410
  - tahead\_get\_trace\_level, 408, 411
  - trace\_ext\_prefix, 408, 412
  - trace\_level, 409, 412
- user\_ta\_header\_defines.h
  - TA\_CURRENT\_TA\_EXT\_PROPERTIES, 413, 415
  - TA\_DATA\_SIZE, 413, 415
  - TA\_DESCRIPTION, 413, 415
  - TA\_FLAGS, 413, 415
  - TA\_STACK\_SIZE, 414, 415
  - TA\_UUID, 414, 415
  - TA\_VERSION, 414, 415
- user\_types.h
  - array\_t, 297
  - buffer\_t, 297
  - LOOPS\_PER\_THREAD, 297
- uuid
  - TEE.Identity, 53
- value
  - TEE\_Attribute, 52
  - TEE\_Param, 58
  - TEEC\_Parameter, 64
- vsnprintf
  - vsnprintf.c, 251, 264
- vsnprintf.c
  - .atoi, 248, 254
  - .ftoa, 248, 255
  - .is\_digit, 249, 255
  - .ntoa\_format, 249, 256
  - .ntoa\_long, 249, 256
  - .ntoa\_long\_long, 249, 257
  - .out\_buffer, 249, 258
  - .out\_char, 250, 258
  - .out\_fct, 250, 259
  - .out\_null, 250, 259
  - .putchar, 247, 252
  - .strlen, 250, 259
  - .vsnprintf, 250, 260
  - fctprintf, 250, 260
  - FLAGS\_CHAR, 247, 253
  - FLAGS\_HASH, 247, 253
  - FLAGS\_LEFT, 247, 253
  - FLAGS\_LONG, 247, 253
  - FLAGS\_LONG\_LONG, 247, 253
  - FLAGS\_PLUS, 247, 253
  - FLAGS\_PRECISION, 247, 253
  - FLAGS\_SHORT, 247, 253
  - FLAGS\_SPACE, 247, 253
  - FLAGS\_UPPERCASE, 247, 253
  - FLAGS\_ZEROPAD, 248, 253
  - out\_fct\_type, 248, 254
  - PRINTF\_FTOA\_BUFFER\_SIZE, 248, 254
  - PRINTF\_NTOA\_BUFFER\_SIZE, 248, 254
  - PRINTF\_SUPPORT\_FLOAT, 248, 254
  - PRINTF\_SUPPORT\_LONG\_LONG, 248, 254
  - PRINTF\_SUPPORT\_PTRDIFF\_T, 248, 254
  - putchar, 250, 262
  - snprintf, 251, 262
  - sprintf, 251, 262
  - vsnprintf, 251, 264
- WOLFCRYPT
  - tee\_api\_tee\_types.h, 233, 236
- wolfSSL\_Free
  - tee-internal-api-cryptlib.c, 278
- wolfSSL\_Malloc
  - tee-internal-api-cryptlib.c, 278