Trusted Application Programming Reference on Portable TEE

National Institute of Advanced Industrial Science and Technology

2021-02-19

1 Preparation 1
1.1 Keystone(RISC-V Unleased) 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 scucceeded in IAS Test 4
1.3.3 BIOS Settings 5
1.3.4 Required Packages 5
1.3.5 Build SGX 5
1.3.6 Run sgx-ra-sample 6

2 Building 8
2.1 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

3 Running on Dev Boards 14
3.1 Keystone, Unleased 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 Unleased 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

4 Overview of ta-ref **21**
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

5 How to Program on ta-ref **22**
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

6 API Compare With Full-Set of GP API **28**
6.1 GP API . . . 28

7 Class Index **30**
7.1 Class List . . . 30

8 File Index **31**
8.1 File List . . . 31

9 Class Documentation **35**
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_t Struct Reference . . . 41
9.8.1 Member Data Documentation . . . 42
9.9 list Struct Reference . . . 43
9.9.1 Member Data Documentation . . . 43

9.10 nm_info Struct Reference . . . 43
9.10.1 Member Data Documentation . . . 44
9.11 nonce_t 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 . . . 46
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 pollfd Struct Reference . . . 47
9.16.1 Member Data Documentation . . . 47
9.17 ree_time_t Struct Reference . . . 47
9.17.1 Member Data Documentation . . . 47
9.18 report Struct Reference . . . 48
9.18.1 Member Data Documentation . . . 48
9.19 result Struct Reference . . . 49
9.19.1 Member Data Documentation . . . 49
9.20 sm_report Struct Reference . . . 50
9.20.1 Member Data Documentation . . . 50
9.21 TEE_Attribute Struct Reference . . . 50
9.21.1 Member Data Documentation . . . 51
9.22 TEE_Identity Struct Reference . . . 52
9.22.1 Member Data Documentation . . . 52
9.23 TEE_ObjectInfo Struct Reference . . . 53
9.23.1 Member Data Documentation . . . 53
9.24 TEE_OperationInfo Struct Reference . . . 54
9.24.1 Member Data Documentation . . . 54
9.25 TEE_OperationInfoKey Struct Reference . . . 55
9.25.1 Member Data Documentation . . . 56
9.26 TEE_OperationInfoMultiple Struct Reference . . . 56
9.26.1 Member Data Documentation . . . 56
9.27 TEE_Param Union Reference . . . 57
9.27.1 Member Data Documentation . . . 58
9.28 TEE_SEAID Struct Reference . . . 58
9.28.1 Member Data Documentation . . . 59
9.29 TEE_SEReaderProperties Struct Reference . . . 59
9.29.1 Member Data Documentation . . . 59
9.30 TEE_Time Struct Reference . . . 60
9.30.1 Member Data Documentation . . . 60

9.31 TEE_UUID Struct Reference . . . 60
9.31.1 Member Data Documentation . . . 60
9.32 TEEC_Context Struct Reference . . . 61
9.32.1 Detailed Description . . . 61
9.32.2 Member Data Documentation . . . 61
9.33 TEEC_Operation Struct Reference . . . 62
9.33.1 Detailed Description . . . 62
9.33.2 Member Data Documentation . . . 62
9.34 TEEC_Parameter Union Reference . . . 63
9.34.1 Detailed Description . . . 63
9.34.2 Member Data Documentation . . . 64
9.35 TEEC_RegisteredMemoryReference Struct Reference . . . 64
9.35.1 Detailed Description . . . 65
9.35.2 Member Data Documentation . . . 65
9.36 TEEC_Session Struct Reference . . . 66
9.36.1 Detailed Description . . . 66
9.36.2 Member Data Documentation . . . 66
9.37 TEEC_SharedMemory Struct Reference . . . 66
9.37.1 Detailed Description . . . 67
9.37.2 Member Data Documentation . . . 67
9.38 TEEC_TempMemoryReference Struct Reference . . . 68
9.38.1 Detailed Description . . . 68
9.38.2 Member Data Documentation . . . 68
9.39 TEEC_UUID Struct Reference . . . 69
9.39.1 Detailed Description . . . 69
9.39.2 Member Data Documentation . . . 69
9.40 TEEC_Value Struct Reference . . . 70
9.40.1 Detailed Description . . . 70
9.40.2 Member Data Documentation . . . 70

10 File Documentation **71**

10.1 ta-ref/api/include/compiler.h File Reference . . . 71
10.1.1 Macro Definition Documentation . . . 72
10.2 compiler.h . . . 78
10.3 ta-ref/api/include/report.h File Reference . . . 80
10.3.1 Macro Definition Documentation . . . 81
10.4 report.h . . . 81
10.5 ta-ref/api/include/tee-common.h File Reference . . . 82
10.5.1 Detailed Description . . . 82
10.5.2 Macro Definition Documentation . . . 83
10.6 tee-common.h . . . 83
10.7 ta-ref/api/include/tee-ta-internal.h File Reference . . . 83

10.7.1 Detailed Description . . . 86
10.7.2 Function Documentation . . . 86
10.8 tee-ta-internal.h . . . 108
10.9 ta-ref/api/include/tee_api_defines.h File Reference . . . 110
10.9.1 Macro Definition Documentation . . . 116
10.10 tee_api_defines.h . . . 148
10.11 ta-ref/api/include/tee_api_defines_extensions.h File Reference . . . 153
10.11.1 Macro Definition Documentation . . . 154
10.12 tee_api_defines_extensions.h . . . 157
10.13 ta-ref/api/include/tee_api_types.h File Reference . . . 158
10.13.1 Macro Definition Documentation . . . 160
10.13.2 Typedef Documentation . . . 161
10.13.3 Enumeration Type Documentation . . . 162
10.14 tee_api_types.h . . . 163
10.15 ta-ref/api/include/tee_client_api.h File Reference . . . 166
10.15.1 Macro Definition Documentation . . . 168
10.15.2 Typedef Documentation . . . 174
10.15.3 Function Documentation . . . 174
10.16 tee_client_api.h . . . 178
10.17 ta-ref/api/include/tee_internal_api.h File Reference . . . 181
10.18 tee_internal_api.h . . . 181
10.19 ta-ref/api/include/tee_internal_api_extensions.h File Reference . . . 181
10.19.1 Macro Definition Documentation . . . 182
10.19.2 Function Documentation . . . 182
10.20 tee_internal_api_extensions.h . . . 184
10.21 ta-ref/api/include/tee_ta_api.h File Reference . . . 184
10.21.1 Macro Definition Documentation . . . 185
10.21.2 Function Documentation . . . 185
10.22 tee_ta_api.h . . . 187
10.23 ta-ref/api/include/test_dev_key.h File Reference . . . 189
10.23.1 Variable Documentation . . . 189
10.24 test_dev_key.h . . . 190
10.25 ta-ref/api/include/trace.h File Reference . . . 190
10.25.1 Macro Definition Documentation . . . 192
10.25.2 Function Documentation . . . 194
10.25.3 Variable Documentation . . . 195
10.26 trace.h . . . 196
10.27 ta-ref/api/include/trace_levels.h File Reference . . . 198
10.27.1 Macro Definition Documentation . . . 199
10.28 trace_levels.h . . . 199
10.29 ta-ref/api/keystone/tee-internal-api-machine.c File Reference . . . 200
10.29.1 Function Documentation . . . 201

10.30 ta-ref/api/keystone/tee-internal-api.c File Reference . . . 201
10.30.1 Macro Definition Documentation . . . 202
10.30.2 Function Documentation . . . 203
10.31 ta-ref/api/sgx/tee-internal-api.c File Reference . . . 212
10.31.1 Macro Definition Documentation . . . 213
10.31.2 Function Documentation . . . 214
10.32 ta-ref/api/keystone/tee_api_tee_types.h File Reference . . . 221
10.32.1 Macro Definition Documentation . . . 222
10.33 tee_api_tee_types.h . . . 223
10.34 ta-ref/api/optee/tee_api_tee_types.h File Reference . . . 224
10.35 tee_api_tee_types.h . . . 224
10.36 ta-ref/api/sgx/tee_api_tee_types.h File Reference . . . 224
10.36.1 Macro Definition Documentation . . . 226
10.37 tee_api_tee_types.h . . . 227
10.38 ta-ref/api/keystone/teec_stub.c File Reference . . . 228
10.38.1 Function Documentation . . . 229
10.39 ta-ref/api/keystone/trace.c File Reference . . . 232
10.39.1 Function Documentation . . . 232
10.40 ta-ref/test_gp/keystone/Enclave/trace.c File Reference . . . 234
10.40.1 Function Documentation . . . 234
10.41 ta-ref/test_gp/optee/Enclave/trace.c File Reference . . . 235
10.41.1 Function Documentation . . . 235
10.42 ta-ref/test_gp/sgx/Enclave/trace.c File Reference . . . 236
10.42.1 Function Documentation . . . 237
10.43 ta-ref/api/tee-internal-api-cryptlib.c File Reference . . . 237
10.43.1 Macro Definition Documentation . . . 239
10.43.2 Function Documentation . . . 240
10.44 ta-ref/benchmark/bench.c File Reference . . . 251
10.44.1 Function Documentation . . . 252
10.44.2 Variable Documentation . . . 254
10.45 ta-ref/benchmark/bench.h File Reference . . . 255
10.45.1 Macro Definition Documentation . . . 256
10.45.2 Function Documentation . . . 256
10.46 bench.h . . . 258
10.47 ta-ref/benchmark/cpu_bench.c File Reference . . . 258
10.47.1 Function Documentation . . . 259
10.48 ta-ref/benchmark/include/config_bench.h File Reference . . . 260
10.48.1 Macro Definition Documentation . . . 261
10.48.2 Enumeration Type Documentation . . . 261
10.48.3 Function Documentation . . . 261
10.49 config_bench.h . . . 262
10.50 ta-ref/benchmark/io_bench.c File Reference . . . 262

10.50.1 Macro Definition Documentation . . . 263
10.50.2 Function Documentation . . . 263
10.51 ta-ref/benchmark/keystone/tee_def.h File Reference . . . 264
10.51.1 Function Documentation . . . 265
10.51.2 Variable Documentation . . . 265
10.52 tee_def.h . . . 265
10.53 ta-ref/benchmark/optee/tee_def.h File Reference . . . 266
10.53.1 Macro Definition Documentation . . . 266
10.53.2 Function Documentation . . . 266
10.53.3 Variable Documentation . . . 266
10.54 tee_def.h . . . 266
10.55 ta-ref/benchmark/sgx/tee_def.h File Reference . . . 267
10.55.1 Function Documentation . . . 267
10.55.2 Variable Documentation . . . 267
10.56 tee_def.h . . . 268
10.57 ta-ref/benchmark/memory_bench.c File Reference . . . 268
10.57.1 Macro Definition Documentation . . . 269
10.57.2 Function Documentation . . . 269
10.58 ta-ref/benchmark/time_test.c File Reference . . . 270
10.58.1 Function Documentation . . . 270
10.59 ta-ref/docs/building.md File Reference . . . 271
10.60 ta-ref/docs/gp_api.md File Reference . . . 271
10.61 ta-ref/docs/how_to_program_on_ta-ref.md File Reference . . . 271
10.62 ta-ref/docs/overview_of_ta-ref.md File Reference . . . 271
10.63 ta-ref/docs/preparation.md File Reference . . . 271
10.64 ta-ref/docs/running_on_dev_boards.md File Reference . . . 271
10.65 ta-ref/edger/edger8r/user_types.h File Reference . . . 271
10.65.1 Macro Definition Documentation . . . 271
10.65.2 Typedef Documentation . . . 272
10.66 user_types.h . . . 272
10.67 ta-ref/edger/keyedge/Enclave_t.c File Reference . . . 272
10.68 ta-ref/edger/keyedge/Enclave_t.h File Reference . . . 273
10.69 Enclave_t.h . . . 273
10.70 ta-ref/edger/optee/Enclave_t.h File Reference . . . 274
10.71 Enclave_t.h . . . 274
10.72 ta-ref/edger/keyedge/Enclave_u.c File Reference . . . 274
10.73 ta-ref/edger/keyedge/Enclave_u.h File Reference . . . 274
10.73.1 Macro Definition Documentation . . . 275
10.73.2 Function Documentation . . . 275
10.74 Enclave_u.h . . . 276
10.75 ta-ref/edger/keyedge/ocalls.h File Reference . . . 276
10.75.1 Macro Definition Documentation . . . 277

10.75.2 Typedef Documentation . . . 278
10.75.3 Function Documentation . . . 278
10.76 ocalls.h . . . 281
10.77 ta-ref/edger/optee/Enclave.h File Reference . . . 282
10.77.1 Macro Definition Documentation . . . 282
10.78 Enclave.h . . . 282
10.79 ta-ref/test_gp/sgx/Enclave/Enclave.h File Reference . . . 283
10.79.1 Function Documentation . . . 284
10.80 Enclave.h . . . 286
10.81 ta-ref/gp/asymmetric_key.c File Reference . . . 286
10.81.1 Macro Definition Documentation . . . 287
10.81.2 Function Documentation . . . 287
10.82 ta-ref/gp/include/config_ref_ta.h File Reference . . . 288
10.82.1 Macro Definition Documentation . . . 289
10.82.2 Function Documentation . . . 289
10.83 config_ref_ta.h . . . 290
10.84 ta-ref/gp/include/gp_test.h File Reference . . . 291
10.84.1 Function Documentation . . . 291
10.85 gp_test.h . . . 293
10.86 ta-ref/gp/invoke_command.c File Reference . . . 293
10.86.1 Macro Definition Documentation . . . 294
10.87 ta-ref/gp/message_digest.c File Reference . . . 294
10.87.1 Macro Definition Documentation . . . 295
10.87.2 Function Documentation . . . 295
10.88 ta-ref/gp/random.c File Reference . . . 295
10.88.1 Function Documentation . . . 296
10.89 ta-ref/gp/secure_stoage.c File Reference . . . 296
10.89.1 Macro Definition Documentation . . . 296
10.89.2 Function Documentation . . . 297
10.90 ta-ref/gp/symmetric_key.c File Reference . . . 297
10.90.1 Macro Definition Documentation . . . 298
10.90.2 Function Documentation . . . 298
10.91 ta-ref/gp/symmetric_key_gcm.c File Reference . . . 298
10.91.1 Macro Definition Documentation . . . 299
10.91.2 Function Documentation . . . 299
10.92 ta-ref/gp/time.c File Reference . . . 299
10.92.1 Function Documentation . . . 300
10.93 ta-ref/profiler/analyzer/analyzer.c File Reference . . . 300
10.93.1 Macro Definition Documentation . . . 301
10.93.2 Function Documentation . . . 301
10.94 ta-ref/profiler/analyzer/analyzer.h File Reference . . . 302
10.95 analyzer.h . . . 303

10.96 ta-ref/profiler/analyzer/nm_parse.c File Reference . . . 303
10.96.1 Macro Definition Documentation . . . 304
10.96.2 Function Documentation . . . 304
10.96.3 Variable Documentation . . . 306
10.97 ta-ref/profiler/analyzer/nm_parse.h File Reference . . . 306
10.97.1 Macro Definition Documentation . . . 307
10.97.2 Function Documentation . . . 307
10.98 nm_parse.h . . . 308
10.99 ta-ref/profiler/analyzer/stack.h File Reference . . . 308
10.99.1 Macro Definition Documentation . . . 310
10.99.2 Function Documentation . . . 310
10.99.3 Variable Documentation . . . 310
10.100 stack.h . . . 311
10.101 ta-ref/profiler/app/tools.c File Reference . . . 311
10.101.1 Function Documentation . . . 311
10.102 ta-ref/profiler/keystone/Enclave/tools.c File Reference . . . 312
10.102.1 Function Documentation . . . 312
10.103 ta-ref/profiler/optee/Enclave/tools.c File Reference . . . 313
10.103.1 Function Documentation . . . 313
10.104 ta-ref/profiler/sgx/Enclave/tools.c File Reference . . . 314
10.104.1 Function Documentation . . . 314
10.105 ta-ref/test_gp/tools.c File Reference . . . 315
10.105.1 Function Documentation . . . 315
10.106 ta-ref/profiler/keystone/tee_config.h File Reference . . . 317
10.106.1 Function Documentation . . . 317
10.106.2 Variable Documentation . . . 317
10.107 tee_config.h . . . 318
10.108 ta-ref/profiler/optee/tee_config.h File Reference . . . 318
10.108.1 Macro Definition Documentation . . . 319
10.108.2 Function Documentation . . . 319
10.108.3 Variable Documentation . . . 319
10.109 tee_config.h . . . 319
10.110 ta-ref/profiler/sgx/tee_config.h File Reference . . . 320
10.110.1 Function Documentation . . . 320
10.110.2 Variable Documentation . . . 320
10.111 tee_config.h . . . 321
10.112 ta-ref/profiler/keystone/tee_profiler.c File Reference . . . 321
10.112.1 Function Documentation . . . 321
10.112.2 Variable Documentation . . . 323
10.113 ta-ref/profiler/optee/tee_profiler.c File Reference . . . 323
10.113.1 Function Documentation . . . 323
10.113.2 Variable Documentation . . . 324

10.114 ta-ref/profiler/sgx/tee_profiler.c File Reference . . . 324
10.114.1 Function Documentation . . . 325
10.114.2 Variable Documentation . . . 326
10.115 ta-ref/profiler/keystone/tee_profiler.h File Reference . . . 327
10.115.1 Function Documentation . . . 327
10.116 tee_profiler.h . . . 327
10.117 ta-ref/profiler/optee/tee_profiler.h File Reference . . . 328
10.117.1 Function Documentation . . . 328
10.118 tee_profiler.h . . . 328
10.119 ta-ref/profiler/sgx/tee_profiler.h File Reference . . . 329
10.119.1 Function Documentation . . . 329
10.120 tee_profiler.h . . . 329
10.121 ta-ref/profiler/profiler.c File Reference . . . 330
10.121.1 Function Documentation . . . 330
10.121.2 Variable Documentation . . . 332
10.122 ta-ref/profiler/profiler.h File Reference . . . 332
10.122.1 Function Documentation . . . 333
10.123 profiler.h . . . 333
10.124 ta-ref/profiler/profiler_attrs.h File Reference . . . 333
10.124.1 Macro Definition Documentation . . . 334
10.125 profiler_attrs.h . . . 334
10.126 ta-ref/profiler/profiler_data.h File Reference . . . 334
10.126.1 Macro Definition Documentation . . . 335
10.126.2 Typedef Documentation . . . 336
10.126.3 Enumeration Type Documentation . . . 336
10.126.4 Function Documentation . . . 336
10.126.5 Variable Documentation . . . 336
10.127 profiler_data.h . . . 337
10.128 ta-ref/test_gp/crt.c File Reference . . . 337
10.128.1 Function Documentation . . . 338
10.128.2 Variable Documentation . . . 338
10.129 ta-ref/test_gp/optee/Enclave/crt.c File Reference . . . 339
10.129.1 Macro Definition Documentation . . . 340
10.129.2 Function Documentation . . . 340
10.129.3 Variable Documentation . . . 344
10.130 ta-ref/test_gp/include/crt.h File Reference . . . 344
10.130.1 Function Documentation . . . 345
10.131 crt.h . . . 346
10.132 ta-ref/test_gp/include/ocall_wrapper.h File Reference . . . 346
10.132.1 Function Documentation . . . 346
10.133 ocall_wrapper.h . . . 347
10.134 ta-ref/test_gp/include/random.h File Reference . . . 347

10.135 random.h . . . 348
10.136 ta-ref/test_gp/include/tools.h File Reference . . . 348
10.136.1 Function Documentation . . . 348
10.137 tools.h . . . 350
10.138 ta-ref/test_gp/keystone/Enclave/ocall_wrapper.c File Reference . . . 350
10.138.1 Function Documentation . . . 350
10.139 ta-ref/test_gp/sgx/Enclave/ocall_wrapper.c File Reference . . . 351
10.139.1 Function Documentation . . . 351
10.140 ta-ref/test_gp/keystone/Enclave/startup.c File Reference . . . 352
10.140.1 Function Documentation . . . 352
10.141 ta-ref/test_gp/sgx/Enclave/startup.c File Reference . . . 353
10.141.1 Function Documentation . . . 353
10.142 ta-ref/test_gp/vsnprintf.c File Reference . . . 354
10.142.1 Macro Definition Documentation . . . 355
10.142.2 Typedef Documentation . . . 357
10.142.3 Function Documentation . . . 357
10.143 ta-ref/test_gp/keystone/App/App.cpp File Reference . . . 366
10.143.1 Function Documentation . . . 366
10.143.2 Variable Documentation . . . 367
10.144 ta-ref/test_gp/sgx/App/App.cpp File Reference . . . 367
10.144.1 Macro Definition Documentation . . . 368
10.144.2 Function Documentation . . . 368
10.145 ta-ref/test_hello/keystone/App/App.cpp File Reference . . . 369
10.145.1 Function Documentation . . . 370
10.145.2 Variable Documentation . . . 370
10.146 ta-ref/test_hello/sgx/App/App.cpp File Reference . . . 370
10.146.1 Macro Definition Documentation . . . 371
10.146.2 Function Documentation . . . 371
10.147 ta-ref/test_gp/keystone/App/App_ocalls.cpp File Reference . . . 372
10.147.1 Macro Definition Documentation . . . 373
10.147.2 Function Documentation . . . 373
10.148 ta-ref/test_gp/sgx/App/App_ocalls.cpp File Reference . . . 377
10.148.1 Macro Definition Documentation . . . 378
10.148.2 Function Documentation . . . 378
10.149 ta-ref/test_hello/keystone/App/App_ocalls.cpp File Reference . . . 381
10.149.1 Function Documentation . . . 381
10.150 ta-ref/test_hello/sgx/App/App_ocalls.cpp File Reference . . . 385
10.150.1 Function Documentation . . . 386
10.151 ta-ref/test_gp/keystone/Enclave/Enclave.c File Reference . . . 388
10.151.1 Function Documentation . . . 389
10.152 ta-ref/test_gp/optee/Enclave/Enclave.c File Reference . . . 389
10.152.1 Function Documentation . . . 390

10.153 ta-ref/test_gp/sgx/Enclave/Enclave.c File Reference . . . 390
10.153.1 Function Documentation . . . 391
10.154 ta-ref/test_hello/keystone/Enclave/Enclave.c File Reference . . . 391
10.154.1 Macro Definition Documentation . . . 392
10.154.2 Function Documentation . . . 392
10.155 ta-ref/test_hello/optee/Enclave/Enclave.c File Reference . . . 393
10.155.1 Macro Definition Documentation . . . 393
10.155.2 Function Documentation . . . 394
10.155.3 Variable Documentation . . . 397
10.156 ta-ref/test_hello/sgx/Enclave/Enclave.c File Reference . . . 397
10.156.1 Macro Definition Documentation . . . 397
10.156.2 Function Documentation . . . 398
10.157 ta-ref/test_gp/optee/App/main.c File Reference . . . 398
10.157.1 Macro Definition Documentation . . . 398
10.157.2 Function Documentation . . . 399
10.158 ta-ref/test_hello/optee/App/main.c File Reference . . . 400
10.158.1 Macro Definition Documentation . . . 400
10.158.2 Function Documentation . . . 401
10.158.3 Variable Documentation . . . 401
10.159 ta-ref/test_gp/optee/Enclave/user_ta_header.c File Reference . . . 401
10.159.1 Macro Definition Documentation . . . 402
10.159.2 Function Documentation . . . 402
10.159.3 Variable Documentation . . . 403
10.160 ta-ref/test_hello/optee/Enclave/user_ta_header.c File Reference . . . 404
10.160.1 Macro Definition Documentation . . . 405
10.160.2 Function Documentation . . . 405
10.160.3 Variable Documentation . . . 406
10.161 ta-ref/test_gp/optee/Enclave/user_ta_header_defines.h File Reference . . . 407
10.161.1 Macro Definition Documentation . . . 408
10.162 user_ta_header_defines.h . . . 409
10.163 ta-ref/test_hello/optee/Enclave/user_ta_header_defines.h File Reference . . . 410
10.163.1 Macro Definition Documentation . . . 411
10.164 user_ta_header_defines.h . . . 412
10.165 ta-ref/test_gp/sgx/App/App.h File Reference . . . 412
10.165.1 Macro Definition Documentation . . . 413
10.165.2 Variable Documentation . . . 414
10.166 App.h . . . 414
10.167 ta-ref/test_hello/sgx/App/App.h File Reference . . . 415
10.167.1 Macro Definition Documentation . . . 416
10.167.2 Variable Documentation . . . 416
10.168 App.h . . . 416
10.169 ta-ref/test_gp/sgx/App/App_ocalls.h File Reference . . . 417

10.169.1 Typedef Documentation . . . 418
10.169.2 Function Documentation . . . 418
10.170 App_ocalls.h . . . 425
10.171 ta-ref/test_hello/sgx/App/App_ocalls.h File Reference . . . 426
10.171.1 Typedef Documentation . . . 427
10.171.2 Function Documentation . . . 427
10.172 App_ocalls.h . . . 433
10.173 ta-ref/test_gp/sgx/App/types.h File Reference . . . 433
10.173.1 Typedef Documentation . . . 434
10.173.2 Variable Documentation . . . 434
10.174 types.h . . . 435
10.175 ta-ref/test_hello/sgx/App/types.h File Reference . . . 436
10.175.1 Typedef Documentation . . . 437
10.175.2 Variable Documentation . . . 437
10.176 types.h . . . 437

Index **439**

1 Preparation

1.1 Keystone(RISC-V Unleased)

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 lbzip2 python pkg-config libglib2.0-dev libpixman-1-dev libssl-dev screen
       device-tree-compiler expect makeself 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-gnueabihf.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/knknkn1162/manifest.git -m qemu_v8.xml -b 3.9.0
~/bin/repo sync -j4 --no-clone-bundle
ln -s ~/toolchains toolchains
cd build
make
```

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

Run the script as follows

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

If build is successfull, 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 scucceeded 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

```
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 experianced 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 modprove 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”

```
./run-server
Listening for connections on port 7777
Waiting for a client to connect...
Connection from 127.0.0.1
Waiting for msg0||msg1
---- Copy/Paste Msg2 Below to Client -----------------------------------------
44f2e22125f052e0118018febceefd469f1f4b73de22b34ebdf27003605963946bb862980be691a2f532d8e66abb2ce4eaadf778593b1ce7a14b2759f133b
------------------------------------------------------------------------------
Waiting for msg3
+++ POST data written to /tmp/wgetpostwnMRxU
---- Copy/Paste Msg4 Below to Client -----------------------------------------
030000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
------------------------------------------------------------------------------
Waiting for a client to connect...
```

Open another terminal and run ”run-client”

```
./run-client
---- Copy/Paste Msg0||Msg1 Below to SP ---------------------------------------
00000000a7fa6ed63bec97891885abc2e2e80bd4bb2bd5bb32a7e142337f486bb9f6e76a9db59aa9aaac50cd24c3625451a79bce7c51e24447981444cf516
------------------------------------------------------------------------------
Waiting for msg2
---- Copy/Paste Msg3 Below to SP ---------------------------------------------
787d992031b5ed7d57f149aec7f04912a7fa6ed63bec97891885abc2e2e80bd4bb2bd5bb32a7e142337f486bb9f6e76a9db59aa9aaac50cd24c3625451a79
------------------------------------------------------------------------------
---- 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 reqired 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: 9b04c091da96b997afb8f2585d608aebe9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE_AllocateTransientObject(): start
TEE_GenerateKey(): start
TEE_GenerateRandom(0x000000003FFFFD88, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE_AllocateOperation(): start
TEE_GenerateRandom(0x000000003FFFFED0, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE_CipherInit(): start
TEE_CipherUpdate(): start
TEE_FreeOperation(): start
@cipher:
       e94431cd22a6029185d0dbb1a17b5d62843bfeef25591583d2d668ec6fed1c692f88ce4754d690c346c8d9f2726630e0386abf4e45699a2ca2b34b
TEE_AllocateOperation(): start
TEE_CipherInit(): start
TEE_CipherUpdate(): start
TEE_FreeOperation(): start
TEE_FreeTransientObject(): start
decrypted to:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
verify ok
TEE_AllocateTransientObject(): start
TEE_GenerateKey(): start
TEE_GenerateRandom(0x000000003FFFFC68, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE_AllocateOperation(): start
TEE_GenerateRandom(0x000000003FFFFEC8, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE_AEInit(): start
TEE_AEEncryptFinal(): start
TEE_FreeOperation(): start
@cipher:
       c23e9ce04589e80a66debe23a788ae5393bdcd8e875e87e1bcf2b2d998f6418ccc6ee4ab112fdbfc5175868691efb40781a318ff439d30b49cc9f72
@tag: a551f999317b3fbd1eea7b622ce2caee
TEE_AllocateOperation(): start
TEE_AEInit(): start
TEE_AEDecryptFinal(): start
TEE_FreeOperation(): start
TEE_FreeTransientObject(): start
decrypted to:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
verify ok
TEE_AllocateOperation(): start
TEE_FreeOperation(): start
TEE_DigestDoFinal(): start
TEE_FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aebe9c4a904f7d52c8f28c7e4d2dd9fba5f
```

```
TEE_AllocateOperation(): start
TEE_AllocateTransientObject(): start
TEE_InitValueAttribute(): start
TEE_GenerateKey(): start
TEE_GenerateRandom(0x000000003FFFFE28, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE_AsymmetricSignDigest(): start
TEE_FreeOperation(): start
@signature:
       d6e6b6e54db8b6a62fc1927886938bead27f4813f19ce77182e3016b5426bcad067ca98cd75f9dfddafe9eb0655c48df992d3ad674db69d831f26a
TEE_AllocateOperation(): start
TEE_AsymmetricVerifyDigest(): start
TEE_FreeOperation(): start
@@TEE_FreeOperation:
TEE_FreeTransientObject(): start
verify ok
main end
```

2.5 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:
      30a558176172c53be4a2ac320776de105da79c29726879fe67d06b629f065731285f8a90f8a521ce34eceea51e15e928d157ea10d149bb687dd78be
decrypted to:
      000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a
verify ok
@cipher:
      ff409d8fe203bf0d81de36832b86c702f07edd343f408d3a2fb5ab347b4f72b10031efff0c17b7e0bc56c3f2f95f53c0d731ed87eb3e1187b6714a2
@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:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3
verify ok
TEE_AllocateOperation(): start
TEE_FreeOperation(): start
TEE_DigestDoFinal(): start
TEE_FreeOperation(): start
hash: 9b04c091da96b997afb8f2585d608aebe9c4a904f7d52c8f28c7e4d2dd9fba5f
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:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3
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:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3
```

```
verify ok
TEE_AllocateOperation(): start
TEE_FreeOperation(): start
TEE_DigestDoFinal(): start
TEE_FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aebe9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE_AllocateOperation(): start
TEE_AllocateTransientObject(): start
TEE_InitValueAttribute(): start
TEE_GenerateKey(): start
TEE_GenerateRandom(): start
TEE_AsymmetricSignDigest(): start
TEE_FreeOperation(): start
@signature:
       100b392ce043e9b8dc703088f505dd3083ec47bfcb8d59d968a66b54e80464d684d56dc9c44336f08fd9630979863a2d8fb7cd672a819ef609357e
TEE_AllocateOperation(): start
TEE_AsymmetricVerifyDigest(): start
TEE_FreeOperation(): start
@@TEE_FreeOperation:
TEE_FreeTransientObject(): start
verify ok
main end
Info: Enclave successfully returned.
```

3 Running on Dev Boards

3.1 Keystone, Unleased

Make sure Keystone and other dependant sources have been built

3.1.1 Preparation of rootfs on SD Card

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

Prerequisites: libncursesw5-dev, libpopt-dev

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

3.1.1.1 Create SD-card partition manually

```
sudo ./gdisk /dev/mmcblk0
GPT fdisk (gdisk) version 1.0.4
Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present
Found valid GPT with protective MBR; using GPT.
Command (? for help): n
Partition number (1-128, default 1): 1
First sector (34-15523806, default = 2048) or {+-}size{KMGTP}:
Last sector (2048-15523806, default = 15523806) or {+-}size{KMGTP}: 67583
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300): 5202
Changed type of partition to 'SiFive bare-metal (or stage 2 loader)'
Command (? for help): n
Partition number (2-128, default 2): 4
First sector (34-15523806, default = 67584) or {+-}size{KMGTP}:
Last sector (67584-15523806, default = 15523806) or {+-}size{KMGTP}: 67839
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300): 5201
Changed type of partition to 'SiFive FSBL (first-stage bootloader)'
Command (? for help): n
Partition number (2-128, default 2):
First sector (34-15523806, default = 69632) or {+-}size{KMGTP}: 264192
```

```
Last sector (264192-15523806, default = 15523806) or {+-}size{KMGTP}:
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300): 8300
Changed type of partition to 'Linux filesystem'
Command (? for help): p
Disk /dev/mmcblk0: 15523840 sectors, 7.4 GiB
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 11A0F8F6-D5DE-4993-8C0D-D543DFBA17AD
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 15523806
Partitions will be aligned on 2048-sector boundaries
Total free space is 198366 sectors (96.9 MiB)
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048           67583   32.0 MiB    5202  SiFive bare-metal (...
   2          264192        15523806   7.3 GiB     8300  Linux filesystem
   4           67584           67839   128.0 KiB   5201  SiFive FSBL (first-...
Command (? for help): i
Partition number (1-4): 4
Partition GUID code: 5B193300-FC78-40CD-8002-E86C45580B47 (SiFive FSBL (first-stage bootloader))
Partition unique GUID: FC1FBC7C-EC94-4B0A-9DAF-0ED85452B885
First sector: 67584 (at 33.0 MiB)
Last sector: 67839 (at 33.1 MiB)
Partition size: 256 sectors (128.0 KiB)
Attribute flags: 0000000000000000
Partition name: 'SiFive FSBL (first-stage bootloader)'
Command (? for help): i
Partition number (1-4): 1
Partition GUID code: 2E54B353-1271-4842-806F-E436D6AF6985 (SiFive bare-metal (or stage 2 loader))
Partition unique GUID: 2FFF07EF-E44A-4278-A16D-C29697C6653D
First sector: 2048 (at 1024.0 KiB)
Last sector: 67583 (at 33.0 MiB)
Partition size: 65536 sectors (32.0 MiB)
Attribute flags: 0000000000000000
Partition name: 'SiFive bare-metal (or stage 2 loader'
Command (? for help): wq
Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!
Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/mmcblk1.
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.
```

3.1.1.2 Write boot and rootfs files into SD-card

Build FSBL for hifive-Unleased board

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

Writing fsbl.bin and bbl.bin

```
sudo dd if=freedom-u540-c000-bootloader/fsbl.bin of=/dev/mmcblk0p4 bs=4096 conv=fsync
sudo dd if=$KEYSTONE_DIR/hifive-work/bbl.bin of=/dev/mmcblk0p1 bs=4096 conv=fsync
```

Once files written, insert the SD-card into unleased

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 unleased board.

3.1.3 Check test_hello and test_gp on Unleased

1. Insert SD-card into unleased 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:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3
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:
       e94431cd22a6029185d0dbb1a17b5d62843bfeef25591583d2d668ec6fed1c692f88ce4754d690c346c8d9f2726630e0386abf4e45699a2ca2b34b3
TEE_AllocateOperation(): start
TEE_CipherInit(): start
TEE_CipherUpdate(): start
TEE_FreeOperation(): start
TEE_FreeTransientObject(): start
decrypted to:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3
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:
       c23e9ce04589e80a66debe23a788ae5393bdcd8e875e87e1bcf2b2d998f6418ccc6ee4ab112fdbfc5175868691efb40781a318ff439d30b49cc9f72
@tag: a551f999317b3fbd1eea7b622ce2caee
TEE_AllocateOperation(): start
TEE_AEInit(): start
TEE_AEDecryptFinal(): start
TEE_FreeOperation(): start
TEE_FreeTransientObject(): start
decrypted to:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3
verify ok
TEE_AllocateOperation(): start
TEE_FreeOperation(): start
TEE_DigestDoFinal(): start
TEE_FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aebe9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE_AllocateOperation(): start
TEE_AllocateTransientObject(): start
TEE_InitValueAttribute(): start
TEE_GenerateKey(): start
TEE_GenerateRandom(0x000000003FFFFE28, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE_AsymmetricSignDigest(): start
TEE_FreeOperation(): start
@signature:
       d6e6b6e54db8b6a62fc1927886938bead27f4813f19ce77182e3016b5426bcad067ca98cd75f9dfddafe9eb0655c48df992d3ad674db69d831f26ae
TEE_AllocateOperation(): start
TEE_AsymmetricVerifyDigest(): start
TEE_FreeOperation(): start
@@TEE_FreeOperation:
TEE_FreeTransientObject(): start
verify ok
main end
```

Test is successful.

3.2 OPTEE, RPI3

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

3.2.1 Preparation of rootfs on SD Card

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

```
make img-help
$ fdisk /dev/sdx   # where sdx is the name of your sd-card
   > p             # prints partition table
   > d             # repeat until all partitions are deleted
   > n             # create a new partition
   > p             # create primary
   > 1             # make it the first partition
   > <enter>       # use the default sector
   > +32M          # create a boot partition with 32MB of space
   > n             # create rootfs partition
   > p
   > 2
   > <enter>
   > <enter>       # fill the remaining disk, adjust size to fit your needs
   > t             # change partition type
   > 1             # select first partition
   > e             # use type 'e' (FAT16)
   > a             # make partition bootable
   > 1             # select first partition
   > p             # double check everything looks right
   > w             # write partition table to disk.
```

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

Write boot file

```
$ mkfs.vfat -F16 -n BOOT /dev/mmcblk0p1
```

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

Write rootfs

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

If you use CI from AIST, download rpi3_sdimage as follows

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

Now SD-card is ready to boot RPI3.

3.2.2 Copying binaries of test_hello and test_gp to rootfs partition

Copying test_hello & test_gp

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

3.2.3 Check test_hello and test_gp

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

Login to Serial console and enter ”root” as username

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

test_hello:

```
cp /home/gitlab/out/test_hello/
cp a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta /home/gitlab/out/
ln -s /home/gitlab/out/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
       /lib64/optee_armtz/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
./optee_ref_ta
--- enclave log start---
ecall_ta_main() start
hello world!
ecall_ta_main() end
--- enclave log end---
```

If executed successfully, you see above messages

test_gp:

```
cd /home/gitlab/out/test_gp/
cp a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta /home/gitlab/out/
```

```
ln -s /home/gitlab/out/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
       /lib64/optee_armtz/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
./optee_ref_ta
start TEEC_InvokeCommand
--- enclave log start---
ecall_ta_main() start
@random: fe0c7d3eefb9bd5e63b8a0cce29af7eb
@GP REE time 1612156259 sec 390 millis
@GP System time 249187 sec 954 millis
256 bytes read:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3
verify ok
hash: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@cipher:
       30a558176172c53be4a2ac320776de105da79c29726879fe67d06b629f065731285f8a90f8a521ce34eceea51e15e928d157ea10d149bb687dd78be
decrypted to:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3
verify ok
@cipher:
       ff409d8fe203bf0d81de36832b86c702f07edd343f408d3a2fb5ab347b4f72b10031efff0c17b7e0bc56c3f2f95f53c0d731ed87eb3e1187b6714a2
@tag: 9b357baf76d2632fa7d16231640d6324
decrypted to:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3
verify ok
@digest: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@signature:
       719fa9898f3423b754675b835268f9b2368b77a429eeabf7369d60d135dee08158c3902fd2ed3c1bf17cb34e76f2ba25da915fa3970c757962f7531
@@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
```

```
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:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3
verify ok
TEE_AllocateOperation(): start
TEE_FreeOperation(): start
TEE_DigestDoFinal(): start
TEE_FreeOperation(): start
hash: 9b04c091da96b997afb8f2585d608aebe9c4a904f7d52c8f28c7e4d2dd9fba5f
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:
       7427bff21e729a824a239e25332ebd455d18fa6aec1ec6618b77c252f768e0a9345608b0135727568867ce5b0fac872f6647787861b882208402811
TEE_AllocateOperation(): start
TEE_CipherInit(): start
TEE_CipherUpdate(): start
TEE_FreeOperation(): start
TEE_FreeTransientObject(): start
decrypted to:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3
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:
       e33f34122c80b9a10002725e4e21542256da7c7cd3f6dd1b62b71cf8308f9e4a0daa50b29880a8f76707c4ed432549c4da9e68e7930189d2127fdd7
@tag: 4c920ce2aef079e468ab24e25730d9d2
TEE_AllocateOperation(): start
TEE_AEInit(): start
TEE_AEDecryptFinal(): start
TEE_FreeOperation(): start
TEE_FreeTransientObject(): start
decrypted to:
       000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3
verify ok
TEE_AllocateOperation(): start
TEE_FreeOperation(): start
TEE_DigestDoFinal(): start
TEE_FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aebe9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE_AllocateOperation(): start
TEE_AllocateTransientObject(): start
TEE_InitValueAttribute(): start
TEE_GenerateKey(): start
TEE_GenerateRandom(): start
TEE_AsymmetricSignDigest(): start
TEE_FreeOperation(): start
@signature:
       100b392ce043e9b8dc703088f505dd3083ec47bfcb8d59d968a66b54e80464d684d56dc9c44336f08fd9630979863a2d8fb7cd672a819ef609357e9
TEE_AllocateOperation(): start
TEE_AsymmetricVerifyDigest(): start
TEE_FreeOperation(): start
@@TEE_FreeOperation:
TEE_FreeTransientObject(): start
verify ok
main end
Info: Enclave successfully returned.
```

4 Overview of ta-ref

4.1 Features

4.1.1 What we did on RISC-V

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

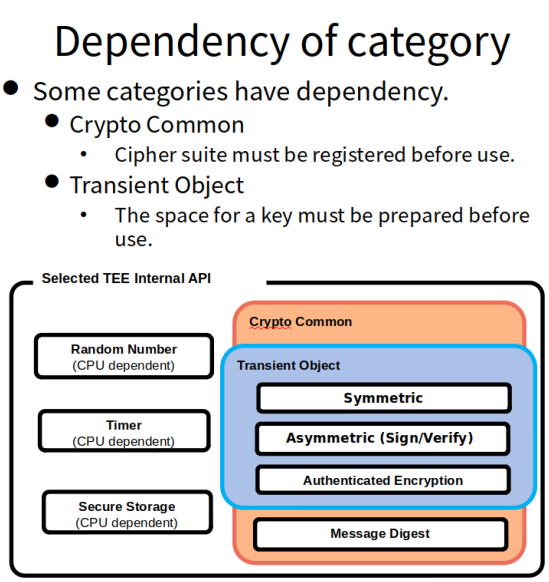
4.1.2 Separate GP TEE Internal API

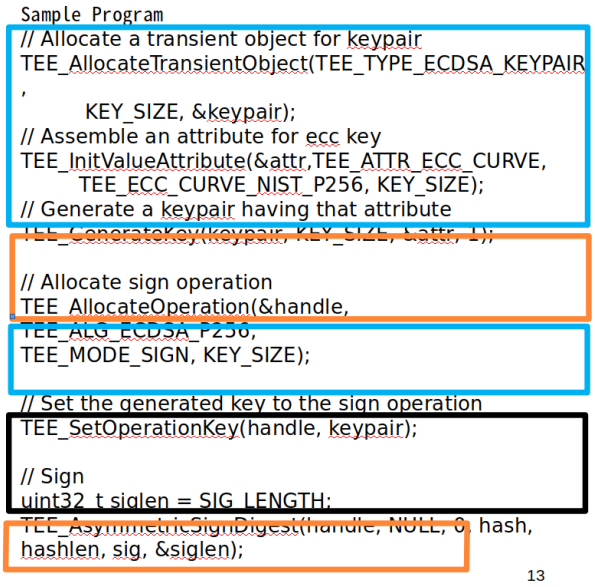
- CPU architecture dependent
  - Random Generator, Time, Secure Storage, Transient Object(TEE_GenerateKey)
- CPU architecture independent(Crypto)
  - Transient Object(exclude TEE_GenerateKey), Crypto Common, Authenticated Encryption, Symmetric/↩ Asymmetric Cipher, Message Digest

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

4.2 Diagram

4.2.1 Dependency of category





5 How to Program on ta-ref

5.1 Time Functions

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

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

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

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

5.2 Random Functions

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

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

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

5.3 Hash Functions

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

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

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

5.4 Symmetric Crypto Functions

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

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

```
    // Encrypt test data
    rv = TEE_AllocateOperation(&handle, TEE_ALG_AES_CBC_NOPAD, TEE_MODE_ENCRYPT, 256);
    GP_ASSERT(rv, "TEE_AllocateOperation fails");
    rv = TEE_SetOperationKey(handle, key);
    GP_ASSERT(rv, "TEE_SetOperationKey fails");
    TEE_GenerateRandom(iv, sizeof(iv));
    TEE_CipherInit(handle, iv, sizeof(iv));
    //GP_ASSERT(rv, "TEE_AEInit fails");
    outlen = CIPHER_LENGTH;
    rv = TEE_CipherUpdate(handle, data, CIPHER_LENGTH, out, &outlen);
    GP_ASSERT(rv, "TEE_CipherUpdate fails");
    TEE_FreeOperation(handle);
    // Dump encrypted data
    tee_printf("@cipher: ");
    for (int i = 0; i < CIPHER_LENGTH; i++) {
        tee_printf ("%02x", out[i]);
    }
    tee_printf("\n");
    // Decrypt it
    rv= TEE_AllocateOperation(&handle, TEE_ALG_AES_CBC_NOPAD, TEE_MODE_DECRYPT, 256);
    GP_ASSERT(rv, "TEE_AllocateOperation fails");
    rv = TEE_SetOperationKey(handle, key);
    GP_ASSERT(rv, "TEE_SetOperationKey fails");
    TEE_CipherInit(handle, iv, sizeof(iv));
    //GP_ASSERT(rv, "TEE_AEInit fails");
    outlen = CIPHER_LENGTH;
    rv = TEE_CipherUpdate(handle, out, CIPHER_LENGTH, out, &outlen);
    GP_ASSERT(rv, "TEE_CipherUpdate fails");
    TEE_FreeOperation(handle);
    TEE_FreeTransientObject(key);
    // Dump data
    tee_printf("decrypted to: ");
    for (int i = 0; i < CIPHER_LENGTH; i++) {
        tee_printf ("%02x", out[i]);
    }
    tee_printf("\n");
    // Verify decrypted data against original one
    int verify_ok;
    verify_ok = !memcmp(out, data, CIPHER_LENGTH);
    if (verify_ok) {
        tee_printf("verify ok\n");
    } else {
        tee_printf("verify fails\n");
    }
}
--- AE decrypt and verify end ---
```

5.5 Asymmetric Crypto Functions

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

```
--- Asymmetric Key sign start ---
void gp_asymmetric_key_sign_test(void)
{
    static unsigned char data[256] = {
      // 0x00,0x01,...,0xff
#include "test.dat"
    };
    unsigned char hash[SHA_LENGTH];
    unsigned char sig[SIG_LENGTH];
    TEE_OperationHandle handle;
    uint32_t hashlen = SHA_LENGTH;
    TEE_Result rv;

    // Take hash of test data
    /* Calculate hash */
    /* sha3_init() in sha3.c */
    rv = TEE_AllocateOperation(&handle, TEE_ALG_SHA256, TEE_MODE_DIGEST, SHA_LENGTH);
    GP_ASSERT(rv, "TEE_AllocateOperation fails");
    /* sha3_update() in sha3.c */
    TEE_DigestUpdate(handle, data, sizeof(data));

    /* sha3_final() in sha3.c */
    rv = TEE_DigestDoFinal(handle, NULL, 0, hash, &hashlen);
    GP_ASSERT(rv, "TEE_DigestDoFinal fails");
    /* free up */
    TEE_FreeOperation(handle);
```

```
    /* Get the signature */
    // Dump hashed data
    tee_printf("@digest: ");
    for (int i = 0; i < SHA_LENGTH; i++) {
      tee_printf ("%02x", hash[i]);
    }
    tee_printf("\n");
    uint32_t siglen = SIG_LENGTH;
    TEE_ObjectHandle keypair;
    // Sign hashed data with the generated keys
    /* set ecdsa_p256 key */
    rv = TEE_AllocateOperation(&handle, TEE_ALG_ECDSA_P256, TEE_MODE_SIGN, 256);
    GP_ASSERT(rv, "TEE_AllocateOperation fails");
    // Generate keypair
    rv = TEE_AllocateTransientObject(TEE_TYPE_ECDSA_KEYPAIR, 256, &keypair);
    GP_ASSERT(rv, "TEE_AllocateTransientObject fails");
    TEE_Attribute attr;
    TEE_InitValueAttribute(&attr,
               TEE_ATTR_ECC_CURVE,
               TEE_ECC_CURVE_NIST_P256,
               256);
    rv = TEE_GenerateKey(keypair, 256, &attr, 1);
    GP_ASSERT(rv, "TEE_GenerateKey fails");
    rv = TEE_SetOperationKey(handle, keypair);
    GP_ASSERT(rv, "TEE_SetOperationKey fails");
    /* Keystone has ecdsa_p256_sign() Equivalent in openssl is EVP_DigestSign() */
    rv = TEE_AsymmetricSignDigest(handle, NULL, 0, hash, hashlen, sig, &siglen);
    GP_ASSERT(rv, "TEE_AsymmetricSignDigest fails");

    /* free up */
    TEE_FreeOperation(handle);
    /* Get the signature */
    // Dump signature
    tee_printf("@signature: ");
    for (uint32_t i = 0; i < siglen; i++) {
      tee_printf ("%02x", sig[i]);
    }
    tee_printf("\n");
    // Verify signature against hashed data
    /* set ecdsa_p256 key */
    rv = TEE_AllocateOperation(&handle, TEE_ALG_ECDSA_P256, TEE_MODE_VERIFY, 256);
    GP_ASSERT(rv, "TEE_AllocateOperation fails");
    rv = TEE_SetOperationKey(handle, keypair);
    GP_ASSERT(rv, "TEE_SetOperationKey fails");
    /* Keystone has ecdsa_p256_verify() Equivalent in openssl is EVP_DigestVerify() */
    TEE_Result verify_ok;
    verify_ok = TEE_AsymmetricVerifyDigest(handle, NULL, 0, hash, hashlen, sig, siglen);
    /* free up */
    TEE_FreeOperation(handle);
    tee_printf("@@TEE_FreeOperation: \n");
    TEE_FreeTransientObject(keypair);

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

5.6 Asymmetric Crypto Gcm Functions

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

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

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

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

5.7 Open, Read, Write, Close On Secure Storage

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

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

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

6 API Compare With Full-Set of GP API

6.1 GP API

API Functions by Category

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

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

Asymmetric
- TEE_AsymmetricDecrypt
- TEE_AsymmetricEncrypt
- TEE_AsymmetricSignDigest
- TEE_AsymmetricVerifyDigest

Authenticated Encryption
- TEE_AEDecryptFinal
- TEE_AEEncryptFinal
- TEE_AEInit
- TEE_AEUpdate
- TEE_AEUpdateAAD

Basic Arithmetic
- TEE_BigIntAdd
- TEE_BigIntDiv
- TEE_BigIntMul
- TEE_BigIntNeg
- TEE_BigIntSquare
- TEE_BigIntSub

Cancellation
- TEE_GetCancellationFlag
- TEE_MaskCancellation
- TEE_UnmaskCancellation

Converter
- TEE_BigIntConvertFromOctetString
- TEE_BigIntConvertFromS32
- TEE_BigIntConvertToOctetString
- TEE_BigIntConvertToS32

Data Stream Access
- TEE_ReadObjectData
- TEE_SeekObjectData
- TEE_TruncateObjectData
- TEE_WriteObjectData

Deprecated
- TEE_CloseAndDeletePersistentObject
- TEE_CopyObjectAttributes
- TEE_GetObjectInfo,
- TEE_RestrictObjectUsage

Fast Modular Multiplication
- TEE_BigIntComputeFMM
- TEE_BigIntConvertFromFMM
- TEE_BigIntConvertToFMM

Generic Object
- TEE_CloseObject
- TEE_GetObjectBufferAttribute
- TEE_GetObjectInfo (deprecated)
- TEE_GetObjectInfo1
- TEE_GetObjectValueAttribute
- TEE_RestrictObjectUsage (deprecated)
- TEE_RestrictObjectUsage1

Generic Operation
- TEE_AllocateOperation
- TEE_CopyOperation
- TEE_FreeOperation
- TEE_GetOperationInfo
- TEE_GetOperationInfoMultiple
- TEE_IsAlgorithmSupported
- TEE_ResetOperation
- TEE_SetOperationKey
- TEE_SetOperationKey2

Initialization
- TEE_BigIntInit
- TEE_BigIntInitFMM
- TEE_BigIntInitFMMContext

Internal Client API
- TEE_CloseTASession
- TEE_InvokeTACommand
- TEE_OpenTASession

Key Derivation
- TEE_DeriveKey

Logical Operation
- TEE_BigIntCmp
- TEE_BigIntCmpS32
- TEE_BigIntGetBit
- TEE_BigIntGetBitCount
- TEE_BigIntShiftRight

MAC
- TEE_MACCompareFinal
- TEE_MACComputeFinal
- TEE_MACInit
- TEE_MACUpdate

Memory Allocation and Size of Objects
- TEE_BigIntFMMContextSizeInU32
- TEE_BigIntFMMSizeInU32
- TEE_BigIntSizeInU32 (macro)

Memory Management
- TEE_CheckMemoryAccessRights
- TEE_Free
- TEE_GetInstanceData
- TEE_Malloc
- TEE_MemCompare
- TEE_MemFill
- TEE_MemMove
- TEE_Realloc
- TEE_SetInstanceData

Message Digest
- TEE_DigestDoFinal
- TEE_DigestUpdate

Modular Arithmetic
- TEE_BigIntAddMod
- TEE_BigIntInvMod
- TEE_BigIntMod
- TEE_BigIntMulMod
- TEE_BigIntSquareMod
- TEE_BigIntSubMod

- Other Arithmetic
  - TEE_BigIntComputeExtendedGcd
  - TEE_BigIntIsProbablePrime
  - TEE_BigIntRelativePrime
- Panic Function
  - TEE_Panic
- Persistent Object
  - TEE_CloseAndDeletePersistentObject (deprecated)
  - TEE_CloseAndDeletePersistentObject1
  - TEE_CreatePersistentObject
  - TEE_OpenPersistentObject
  - TEE_RenamePersistentObject
- Persistent Object Enumeration *
  - TEE_AllocatePersistentObjectEnumerator
  - TEE_FreePersistentObjectEnumerator
  - TEE_GetNextPersistentObject
  - TEE_ResetPersistentObjectEnumerator
  - TEE_StartPersistentObjectEnumerator
- Property Access
  - TEE_AllocatePropertyEnumerator
  - TEE_FreePropertyEnumerator
  - TEE_GetNextProperty
  - TEE_GetPropertyAsBinaryBlock
  - TEE_GetPropertyAsBool
  - TEE_GetPropertyAsIdentity
  - TEE_GetPropertyAsString
  - TEE_GetPropertyAsU32
  - TEE_GetPropertyAsU64
  - TEE_GetPropertyAsUUID
  - TEE_GetPropertyName
  - TEE_ResetPropertyEnumerator
  - TEE_StartPropertyEnumerator
- Random Data Generation
  - TEE_GenerateRandom
- Symmetric Cipher
  - TEE_CipherDoFinal
  - TEE_CipherInit
  - TEE_CipherUpdate
- TA Interface
  - TA_CloseSessionEntryPoint
  - TA_CreateEntryPoint
  - TA_DestroyEntryPoint
  - TA_InvokeCommandEntryPoint
  - TA_OpenSessionEntryPoint
- Time
  - TEE_GetREETime
  - TEE_GetSystemTime
  - TEE_GetTAPersistentTime
  - TEE_SetTAPersistentTime
  - TEE_Wait
- Transient Object
  - TEE_AllocateTransientObject
  - TEE_CopyObjectAttributes (deprecated)
  - TEE_CopyObjectAttributes1
  - TEE_FreeTransientObject
  - TEE_GenerateKey
  - TEE_InitRefAttribute
  - TEE_InitValueAttribute
  - TEE_PopulateTransientObject
  - TEE_ResetTransientObject

7 Class Index

7.1 Class List

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

__profiler_data 35
__profiler_header 36
__TEE_ObjectHandle 36
__TEE_OperationHandle 37
_sgx_errlist_t 39
addrinfo 39
enclave_report 41
invoke_command_t 41
list 43
nm_info 43
nonce_t 44
ob16_t 44
ob196_t 45
ob256_t 46
out_fct_wrap_type 46
pollfd 47
ree_time_t 47
report 48
result 49
sm_report 50
TEE_Attribute 50
TEE_Identity 52
TEE_ObjectInfo 53
TEE_OperationInfo 54
TEE_OperationInfoKey 55
TEE_OperationInfoMultiple 56
TEE_Param 57
TEE_SEAID 58

TEE_SEReaderProperties 59

TEE_Time 60

TEE_UUID 60

TEEC_Context 61

TEEC_Operation 62

TEEC_Parameter 63

TEEC_RegisteredMemoryReference 64

TEEC_Session 66

TEEC_SharedMemory 66

TEEC_TempMemoryReference 68

TEEC_UUID 69

TEEC_Value 70

8 File Index

8.1 File List

Here is a list of all files with brief descriptions:

ta-ref/api/tee-internal-api-cryptlib.c 237

ta-ref/api/include/compiler.h 71

ta-ref/api/include/report.h 80

ta-ref/api/include/tee-common.h
Common type and definitions of RISC-V TEE 82

ta-ref/api/include/tee-ta-internal.h
Candidate API list for Global Platform like RISC-V TEE 83

ta-ref/api/include/tee_api_defines.h 110

ta-ref/api/include/tee_api_defines_extensions.h 153

ta-ref/api/include/tee_api_types.h 158

ta-ref/api/include/tee_client_api.h 166

ta-ref/api/include/tee_internal_api.h 181

ta-ref/api/include/tee_internal_api_extensions.h 181

ta-ref/api/include/tee_ta_api.h 184

ta-ref/api/include/test_dev_key.h 189

ta-ref/api/include/trace.h 190

ta-ref/api/include/trace_levels.h 198

ta-ref/api/keystone/tee-internal-api-machine.c 200

ta-ref/api/keystone/tee-internal-api.c 201

ta-ref/api/keystone/tee_api_tee_types.h 221

ta-ref/api/keystone/teec_stub.c 228

ta-ref/api/keystone/trace.c 232

ta-ref/api/optee/tee_api_tee_types.h 224

ta-ref/api/sgx/tee-internal-api.c 212

ta-ref/api/sgx/tee_api_tee_types.h 224

ta-ref/benchmark/bench.c 251

ta-ref/benchmark/bench.h 255

ta-ref/benchmark/cpu_bench.c 258

ta-ref/benchmark/io_bench.c 262

ta-ref/benchmark/memory_bench.c 268

ta-ref/benchmark/time_test.c 270

ta-ref/benchmark/include/config_bench.h 260

ta-ref/benchmark/keystone/tee_def.h 264

ta-ref/benchmark/optee/tee_def.h 266

ta-ref/benchmark/sgx/tee_def.h 267

ta-ref/edger/edger8r/user_types.h 271

ta-ref/edger/keyedge/Enclave_t.c 272

ta-ref/edger/keyedge/Enclave_t.h 273

ta-ref/edger/keyedge/Enclave_u.c 274

ta-ref/edger/keyedge/Enclave_u.h 274

ta-ref/edger/keyedge/ocalls.h 276

ta-ref/edger/optee/Enclave.h 282

ta-ref/edger/optee/Enclave_t.h 274

ta-ref/gp/asymmetric_key.c 286

ta-ref/gp/invoke_command.c 293

ta-ref/gp/message_digest.c 294

ta-ref/gp/random.c 295

ta-ref/gp/secure_stoage.c 296

ta-ref/gp/symmetric_key.c 297

ta-ref/gp/symmetric_key_gcm.c 298

ta-ref/gp/time.c 299

ta-ref/gp/include/config_ref_ta.h 288

ta-ref/gp/include/gp_test.h 291

ta-ref/profiler/profiler.c 330

ta-ref/profiler/profiler.h 332

ta-ref/profiler/profiler_attrs.h 333

ta-ref/profiler/profiler_data.h 334

ta-ref/profiler/analyzer/analyzer.c 300

ta-ref/profiler/analyzer/analyzer.h 302

ta-ref/profiler/analyzer/nm_parse.c 303

ta-ref/profiler/analyzer/nm_parse.h 306

ta-ref/profiler/analyzer/stack.h 308

ta-ref/profiler/app/tools.c 311

ta-ref/profiler/keystone/tee_config.h 317

ta-ref/profiler/keystone/tee_profiler.c 321

ta-ref/profiler/keystone/tee_profiler.h 327

ta-ref/profiler/keystone/Enclave/tools.c 312

ta-ref/profiler/optee/tee_config.h 318

ta-ref/profiler/optee/tee_profiler.c 323

ta-ref/profiler/optee/tee_profiler.h 328

ta-ref/profiler/optee/Enclave/tools.c 313

ta-ref/profiler/sgx/tee_config.h 320

ta-ref/profiler/sgx/tee_profiler.c 324

ta-ref/profiler/sgx/tee_profiler.h 329

ta-ref/profiler/sgx/Enclave/tools.c 314

ta-ref/test_gp/crt.c 337

ta-ref/test_gp/tools.c 315

ta-ref/test_gp/vsnprintf.c 354

ta-ref/test_gp/include/crt.h 344

ta-ref/test_gp/include/ocall_wrapper.h 346

ta-ref/test_gp/include/random.h 347

ta-ref/test_gp/include/tools.h 348

ta-ref/test_gp/keystone/App/App.cpp 366

ta-ref/test_gp/keystone/App/App_ocalls.cpp 372

ta-ref/test_gp/keystone/Enclave/Enclave.c 388

ta-ref/test_gp/keystone/Enclave/ocall_wrapper.c 350

ta-ref/test_gp/keystone/Enclave/startup.c 352

ta-ref/test_gp/keystone/Enclave/trace.c 234

ta-ref/test_gp/optee/App/main.c 398

ta-ref/test_gp/optee/Enclave/crt.c 339

ta-ref/test_gp/optee/Enclave/Enclave.c 389

ta-ref/test_gp/optee/Enclave/trace.c 235

ta-ref/test_gp/optee/Enclave/user_ta_header.c 401

ta-ref/test_gp/optee/Enclave/user_ta_header_defines.h 407

ta-ref/test_gp/sgx/App/App.cpp 367

ta-ref/test_gp/sgx/App/App.h 412

ta-ref/test_gp/sgx/App/App_ocalls.cpp 377

ta-ref/test_gp/sgx/App/App_ocalls.h 417

ta-ref/test_gp/sgx/App/types.h 433

ta-ref/test_gp/sgx/Enclave/Enclave.c 390

ta-ref/test_gp/sgx/Enclave/Enclave.h 283

ta-ref/test_gp/sgx/Enclave/ocall_wrapper.c 351

ta-ref/test_gp/sgx/Enclave/startup.c 353

ta-ref/test_gp/sgx/Enclave/trace.c 236

ta-ref/test_hello/keystone/App/App.cpp 369

ta-ref/test_hello/keystone/App/App_ocalls.cpp 381

ta-ref/test_hello/keystone/Enclave/Enclave.c 391

ta-ref/test_hello/optee/App/main.c 400

ta-ref/test_hello/optee/Enclave/Enclave.c 393

ta-ref/test_hello/optee/Enclave/user_ta_header.c 404

ta-ref/test_hello/optee/Enclave/user_ta_header_defines.h 410

ta-ref/test_hello/sgx/App/App.cpp 370

ta-ref/test_hello/sgx/App/App.h 415

ta-ref/test_hello/sgx/App/App_ocalls.cpp 385

ta-ref/test_hello/sgx/App/App_ocalls.h 426

ta-ref/test_hello/sgx/App/types.h 436

ta-ref/test_hello/sgx/Enclave/Enclave.c 397

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`

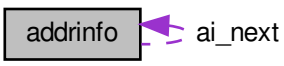
The documentation for this struct was generated from the following files:

- 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_t Struct Reference

`#include <ocalls.h>`

Public Attributes

- unsigned int commandID
- char params0_buffer [256]
- unsigned int params0_size
- int param1_fd
- char params1_buffer [256]
- unsigned int params1_size

9.8.1 Member Data Documentation

9.8.1.1 commandID `unsigned int invoke_command_t::commandID`

9.8.1.2 param1_fd `int invoke_command_t::param1_fd`

9.8.1.3 params0_buffer `char invoke_command_t::params0_buffer[256]`

9.8.1.4 params0_size `unsigned int invoke_command_t::params0_size`

9.8.1.5 params1_buffer `char invoke_command_t::params1_buffer[256]`

9.8.1.6 params1_size `unsigned int invoke_command_t::params1_size`

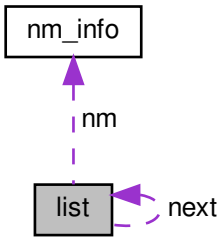
The documentation for this struct was generated from the following file:

- ta-ref/edger/keyedge/ocalls.h

9.9 list Struct Reference

`#include <nm_parse.h>`

Collaboration diagram for list:



Public Attributes

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

9.9.1 Member Data Documentation

9.9.1.1 addr `uintptr_t list::addr`

9.9.1.2 next `struct list* list::next`

9.9.1.3 nm `struct nm_info* list::nm`

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

- ta-ref/profiler/analyzer/nm_parse.h

9.10 nm_info Struct Reference

`#include <nm_parse.h>`

Public Attributes

- char type
- char func_name [256]

9.10.1 Member Data Documentation

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

9.10.1.2 type `char nm_info::type`

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

- ta-ref/profiler/analyzer/nm_parse.h

9.11 nonce_t Struct Reference

`#include <ocalls.h>`

Public Attributes

- unsigned char nonce [NONCE_SIZE]

9.11.1 Member Data Documentation

9.11.1.1 nonce `unsigned char nonce_t::nonce[NONCE_SIZE]`

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

- ta-ref/edger/keyedge/ocalls.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 file:

- ta-ref/test_gp/vsnprintf.c

9.16 pollfd Struct Reference

`#include <tee_api_types.h>`

Public Attributes

- int fd
- short int events
- short int revents

9.16.1 Member Data Documentation

9.16.1.1 events `short int pollfd::events`

9.16.1.2 fd `int pollfd::fd`

9.16.1.3 revents `short int pollfd::revents`

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

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

9.17 ree_time_t Struct Reference

`#include <ocalls.h>`

Public Attributes

- unsigned int seconds
- unsigned int millis

9.17.1 Member Data Documentation

9.17.1.1 millis `unsigned int ree_time_t::millis`

9.17.1.2 seconds `unsigned int ree_time_t::seconds`

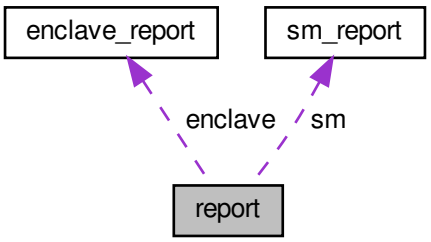
The documentation for this struct was generated from the following files:

- ta-ref/edger/keyedge/ocalls.h
- ta-ref/test_hello/sgx/App/App_ocalls.h
- ta-ref/test_gp/sgx/App/App_ocalls.h

9.18 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]

9.18.1 Member Data Documentation

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

9.18.1.2 enclave `struct enclave_report report::enclave`

9.18.1.3 sm `struct sm_report report::sm`

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

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

9.19 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

9.19.1 Member Data Documentation

9.19.1.1 callee `uintptr_t result::callee`

9.19.1.2 depth `size_t result::depth`

9.19.1.3 end `__profiler_nsec_t result::end`

9.19.1.4 end_hartid `uint8_t result::end_hartid`

9.19.1.5 idx `size_t result::idx`

9.19.1.6 start __profiler_nsec_t result::start

9.19.1.7 start_hartid uint8_t result::start_hartid

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

- ta-ref/profiler/analyzer/analyzer.h

9.20 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]

9.20.1 Member Data Documentation

9.20.1.1 hash uint8_t sm_report::hash[MDSIZE]

9.20.1.2 public_key uint8_t sm_report::public_key[PUBLIC_KEY_SIZE]

9.20.1.3 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

9.21 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

9.21.1 Member Data Documentation

9.21.1.1 a `uint32_t TEE_Attribute::a`

9.21.1.2 attributeID `uint32_t TEE_Attribute::attributeID`

9.21.1.3 b `uint32_t TEE_Attribute::b`

9.21.1.4 buffer `void* TEE_Attribute::buffer`

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

9.21.1.6 length `uint32_t TEE_Attribute::length`

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

9.21.1.8 struct { ... } TEE_Attribute::value

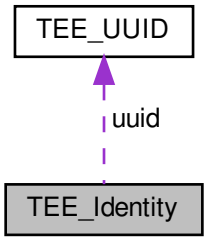
The documentation for this struct was generated from the following file:

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

9.22 TEE_Identity Struct Reference

`#include <tee_api_types.h>`

Collaboration diagram for TEE_Identity:



Public Attributes

- uint32_t login
- TEE_UUID uuid

9.22.1 Member Data Documentation

9.22.1.1 login uint32_t TEE_Identity::login

9.22.1.2 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

9.23 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

9.23.1 Member Data Documentation

9.23.1.1 `__extension__ union { ... } TEE_ObjectInfo::@3`

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

9.23.1.3 dataPosition `uint32_t TEE_ObjectInfo::dataPosition`

9.23.1.4 dataSize `uint32_t TEE_ObjectInfo::dataSize`

9.23.1.5 handleFlags `uint32_t TEE_ObjectInfo::handleFlags`

9.23.1.6 keySize `uint32_t TEE_ObjectInfo::keySize`

9.23.1.7 maxKeySize `uint32_t TEE_ObjectInfo::maxKeySize`

9.23.1.8 maxObjectSize `uint32_t TEE_ObjectInfo::maxObjectSize`

9.23.1.9 objectSize `uint32_t TEE_ObjectInfo::objectSize`

9.23.1.10 objectType `uint32_t TEE_ObjectInfo::objectType`

9.23.1.11 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

9.24 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

9.24.1 Member Data Documentation

9.24.1.1 algorithm `uint32_t TEE_OperationInfo::algorithm`

9.24.1.2 digestLength `uint32_t TEE_OperationInfo::digestLength`

9.24.1.3 handleState `uint32_t TEE_OperationInfo::handleState`

9.24.1.4 keySize `uint32_t TEE_OperationInfo::keySize`

9.24.1.5 maxKeySize `uint32_t TEE_OperationInfo::maxKeySize`

9.24.1.6 mode `uint32_t TEE_OperationInfo::mode`

9.24.1.7 operationClass `uint32_t TEE_OperationInfo::operationClass`

9.24.1.8 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

9.25 TEE_OperationInfoKey Struct Reference

`#include <tee_api_types.h>`

Public Attributes

- uint32_t keySize
- uint32_t requiredKeyUsage

9.25.1 Member Data Documentation

9.25.1.1 keySize `uint32_t TEE_OperationInfoKey::keySize`

9.25.1.2 requiredKeyUsage `uint32_t TEE_OperationInfoKey::requiredKeyUsage`

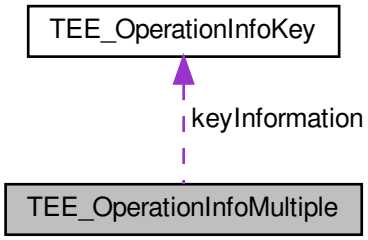
The documentation for this struct was generated from the following file:

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

9.26 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 [ ]

9.26.1 Member Data Documentation

9.26.1.1 algorithm `uint32_t TEE_OperationInfoMultiple::algorithm`

9.26.1.2 digestLength `uint32_t TEE_OperationInfoMultiple::digestLength`

9.26.1.3 handleState `uint32_t TEE_OperationInfoMultiple::handleState`

9.26.1.4 keyInformation `TEE_OperationInfoKey TEE_OperationInfoMultiple::keyInformation[]`

9.26.1.5 maxKeySize `uint32_t TEE_OperationInfoMultiple::maxKeySize`

9.26.1.6 mode `uint32_t TEE_OperationInfoMultiple::mode`

9.26.1.7 numberOfKeys `uint32_t TEE_OperationInfoMultiple::numberOfKeys`

9.26.1.8 operationClass `uint32_t TEE_OperationInfoMultiple::operationClass`

9.26.1.9 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

9.27 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

9.27.1 Member Data Documentation

9.27.1.1 a `uint32_t TEE_Param::a`

9.27.1.2 b `uint32_t TEE_Param::b`

9.27.1.3 buffer `void* TEE_Param::buffer`

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

9.27.1.5 size `uint32_t TEE_Param::size`

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

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

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

9.28 TEE_SEAID Struct Reference

`#include <tee_api_types.h>`

Public Attributes

- uint8_t * buffer
- size_t bufferLen

9.28.1 Member Data Documentation

9.28.1.1 buffer `uint8_t* TEE_SEAID::buffer`

9.28.1.2 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

9.29 TEE_SEReaderProperties Struct Reference

`#include <tee_api_types.h>`

Public Attributes

- bool sePresent
- bool teeOnly
- bool selectResponseEnable

9.29.1 Member Data Documentation

9.29.1.1 selectResponseEnable `bool TEE_SEReaderProperties::selectResponseEnable`

9.29.1.2 sePresent `bool TEE_SEReaderProperties::sePresent`

9.29.1.3 teeOnly `bool TEE_SEReaderProperties::teeOnly`

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

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

9.30 TEE_Time Struct Reference

`#include <tee_api_types.h>`

Public Attributes

- uint32_t seconds
- uint32_t millis

9.30.1 Member Data Documentation

9.30.1.1 millis `uint32_t TEE_Time::millis`

9.30.1.2 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

9.31 TEE_UUID Struct Reference

`#include <tee_api_types.h>`

Public Attributes

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

9.31.1 Member Data Documentation

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

9.31.1.2 timeHiAndVersion `uint16_t TEE_UUID::timeHiAndVersion`

9.31.1.3 timeLow `uint32_t TEE_UUID::timeLow`

9.31.1.4 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

9.32 TEEC_Context Struct Reference

`#include <tee_client_api.h>`

Public Attributes

- int fd
- bool reg_mem

9.32.1 Detailed Description

struct TEEC_Context - Represents a connection between a client application and a TEE.

9.32.2 Member Data Documentation

9.32.2.1 fd `int TEEC_Context::fd`

9.32.2.2 reg_mem `bool TEEC_Context::reg_mem`

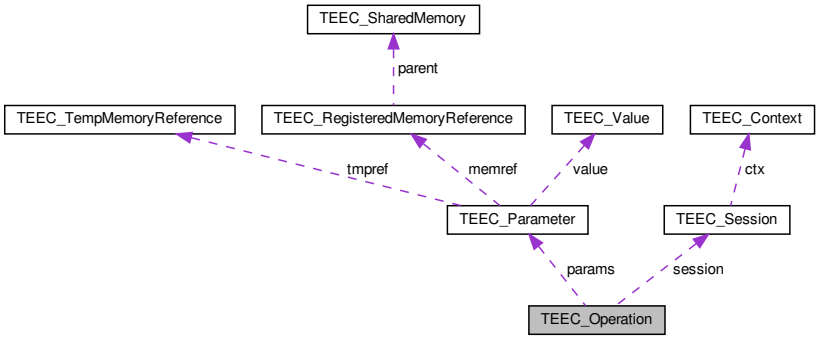
The documentation for this struct was generated from the following file:

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

9.33 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

9.33.1 Detailed Description

struct TEEC_Operation - Holds information and memory references used in TEEC_InvokeCommand().

Parameters

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

9.33.2 Member Data Documentation

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

9.33.2.2 paramTypes `uint32_t TEEC_Operation::paramTypes`

9.33.2.3 session `TEEC_Session* TEEC_Operation::session`

9.33.2.4 started `uint32_t TEEC_Operation::started`

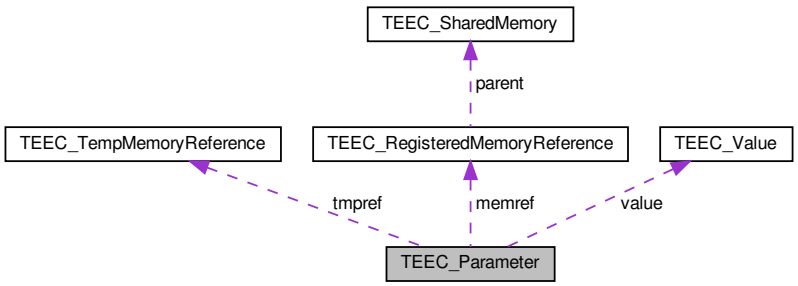
The documentation for this struct was generated from the following file:

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

9.34 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

9.34.1 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

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

9.34.2 Member Data Documentation

9.34.2.1 memref `TEEC_RegisteredMemoryReference TEEC_Parameter::memref`

9.34.2.2 tmpref `TEEC_TempMemoryReference TEEC_Parameter::tmpref`

9.34.2.3 value `TEEC_Value TEEC_Parameter::value`

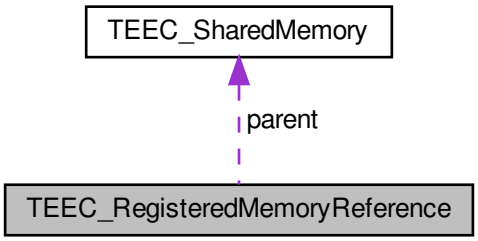
The documentation for this union was generated from the following file:

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

9.35 TEEC_RegisteredMemoryReference Struct Reference

`#include <tee_client_api.h>`

Collaboration diagram for TEEC_RegisteredMemoryReference:



Public Attributes

- TEEC_SharedMemory * parent
- size_t size
- size_t offset

9.35.1 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

parent	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
size	The size, in bytes, of the memory buffer.
offset	The offset, in bytes, of the referenced memory region from the start of the shared memory block.

9.35.2 Member Data Documentation

9.35.2.1 offset `size_t TEEC_RegisteredMemoryReference::offset`

9.35.2.2 parent `TEEC_SharedMemory* TEEC_RegisteredMemoryReference::parent`

9.35.2.3 size `size_t TEEC_RegisteredMemoryReference::size`

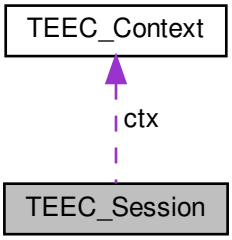
The documentation for this struct was generated from the following file:

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

9.36 TEEC_Session Struct Reference

`#include <tee_client_api.h>`

Collaboration diagram for TEEC_Session:



Public Attributes

- TEEC_Context * ctx
- uint32_t session_id

9.36.1 Detailed Description

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

9.36.2 Member Data Documentation

9.36.2.1 ctx `TEEC_Context* TEEC_Session::ctx`

9.36.2.2 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

9.37 TEEC_SharedMemory Struct Reference

`#include <tee_client_api.h>`

Public Attributes

- void * buffer
- size_t size
- uint32_t flags
- int id
- size_t alloced_size
- void * shadow_buffer
- int registered_fd
- bool buffer_allocated

9.37.1 Detailed Description

struct TEEC_SharedMemory - Memory to transfer data between a client application and trusted code.

Parameters

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

9.37.2 Member Data Documentation

9.37.2.1 alloced_size `size_t TEEC_SharedMemory::alloced_size`

9.37.2.2 buffer `void* TEEC_SharedMemory::buffer`

9.37.2.3 buffer_allocated `bool TEEC_SharedMemory::buffer_allocated`

9.37.2.4 flags `uint32_t TEEC_SharedMemory::flags`

9.37.2.5 id `int TEEC_SharedMemory::id`

9.37.2.6 registered_fd `int TEEC_SharedMemory::registered_fd`

9.37.2.7 shadow_buffer `void* TEEC_SharedMemory::shadow_buffer`

9.37.2.8 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

9.38 TEEC_TempMemoryReference Struct Reference

`#include <tee_client_api.h>`

Public Attributes

- void * buffer
- size_t size

9.38.1 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

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

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

9.38.2 Member Data Documentation

9.38.2.1 buffer `void* TEEC_TempMemoryReference::buffer`

9.38.2.2 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

9.39 TEEC_UUID Struct Reference

`#include <tee_client_api.h>`

Public Attributes

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

9.39.1 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.

9.39.2 Member Data Documentation

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

9.39.2.2 timeHiAndVersion `uint16_t TEEC_UUID::timeHiAndVersion`

9.39.2.3 timeLow `uint32_t TEEC_UUID::timeLow`

9.39.2.4 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

9.40 TEEC_Value Struct Reference

`#include <tee_client_api.h>`

Public Attributes

- uint32_t a
- uint32_t b

9.40.1 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

a	The first integer value.
b	The second second value.

9.40.2 Member Data Documentation

9.40.2.1 a `uint32_t TEEC_Value::a`

9.40.2.2 b `uint32_t TEEC_Value::b`

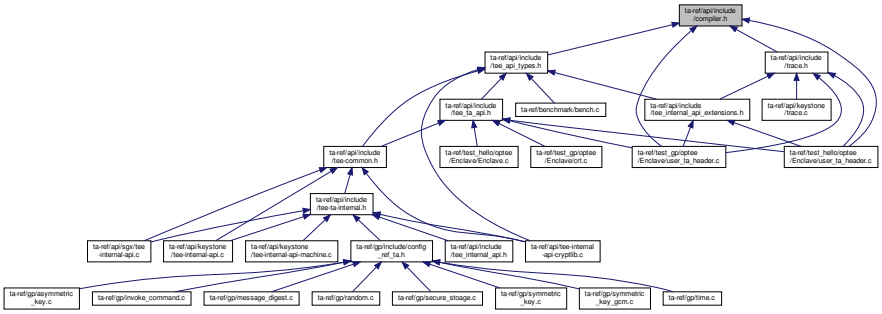
The documentation for this struct was generated from the following file:

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

10 File Documentation

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

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



Macros

- #define __deprecated __attribute__((deprecated))
- #define __packed __attribute__((packed))
- #define __weak __attribute__((weak))
- #define __noreturn __attribute__((noreturn))
- #define __pure __attribute__((pure))
- #define __aligned(x) __attribute__((aligned(x)))
- #define __printf(a, b) __attribute__((format(printf, a, b)))
- #define __noinline __attribute__((noinline))
- #define __attr_const __attribute__((__const__))
- #define __unused __attribute__((unused))
- #define __maybe_unused __attribute__((unused))
- #define __used __attribute__((__used__))
- #define __must_check __attribute__((warn_unused_result))
- #define __cold __attribute__((__cold__))
- #define __section(x) __attribute__((section(x)))
- #define __data __section(".data")
- #define __bss __section(".bss")
- #define __rodata __section(".rodata")
- #define __rodata_unpaged __section(".rodata.__unpaged")
- #define __early_ta __section(".rodata.early_ta")
- #define __noprof __attribute__((no_instrument_function))
- #define __compiler_bswap64(x) __builtin_bswap64((x))
- #define __compiler_bswap32(x) __builtin_bswap32((x))
- #define __compiler_bswap16(x) __builtin_bswap16((x))
- #define __GCC_VERSION
- #define __INTOF_HALF_MAX_SIGNED(type) ((type)1 << (sizeof(type)*8-2))
- #define __INTOF_MAX_SIGNED(type)
- #define __INTOF_MIN_SIGNED(type) (-1 - __INTOF_MAX_SIGNED(type))
- #define __INTOF_MIN(type) ((type)-1 < 1?__INTOF_MIN_SIGNED(type):(type)0)
- #define __INTOF_MAX(type) ((type)~__INTOF_MIN(type))
- #define __INTOF_ASSIGN(dest, src)
- #define __INTOF_ADD(c, a, b)

- #define __INTOF_SUB(c, a, b)
- #define __intof_mul_negate ((__intof_oa < 1) != (__intof_ob < 1))
- #define __intof_mul_hshift (sizeof(uintmax_t) * 8 / 2)
- #define __intof_mul_hmask (UINTMAX_MAX >> __intof_mul_hshift)
- #define __intof_mul_a0 ((uintmax_t)(__intof_a) >> __intof_mul_hshift)
- #define __intof_mul_b0 ((uintmax_t)(__intof_b) >> __intof_mul_hshift)
- #define __intof_mul_a1 ((uintmax_t)(__intof_a) & __intof_mul_hmask)
- #define __intof_mul_b1 ((uintmax_t)(__intof_b) & __intof_mul_hmask)
- #define __intof_mul_t
- #define __INTOF_MUL(c, a, b)
- #define __compiler_add_overflow(a, b, res) __INTOF_ADD(*(res), (a), (b))
- #define __compiler_sub_overflow(a, b, res) __INTOF_SUB(*(res), (a), (b))
- #define __compiler_mul_overflow(a, b, res) __INTOF_MUL(*(res), (a), (b))
- #define __compiler_compare_and_swap(p, oval, nval)
- #define __compiler_atomic_load(p) __atomic_load_n((p), __ATOMIC_RELAXED)
- #define __compiler_atomic_store(p, val) __atomic_store_n((p), (val), __ATOMIC_RELAXED)

10.1.1 Macro Definition Documentation

10.1.1.1 __aligned

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

10.1.1.2 __attr_const

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

10.1.1.3 __bss

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

10.1.1.4 __cold

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

10.1.1.5 __compiler_add_overflow

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

10.1.1.6 __compiler_atomic_load

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

10.1.1.7 __compiler_atomic_store

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

10.1.1.8 __compiler_bswap16

```
#define __compiler_bswap16(
        x ) __builtin_bswap16((x))
```

10.1.1.9 __compiler_bswap32

```
#define __compiler_bswap32(
        x ) __builtin_bswap32((x))
```

10.1.1.10 __compiler_bswap64

```
#define __compiler_bswap64(
        x ) __builtin_bswap64((x))
```

10.1.1.11 __compiler_compare_and_swap

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

Value:

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

__HAVE_BUILTIN_OVERFLOW

10.1.1.12 __compiler_mul_overflow

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

10.1.1.13 __compiler_sub_overflow

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

10.1.1.14 __data

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

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

10.1.1.16 __early_ta `#define __early_ta __section(".rodata.early_ta")`

10.1.1.17 __GCC_VERSION `#define __GCC_VERSION`

Value:

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

10.1.1.18 __INTOF_ADD `#define __INTOF_ADD(`

c,

a,

b)

Value:

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

10.1.1.19 __INTOF_ASSIGN `#define __INTOF_ASSIGN(`

dest,

src)

Value:

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

10.1.1.20 __INTOF_HALF_MAX_SIGNED `#define __INTOF_HALF_MAX_SIGNED(`

*type) ((type)1 << (sizeof(type)*8-2))*

__HAVE_BUILTIN_OVERFLOW

10.1.1.21 __INTOF_MAX `#define __INTOF_MAX(`

type) ((type)~__INTOF_MIN(type))

10.1.1.22 __INTOF_MAX_SIGNED #define __INTOF_MAX_SIGNED(

type)

Value:

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

10.1.1.23 __INTOF_MIN #define __INTOF_MIN(

type) ((type)-1 < 1?__INTOF_MIN_SIGNED(type):(type)0)

10.1.1.24 __INTOF_MIN_SIGNED #define __INTOF_MIN_SIGNED(

type) (-1 - __INTOF_MAX_SIGNED(type))

10.1.1.25 __INTOF_MUL #define __INTOF_MUL(

c,

a,

b)

Value:

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

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

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

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

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

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

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

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

10.1.1.33 __intof_mul_t `#define __intof_mul_t`

Value:

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

10.1.1.34 __INTOF_SUB `#define __INTOF_SUB(`

c,

a,

b)

Value:

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

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

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

10.1.1.37 __noinline

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

10.1.1.38 __noprof

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

10.1.1.39 __noreturn

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

10.1.1.40 __packed

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

10.1.1.41 __printf

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

10.1.1.42 __pure

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

10.1.1.43 __rodata

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

10.1.1.44 __rodata_unpaged

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

10.1.1.45 __section

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

10.1.1.46 __unused

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

10.1.1.47 __used

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

10.1.1.48 __weak #define __weak __attribute__((weak))

10.2 compiler.h

Go to the documentation of this file.

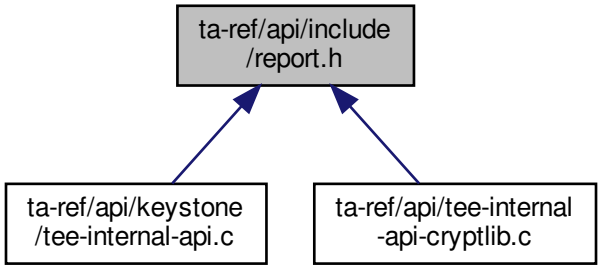
```
1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10  *
11  * 2. Redistributions in binary form must reproduce the above copyright notice,
12  * this list of conditions and the following disclaimer in the documentation
13  * and/or other materials provided with the distribution.
14  *
15  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25  * POSSIBILITY OF SUCH DAMAGE.
26  */
27
28 #ifndef COMPILER_H
29 #define COMPILER_H
30
31 /*
32  * Macros that should be used instead of using __attribute__ directly to
33  * ease portability and make the code easier to read.
34  */
35
36 #define __deprecated    __attribute__((deprecated))
37 #define __packed    __attribute__((packed))
38 #define __weak      __attribute__((weak))
39 #define __noreturn  __attribute__((noreturn))
40 #define __pure      __attribute__((pure))
41 #define __aligned(x)    __attribute__((aligned(x)))
42 #define __printf(a, b)  __attribute__((format(printf, a, b)))
43 #define __noinline  __attribute__((noinline))
44 #define __attr_const    __attribute__((__const__))
45 #define __unused    __attribute__((unused))
46 #define __maybe_unused  __attribute__((unused))
47 #define __used      __attribute__((__used__))
48 #define __must_check    __attribute__((warn_unused_result))
49 #define __cold      __attribute__((__cold__))
50 #define __section(x)    __attribute__((section(x)))
51 #define __data      __section(".data")
52 #define __bss       __section(".bss")
53 #define __rodata    __section(".rodata")
54 #define __rodata_unpaged __section(".rodata.__unpaged")
55 #define __early_ta  __section(".rodata.early_ta")
56 #define __noprof    __attribute__((no_instrument_function))
57
58 #define __compiler_bswap64(x)   __builtin_bswap64((x))
59 #define __compiler_bswap32(x)   __builtin_bswap32((x))
60 #define __compiler_bswap16(x)   __builtin_bswap16((x))
61
62 #define __GCC_VERSION (__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + \
63                __GNUC_PATCHLEVEL__)
64
65 #if __GCC_VERSION >= 50100 && !defined(__CHECKER__)
66 #define __HAVE_BUILTIN_OVERFLOW 1
67 #endif
68
69 #ifdef __HAVE_BUILTIN_OVERFLOW
70 #define __compiler_add_overflow(a, b, res) \
71     __builtin_add_overflow((a), (b), (res))
72
73 #define __compiler_sub_overflow(a, b, res) \
74     __builtin_sub_overflow((a), (b), (res))
75
76 #define __compiler_mul_overflow(a, b, res) \
77     __builtin_mul_overflow((a), (b), (res))
```

```
78 #else
80 /*
81  * Copied/inspired from https://www.fefe.de/intof.html
82  */
83 #define __INTOF_HALF_MAX_SIGNED(type) ((type)1 << (sizeof(type)*8-2))
84 #define __INTOF_MAX_SIGNED(type) (__INTOF_HALF_MAX_SIGNED(type) - 1 + \
85                 __INTOF_HALF_MAX_SIGNED(type))
86 #define __INTOF_MIN_SIGNED(type) (-1 - __INTOF_MAX_SIGNED(type))
87
88 #define __INTOF_MIN(type) ((type)-1 < 1?__INTOF_MIN_SIGNED(type):(type)0)
89 #define __INTOF_MAX(type) ((type)~__INTOF_MIN(type))
90
91 #define __INTOF_ASSIGN(dest, src) (__extension__({ \
92     typeof(src) __intof_x = (src); \
93     typeof(dest) __intof_y = __intof_x; \
94     (((uintmax_t)__intof_x == (uintmax_t)__intof_y) && \
95      ((__intof_x < 1) == (__intof_y < 1)) ? \
96         (void)((dest) = __intof_y) , 0 : 1); \
97 }))
98
99 #define __INTOF_ADD(c, a, b) (__extension__({ \
100     typeof(a) __intofa_a = (a); \
101     typeof(b) __intofa_b = (b); \
102     \
103     __intofa_b < 1 ? \
104         ((__INTOF_MIN(typeof(c)) - __intofa_b <= __intofa_a) ? \
105             __INTOF_ASSIGN((c), __intofa_a + __intofa_b) : 1) : \
106         ((__INTOF_MAX(typeof(c)) - __intofa_b >= __intofa_a) ? \
107             __INTOF_ASSIGN((c), __intofa_a + __intofa_b) : 1); \
108 }))
109
110 #define __INTOF_SUB(c, a, b) (__extension__({ \
111     typeof(a) __intofs_a = a; \
112     typeof(b) __intofs_b = b; \
113     \
114     __intofs_b < 1 ? \
115         ((__INTOF_MAX(typeof(c)) + __intofs_b >= __intofs_a) ? \
116             __INTOF_ASSIGN((c), __intofs_a - __intofs_b) : 1) : \
117         ((__INTOF_MIN(typeof(c)) + __intofs_b <= __intofs_a) ? \
118             __INTOF_ASSIGN((c), __intofs_a - __intofs_b) : 1); \
119 }))
120
121 /*
122  * Dealing with detecting overflow in multiplication of integers.
123  *
124  * First step is to remove two corner cases with the minum signed integer
125  * which can't be represented as a positive integer + sign.
126  * Multiply with 0 or 1 can't overflow, no checking needed of the operation,
127  * only if it can be assigned to the result.
128  *
129  * After the corner cases are eliminated we convert the two factors to
130  * positive unsigned values, keeping track of the original in another
131  * variable which is used at the end to determine the sign of the product.
132  *
133  * The two terms (a and b) are divided into upper and lower half (x1 upper
134  * and x0 lower), so the product is:
135  * ((a1 << hshift) + a0) * ((b1 << hshift) + b0)
136  * which also is:
137  * ((a1 * b1) << (hshift * 2)) +                (T1)
138  * ((a1 * b0 + a0 * b1) << hshift) +                (T2)
139  * (a0 * b0)                                (T3)
140  *
141  * From this we can tell and (a1 * b1) has to be 0 or we'll overflow, that
142  * is, at least one of a1 or b1 has to be 0. Once this has been checked the
143  * addition: ((a1 * b0) << hshift) + ((a0 * b1) << hshift)
144  * isn't an addition as one of the terms will be 0.
145  *
146  * Since each factor in: (a0 * b0)
147  * only uses half the capicity of the underlaying type it can't overflow
148  *
149  * The addition of T2 and T3 can overflow so we use __INTOF_ADD() to
150  * perform that addition. If the addition succeeds without overflow the
151  * result is assigned the required sign and checked for overflow again.
152  */
153
154 #define __intof_mul_negate  ((__intof_oa < 1) != (__intof_ob < 1))
155 #define __intof_mul_hshift  (sizeof(uintmax_t) * 8 / 2)
156 #define __intof_mul_hmask   (UINTMAX_MAX >> __intof_mul_hshift)
157 #define __intof_mul_a0      ((uintmax_t)(__intof_a) >> __intof_mul_hshift)
158 #define __intof_mul_b0      ((uintmax_t)(__intof_b) >> __intof_mul_hshift)
159 #define __intof_mul_a1      ((uintmax_t)(__intof_a) & __intof_mul_hmask)
160 #define __intof_mul_b1      ((uintmax_t)(__intof_b) & __intof_mul_hmask)
161 #define __intof_mul_t       (__intof_mul_a1 * __intof_mul_b0 + \
162                 __intof_mul_a0 * __intof_mul_b1)
163
164 #define __INTOF_MUL(c, a, b) (__extension__({ \
165     typeof(a) __intof_oa = (a); \
```

```
166     typeof(a) __intof_a = __intof_oa < 1 ? -__intof_oa : __intof_oa; \
167     typeof(b) __intof_ob = (b); \
168     typeof(b) __intof_b = __intof_ob < 1 ? -__intof_ob : __intof_ob; \
169     typeof(c) __intof_c; \
170     \
171     __intof_oa == 0 || __intof_ob == 0 || \
172     __intof_oa == 1 || __intof_ob == 1 ? \
173         __INTOF_ASSIGN((c), __intof_oa * __intof_ob) : \
174     (__intof_mul_a0 && __intof_mul_b0) || \
175      __intof_mul_t > __intof_mul_hmask ?  1 : \
176     __INTOF_ADD((__intof_c), __intof_mul_t << __intof_mul_hshift, \
177                 __intof_mul_a1 * __intof_mul_b1) ? 1 : \
178     __intof_mul_negate ? __INTOF_ASSIGN((c), -__intof_c) : \
179                 __INTOF_ASSIGN((c), __intof_c); \
180 }))
181
182 #define __compiler_add_overflow(a, b, res) __INTOF_ADD(*(res), (a), (b))
183 #define __compiler_sub_overflow(a, b, res) __INTOF_SUB(*(res), (a), (b))
184 #define __compiler_mul_overflow(a, b, res) __INTOF_MUL(*(res), (a), (b))
185
186 #endif
188 #define __compiler_compare_and_swap(p, oval, nval) \
189     __atomic_compare_exchange_n((p), (oval), (nval), true, \
190                     __ATOMIC_ACQUIRE, __ATOMIC_RELAXED) \
191
192 #define __compiler_atomic_load(p) __atomic_load_n((p), __ATOMIC_RELAXED)
193 #define __compiler_atomic_store(p, val) \
194     __atomic_store_n((p), (val), __ATOMIC_RELAXED)
195
196 #endif /*COMPILER_H*/
```

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

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



Classes

- struct enclave_report
- struct sm_report
- struct report

Macros

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

10.3.1 Macro Definition Documentation

10.3.1.1 ATTEST_DATA_MAXLEN #define ATTEST_DATA_MAXLEN 1024

10.3.1.2 MDSIZE #define MDSIZE 64

10.3.1.3 PUBLIC_KEY_SIZE #define PUBLIC_KEY_SIZE 32

10.3.1.4 SIGNATURE_SIZE #define SIGNATURE_SIZE 64

10.4 report.h

Go to the documentation of this file.

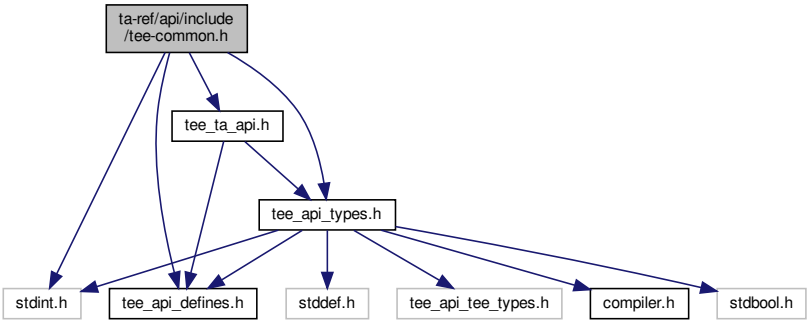
```
1 #ifndef _REPORT_H
2 #define _REPORT_H
3
4 #define MDSIZE  64
5 #define SIGNATURE_SIZE  64
6 #define PUBLIC_KEY_SIZE 32
7 #define ATTEST_DATA_MAXLEN  1024
8
9 /* attestation reports */
10 struct enclave_report
11 {
12   uint8_t hash[MDSIZE];
13   uint64_t data_len;
14   uint8_t data[ATTEST_DATA_MAXLEN];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

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

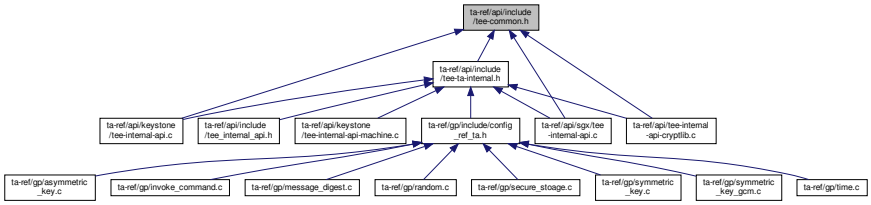
Common type and definitions of RISC-V TEE.

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

Include dependency graph for tee-common.h:



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



Macros

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

10.5.1 Detailed Description

Common type and definitions of RISC-V TEE.

draft RISC-V Internal TEE API

Author

Akira Tsukamoto, AIST

Date

2019/09/25

10.5.2 Macro Definition Documentation

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

10.6 tee-common.h

Go to the documentation of this file.

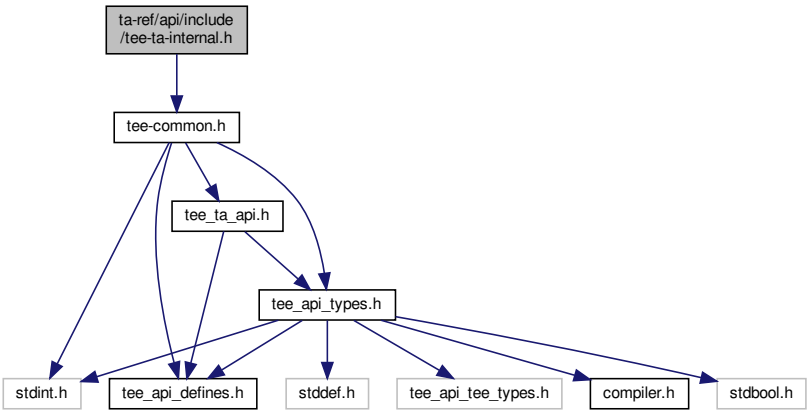
```
1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                          and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10  *
11  * 1. Redistributions of source code must retain the above copyright notice,
12  * this list of conditions and the following disclaimer.
13  *
14  * 2. Redistributions in binary form must reproduce the above copyright notice,
15  * this list of conditions and the following disclaimer in the documentation
16  * and/or other materials provided with the distribution.
17  *
18  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28  * POSSIBILITY OF SUCH DAMAGE.
29  */
38 #ifndef TEE_COMMON_H
39 #define TEE_COMMON_H
40
41 #include <stdint.h>
42
43 #ifdef __cplusplus
44 extern "C" {
45 #endif
46
47 #ifdef DEBUG
48 #define pr_deb(...)       do { printf(__VA_ARGS__); } while (0)
49 #else
50 #define pr_deb(...)       do { } while (0)
51 #endif /* DEBUG */
52
53 //#include <tee_api.h>
54 #include <tee_api_defines.h>
55 #include <tee_api_types.h>
56 #include <tee_ta_api.h>
57
58 //typedef uint32_t TEE_Result;
59
60 #ifdef __cplusplus
61 }
62 #endif
63
64 #endif /* TEE_COMMON_H */
```

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

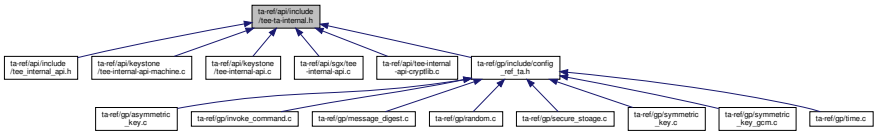
Candidate API list for Global Platform like RISC-V TEE.

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

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



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



Functions

- void __attribute__ ((noreturn)) TEE_Panic(unsigned long code)
- void TEE_GetREETime (TEE_Time *time)

  Core Functions, Time Functions.
- void TEE_GetSystemTime (TEE_Time *time)

  Core Functions, Time Functions.
- TEE_Result GetRelTimeStart (uint64_t start)

  Core Functions, Time Functions.
- TEE_Result GetRelTimeEnd (uint64_t end)

  Core Functions, Time Functions.
- TEE_Result TEE_CreatePersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle attributes, const void *initialData, uint32_t initialDataLen, TEE_ObjectHandle *object)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- TEE_Result TEE_OpenPersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle *object)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- TEE_Result TEE_GetObjectInfo1 (TEE_ObjectHandle object, TEE_ObjectInfo *objectInfo)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- TEE_Result TEE_WriteObjectData (TEE_ObjectHandle object, const void *buffer, uint32_t size)

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

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

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- void TEE_CloseObject (TEE_ObjectHandle object)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- void TEE_GenerateRandom (void *randomBuffer, uint32_t randomBufferLen)

  Crypto, common.
- TEE_Result TEE_AllocateOperation (TEE_OperationHandle *operation, uint32_t algorithm, uint32_t mode, uint32_t maxKeySize)

  Crypto, for all Crypto Functions.
- void TEE_FreeOperation (TEE_OperationHandle operation)

  Crypto, for all Crypto Functions.
- void TEE_DigestUpdate (TEE_OperationHandle operation, const void *chunk, uint32_t chunkSize)

  Crypto, Message Digest Functions.
- TEE_Result TEE_DigestDoFinal (TEE_OperationHandle operation, const void *chunk, uint32_t chunkLen, void *hash, uint32_t *hashLen)
- TEE_Result TEE_SetOperationKey (TEE_OperationHandle operation, TEE_ObjectHandle key)

  Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- TEE_Result TEE_AEInit (TEE_OperationHandle operation, const void *nonce, uint32_t nonceLen, uint32_t tagLen, uint32_t AADLen, uint32_t payloadLen)

  Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- TEE_Result TEE_AEUpdate (TEE_OperationHandle operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)

  Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- void TEE_AEUpdateAAD (TEE_OperationHandle operation, const void *AADdata, uint32_t AADdataLen)

  Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- TEE_Result TEE_AEEncryptFinal (TEE_OperationHandle operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen, void *tag, uint32_t *tagLen)

  Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- TEE_Result TEE_AEDecryptFinal (TEE_OperationHandle operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen, void *tag, uint32_t tagLen)

  Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- void TEE_CipherInit (TEE_OperationHandle operation, const void *nonce, uint32_t nonceLen)

  Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- TEE_Result TEE_CipherUpdate (TEE_OperationHandle operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)

  Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- TEE_Result TEE_GenerateKey (TEE_ObjectHandle object, uint32_t keySize, const TEE_Attribute *params, uint32_t paramCount)

  Crypto, Asymmetric key Verification Functions.
- TEE_Result TEE_AllocateTransientObject (TEE_ObjectType objectType, uint32_t maxKeySize, TEE_ObjectHandle *object)

  Crypto, Asymmetric key Verification Functions.
- void TEE_InitRefAttribute (TEE_Attribute *attr, uint32_t attributeID, const void *buffer, uint32_t length)

  Crypto, Asymmetric key Verification Functions.
- void TEE_InitValueAttribute (TEE_Attribute *attr, uint32_t attributeID, uint32_t a, uint32_t b)

  Crypto, Asymmetric key Verification Functions.
- void TEE_FreeTransientObject (TEE_ObjectHandle object)

  Crypto, Asymmetric key Verification Functions.
- TEE_Result TEE_AsymmetricSignDigest (TEE_OperationHandle operation, const TEE_Attribute *params, uint32_t paramCount, const void *digest, uint32_t digestLen, void *signature, uint32_t *signatureLen)

  Crypto, Asymmetric key Verification Functions.
- TEE_Result TEE_AsymmetricVerifyDigest (TEE_OperationHandle operation, const TEE_Attribute *params, uint32_t paramCount, const void *digest, uint32_t digestLen, const void *signature, uint32_t signatureLen)

  Crypto, Asymmetric key Verification Functions.

10.7.1 Detailed Description

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

draft RISC-V Internal TEE API

Author

Akira Tsukamoto, AIST

Date

2019/09/25

10.7.2 Function Documentation

10.7.2.1 __attribute__() `void __attribute__ ( (noreturn) )`

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

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

Parameters

code	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

ec	An informative panic code defined by the TA. May be displayed in traces if traces are available.

10.7.2.2 GetRelTimeEnd() `TEE_Result GetRelTimeEnd (`

```
        uint64_t end )
```

Core Functions, Time Functions.

Return the elapsed.

GetRelTimeEnd() - finds the real time of the end timing.

This function prints the ending time.

Parameters

end	End timing

Returns

0 If success

GetRelTimeStart() - find the real time of the end timing.

This function prints the End timing.

Parameters

end	End timing

Returns

0 if success else error occured

10.7.2.3 GetRelTimeStart()

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

Core Functions, Time Functions.

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

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

GetRelTimeStart() - Gets the real time of the start timing.

This function prints the starting time.

Parameters

start	Start timing

Returns

0 on success

GetRelTimeStart() - Gets the real time of the start timing.

Ths function prints the start timing.

Parameters

start	start timing

Returns

0 if success else error occured.

10.7.2.4 TEE_AEDecryptFinal()

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

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

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

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

Parameters

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

Returns

0 on success.

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

TEE_ERROR_MAC_INVALID If the computed tag does not match the supplied tag

10.7.2.5 TEE_AEEncryptFinal()

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

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

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

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

Parameters

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

Returns

0 on success.

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

10.7.2.6 TEE_AEInit()

```
TEE_Result TEE_AEInit (
        TEE_OperationHandle operation,
        const void * nonce,
        uint32_t nonceLen,
        uint32_t tagLen,
```

```
            uint32_t AADLen,
            uint32_t payloadLen )
```

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

TEE_AEInit() - Initializes an Authentication Encryption operation.

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

Parameters

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

Returns

0 on success.

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

10.7.2.7 TEE_AEUpdate()

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

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

TEE_AEUpdate() - Accumulates data for an Authentication Encryption operation

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

Parameters

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

Returns

0 on success.

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

10.7.2.8 TEE_AEUpdateAAD()

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

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

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

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

Parameters

operation	Handle on the AE operation
AADdata	Input buffer containing the chunk of AAD
AADdataLen	length of the chunk of AAD.

10.7.2.9 TEE_AllocateOperation()

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

Crypto, for all Crypto Functions.

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

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

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

Parameters

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

Returns

0 in case of success

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

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

10.7.2.10 TEE_AllocateTransientObject()

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

Crypto, Asymmetric key Verification Functions.

Create object storing asymmetric key.

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

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

Parameters

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

Returns

0 on success

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

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

10.7.2.11 TEE_AsymmetricSignDigest()

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

Crypto, Asymmetric key Verification Functions.

Sign a message digest within an asymmetric key operation.
Keystone has ed25519_sign().
Equivalent in openssl is EVP_DigestSign().

TEE_AsymmetricSignDigest() - Signs a message digest within an asymmetric operation.

Parameters

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

Returns

0 on sccess

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

10.7.2.12 TEE_AsymmetricVerifyDigest()

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

Crypto, Asymmetric key Verification Functions.

Verifies a message digest signature within an asymmetric key operation.
Keystone has ed25519_verify().
Equivalent in openssl is EVP_DigestVerify().

TEE_AsymmetricVerifyDigest() - verifies a message digest signature within an asymmetric operation.

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

Parameters

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

Returns

TEE_SUCCESS on success

TEE_ERROR_SIGNATURE_INVALID if the signature is invalid.

10.7.2.13 TEE_CipherInit() `void TEE_CipherInit (`

```
TEE_OperationHandle operation,
const void * nonce,
uint32_t nonceLen )
```

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CBC.

TEE_CipherInit() - starts the symmetric cipher operation.

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

Parameters

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

10.7.2.14 TEE_CipherUpdate() `TEE_Result TEE_CipherUpdate (`

```
TEE_OperationHandle operation,
const void * srcData,
uint32_t srcLen,
void * destData,
uint32_t * destLen )
```

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CBC.

TEE_CipherUpdate() - encrypts or decrypts input data.

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

Parameters

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

10.7.2.15 TEE_CloseObject() `void TEE_CloseObject ( TEE_ObjectHandle object )`

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

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

TEE_CloseObject() - Closes an opened object handle.

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

Parameters

object	Handle of the object.

Returns

TEE_SUCCESS if success else error occured.

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

object	Handle of the object

Returns

TEE_SUCCESS if success else error occured.

10.7.2.16 TEE_CreatePersistentObject()

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

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

Create persistent object (key, key-pair or Data).
For the people who have not written code on GP then probably do not need to care the meaning of what is Persistent Object is, since the following are enough to use secure storage feature.

TEE_CreatePersistentObject() - Creates a persistent object with initial attributes.

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

Parameters

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

Returns

0 if success else error occured.

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

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

Returns

0 if success, else error occured.

10.7.2.17 TEE_DigestDoFinal()

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

Function accumulates message data for hashing.

TEE_DigestDoFinal() - Finalizes the message digest operation and produces the message hash.

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

Parameters

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

10.7.2.18 TEE_DigestUpdate()

```
void TEE_DigestUpdate (
    TEE_OperationHandle operation,
    const void * chunk,
    uint32_t chunkSize )
```

Crypto, Message Digest Functions.

Function accumulates message data for hashing.

TEE_DigestUpdate()- Accumulates message data for hashing.

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

Parameters

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

10.7.2.19 TEE_FreeOperation()

```
void TEE_FreeOperation (
    TEE_OperationHandle operation )
```

Crypto, for all Crypto Functions.

All Crypto Functions use TEE_OperationHandle∗ operation instances.
Destroy Crypto instance.

TEE_FreeOperation() - Deallocates all resources associated with an operation handle.

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

Parameters

operation	Reference to operation handle.

Returns

nothing after the operation free.

10.7.2.20 TEE_FreeTransientObject()

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

Crypto, Asymmetric key Verification Functions.

Destroy object storing asymmetric key.

TEE_FreeTransientObject() - Deallocates a transient object previously allocated with TEE_AllocateTransientObject .

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

Parameters

object	Handle on the object to free.

10.7.2.21 TEE_GenerateKey()

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

Crypto, Asymmetric key Verification Functions.

Generate asymmetric keypair.

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

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

Parameters

object	Handle on an uninitialized transient key to populate with the generated key.
keySize	Requested key size shall be less than or equal to the maximum key size specified when the object container was created
params	Parameters for the key generation.
paramCount	The values of all parameters are copied nto the object so that the params array and all the memory buffers it points to may be freed after this routine returns without affecting the object.

Returns

0 on succes

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

10.7.2.22 TEE_GenerateRandom()

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

Crypto, common.

Random Data Generation Function. The quality of the random is implementation dependent.
I am not sure this should be in Keystone or not, but it is very handy.
Good to have adding a way to check the quality of the random implementation.

ocall_getrandom() - For getting random data.

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

Parameters

buf	character type buffer
len	size of the buffer
flags	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

randomBuffer	Reference to generated random data
randomBufferLen	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

randomBuffer	Reference to generated random data
randomBufferLen	Byte length of requested random data

10.7.2.23 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

objectInfo	Pointer to a structure filled with the object information
object	Handle of the object

Returns

0 if success else error occured.

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

objectInfo	Pointer to a structure filled with the object information
object	Handle of the object

Returns

0 if success else error occured.

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

Core Functions, Time Functions.

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

TEE_GetREETime() - Retrieves the current REE system time.

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

Parameters

time	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

time	Filled with the number of seconds and milliseconds.

10.7.2.25 TEE_GetSystemTime()

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

Core Functions, Time Functions.

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

TEE_GetSystemTime() - Retrieves the current system time.

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

Parameters

time	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

time	Filled with the number of seconds and milliseconds.

10.7.2.26 TEE_InitRefAttribute()

```
void TEE_InitRefAttribute (
    TEE_Attribute * attr,
    uint32_t attributeID,
```

```
const void * buffer,
uint32_t length )
```

Crypto, Asymmetric key Verification Functions.

Storing asymmetric key.

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

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

Parameters

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

10.7.2.27 TEE_InitValueAttribute() `void TEE_InitValueAttribute (`

```
TEE_Attribute * attr,
uint32_t attributeID,
uint32_t a,
uint32_t b )
```

Crypto, Asymmetric key Verification Functions.

Storing asymmetric key.

TEE_InitValueAttribute() - The helper function can be used to populate a single attribute either with a reference to a buffer or with integer values.

Parameters

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

10.7.2.28 TEE_OpenPersistentObject() `TEE_Result TEE_OpenPersistentObject (`

```
uint32_t storageID,
const void * objectID,
uint32_t objectIDLen,
```

```
uint32_t flags,
TEE_ObjectHandle * object )
```

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

Open persistent object.

TEE_OpenPersistentObject() - Opens a handle on an existing persistent object.

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

Parameters

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

Returns

0 if success else error occured.

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

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

Returns

0 if success, else error occured.

10.7.2.29 TEE_ReadObjectData() TEE_Result TEE_ReadObjectData (

```
TEE_ObjectHandle object,
void * buffer,
uint32_t size,
uint32_t * count )
```

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

Read object.

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

The bytes are read starting at the position in the data stream currently stored in the object handle. The handle's position is incremented by the number of bytes actually read. On completion of TEE_ReadObjectData sets the number of bytes actually read in the "uint32_t" pointed to by count. The value written to ∗count may be less than size if the number of bytes until the end-of3067 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

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

Returns

TEE_SUCCESS if success else error occured.

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-of3067 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

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

Returns

TEE_SUCCESS if success, else error occured.

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

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Set symmetric key used in operation.

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

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

Parameters

operation	Operation handle.
key	A handle on a key object.

Returns

0 on success return

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

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

10.7.2.31 TEE_WriteObjectData()

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

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

Write object.

TEE_WriteObjectData() - Writes the buffer data in to persistent objects.

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

Parameters

object	Handle of the object
buffer	The buffer containing the data to be written
size	The number of bytes to write

Returns

TEE_SUCCESS if success else error occured.

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

object	Handle of the object
buffer	The buffer containing the data to be written
size	The number of bytes to write

Returns

TEE_SUCCESS if success else error occured.

10.8 tee-ta-internal.h

Go to the documentation of this file.

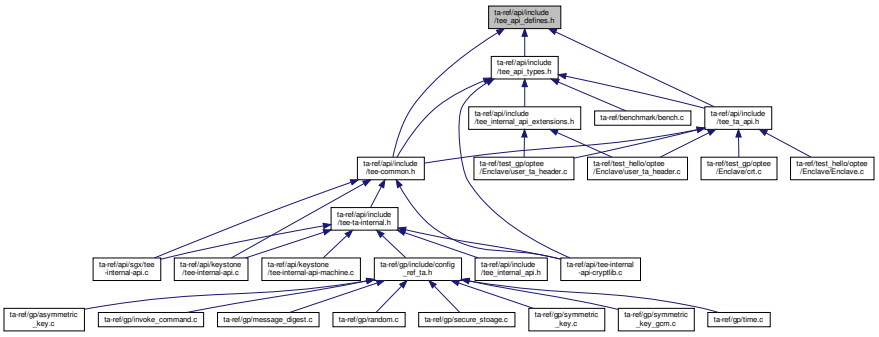
```
1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                          and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10  *
11  * 1. Redistributions of source code must retain the above copyright notice,
12  * this list of conditions and the following disclaimer.
13  *
14  * 2. Redistributions in binary form must reproduce the above copyright notice,
15  * this list of conditions and the following disclaimer in the documentation
16  * and/or other materials provided with the distribution.
17  *
18  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28  * POSSIBILITY OF SUCH DAMAGE.
29  */
37 #ifndef TA_INTERNAL_TEE_H
38 #define TA_INTERNAL_TEE_H
39
40 #include "tee-common.h"
41
42 #ifdef __cplusplus
43 extern "C" {
44 #endif
45
46 void __attribute__((noreturn)) TEE_Panic(unsigned long code);
47
49
53 void TEE_GetREETime(TEE_Time *time);
54
56
58 /* Wall clock time is important for verifying certificates. */
59 void TEE_GetSystemTime(TEE_Time *time);
60
62
69 /* Start timer */
70 TEE_Result GetRelTimeStart(uint64_t start);
71
73
76 TEE_Result GetRelTimeEnd(uint64_t end);
77
```

```
79
85 TEE_Result TEE_CreatePersistentObject(uint32_t storageID, const void *objectID,
86                                       uint32_t objectIDLen, uint32_t flags,
87                                       TEE_ObjectHandle attributes,
88                                       const void *initialData,
89                                       uint32_t initialDataLen,
90                                       TEE_ObjectHandle *object);
92
93 TEE_Result TEE_OpenPersistentObject(uint32_t storageID, const void *objectID,
94                                     uint32_t objectIDLen, uint32_t flags,
95                                     TEE_ObjectHandle *object);
97
98 TEE_Result TEE_GetObjectInfo1(TEE_ObjectHandle object, TEE_ObjectInfo *objectInfo);
100
101 TEE_Result TEE_WriteObjectData(TEE_ObjectHandle object, const void *buffer,
102                                uint32_t size);
105 TEE_Result TEE_ReadObjectData(TEE_ObjectHandle object, void *buffer,
106                               uint32_t size, uint32_t *count);
108
109 void TEE_CloseObject(TEE_ObjectHandle object);
110
111
113
119 void TEE_GenerateRandom(void *randomBuffer, uint32_t randomBufferLen);
120
122
124 TEE_Result TEE_AllocateOperation(TEE_OperationHandle *operation,
125                                  uint32_t algorithm, uint32_t mode,
126                                  uint32_t maxKeySize);
128
130 void TEE_FreeOperation(TEE_OperationHandle operation);
131
132
134
135 void TEE_DigestUpdate(TEE_OperationHandle operation,
136                       const void *chunk, uint32_t chunkSize);
138 TEE_Result TEE_DigestDoFinal(TEE_OperationHandle operation, const void *chunk,
139                              uint32_t chunkLen, void *hash, uint32_t *hashLen);
140
142
143 TEE_Result TEE_SetOperationKey(TEE_OperationHandle operation,
144                    TEE_ObjectHandle key);
146
147 TEE_Result TEE_AEInit(TEE_OperationHandle operation, const void *nonce,
148                       uint32_t nonceLen, uint32_t tagLen, uint32_t AADLen,
149                       uint32_t payloadLen);
151
152 TEE_Result TEE_AEUpdate(TEE_OperationHandle operation, const void *srcData,
153                         uint32_t srcLen, void *destData, uint32_t *destLen);
155
156 void TEE_AEUpdateAAD(TEE_OperationHandle operation, const void *AADdata,
157              uint32_t AADdataLen);
159
160 TEE_Result TEE_AEEncryptFinal(TEE_OperationHandle operation,
161                               const void *srcData, uint32_t srcLen,
162                               void *destData, uint32_t *destLen, void *tag,
163                               uint32_t *tagLen);
165
166 TEE_Result TEE_AEDecryptFinal(TEE_OperationHandle operation,
167                               const void *srcData, uint32_t srcLen,
168                               void *destData, uint32_t *destLen, void *tag,
169                               uint32_t tagLen);
170
172
173 void TEE_CipherInit(TEE_OperationHandle operation, const void *nonce,
174            uint32_t nonceLen);
176
177 TEE_Result TEE_CipherUpdate(TEE_OperationHandle operation, const void *srcData,
178                 uint32_t srcLen, void *destData, uint32_t *destLen);
179
181
182 TEE_Result TEE_GenerateKey(TEE_ObjectHandle object, uint32_t keySize,
183                const TEE_Attribute *params, uint32_t paramCount);
185
186 TEE_Result TEE_AllocateTransientObject(TEE_ObjectType objectType,
187                                        uint32_t maxKeySize,
188                                        TEE_ObjectHandle *object);
190
191 void TEE_InitRefAttribute(TEE_Attribute *attr, uint32_t attributeID,
192                           const void *buffer, uint32_t length);
194
195 void TEE_InitValueAttribute(TEE_Attribute *attr, uint32_t attributeID,
196                 uint32_t a, uint32_t b);
198
199 void TEE_FreeTransientObject(TEE_ObjectHandle object);
200
202
```

```
206 TEE_Result TEE_AsymmetricSignDigest(TEE_OperationHandle operation,
207                                     const TEE_Attribute *params,
208                                     uint32_t paramCount, const void *digest,
209                                     uint32_t digestLen, void *signature,
210                                     uint32_t *signatureLen);
212
216 TEE_Result TEE_AsymmetricVerifyDigest(TEE_OperationHandle operation,
217                                       const TEE_Attribute *params,
218                                       uint32_t paramCount, const void *digest,
219                                       uint32_t digestLen, const void *signature,
220                                       uint32_t signatureLen);
221
222 #ifdef __cplusplus
223 }
224 #endif
225
226 #endif /* TA_INTERNAL_TEE_H */
```

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

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



Macros

- #define TEE_INT_CORE_API_SPEC_VERSION 0x0000000A
- #define TEE_HANDLE_NULL 0
- #define TEE_TIMEOUT_INFINITE 0xFFFFFFFF
- #define TEE_SUCCESS 0x00000000
- #define TEE_ERROR_CORRUPT_OBJECT 0xF0100001
- #define TEE_ERROR_CORRUPT_OBJECT_2 0xF0100002
- #define TEE_ERROR_STORAGE_NOT_AVAILABLE 0xF0100003
- #define TEE_ERROR_STORAGE_NOT_AVAILABLE_2 0xF0100004
- #define TEE_ERROR_GENERIC 0xFFFF0000
- #define TEE_ERROR_ACCESS_DENIED 0xFFFF0001
- #define TEE_ERROR_CANCEL 0xFFFF0002
- #define TEE_ERROR_ACCESS_CONFLICT 0xFFFF0003
- #define TEE_ERROR_EXCESS_DATA 0xFFFF0004
- #define TEE_ERROR_BAD_FORMAT 0xFFFF0005
- #define TEE_ERROR_BAD_PARAMETERS 0xFFFF0006
- #define TEE_ERROR_BAD_STATE 0xFFFF0007
- #define TEE_ERROR_ITEM_NOT_FOUND 0xFFFF0008
- #define TEE_ERROR_NOT_IMPLEMENTED 0xFFFF0009
- #define TEE_ERROR_NOT_SUPPORTED 0xFFFF000A
- #define TEE_ERROR_NO_DATA 0xFFFF000B
- #define TEE_ERROR_OUT_OF_MEMORY 0xFFFF000C

- #define TEE_ERROR_BUSY 0xFFFF000D
- #define TEE_ERROR_COMMUNICATION 0xFFFF000E
- #define TEE_ERROR_SECURITY 0xFFFF000F
- #define TEE_ERROR_SHORT_BUFFER 0xFFFF0010
- #define TEE_ERROR_EXTERNAL_CANCEL 0xFFFF0011
- #define TEE_ERROR_OVERFLOW 0xFFFF300F
- #define TEE_ERROR_TARGET_DEAD 0xFFFF3024
- #define TEE_ERROR_STORAGE_NO_SPACE 0xFFFF3041
- #define TEE_ERROR_MAC_INVALID 0xFFFF3071
- #define TEE_ERROR_SIGNATURE_INVALID 0xFFFF3072
- #define TEE_ERROR_TIME_NOT_SET 0xFFFF5000
- #define TEE_ERROR_TIME_NEEDS_RESET 0xFFFF5001
- #define TEE_PARAM_TYPE_NONE 0
- #define TEE_PARAM_TYPE_VALUE_INPUT 1
- #define TEE_PARAM_TYPE_VALUE_OUTPUT 2
- #define TEE_PARAM_TYPE_VALUE_INOUT 3
- #define TEE_PARAM_TYPE_MEMREF_INPUT 5
- #define TEE_PARAM_TYPE_MEMREF_OUTPUT 6
- #define TEE_PARAM_TYPE_MEMREF_INOUT 7
- #define TEE_LOGIN_PUBLIC 0x00000000
- #define TEE_LOGIN_USER 0x00000001
- #define TEE_LOGIN_GROUP 0x00000002
- #define TEE_LOGIN_APPLICATION 0x00000004
- #define TEE_LOGIN_APPLICATION_USER 0x00000005
- #define TEE_LOGIN_APPLICATION_GROUP 0x00000006
- #define TEE_LOGIN_TRUSTED_APP 0xF0000000
- #define TEE_ORIGIN_API 0x00000001
- #define TEE_ORIGIN_COMMS 0x00000002
- #define TEE_ORIGIN_TEE 0x00000003
- #define TEE_ORIGIN_TRUSTED_APP 0x00000004
- #define TEE_PROPSET_TEE_IMPLEMENTATION (TEE_PropSetHandle)0xFFFFFFFD
- #define TEE_PROPSET_CURRENT_CLIENT (TEE_PropSetHandle)0xFFFFFFFE
- #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_AEDECRYPTFINAL 0x00001001
- #define TEE_PANIC_ID_TEE_AEENCRYPTFINAL 0x00001002
- #define TEE_PANIC_ID_TEE_AEINIT 0x00001003
- #define TEE_PANIC_ID_TEE_AEUPDATE 0x00001004
- #define TEE_PANIC_ID_TEE_AEUPDATEAAD 0x00001005
- #define TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT 0x00001101
- #define TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT 0x00001102
- #define TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST 0x00001103
- #define TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST 0x00001104
- #define TEE_PANIC_ID_TEE_DERIVEKEY 0x00001201
- #define TEE_PANIC_ID_TEE_GENERATERANDOM 0x00001301

- #define TEE_PANIC_ID_TEE_GETREETIME 0x00001401
- #define TEE_PANIC_ID_TEE_GETSYSTEMTIME 0x00001402
- #define TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME 0x00001403
- #define TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME 0x00001404
- #define TEE_PANIC_ID_TEE_WAIT 0x00001405
- #define TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32 0x00001501
- #define TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32 0x00001502
- #define TEE_PANIC_ID_TEE_BIGINTINIT 0x00001601
- #define TEE_PANIC_ID_TEE_BIGINTINITFMM 0x00001602
- #define TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT 0x00001603
- #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING 0x00001701
- #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32 0x00001702
- #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING 0x00001703
- #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32 0x00001704
- #define TEE_PANIC_ID_TEE_BIGINTCMP 0x00001801
- #define TEE_PANIC_ID_TEE_BIGINTCMPS32 0x00001802
- #define TEE_PANIC_ID_TEE_BIGINTGETBIT 0x00001803
- #define TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT 0x00001804
- #define TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT 0x00001805
- #define TEE_PANIC_ID_TEE_BIGINTADD 0x00001901
- #define TEE_PANIC_ID_TEE_BIGINTDIV 0x00001902
- #define TEE_PANIC_ID_TEE_BIGINTMUL 0x00001903
- #define TEE_PANIC_ID_TEE_BIGINTNEG 0x00001904
- #define TEE_PANIC_ID_TEE_BIGINTSQUARE 0x00001905
- #define TEE_PANIC_ID_TEE_BIGINTSUB 0x00001906
- #define TEE_PANIC_ID_TEE_BIGINTADDMOD 0x00001A01
- #define TEE_PANIC_ID_TEE_BIGINTINVMOD 0x00001A02
- #define TEE_PANIC_ID_TEE_BIGINTMOD 0x00001A03
- #define TEE_PANIC_ID_TEE_BIGINTMULMOD 0x00001A04
- #define TEE_PANIC_ID_TEE_BIGINTSQUAREMOD 0x00001A05
- #define TEE_PANIC_ID_TEE_BIGINTSUBMOD 0x00001A06
- #define TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD 0x00001B01
- #define TEE_PANIC_ID_TEE_BIGINTISPROBABLEPRIME 0x00001B02
- #define TEE_PANIC_ID_TEE_BIGINTRELATIVEPRIME 0x00001B03
- #define TEE_PANIC_ID_TEE_BIGINTCOMPUTEFMM 0x00001C01
- #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMFMM 0x00001C02
- #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOFMM 0x00001C03
- #define TEE_PARAM_TYPES(t0, t1, t2, t3) ((t0) | ((t1) << 4) | ((t2) << 8) | ((t3) << 12))
- #define TEE_PARAM_TYPE_GET(t, i) ((((uint32_t)t) >> ((i)*4)) & 0xF)
- #define TEE_PARAM_TYPE_SET(t, i) (((uint32_t)(t) & 0xF) << ((i)*4))
- #define TEE_NUM_PARAMS 4
- #define TEE_BigIntSizeInU32(n) ((((n)+31)/32)+2)

10.9.1 Macro Definition Documentation

10.9.1.1 TEE_ALG_AES_CBC_MAC_NOPAD `#define TEE_ALG_AES_CBC_MAC_NOPAD 0x30000110`

10.9.1.2 TEE_ALG_AES_CBC_MAC_PKCS5 #define TEE_ALG_AES_CBC_MAC_PKCS5 0x30000510

10.9.1.3 TEE_ALG_AES_CBC_NOPAD #define TEE_ALG_AES_CBC_NOPAD 0x10000110

10.9.1.4 TEE_ALG_AES_CCM #define TEE_ALG_AES_CCM 0x40000710

10.9.1.5 TEE_ALG_AES_CMAC #define TEE_ALG_AES_CMAC 0x30000610

10.9.1.6 TEE_ALG_AES_CTR #define TEE_ALG_AES_CTR 0x10000210

10.9.1.7 TEE_ALG_AES_CTS #define TEE_ALG_AES_CTS 0x10000310

10.9.1.8 TEE_ALG_AES_ECB_NOPAD #define TEE_ALG_AES_ECB_NOPAD 0x10000010

10.9.1.9 TEE_ALG_AES_GCM #define TEE_ALG_AES_GCM 0x40000810

10.9.1.10 TEE_ALG_AES_XTS #define TEE_ALG_AES_XTS 0x10000410

10.9.1.11 TEE_ALG_DES3_CBC_MAC_NOPAD #define TEE_ALG_DES3_CBC_MAC_NOPAD 0x30000113

10.9.1.12 TEE_ALG_DES3_CBC_MAC_PKCS5 #define TEE_ALG_DES3_CBC_MAC_PKCS5 0x30000513

10.9.1.13 TEE_ALG_DES3_CBC_NOPAD

```
#define TEE_ALG_DES3_CBC_NOPAD 0x10000113
```

10.9.1.14 TEE_ALG_DES3_ECB_NOPAD

```
#define TEE_ALG_DES3_ECB_NOPAD 0x10000013
```

10.9.1.15 TEE_ALG_DES_CBC_MAC_NOPAD

```
#define TEE_ALG_DES_CBC_MAC_NOPAD 0x30000111
```

10.9.1.16 TEE_ALG_DES_CBC_MAC_PKCS5

```
#define TEE_ALG_DES_CBC_MAC_PKCS5 0x30000511
```

10.9.1.17 TEE_ALG_DES_CBC_NOPAD

```
#define TEE_ALG_DES_CBC_NOPAD 0x10000111
```

10.9.1.18 TEE_ALG_DES_ECB_NOPAD

```
#define TEE_ALG_DES_ECB_NOPAD 0x10000011
```

10.9.1.19 TEE_ALG_DH_DERIVE_SHARED_SECRET

```
#define TEE_ALG_DH_DERIVE_SHARED_SECRET 0x80000032
```

10.9.1.20 TEE_ALG_DSA_SHA1

```
#define TEE_ALG_DSA_SHA1 0x70002131
```

10.9.1.21 TEE_ALG_DSA_SHA224

```
#define TEE_ALG_DSA_SHA224 0x70003131
```

10.9.1.22 TEE_ALG_DSA_SHA256

```
#define TEE_ALG_DSA_SHA256 0x70004131
```

10.9.1.23 TEE_ALG_ECDH_P192

```
#define TEE_ALG_ECDH_P192 0x80001042
```

10.9.1.24 TEE_ALG_ECDH_P224

```
#define TEE_ALG_ECDH_P224 0x80002042
```

10.9.1.25 TEE_ALG_ECDH_P256

```
#define TEE_ALG_ECDH_P256 0x80003042
```

10.9.1.26 TEE_ALG_ECDH_P384

```
#define TEE_ALG_ECDH_P384 0x80004042
```

10.9.1.27 TEE_ALG_ECDH_P521

```
#define TEE_ALG_ECDH_P521 0x80005042
```

10.9.1.28 TEE_ALG_ECDSA_P192

```
#define TEE_ALG_ECDSA_P192 0x70001041
```

10.9.1.29 TEE_ALG_ECDSA_P224

```
#define TEE_ALG_ECDSA_P224 0x70002041
```

10.9.1.30 TEE_ALG_ECDSA_P256

```
#define TEE_ALG_ECDSA_P256 0x70003041
```

10.9.1.31 TEE_ALG_ECDSA_P384

```
#define TEE_ALG_ECDSA_P384 0x70004041
```

10.9.1.32 TEE_ALG_ECDSA_P521

```
#define TEE_ALG_ECDSA_P521 0x70005041
```

10.9.1.33 TEE_ALG_HMAC_MD5

```
#define TEE_ALG_HMAC_MD5 0x30000001
```

10.9.1.34 TEE_ALG_HMAC_SHA1

```
#define TEE_ALG_HMAC_SHA1 0x30000002
```

10.9.1.35 TEE_ALG_HMAC_SHA224 #define TEE_ALG_HMAC_SHA224 0x30000003

10.9.1.36 TEE_ALG_HMAC_SHA256 #define TEE_ALG_HMAC_SHA256 0x30000004

10.9.1.37 TEE_ALG_HMAC_SHA384 #define TEE_ALG_HMAC_SHA384 0x30000005

10.9.1.38 TEE_ALG_HMAC_SHA512 #define TEE_ALG_HMAC_SHA512 0x30000006

10.9.1.39 TEE_ALG_MD5 #define TEE_ALG_MD5 0x50000001

10.9.1.40 TEE_ALG_MD5SHA1 #define TEE_ALG_MD5SHA1 0x5000000F

10.9.1.41 TEE_ALG_RSA_NOPAD #define TEE_ALG_RSA_NOPAD 0x60000030

10.9.1.42 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1 0x60210230

10.9.1.43 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_↩
SHA224 0x60310230

10.9.1.44 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_↩
SHA256 0x60410230

10.9.1.45 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_↩
SHA384 0x60510230

10.9.1.46 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512

```
#define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512 0x60610230
```

10.9.1.47 TEE_ALG_RSAES_PKCS1_V1_5

```
#define TEE_ALG_RSAES_PKCS1_V1_5 0x60000130
```

10.9.1.48 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1

```
#define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1 0x70212930
```

10.9.1.49 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224

```
#define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224 0x70313930
```

10.9.1.50 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256

```
#define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256 0x70414930
```

10.9.1.51 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384

```
#define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384 0x70515930
```

10.9.1.52 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512

```
#define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512 0x70616930
```

10.9.1.53 TEE_ALG_RSASSA_PKCS1_V1_5_MD5

```
#define TEE_ALG_RSASSA_PKCS1_V1_5_MD5 0x70001830
```

10.9.1.54 TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1

```
#define TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1 0x7000F830
```

10.9.1.55 TEE_ALG_RSASSA_PKCS1_V1_5_SHA1

```
#define TEE_ALG_RSASSA_PKCS1_V1_5_SHA1 0x70002830
```

10.9.1.56 TEE_ALG_RSASSA_PKCS1_V1_5_SHA224 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA224 0x70003830

10.9.1.57 TEE_ALG_RSASSA_PKCS1_V1_5_SHA256 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA256 0x70004830

10.9.1.58 TEE_ALG_RSASSA_PKCS1_V1_5_SHA384 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA384 0x70005830

10.9.1.59 TEE_ALG_RSASSA_PKCS1_V1_5_SHA512 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA512 0x70006830

10.9.1.60 TEE_ALG_SHA1 #define TEE_ALG_SHA1 0x50000002

10.9.1.61 TEE_ALG_SHA224 #define TEE_ALG_SHA224 0x50000003

10.9.1.62 TEE_ALG_SHA256 #define TEE_ALG_SHA256 0x50000004

10.9.1.63 TEE_ALG_SHA384 #define TEE_ALG_SHA384 0x50000005

10.9.1.64 TEE_ALG_SHA512 #define TEE_ALG_SHA512 0x50000006

10.9.1.65 TEE_ATTR_BIT_PROTECTED #define TEE_ATTR_BIT_PROTECTED (1 << 28)

10.9.1.66 TEE_ATTR_BIT_VALUE #define TEE_ATTR_BIT_VALUE (1 << 29)

10.9.1.67 TEE_ATTR_DH_BASE

```
#define TEE_ATTR_DH_BASE 0xD0001232
```

10.9.1.68 TEE_ATTR_DH_PRIME

```
#define TEE_ATTR_DH_PRIME 0xD0001032
```

10.9.1.69 TEE_ATTR_DH_PRIVATE_VALUE

```
#define TEE_ATTR_DH_PRIVATE_VALUE 0xC0000232
```

10.9.1.70 TEE_ATTR_DH_PUBLIC_VALUE

```
#define TEE_ATTR_DH_PUBLIC_VALUE 0xD0000132
```

10.9.1.71 TEE_ATTR_DH_SUBPRIME

```
#define TEE_ATTR_DH_SUBPRIME 0xD0001132
```

10.9.1.72 TEE_ATTR_DH_X_BITS

```
#define TEE_ATTR_DH_X_BITS 0xF0001332
```

10.9.1.73 TEE_ATTR_DSA_BASE

```
#define TEE_ATTR_DSA_BASE 0xD0001231
```

10.9.1.74 TEE_ATTR_DSA_PRIME

```
#define TEE_ATTR_DSA_PRIME 0xD0001031
```

10.9.1.75 TEE_ATTR_DSA_PRIVATE_VALUE

```
#define TEE_ATTR_DSA_PRIVATE_VALUE 0xC0000231
```

10.9.1.76 TEE_ATTR_DSA_PUBLIC_VALUE

```
#define TEE_ATTR_DSA_PUBLIC_VALUE 0xD0000131
```

10.9.1.77 TEE_ATTR_DSA_SUBPRIME

```
#define TEE_ATTR_DSA_SUBPRIME 0xD0001131
```

10.9.1.78 TEE_ATTR_ECC_CURVE

```
#define TEE_ATTR_ECC_CURVE 0xF0000441
```

10.9.1.79 TEE_ATTR_ECC_PRIVATE_VALUE

```
#define TEE_ATTR_ECC_PRIVATE_VALUE 0xC0000341
```

10.9.1.80 TEE_ATTR_ECC_PUBLIC_VALUE_X

```
#define TEE_ATTR_ECC_PUBLIC_VALUE_X 0xD0000141
```

10.9.1.81 TEE_ATTR_ECC_PUBLIC_VALUE_Y

```
#define TEE_ATTR_ECC_PUBLIC_VALUE_Y 0xD0000241
```

10.9.1.82 TEE_ATTR_RSA_COEFFICIENT

```
#define TEE_ATTR_RSA_COEFFICIENT 0xC0000830
```

10.9.1.83 TEE_ATTR_RSA_EXPONENT1

```
#define TEE_ATTR_RSA_EXPONENT1 0xC0000630
```

10.9.1.84 TEE_ATTR_RSA_EXPONENT2

```
#define TEE_ATTR_RSA_EXPONENT2 0xC0000730
```

10.9.1.85 TEE_ATTR_RSA_MODULUS

```
#define TEE_ATTR_RSA_MODULUS 0xD0000130
```

10.9.1.86 TEE_ATTR_RSA_OAEP_LABEL

```
#define TEE_ATTR_RSA_OAEP_LABEL 0xD0000930
```

10.9.1.87 TEE_ATTR_RSA_PRIME1

```
#define TEE_ATTR_RSA_PRIME1 0xC0000430
```

10.9.1.88 TEE_ATTR_RSA_PRIME2

```
#define TEE_ATTR_RSA_PRIME2 0xC0000530
```

10.9.1.89 TEE_ATTR_RSA_PRIVATE_EXPONENT

```
#define TEE_ATTR_RSA_PRIVATE_EXPONENT 0xC0000330
```

10.9.1.90 TEE_ATTR_RSA_PSS_SALT_LENGTH

```
#define TEE_ATTR_RSA_PSS_SALT_LENGTH 0xF0000A30
```

10.9.1.91 TEE_ATTR_RSA_PUBLIC_EXPONENT

```
#define TEE_ATTR_RSA_PUBLIC_EXPONENT 0xD0000230
```

10.9.1.92 TEE_ATTR_SECRET_VALUE

```
#define TEE_ATTR_SECRET_VALUE 0xC0000000
```

10.9.1.93 TEE_BigIntSizeInU32

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

10.9.1.94 TEE_DATA_FLAG_ACCESS_READ

```
#define TEE_DATA_FLAG_ACCESS_READ 0x00000001
```

10.9.1.95 TEE_DATA_FLAG_ACCESS_WRITE

```
#define TEE_DATA_FLAG_ACCESS_WRITE 0x00000002
```

10.9.1.96 TEE_DATA_FLAG_ACCESS_WRITE_META

```
#define TEE_DATA_FLAG_ACCESS_WRITE_META 0x00000004
```

10.9.1.97 TEE_DATA_FLAG_OVERWRITE

```
#define TEE_DATA_FLAG_OVERWRITE 0x00000400
```

10.9.1.98 TEE_DATA_FLAG_SHARE_READ

```
#define TEE_DATA_FLAG_SHARE_READ 0x00000010
```

10.9.1.99 TEE_DATA_FLAG_SHARE_WRITE

```
#define TEE_DATA_FLAG_SHARE_WRITE 0x00000020
```

10.9.1.100 TEE_DATA_MAX_POSITION

```
#define TEE_DATA_MAX_POSITION 0xFFFFFFFF
```

10.9.1.101 TEE_ECC_CURVE_NIST_P192

```
#define TEE_ECC_CURVE_NIST_P192 0x00000001
```

10.9.1.102 TEE_ECC_CURVE_NIST_P224

```
#define TEE_ECC_CURVE_NIST_P224 0x00000002
```

10.9.1.103 TEE_ECC_CURVE_NIST_P256

```
#define TEE_ECC_CURVE_NIST_P256 0x00000003
```

10.9.1.104 TEE_ECC_CURVE_NIST_P384

```
#define TEE_ECC_CURVE_NIST_P384 0x00000004
```

10.9.1.105 TEE_ECC_CURVE_NIST_P521

```
#define TEE_ECC_CURVE_NIST_P521 0x00000005
```

10.9.1.106 TEE_ERROR_ACCESS_CONFLICT

```
#define TEE_ERROR_ACCESS_CONFLICT 0xFFFF0003
```

10.9.1.107 TEE_ERROR_ACCESS_DENIED

```
#define TEE_ERROR_ACCESS_DENIED 0xFFFF0001
```

10.9.1.108 TEE_ERROR_BAD_FORMAT

```
#define TEE_ERROR_BAD_FORMAT 0xFFFF0005
```

10.9.1.109 TEE_ERROR_BAD_PARAMETERS

```
#define TEE_ERROR_BAD_PARAMETERS 0xFFFF0006
```

10.9.1.110 TEE_ERROR_BAD_STATE

```
#define TEE_ERROR_BAD_STATE 0xFFFF0007
```

10.9.1.111 TEE_ERROR_BUSY

```
#define TEE_ERROR_BUSY 0xFFFF000D
```

10.9.1.112 TEE_ERROR_CANCEL

```
#define TEE_ERROR_CANCEL 0xFFFF0002
```

10.9.1.113 TEE_ERROR_COMMUNICATION

```
#define TEE_ERROR_COMMUNICATION 0xFFFF000E
```

10.9.1.114 TEE_ERROR_CORRUPT_OBJECT

```
#define TEE_ERROR_CORRUPT_OBJECT 0xF0100001
```

10.9.1.115 TEE_ERROR_CORRUPT_OBJECT_2

```
#define TEE_ERROR_CORRUPT_OBJECT_2 0xF0100002
```

10.9.1.116 TEE_ERROR_EXCESS_DATA

```
#define TEE_ERROR_EXCESS_DATA 0xFFFF0004
```

10.9.1.117 TEE_ERROR_EXTERNAL_CANCEL

```
#define TEE_ERROR_EXTERNAL_CANCEL 0xFFFF0011
```

10.9.1.118 TEE_ERROR_GENERIC

```
#define TEE_ERROR_GENERIC 0xFFFF0000
```

10.9.1.119 TEE_ERROR_ITEM_NOT_FOUND

```
#define TEE_ERROR_ITEM_NOT_FOUND 0xFFFF0008
```

10.9.1.120 TEE_ERROR_MAC_INVALID

```
#define TEE_ERROR_MAC_INVALID 0xFFFF3071
```

10.9.1.121 TEE_ERROR_NO_DATA

```
#define TEE_ERROR_NO_DATA 0xFFFF000B
```

10.9.1.122 TEE_ERROR_NOT_IMPLEMENTED

```
#define TEE_ERROR_NOT_IMPLEMENTED 0xFFFF0009
```

10.9.1.123 TEE_ERROR_NOT_SUPPORTED

```
#define TEE_ERROR_NOT_SUPPORTED 0xFFFF000A
```

10.9.1.124 TEE_ERROR_OUT_OF_MEMORY

```
#define TEE_ERROR_OUT_OF_MEMORY 0xFFFF000C
```

10.9.1.125 TEE_ERROR_OVERFLOW

```
#define TEE_ERROR_OVERFLOW 0xFFFF300F
```

10.9.1.126 TEE_ERROR_SECURITY

```
#define TEE_ERROR_SECURITY 0xFFFF000F
```

10.9.1.127 TEE_ERROR_SHORT_BUFFER

```
#define TEE_ERROR_SHORT_BUFFER 0xFFFF0010
```

10.9.1.128 TEE_ERROR_SIGNATURE_INVALID

```
#define TEE_ERROR_SIGNATURE_INVALID 0xFFFF3072
```

10.9.1.129 TEE_ERROR_STORAGE_NO_SPACE

```
#define TEE_ERROR_STORAGE_NO_SPACE 0xFFFF3041
```

10.9.1.130 TEE_ERROR_STORAGE_NOT_AVAILABLE

```
#define TEE_ERROR_STORAGE_NOT_AVAILABLE 0x↩
F0100003
```

10.9.1.131 TEE_ERROR_STORAGE_NOT_AVAILABLE_2

```
#define TEE_ERROR_STORAGE_NOT_AVAILABLE_2 0x↩
F0100004
```

10.9.1.132 TEE_ERROR_TARGET_DEAD

```
#define TEE_ERROR_TARGET_DEAD 0xFFFF3024
```

10.9.1.133 TEE_ERROR_TIME_NEEDS_RESET

```
#define TEE_ERROR_TIME_NEEDS_RESET 0xFFFF5001
```

10.9.1.134 TEE_ERROR_TIME_NOT_SET

```
#define TEE_ERROR_TIME_NOT_SET 0xFFFF5000
```

10.9.1.135 TEE_HANDLE_FLAG_EXPECT_TWO_KEYS

```
#define TEE_HANDLE_FLAG_EXPECT_TWO_KEYS 0x00080000
```

10.9.1.136 TEE_HANDLE_FLAG_INITIALIZED

```
#define TEE_HANDLE_FLAG_INITIALIZED 0x00020000
```

10.9.1.137 TEE_HANDLE_FLAG_KEY_SET

```
#define TEE_HANDLE_FLAG_KEY_SET 0x00040000
```

10.9.1.138 TEE_HANDLE_FLAG_PERSISTENT

```
#define TEE_HANDLE_FLAG_PERSISTENT 0x00010000
```

10.9.1.139 TEE_HANDLE_NULL

```
#define TEE_HANDLE_NULL 0
```

10.9.1.140 TEE_INT_CORE_API_SPEC_VERSION

```
#define TEE_INT_CORE_API_SPEC_VERSION 0x0000000A
```

10.9.1.141 TEE_LOGIN_APPLICATION

```
#define TEE_LOGIN_APPLICATION 0x00000004
```

10.9.1.142 TEE_LOGIN_APPLICATION_GROUP

```
#define TEE_LOGIN_APPLICATION_GROUP 0x00000006
```

10.9.1.143 TEE_LOGIN_APPLICATION_USER

```
#define TEE_LOGIN_APPLICATION_USER 0x00000005
```

10.9.1.144 TEE_LOGIN_GROUP

```
#define TEE_LOGIN_GROUP 0x00000002
```

10.9.1.145 TEE_LOGIN_PUBLIC

```
#define TEE_LOGIN_PUBLIC 0x00000000
```

10.9.1.146 TEE_LOGIN_TRUSTED_APP

```
#define TEE_LOGIN_TRUSTED_APP 0xF0000000
```

10.9.1.147 TEE_LOGIN_USER

```
#define TEE_LOGIN_USER 0x00000001
```

10.9.1.148 TEE_MALLOC_FILL_ZERO

```
#define TEE_MALLOC_FILL_ZERO 0x00000000
```

10.9.1.149 TEE_MEMORY_ACCESS_ANY_OWNER

```
#define TEE_MEMORY_ACCESS_ANY_OWNER 0x00000004
```

10.9.1.150 TEE_MEMORY_ACCESS_READ

```
#define TEE_MEMORY_ACCESS_READ 0x00000001
```

10.9.1.151 TEE_MEMORY_ACCESS_WRITE

```
#define TEE_MEMORY_ACCESS_WRITE 0x00000002
```

10.9.1.152 TEE_NUM_PARAMS

```
#define TEE_NUM_PARAMS 4
```

10.9.1.153 TEE_OBJECT_ID_MAX_LEN

```
#define TEE_OBJECT_ID_MAX_LEN 64
```

10.9.1.154 TEE_OPERATION_AE

```
#define TEE_OPERATION_AE 4
```

10.9.1.155 TEE_OPERATION_ASYMMETRIC_CIPHER

```
#define TEE_OPERATION_ASYMMETRIC_CIPHER 6
```

10.9.1.156 TEE_OPERATION_ASYMMETRIC_SIGNATURE

```
#define TEE_OPERATION_ASYMMETRIC_SIGNATURE 7
```

10.9.1.157 TEE_OPERATION_CIPHER

```
#define TEE_OPERATION_CIPHER 1
```

10.9.1.158 TEE_OPERATION_DIGEST

```
#define TEE_OPERATION_DIGEST 5
```

10.9.1.159 TEE_OPERATION_KEY_DERIVATION

```
#define TEE_OPERATION_KEY_DERIVATION 8
```

10.9.1.160 TEE_OPERATION_MAC

```
#define TEE_OPERATION_MAC 3
```

10.9.1.161 TEE_OPERATION_STATE_ACTIVE

```
#define TEE_OPERATION_STATE_ACTIVE 0x00000001
```

10.9.1.162 TEE_OPERATION_STATE_INITIAL

```
#define TEE_OPERATION_STATE_INITIAL 0x00000000
```

10.9.1.163 TEE_ORIGIN_API

```
#define TEE_ORIGIN_API 0x00000001
```

10.9.1.164 TEE_ORIGIN_COMMS

```
#define TEE_ORIGIN_COMMS 0x00000002
```

10.9.1.165 TEE_ORIGIN_TEE

```
#define TEE_ORIGIN_TEE 0x00000003
```

10.9.1.166 TEE_ORIGIN_TRUSTED_APP

```
#define TEE_ORIGIN_TRUSTED_APP 0x00000004
```

10.9.1.167 TEE_PANIC_ID_TA_CLOSESESSIONENTRYPOINT

```
#define TEE_PANIC_ID_TA_CLOSESESSIONENTRYPOINT 0x0000010
```

10.9.1.168 TEE_PANIC_ID_TA_CREATEENTRYPOINT

```
#define TEE_PANIC_ID_TA_CREATEENTRYPOINT 0x00000102
```

10.9.1.169 TEE_PANIC_ID_TA_DESTROYENTRYPOINT

```
#define TEE_PANIC_ID_TA_DESTROYENTRYPOINT 0x00000103
```

10.9.1.170 TEE_PANIC_ID_TA_INVOKECOMMANDENTRYPOINT

```
#define TEE_PANIC_ID_TA_INVOKECOMMANDENTRYPOINT 0x0000
```

10.9.1.171 TEE_PANIC_ID_TA_OPENSESSIONENTRYPOINT

```
#define TEE_PANIC_ID_TA_OPENSESSIONENTRYPOINT 0x00000105
```

10.9.1.172 TEE_PANIC_ID_TEE_AEDECRYPTFINAL

```
#define TEE_PANIC_ID_TEE_AEDECRYPTFINAL 0x00001001
```

10.9.1.173 TEE_PANIC_ID_TEE_AEENCRYPTFINAL

```
#define TEE_PANIC_ID_TEE_AEENCRYPTFINAL 0x00001002
```

10.9.1.174 TEE_PANIC_ID_TEE_AEINIT

```
#define TEE_PANIC_ID_TEE_AEINIT 0x00001003
```

10.9.1.175 TEE_PANIC_ID_TEE_AEUPDATE

```
#define TEE_PANIC_ID_TEE_AEUPDATE 0x00001004
```

10.9.1.176 TEE_PANIC_ID_TEE_AEUPDATEAAD

```
#define TEE_PANIC_ID_TEE_AEUPDATEAAD 0x00001005
```

10.9.1.177 TEE_PANIC_ID_TEE_ALLOCATEOPERATION

```
#define TEE_PANIC_ID_TEE_ALLOCATEOPERATION 0x00000↩
C01
```

10.9.1.178 TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR

```
#define TEE_PANIC_ID_↩
TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR 0x00000A01
```

10.9.1.179 TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERATOR

```
#define TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERA
```

10.9.1.180 TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT

```
#define TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT 0x00
```

10.9.1.181 TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT

```
#define TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT 0x00001101
```

10.9.1.182 TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT

```
#define TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT 0x00001102
```

10.9.1.183 TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST

```
#define TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST 0x00001103
```

10.9.1.184 TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST

```
#define TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST 0x00001
```

10.9.1.185 TEE_PANIC_ID_TEE_BIGINTADD

```
#define TEE_PANIC_ID_TEE_BIGINTADD 0x00001901
```

10.9.1.186 TEE_PANIC_ID_TEE_BIGINTADDMOD

```
#define TEE_PANIC_ID_TEE_BIGINTADDMOD 0x00001A01
```

10.9.1.187 TEE_PANIC_ID_TEE_BIGINTCMP

```
#define TEE_PANIC_ID_TEE_BIGINTCMP 0x00001801
```

10.9.1.188 TEE_PANIC_ID_TEE_BIGINTCMPS32 #define TEE_PANIC_ID_TEE_BIGINTCMPS32 0x00001802

10.9.1.189 TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD #define TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD 0x
B01

10.9.1.190 TEE_PANIC_ID_TEE_BIGINTCOMPUTEFMM #define TEE_PANIC_ID_TEE_BIGINTCOMPUTEFMM 0x00001↩
C01

10.9.1.191 TEE_PANIC_ID_TEE_BIGINTCONVERTFROMFMM #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMFMM 0x00001↩
C02

10.9.1.192 TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING #define TEE_PANIC_ID_TEE_↩
BIGINTCONVERTFROMOCTETSTRING 0x00001701

10.9.1.193 TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32 #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32 0x00001702

10.9.1.194 TEE_PANIC_ID_TEE_BIGINTCONVERTTOFMM #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOFMM 0x00001↩
C03

10.9.1.195 TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRIN

10.9.1.196 TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32 #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32 0x00001704

10.9.1.197 TEE_PANIC_ID_TEE_BIGINTDIV #define TEE_PANIC_ID_TEE_BIGINTDIV 0x00001902

10.9.1.198 TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32

```
#define TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32 0x
```

10.9.1.199 TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32

```
#define TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32 0x00001502
```

10.9.1.200 TEE_PANIC_ID_TEE_BIGINTGETBIT

```
#define TEE_PANIC_ID_TEE_BIGINTGETBIT 0x00001803
```

10.9.1.201 TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT

```
#define TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT 0x00001804
```

10.9.1.202 TEE_PANIC_ID_TEE_BIGINTINIT

```
#define TEE_PANIC_ID_TEE_BIGINTINIT 0x00001601
```

10.9.1.203 TEE_PANIC_ID_TEE_BIGINTINITFMM

```
#define TEE_PANIC_ID_TEE_BIGINTINITFMM 0x00001602
```

10.9.1.204 TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT

```
#define TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT 0x00001603
```

10.9.1.205 TEE_PANIC_ID_TEE_BIGINTINVMOD

```
#define TEE_PANIC_ID_TEE_BIGINTINVMOD 0x00001A02
```

10.9.1.206 TEE_PANIC_ID_TEE_BIGINTISPROBABLEPRIME

```
#define TEE_PANIC_ID_TEE_BIGINTISPROBABLEPRIME 0x00001↩
B02
```

10.9.1.207 TEE_PANIC_ID_TEE_BIGINTMOD

```
#define TEE_PANIC_ID_TEE_BIGINTMOD 0x00001A03
```

10.9.1.208 TEE_PANIC_ID_TEE_BIGINTMUL

```
#define TEE_PANIC_ID_TEE_BIGINTMUL 0x00001903
```

10.9.1.209 TEE_PANIC_ID_TEE_BIGINTMULMOD

```
#define TEE_PANIC_ID_TEE_BIGINTMULMOD 0x00001A04
```

10.9.1.210 TEE_PANIC_ID_TEE_BIGINTNEG

```
#define TEE_PANIC_ID_TEE_BIGINTNEG 0x00001904
```

10.9.1.211 TEE_PANIC_ID_TEE_BIGINTRELATIVEPRIME

```
#define TEE_PANIC_ID_TEE_BIGINTRELATIVEPRIME 0x00001↩
B03
```

10.9.1.212 TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT

```
#define TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT 0x00001805
```

10.9.1.213 TEE_PANIC_ID_TEE_BIGINTSQUARE

```
#define TEE_PANIC_ID_TEE_BIGINTSQUARE 0x00001905
```

10.9.1.214 TEE_PANIC_ID_TEE_BIGINTSQUAREMOD

```
#define TEE_PANIC_ID_TEE_BIGINTSQUAREMOD 0x00001↩
A05
```

10.9.1.215 TEE_PANIC_ID_TEE_BIGINTSUB

```
#define TEE_PANIC_ID_TEE_BIGINTSUB 0x00001906
```

10.9.1.216 TEE_PANIC_ID_TEE_BIGINTSUBMOD

```
#define TEE_PANIC_ID_TEE_BIGINTSUBMOD 0x00001A06
```

10.9.1.217 TEE_PANIC_ID_TEE_CHECKMEMORYACCESSRIGHTS

```
#define TEE_PANIC_ID_TEE_CHECKMEMORYACCESSRIGHTS 0x0
```

10.9.1.218 TEE_PANIC_ID_TEE_CIPHERDOFINAL

```
#define TEE_PANIC_ID_TEE_CIPHERDOFINAL 0x00000E01
```

10.9.1.219 TEE_PANIC_ID_TEE_CIPHERINIT

```
#define TEE_PANIC_ID_TEE_CIPHERINIT 0x00000E02
```

10.9.1.220 TEE_PANIC_ID_TEE_CIPHERUPDATE

```
#define TEE_PANIC_ID_TEE_CIPHERUPDATE 0x00000E03
```

10.9.1.221 TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT

```
#define TEE_PANIC_ID_TEE_↩
CLOSEANDDELETEPERSISTENTOBJECT 0x00000901
```

10.9.1.222 TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT1

```
#define TEE_PANIC_ID_TEE_↩
CLOSEANDDELETEPERSISTENTOBJECT1 0x00000905
```

10.9.1.223 TEE_PANIC_ID_TEE_CLOSEOBJECT

```
#define TEE_PANIC_ID_TEE_CLOSEOBJECT 0x00000701
```

10.9.1.224 TEE_PANIC_ID_TEE_CLOSETASESSION

```
#define TEE_PANIC_ID_TEE_CLOSETASESSION 0x00000401
```

10.9.1.225 TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES

```
#define TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES 0x00000802
```

10.9.1.226 TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES1

```
#define TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES1 0x0000080
```

10.9.1.227 TEE_PANIC_ID_TEE_COPYOPERATION

```
#define TEE_PANIC_ID_TEE_COPYOPERATION 0x00000C02
```

10.9.1.228 TEE_PANIC_ID_TEE_CREATEPERSISTENTOBJECT

```
#define TEE_PANIC_ID_TEE_CREATEPERSISTENTOBJECT 0x00000
```

10.9.1.229 TEE_PANIC_ID_TEE_DERIVEKEY

```
#define TEE_PANIC_ID_TEE_DERIVEKEY 0x00001201
```

10.9.1.230 TEE_PANIC_ID_TEE_DIGESTDOFINAL

```
#define TEE_PANIC_ID_TEE_DIGESTDOFINAL 0x00000D01
```

10.9.1.231 TEE_PANIC_ID_TEE_DIGESTUPDATE

```
#define TEE_PANIC_ID_TEE_DIGESTUPDATE 0x00000D02
```

10.9.1.232 TEE_PANIC_ID_TEE_FREE

```
#define TEE_PANIC_ID_TEE_FREE 0x00000602
```

10.9.1.233 TEE_PANIC_ID_TEE_FREEOPERATION

```
#define TEE_PANIC_ID_TEE_FREEOPERATION 0x00000C03
```

10.9.1.234 TEE_PANIC_ID_TEE_FREEPERSISTENTOBJECTENUMERATOR

```
#define TEE_PANIC_ID_TEE_↩
FREEPERSISTENTOBJECTENUMERATOR 0x00000A02
```

10.9.1.235 TEE_PANIC_ID_TEE_FREEPROPERTYENUMERATOR

```
#define TEE_PANIC_ID_TEE_FREEPROPERTYENUMERATOR 0x000
```

10.9.1.236 TEE_PANIC_ID_TEE_FREETRANSIENTOBJECT

```
#define TEE_PANIC_ID_TEE_FREETRANSIENTOBJECT 0x00000803
```

10.9.1.237 TEE_PANIC_ID_TEE_GENERATEKEY

```
#define TEE_PANIC_ID_TEE_GENERATEKEY 0x00000804
```

10.9.1.238 TEE_PANIC_ID_TEE_GENERATERANDOM

```
#define TEE_PANIC_ID_TEE_GENERATERANDOM 0x00001301
```

10.9.1.239 TEE_PANIC_ID_TEE_GETCANCELLATIONFLAG

```
#define TEE_PANIC_ID_TEE_GETCANCELLATIONFLAG 0x00000501
```

10.9.1.240 TEE_PANIC_ID_TEE_GETINSTANCEDATA

```
#define TEE_PANIC_ID_TEE_GETINSTANCEDATA 0x00000603
```

10.9.1.241 TEE_PANIC_ID_TEE_GETNEXTPERSISTENTOBJECT

```
#define TEE_PANIC_ID_TEE_GETNEXTPERSISTENTOBJECT 0x00
A03
```

10.9.1.242 TEE_PANIC_ID_TEE_GETNEXTPROPERTY

```
#define TEE_PANIC_ID_TEE_GETNEXTPROPERTY 0x00000203
```

10.9.1.243 TEE_PANIC_ID_TEE_GETOBJECTBUFFERATTRIBUTE

```
#define TEE_PANIC_ID_TEE_GETOBJECTBUFFERATTRIBUTE 0x
```

10.9.1.244 TEE_PANIC_ID_TEE_GETOBJECTINFO

```
#define TEE_PANIC_ID_TEE_GETOBJECTINFO 0x00000703
```

10.9.1.245 TEE_PANIC_ID_TEE_GETOBJECTINFO1

```
#define TEE_PANIC_ID_TEE_GETOBJECTINFO1 0x00000706
```

10.9.1.246 TEE_PANIC_ID_TEE_GETOBJECTVALUEATTRIBUTE

```
#define TEE_PANIC_ID_TEE_GETOBJECTVALUEATTRIBUTE 0x000
```

10.9.1.247 TEE_PANIC_ID_TEE_GETOPERATIONINFO

```
#define TEE_PANIC_ID_TEE_GETOPERATIONINFO 0x00000↩
C04
```

10.9.1.248 TEE_PANIC_ID_TEE_GETOPERATIONINFOMULTIPLE

```
#define TEE_PANIC_ID_TEE_GETOPERATIONINFOMULTIPLE 0x0
C08
```

10.9.1.249 TEE_PANIC_ID_TEE_GETPROPERTYASBINARYBLOCK

```
#define TEE_PANIC_ID_TEE_GETPROPERTYASBINARYBLOCK 0
```

10.9.1.250 TEE_PANIC_ID_TEE_GETPROPERTYASBOOL

```
#define TEE_PANIC_ID_TEE_GETPROPERTYASBOOL 0x00000205
```

10.9.1.251 TEE_PANIC_ID_TEE_GETPROPERTYASIDENTITY

```
#define TEE_PANIC_ID_TEE_GETPROPERTYASIDENTITY 0x0000020
```

10.9.1.252 TEE_PANIC_ID_TEE_GETPROPERTYASSTRING

```
#define TEE_PANIC_ID_TEE_GETPROPERTYASSTRING 0x00000207
```

10.9.1.253 TEE_PANIC_ID_TEE_GETPROPERTYASU32

```
#define TEE_PANIC_ID_TEE_GETPROPERTYASU32 0x00000208
```

10.9.1.254 TEE_PANIC_ID_TEE_GETPROPERTYASUUID

```
#define TEE_PANIC_ID_TEE_GETPROPERTYASUUID 0x00000209
```

10.9.1.255 TEE_PANIC_ID_TEE_GETPROPERTYNAME

```
#define TEE_PANIC_ID_TEE_GETPROPERTYNAME 0x0000020A
```

10.9.1.256 TEE_PANIC_ID_TEE_GETREETIME

```
#define TEE_PANIC_ID_TEE_GETREETIME 0x00001401
```

10.9.1.257 TEE_PANIC_ID_TEE_GETSYSTEMTIME

```
#define TEE_PANIC_ID_TEE_GETSYSTEMTIME 0x00001402
```

10.9.1.258 TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME

```
#define TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME 0x00001403
```

10.9.1.259 TEE_PANIC_ID_TEE_INITREFATTRIBUTE

```
#define TEE_PANIC_ID_TEE_INITREFATTRIBUTE 0x00000805
```

10.9.1.260 TEE_PANIC_ID_TEE_INITVALUEATTRIBUTE

```
#define TEE_PANIC_ID_TEE_INITVALUEATTRIBUTE 0x00000806
```

10.9.1.261 TEE_PANIC_ID_TEE_INVOKETACOMMAND

```
#define TEE_PANIC_ID_TEE_INVOKETACOMMAND 0x00000402
```

10.9.1.262 TEE_PANIC_ID_TEE_MACCOMPAREFINAL

```
#define TEE_PANIC_ID_TEE_MACCOMPAREFINAL 0x00000↩
F01
```

10.9.1.263 TEE_PANIC_ID_TEE_MACCOMPUTEFINAL

```
#define TEE_PANIC_ID_TEE_MACCOMPUTEFINAL 0x00000↩
F02
```

10.9.1.264 TEE_PANIC_ID_TEE_MACINIT

```
#define TEE_PANIC_ID_TEE_MACINIT 0x00000F03
```

10.9.1.265 TEE_PANIC_ID_TEE_MACUPDATE

```
#define TEE_PANIC_ID_TEE_MACUPDATE 0x00000F04
```

10.9.1.266 TEE_PANIC_ID_TEE_MALLOC

```
#define TEE_PANIC_ID_TEE_MALLOC 0x00000604
```

10.9.1.267 TEE_PANIC_ID_TEE_MASKCANCELLATION

```
#define TEE_PANIC_ID_TEE_MASKCANCELLATION 0x00000502
```

10.9.1.268 TEE_PANIC_ID_TEE_MEMCOMPARE

```
#define TEE_PANIC_ID_TEE_MEMCOMPARE 0x00000605
```

10.9.1.269 TEE_PANIC_ID_TEE_MEMFILL

```
#define TEE_PANIC_ID_TEE_MEMFILL 0x00000606
```

10.9.1.270 TEE_PANIC_ID_TEE_MEMMOVE

```
#define TEE_PANIC_ID_TEE_MEMMOVE 0x00000607
```

10.9.1.271 TEE_PANIC_ID_TEE_OPENPERSISTENTOBJECT

```
#define TEE_PANIC_ID_TEE_OPENPERSISTENTOBJECT 0x00000903
```

10.9.1.272 TEE_PANIC_ID_TEE_OPENTASESSION

```
#define TEE_PANIC_ID_TEE_OPENTASESSION 0x00000403
```

10.9.1.273 TEE_PANIC_ID_TEE_PANIC

```
#define TEE_PANIC_ID_TEE_PANIC 0x00000301
```

10.9.1.274 TEE_PANIC_ID_TEE_POPULATETRANSIENTOBJECT

```
#define TEE_PANIC_ID_TEE_POPULATETRANSIENTOBJECT 0x00
```

10.9.1.275 TEE_PANIC_ID_TEE_READOBJECTDATA

```
#define TEE_PANIC_ID_TEE_READOBJECTDATA 0x00000↩
B01
```

10.9.1.276 TEE_PANIC_ID_TEE_REALLOC

```
#define TEE_PANIC_ID_TEE_REALLOC 0x00000608
```

10.9.1.277 TEE_PANIC_ID_TEE_RENAMEPERSISTENTOBJECT

```
#define TEE_PANIC_ID_TEE_RENAMEPERSISTENTOBJECT 0x0000
```

10.9.1.278 TEE_PANIC_ID_TEE_RESETOPERATION

```
#define TEE_PANIC_ID_TEE_RESETOPERATION 0x00000↩
C05
```

10.9.1.279 TEE_PANIC_ID_TEE_RESETPERSISTENTOBJECTENUMERATOR

```
#define TEE_PANIC_ID_TEE_↩
RESETPERSISTENTOBJECTENUMERATOR 0x00000A04
```

10.9.1.280 TEE_PANIC_ID_TEE_RESETPROPERTYENUMERATOR

```
#define TEE_PANIC_ID_TEE_RESETPROPERTYENUMERATOR 0x0
```

10.9.1.281 TEE_PANIC_ID_TEE_RESETTRANSIENTOBJECT

```
#define TEE_PANIC_ID_TEE_RESETTRANSIENTOBJECT 0x00000808
```

10.9.1.282 TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE

```
#define TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE 0x00000705
```

10.9.1.283 TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE1

```
#define TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE1 0x00000707
```

10.9.1.284 TEE_PANIC_ID_TEE_SEEKOBJECTDATA

```
#define TEE_PANIC_ID_TEE_SEEKOBJECTDATA 0x00000↩
B02
```

10.9.1.285 TEE_PANIC_ID_TEE_SETINSTANCEDATA

```
#define TEE_PANIC_ID_TEE_SETINSTANCEDATA 0x00000609
```

10.9.1.286 TEE_PANIC_ID_TEE_SETOPERATIONKEY

```
#define TEE_PANIC_ID_TEE_SETOPERATIONKEY 0x00000C06
```

10.9.1.287 TEE_PANIC_ID_TEE_SETOPERATIONKEY2

```
#define TEE_PANIC_ID_TEE_SETOPERATIONKEY2 0x00000C07
```

10.9.1.288 TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME

```
#define TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME 0x00001404
```

10.9.1.289 TEE_PANIC_ID_TEE_STARTPERSISTENTOBJECTENUMERATOR

```
#define TEE_PANIC_ID_TEE_STARTPERSISTENTOBJECTENUMERATOR 0x00000A05
```

10.9.1.290 TEE_PANIC_ID_TEE_STARTPROPERTYENUMERATOR

```
#define TEE_PANIC_ID_TEE_STARTPROPERTYENUMERATOR 0x0
```

10.9.1.291 TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA

```
#define TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA 0x00000B03
```

10.9.1.292 TEE_PANIC_ID_TEE_UNMASKCANCELLATION

```
#define TEE_PANIC_ID_TEE_UNMASKCANCELLATION 0x00000503
```

10.9.1.293 TEE_PANIC_ID_TEE_WAIT

```
#define TEE_PANIC_ID_TEE_WAIT 0x00001405
```

10.9.1.294 TEE_PANIC_ID_TEE_WRITEOBJECTDATA

```
#define TEE_PANIC_ID_TEE_WRITEOBJECTDATA 0x00000B04
```

10.9.1.295 TEE_PARAM_TYPE_GET

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

10.9.1.296 TEE_PARAM_TYPE_MEMREF_INOUT

```
#define TEE_PARAM_TYPE_MEMREF_INOUT 7
```

10.9.1.297 TEE_PARAM_TYPE_MEMREF_INPUT

```
#define TEE_PARAM_TYPE_MEMREF_INPUT 5
```

10.9.1.298 TEE_PARAM_TYPE_MEMREF_OUTPUT

```
#define TEE_PARAM_TYPE_MEMREF_OUTPUT 6
```

10.9.1.299 TEE_PARAM_TYPE_NONE

```
#define TEE_PARAM_TYPE_NONE 0
```

10.9.1.300 TEE_PARAM_TYPE_SET

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

10.9.1.301 TEE_PARAM_TYPE_VALUE_INOUT

```
#define TEE_PARAM_TYPE_VALUE_INOUT 3
```

10.9.1.302 TEE_PARAM_TYPE_VALUE_INPUT

```
#define TEE_PARAM_TYPE_VALUE_INPUT 1
```

10.9.1.303 TEE_PARAM_TYPE_VALUE_OUTPUT

```
#define TEE_PARAM_TYPE_VALUE_OUTPUT 2
```

10.9.1.304 TEE_PARAM_TYPES

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

10.9.1.305 TEE_PROPSET_CURRENT_CLIENT

```
#define TEE_PROPSET_CURRENT_CLIENT (TEE_PropSetHandle)0x↩
FFFFFFFE
```

10.9.1.306 TEE_PROPSET_CURRENT_TA

```
#define TEE_PROPSET_CURRENT_TA (TEE_PropSetHandle)0x↩
FFFFFFFF
```

10.9.1.307 TEE_PROPSET_TEE_IMPLEMENTATION

```
#define TEE_PROPSET_TEE_IMPLEMENTATION (TEE_PropSetHandle)0x↩
FFFFFFFD
```

10.9.1.308 TEE_STORAGE_PRIVATE

```
#define TEE_STORAGE_PRIVATE 0x00000001
```

10.9.1.309 TEE_SUCCESS

```
#define TEE_SUCCESS 0x00000000
```

10.9.1.310 TEE_TIMEOUT_INFINITE

```
#define TEE_TIMEOUT_INFINITE 0xFFFFFFFF
```

10.9.1.311 TEE_TYPE_AES

```
#define TEE_TYPE_AES 0xA0000010
```

10.9.1.312 TEE_TYPE_CORRUPTED_OBJECT

```
#define TEE_TYPE_CORRUPTED_OBJECT 0xA00000BE
```

10.9.1.313 TEE_TYPE_DATA

```
#define TEE_TYPE_DATA 0xA00000BF
```

10.9.1.314 TEE_TYPE_DES

```
#define TEE_TYPE_DES 0xA0000011
```

10.9.1.315 TEE_TYPE_DES3

```
#define TEE_TYPE_DES3 0xA0000013
```

10.9.1.316 TEE_TYPE_DH_KEYPAIR

```
#define TEE_TYPE_DH_KEYPAIR 0xA1000032
```

10.9.1.317 TEE_TYPE_DSA_KEYPAIR

```
#define TEE_TYPE_DSA_KEYPAIR 0xA1000031
```

10.9.1.318 TEE_TYPE_DSA_PUBLIC_KEY

```
#define TEE_TYPE_DSA_PUBLIC_KEY 0xA0000031
```

10.9.1.319 TEE_TYPE_ECDH_KEYPAIR

```
#define TEE_TYPE_ECDH_KEYPAIR 0xA1000042
```

10.9.1.320 TEE_TYPE_ECDH_PUBLIC_KEY

```
#define TEE_TYPE_ECDH_PUBLIC_KEY 0xA0000042
```

10.9.1.321 TEE_TYPE_ECDSA_KEYPAIR

```
#define TEE_TYPE_ECDSA_KEYPAIR 0xA1000041
```

10.9.1.322 TEE_TYPE_ECDSA_PUBLIC_KEY

```
#define TEE_TYPE_ECDSA_PUBLIC_KEY 0xA0000041
```

10.9.1.323 TEE_TYPE_GENERIC_SECRET

```
#define TEE_TYPE_GENERIC_SECRET 0xA0000000
```

10.9.1.324 TEE_TYPE_HMAC_MD5

```
#define TEE_TYPE_HMAC_MD5 0xA0000001
```

10.9.1.325 TEE_TYPE_HMAC_SHA1

```
#define TEE_TYPE_HMAC_SHA1 0xA0000002
```

10.9.1.326 TEE_TYPE_HMAC_SHA224

```
#define TEE_TYPE_HMAC_SHA224 0xA0000003
```

10.9.1.327 TEE_TYPE_HMAC_SHA256

```
#define TEE_TYPE_HMAC_SHA256 0xA0000004
```

10.9.1.328 TEE_TYPE_HMAC_SHA384

```
#define TEE_TYPE_HMAC_SHA384 0xA0000005
```

10.9.1.329 TEE_TYPE_HMAC_SHA512

```
#define TEE_TYPE_HMAC_SHA512 0xA0000006
```

10.9.1.330 TEE_TYPE_RSA_KEYPAIR

```
#define TEE_TYPE_RSA_KEYPAIR 0xA1000030
```

10.9.1.331 TEE_TYPE_RSA_PUBLIC_KEY

```
#define TEE_TYPE_RSA_PUBLIC_KEY 0xA0000030
```

10.9.1.332 TEE_USAGE_DECRYPT

```
#define TEE_USAGE_DECRYPT 0x00000004
```

10.9.1.333 TEE_USAGE_DERIVE

```
#define TEE_USAGE_DERIVE 0x00000040
```

10.9.1.334 TEE_USAGE_ENCRYPT

```
#define TEE_USAGE_ENCRYPT 0x00000002
```

10.9.1.335 TEE_USAGE_EXTRACTABLE

```
#define TEE_USAGE_EXTRACTABLE 0x00000001
```

10.9.1.336 TEE_USAGE_MAC

```
#define TEE_USAGE_MAC 0x00000008
```

10.9.1.337 TEE_USAGE_SIGN

```
#define TEE_USAGE_SIGN 0x00000010
```

10.9.1.338 TEE_USAGE_VERIFY

```
#define TEE_USAGE_VERIFY 0x00000020
```

10.10 tee_api_defines.h

Go to the documentation of this file.

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

```
84 #define TEE_LOGIN_APPLICATION           0x00000004
85 #define TEE_LOGIN_APPLICATION_USER      0x00000005
86 #define TEE_LOGIN_APPLICATION_GROUP     0x00000006
87 #define TEE_LOGIN_TRUSTED_APP           0xF0000000
88
89 /* Origin Code Constants */
90 #define TEE_ORIGIN_API                  0x00000001
91 #define TEE_ORIGIN_COMMS                0x00000002
92 #define TEE_ORIGIN_TEE                  0x00000003
93 #define TEE_ORIGIN_TRUSTED_APP          0x00000004
94
95 /* Property Sets pseudo handles */
96 #define TEE_PROPSET_TEE_IMPLEMENTATION  (TEE_PropSetHandle)0xFFFFFFFD
97 #define TEE_PROPSET_CURRENT_CLIENT      (TEE_PropSetHandle)0xFFFFFFFE
98 #define TEE_PROPSET_CURRENT_TA          (TEE_PropSetHandle)0xFFFFFFFF
99
100 /* Memory Access Rights Constants */
101 #define TEE_MEMORY_ACCESS_READ             0x00000001
102 #define TEE_MEMORY_ACCESS_WRITE            0x00000002
103 #define TEE_MEMORY_ACCESS_ANY_OWNER        0x00000004
104
105 /* Memory Management Constant */
106 #define TEE_MALLOC_FILL_ZERO               0x00000000
107
108 /* Other constants */
109 #define TEE_STORAGE_PRIVATE                0x00000001
110
111 #define TEE_DATA_FLAG_ACCESS_READ          0x00000001
112 #define TEE_DATA_FLAG_ACCESS_WRITE         0x00000002
113 #define TEE_DATA_FLAG_ACCESS_WRITE_META    0x00000004
114 #define TEE_DATA_FLAG_SHARE_READ           0x00000010
115 #define TEE_DATA_FLAG_SHARE_WRITE          0x00000020
116 #define TEE_DATA_FLAG_OVERWRITE            0x00000400
117 #define TEE_DATA_MAX_POSITION              0xFFFFFFFF
118 #define TEE_OBJECT_ID_MAX_LEN              64
119 #define TEE_USAGE_EXTRACTABLE              0x00000001
120 #define TEE_USAGE_ENCRYPT                  0x00000002
121 #define TEE_USAGE_DECRYPT                  0x00000004
122 #define TEE_USAGE_MAC                      0x00000008
123 #define TEE_USAGE_SIGN                     0x00000010
124 #define TEE_USAGE_VERIFY                   0x00000020
125 #define TEE_USAGE_DERIVE                   0x00000040
126 #define TEE_HANDLE_FLAG_PERSISTENT         0x00010000
127 #define TEE_HANDLE_FLAG_INITIALIZED        0x00020000
128 #define TEE_HANDLE_FLAG_KEY_SET            0x00040000
129 #define TEE_HANDLE_FLAG_EXPECT_TWO_KEYS    0x00080000
130 #define TEE_OPERATION_CIPHER               1
131 #define TEE_OPERATION_MAC                  3
132 #define TEE_OPERATION_AE                   4
133 #define TEE_OPERATION_DIGEST               5
134 #define TEE_OPERATION_ASYMMETRIC_CIPHER    6
135 #define TEE_OPERATION_ASYMMETRIC_SIGNATURE 7
136 #define TEE_OPERATION_KEY_DERIVATION       8
137 #define TEE_OPERATION_STATE_INITIAL        0x00000000
138 #define TEE_OPERATION_STATE_ACTIVE         0x00000001
139
140 /* Algorithm Identifiers */
141 #define TEE_ALG_AES_ECB_NOPAD                  0x10000010
142 #define TEE_ALG_AES_CBC_NOPAD                  0x10000110
143 #define TEE_ALG_AES_CTR                        0x10000210
144 #define TEE_ALG_AES_CTS                        0x10000310
145 #define TEE_ALG_AES_XTS                        0x10000410
146 #define TEE_ALG_AES_CBC_MAC_NOPAD              0x30000110
147 #define TEE_ALG_AES_CBC_MAC_PKCS5              0x30000510
148 #define TEE_ALG_AES_CMAC                       0x30000610
149 #define TEE_ALG_AES_CCM                        0x40000710
150 #define TEE_ALG_AES_GCM                        0x40000810
151 #define TEE_ALG_DES_ECB_NOPAD                  0x10000011
152 #define TEE_ALG_DES_CBC_NOPAD                  0x10000111
153 #define TEE_ALG_DES_CBC_MAC_NOPAD              0x30000111
154 #define TEE_ALG_DES_CBC_MAC_PKCS5              0x30000511
155 #define TEE_ALG_DES3_ECB_NOPAD                 0x10000013
156 #define TEE_ALG_DES3_CBC_NOPAD                 0x10000113
157 #define TEE_ALG_DES3_CBC_MAC_NOPAD             0x30000113
158 #define TEE_ALG_DES3_CBC_MAC_PKCS5             0x30000513
159 #define TEE_ALG_RSASSA_PKCS1_V1_5_MD5          0x70001830
160 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA1         0x70002830
161 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA224       0x70003830
162 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA256       0x70004830
163 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA384       0x70005830
164 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA512       0x70006830
165 #define TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1      0x7000F830
166 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1     0x70212930
167 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224   0x70313930
168 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256   0x70414930
169 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384   0x70515930
170 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512   0x70616930
```

```
171 #define TEE_ALG_RSAES_PKCS1_V1_5                0x60000130
172 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1      0x60210230
173 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224    0x60310230
174 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256    0x60410230
175 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384    0x60510230
176 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512    0x60610230
177 #define TEE_ALG_RSA_NOPAD                        0x60000030
178 #define TEE_ALG_DSA_SHA1                         0x70002131
179 #define TEE_ALG_DSA_SHA224                       0x70003131
180 #define TEE_ALG_DSA_SHA256                       0x70004131
181 #define TEE_ALG_DH_DERIVE_SHARED_SECRET         0x80000032
182 #define TEE_ALG_MD5                              0x50000001
183 #define TEE_ALG_SHA1                             0x50000002
184 #define TEE_ALG_SHA224                           0x50000003
185 #define TEE_ALG_SHA256                           0x50000004
186 #define TEE_ALG_SHA384                           0x50000005
187 #define TEE_ALG_SHA512                           0x50000006
188 #define TEE_ALG_MD5SHA1                          0x5000000F
189 #define TEE_ALG_HMAC_MD5                        0x30000001
190 #define TEE_ALG_HMAC_SHA1                        0x30000002
191 #define TEE_ALG_HMAC_SHA224                      0x30000003
192 #define TEE_ALG_HMAC_SHA256                      0x30000004
193 #define TEE_ALG_HMAC_SHA384                      0x30000005
194 #define TEE_ALG_HMAC_SHA512                      0x30000006
195 /*
196  * Fix GP Internal Core API v1.1
197  *     "Table 6-12:  Structure of Algorithm Identifier"
198  *     indicates ECDSA have the algorithm "0x41" and ECDH "0x42"
199  * whereas
200  *     "Table 6-11:  List of Algorithm Identifiers" defines
201  *     TEE_ALG_ECDSA_P192 as 0x70001042
202  *
203  * We chose to define TEE_ALG_ECDSA_P192 as 0x70001041 (conform to table 6-12)
204  */
205 #define TEE_ALG_ECDSA_P192                       0x70001041
206 #define TEE_ALG_ECDSA_P224                       0x70002041
207 #define TEE_ALG_ECDSA_P256                       0x70003041
208 #define TEE_ALG_ECDSA_P384                       0x70004041
209 #define TEE_ALG_ECDSA_P521                       0x70005041
210 #define TEE_ALG_ECDH_P192                        0x80001042
211 #define TEE_ALG_ECDH_P224                        0x80002042
212 #define TEE_ALG_ECDH_P256                        0x80003042
213 #define TEE_ALG_ECDH_P384                        0x80004042
214 #define TEE_ALG_ECDH_P521                        0x80005042
215
216 /* Object Types */
217
218 #define TEE_TYPE_AES                          0xA0000010
219 #define TEE_TYPE_DES                          0xA0000011
220 #define TEE_TYPE_DES3                         0xA0000013
221 #define TEE_TYPE_HMAC_MD5                    0xA0000001
222 #define TEE_TYPE_HMAC_SHA1                    0xA0000002
223 #define TEE_TYPE_HMAC_SHA224                  0xA0000003
224 #define TEE_TYPE_HMAC_SHA256                  0xA0000004
225 #define TEE_TYPE_HMAC_SHA384                  0xA0000005
226 #define TEE_TYPE_HMAC_SHA512                  0xA0000006
227 #define TEE_TYPE_RSA_PUBLIC_KEY              0xA0000030
228 #define TEE_TYPE_RSA_KEYPAIR                  0xA1000030
229 #define TEE_TYPE_DSA_PUBLIC_KEY              0xA0000031
230 #define TEE_TYPE_DSA_KEYPAIR                  0xA1000031
231 #define TEE_TYPE_DH_KEYPAIR                   0xA1000032
232 #define TEE_TYPE_ECDSA_PUBLIC_KEY            0xA0000041
233 #define TEE_TYPE_ECDSA_KEYPAIR                0xA1000041
234 #define TEE_TYPE_ECDH_PUBLIC_KEY             0xA0000042
235 #define TEE_TYPE_ECDH_KEYPAIR                 0xA1000042
236 #define TEE_TYPE_GENERIC_SECRET               0xA0000000
237 #define TEE_TYPE_CORRUPTED_OBJECT             0xA00000BE
238 #define TEE_TYPE_DATA                          0xA00000BF
239
240 /* List of Object or Operation Attributes */
241
242 #define TEE_ATTR_SECRET_VALUE                 0xC0000000
243 #define TEE_ATTR_RSA_MODULUS                  0xD0000130
244 #define TEE_ATTR_RSA_PUBLIC_EXPONENT         0xD0000230
245 #define TEE_ATTR_RSA_PRIVATE_EXPONENT        0xC0000330
246 #define TEE_ATTR_RSA_PRIME1                   0xC0000430
247 #define TEE_ATTR_RSA_PRIME2                   0xC0000530
248 #define TEE_ATTR_RSA_EXPONENT1                0xC0000630
249 #define TEE_ATTR_RSA_EXPONENT2                0xC0000730
250 #define TEE_ATTR_RSA_COEFFICIENT              0xC0000830
251 #define TEE_ATTR_DSA_PRIME                    0xD0001031
252 #define TEE_ATTR_DSA_SUBPRIME                 0xD0001131
253 #define TEE_ATTR_DSA_BASE                     0xD0001231
254 #define TEE_ATTR_DSA_PUBLIC_VALUE            0xD0000131
255 #define TEE_ATTR_DSA_PRIVATE_VALUE           0xC0000231
256 #define TEE_ATTR_DH_PRIME                     0xD0001032
257 #define TEE_ATTR_DH_SUBPRIME                  0xD0001132
```

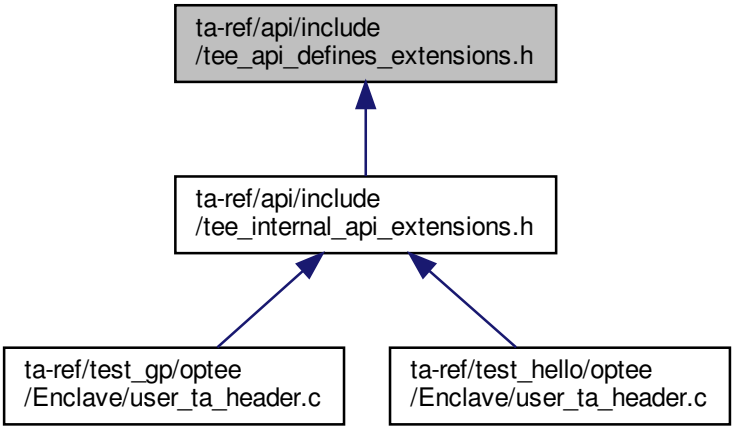
```
258 #define TEE_ATTR_DH_BASE                    0xD0001232
259 #define TEE_ATTR_DH_X_BITS                  0xF0001332
260 #define TEE_ATTR_DH_PUBLIC_VALUE            0xD0000132
261 #define TEE_ATTR_DH_PRIVATE_VALUE           0xC0000232
262 #define TEE_ATTR_RSA_OAEP_LABEL             0xD0000930
263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

```
345 #define TEE_PANIC_ID_TEE_OPENPERSISTENTOBJECT            0x00000903
346 #define TEE_PANIC_ID_TEE_RENAMEPERSISTENTOBJECT          0x00000904
347 #define TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT1 0x00000905
348 /* Persistent Object Enumeration */
349 #define TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR 0x00000A01
350 #define TEE_PANIC_ID_TEE_FREEPERSISTENTOBJECTENUMERATOR     0x00000A02
351 #define TEE_PANIC_ID_TEE_GETNEXTPERSISTENTOBJECT            0x00000A03
352 #define TEE_PANIC_ID_TEE_RESETPERSISTENTOBJECTENUMERATOR    0x00000A04
353 #define TEE_PANIC_ID_TEE_STARTPERSISTENTOBJECTENUMERATOR    0x00000A05
354 /* Data Stream Access */
355 #define TEE_PANIC_ID_TEE_READOBJECTDATA             0x00000B01
356 #define TEE_PANIC_ID_TEE_SEEKOBJECTDATA             0x00000B02
357 #define TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA         0x00000B03
358 #define TEE_PANIC_ID_TEE_WRITEOBJECTDATA            0x00000B04
359 /* Generic Operation */
360 #define TEE_PANIC_ID_TEE_ALLOCATEOPERATION          0x00000C01
361 #define TEE_PANIC_ID_TEE_COPYOPERATION              0x00000C02
362 #define TEE_PANIC_ID_TEE_FREEOPERATION              0x00000C03
363 #define TEE_PANIC_ID_TEE_GETOPERATIONINFO           0x00000C04
364 #define TEE_PANIC_ID_TEE_RESETOPERATION             0x00000C05
365 #define TEE_PANIC_ID_TEE_SETOPERATIONKEY            0x00000C06
366 #define TEE_PANIC_ID_TEE_SETOPERATIONKEY2           0x00000C07
367 #define TEE_PANIC_ID_TEE_GETOPERATIONINFOMULTIPLE   0x00000C08
368 /* Message Digest */
369 #define TEE_PANIC_ID_TEE_DIGESTDOFINAL              0x00000D01
370 #define TEE_PANIC_ID_TEE_DIGESTUPDATE               0x00000D02
371 /* Symmetric Cipher */
372 #define TEE_PANIC_ID_TEE_CIPHERDOFINAL              0x00000E01
373 #define TEE_PANIC_ID_TEE_CIPHERINIT                 0x00000E02
374 #define TEE_PANIC_ID_TEE_CIPHERUPDATE               0x00000E03
375 /* MAC */
376 #define TEE_PANIC_ID_TEE_MACCOMPAREFINAL            0x00000F01
377 #define TEE_PANIC_ID_TEE_MACCOMPUTEFINAL            0x00000F02
378 #define TEE_PANIC_ID_TEE_MACINIT                    0x00000F03
379 #define TEE_PANIC_ID_TEE_MACUPDATE                  0x00000F04
380 /* Authenticated Encryption */
381 #define TEE_PANIC_ID_TEE_AEDECRYPTFINAL             0x00001001
382 #define TEE_PANIC_ID_TEE_AEENCRYPTFINAL             0x00001002
383 #define TEE_PANIC_ID_TEE_AEINIT                     0x00001003
384 #define TEE_PANIC_ID_TEE_AEUPDATE                   0x00001004
385 #define TEE_PANIC_ID_TEE_AEUPDATEAAD                0x00001005
386 /* Asymmetric */
387 #define TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT          0x00001101
388 #define TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT          0x00001102
389 #define TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST       0x00001103
390 #define TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST     0x00001104
391 /* Key Derivation */
392 #define TEE_PANIC_ID_TEE_DERIVEKEY                  0x00001201
393 /* Random Data Generation */
394 #define TEE_PANIC_ID_TEE_GENERATERANDOM             0x00001301
395 /* Time */
396 #define TEE_PANIC_ID_TEE_GETREETIME                 0x00001401
397 #define TEE_PANIC_ID_TEE_GETSYSTEMTIME              0x00001402
398 #define TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME        0x00001403
399 #define TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME        0x00001404
400 #define TEE_PANIC_ID_TEE_WAIT                       0x00001405
401 /* Memory Allocation and Size of Objects */
402 #define TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32  0x00001501
403 #define TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32         0x00001502
404 /* Initialization */
405 #define TEE_PANIC_ID_TEE_BIGINTINIT                 0x00001601
406 #define TEE_PANIC_ID_TEE_BIGINTINITFMM              0x00001602
407 #define TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT       0x00001603
408 /* Converter */
409 #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING 0x00001701
410 #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32         0x00001702
411 #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING   0x00001703
412 #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32           0x00001704
413 /* Logical Operation */
414 #define TEE_PANIC_ID_TEE_BIGINTCMP                  0x00001801
415 #define TEE_PANIC_ID_TEE_BIGINTCMPS32               0x00001802
416 #define TEE_PANIC_ID_TEE_BIGINTGETBIT               0x00001803
417 #define TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT          0x00001804
418 #define TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT           0x00001805
419 /* Basic Arithmetic */
420 #define TEE_PANIC_ID_TEE_BIGINTADD                  0x00001901
421 #define TEE_PANIC_ID_TEE_BIGINTDIV                  0x00001902
422 #define TEE_PANIC_ID_TEE_BIGINTMUL                  0x00001903
423 #define TEE_PANIC_ID_TEE_BIGINTNEG                  0x00001904
424 #define TEE_PANIC_ID_TEE_BIGINTSQUARE               0x00001905
425 #define TEE_PANIC_ID_TEE_BIGINTSUB                  0x00001906
426 /* Modular Arithmetic */
427 #define TEE_PANIC_ID_TEE_BIGINTADDMOD               0x00001A01
428 #define TEE_PANIC_ID_TEE_BIGINTINVMOD               0x00001A02
429 #define TEE_PANIC_ID_TEE_BIGINTMOD                  0x00001A03
430 #define TEE_PANIC_ID_TEE_BIGINTMULMOD               0x00001A04
431 #define TEE_PANIC_ID_TEE_BIGINTSQUAREMOD            0x00001A05
```

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

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

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



Macros

- #define TEE_ALG_HKDF_MD5_DERIVE_KEY 0x800010C0
- #define TEE_ALG_HKDF_SHA1_DERIVE_KEY 0x800020C0
- #define TEE_ALG_HKDF_SHA224_DERIVE_KEY 0x800030C0
- #define TEE_ALG_HKDF_SHA256_DERIVE_KEY 0x800040C0
- #define TEE_ALG_HKDF_SHA384_DERIVE_KEY 0x800050C0
- #define TEE_ALG_HKDF_SHA512_DERIVE_KEY 0x800060C0
- #define TEE_TYPE_HKDF_IKM 0xA10000C0
- #define TEE_ATTR_HKDF_IKM 0xC00001C0
- #define TEE_ATTR_HKDF_SALT 0xD00002C0
- #define TEE_ATTR_HKDF_INFO 0xD00003C0
- #define TEE_ATTR_HKDF_OKM_LENGTH 0xF00004C0
- #define TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY 0x800020C1
- #define TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY 0x800030C1
- #define TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY 0x800040C1
- #define TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY 0x800050C1
- #define TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY 0x800060C1
- #define TEE_TYPE_CONCAT_KDF_Z 0xA10000C1
- #define TEE_ATTR_CONCAT_KDF_Z 0xC00001C1
- #define TEE_ATTR_CONCAT_KDF_OTHER_INFO 0xD00002C1
- #define TEE_ATTR_CONCAT_KDF_DKM_LENGTH 0xF00003C1
- #define TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY 0x800020C2
- #define TEE_TYPE_PBKDF2_PASSWORD 0xA10000C2
- #define TEE_ATTR_PBKDF2_PASSWORD 0xC00001C2
- #define TEE_ATTR_PBKDF2_SALT 0xD00002C2
- #define TEE_ATTR_PBKDF2_ITERATION_COUNT 0xF00003C2
- #define TEE_ATTR_PBKDF2_DKM_LENGTH 0xF00004C2
- #define TEE_STORAGE_PRIVATE_REE 0x80000000
- #define TEE_STORAGE_PRIVATE_RPMB 0x80000100
- #define TEE_STORAGE_PRIVATE_SQL_RESERVED 0x80000200
- #define TEE_MEMORY_ACCESS_NONSECURE 0x10000000
- #define TEE_MEMORY_ACCESS_SECURE 0x20000000

10.11.1 Macro Definition Documentation

10.11.1.1 TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY

```
#define TEE_ALG_CONCAT_KDF_SHA1_DERIVE_↩
KEY 0x800020C1
```

10.11.1.2 TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY

```
#define TEE_ALG_CONCAT_KDF_SHA224_DERIVE_↩
KEY 0x800030C1
```

10.11.1.3 TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY

```
#define TEE_ALG_CONCAT_KDF_SHA256_DERIVE_↩
KEY 0x800040C1
```

10.11.1.4 TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY

```
#define TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY 0x800050C1
```

10.11.1.5 TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY

```
#define TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY 0x800060C1
```

10.11.1.6 TEE_ALG_HKDF_MD5_DERIVE_KEY

```
#define TEE_ALG_HKDF_MD5_DERIVE_KEY 0x800010C0
```

10.11.1.7 TEE_ALG_HKDF_SHA1_DERIVE_KEY

```
#define TEE_ALG_HKDF_SHA1_DERIVE_KEY 0x800020C0
```

10.11.1.8 TEE_ALG_HKDF_SHA224_DERIVE_KEY

```
#define TEE_ALG_HKDF_SHA224_DERIVE_KEY 0x800030C0
```

10.11.1.9 TEE_ALG_HKDF_SHA256_DERIVE_KEY

```
#define TEE_ALG_HKDF_SHA256_DERIVE_KEY 0x800040C0
```

10.11.1.10 TEE_ALG_HKDF_SHA384_DERIVE_KEY

```
#define TEE_ALG_HKDF_SHA384_DERIVE_KEY 0x800050C0
```

10.11.1.11 TEE_ALG_HKDF_SHA512_DERIVE_KEY

```
#define TEE_ALG_HKDF_SHA512_DERIVE_KEY 0x800060C0
```

10.11.1.12 TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY

```
#define TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY 0x800020C2
```

10.11.1.13 TEE_ATTR_CONCAT_KDF_DKM_LENGTH

```
#define TEE_ATTR_CONCAT_KDF_DKM_LENGTH 0xF00003C1
```

10.11.1.14 TEE_ATTR_CONCAT_KDF_OTHER_INFO

```
#define TEE_ATTR_CONCAT_KDF_OTHER_INFO 0xD00002C1
```

10.11.1.15 TEE_ATTR_CONCAT_KDF_Z

```
#define TEE_ATTR_CONCAT_KDF_Z 0xC00001C1
```

10.11.1.16 TEE_ATTR_HKDF_IKM

```
#define TEE_ATTR_HKDF_IKM 0xC00001C0
```

10.11.1.17 TEE_ATTR_HKDF_INFO

```
#define TEE_ATTR_HKDF_INFO 0xD00003C0
```

10.11.1.18 TEE_ATTR_HKDF_OKM_LENGTH

```
#define TEE_ATTR_HKDF_OKM_LENGTH 0xF00004C0
```

10.11.1.19 TEE_ATTR_HKDF_SALT

```
#define TEE_ATTR_HKDF_SALT 0xD00002C0
```

10.11.1.20 TEE_ATTR_PBKDF2_DKM_LENGTH

```
#define TEE_ATTR_PBKDF2_DKM_LENGTH 0xF00004C2
```

10.11.1.21 TEE_ATTR_PBKDF2_ITERATION_COUNT

```
#define TEE_ATTR_PBKDF2_ITERATION_COUNT 0x↩
F00003C2
```

10.11.1.22 TEE_ATTR_PBKDF2_PASSWORD

```
#define TEE_ATTR_PBKDF2_PASSWORD 0xC00001C2
```

10.11.1.23 TEE_ATTR_PBKDF2_SALT

```
#define TEE_ATTR_PBKDF2_SALT 0xD00002C2
```

10.11.1.24 TEE_MEMORY_ACCESS_NONSECURE

```
#define TEE_MEMORY_ACCESS_NONSECURE 0x10000000
```

10.11.1.25 TEE_MEMORY_ACCESS_SECURE

```
#define TEE_MEMORY_ACCESS_SECURE 0x20000000
```

10.11.1.26 TEE_STORAGE_PRIVATE_REE

```
#define TEE_STORAGE_PRIVATE_REE 0x80000000
```

10.11.1.27 TEE_STORAGE_PRIVATE_RPMB

```
#define TEE_STORAGE_PRIVATE_RPMB 0x80000100
```

10.11.1.28 TEE_STORAGE_PRIVATE_SQL_RESERVED

```
#define TEE_STORAGE_PRIVATE_SQL_RESERVED 0x80000200
```

10.11.1.29 TEE_TYPE_CONCAT_KDF_Z

```
#define TEE_TYPE_CONCAT_KDF_Z 0xA10000C1
```

10.11.1.30 TEE_TYPE_HKDF_IKM

```
#define TEE_TYPE_HKDF_IKM 0xA10000C0
```

10.11.1.31 TEE_TYPE_PBKDF2_PASSWORD

```
#define TEE_TYPE_PBKDF2_PASSWORD 0xA10000C2
```

10.12 tee_api_defines_extensions.h

Go to the documentation of this file.

```
1 /*
2  * Copyright (c) 2014, Linaro Limited
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10  *
11  * 2. Redistributions in binary form must reproduce the above copyright notice,
12  * this list of conditions and the following disclaimer in the documentation
13  * and/or other materials provided with the distribution.
14  *
15  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25  * POSSIBILITY OF SUCH DAMAGE.
26  */
27
28 #ifndef TEE_API_DEFINES_EXTENSIONS_H
29 #define TEE_API_DEFINES_EXTENSIONS_H
30
31 /*
32  * HMAC-based Extract-and-Expand Key Derivation Function (HKDF)
33  */
34
35 #define TEE_ALG_HKDF_MD5_DERIVE_KEY     0x800010C0
36 #define TEE_ALG_HKDF_SHA1_DERIVE_KEY    0x800020C0
37 #define TEE_ALG_HKDF_SHA224_DERIVE_KEY  0x800030C0
38 #define TEE_ALG_HKDF_SHA256_DERIVE_KEY  0x800040C0
```

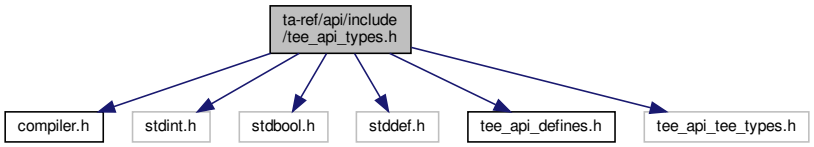
```
39 #define TEE_ALG_HKDF_SHA384_DERIVE_KEY  0x800050C0
40 #define TEE_ALG_HKDF_SHA512_DERIVE_KEY  0x800060C0
41
42 #define TEE_TYPE_HKDF_IKM               0xA10000C0
43
44 #define TEE_ATTR_HKDF_IKM               0xC00001C0
45 #define TEE_ATTR_HKDF_SALT              0xD00002C0
46 #define TEE_ATTR_HKDF_INFO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

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

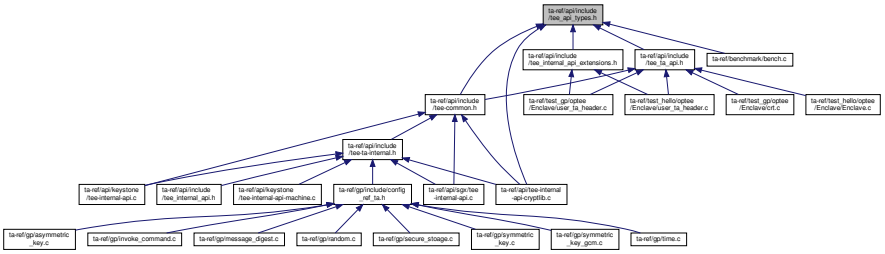
```
#include <compiler.h>
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <tee_api_defines.h>
```

```
#include "tee_api_tee_types.h"
```

Include dependency graph for tee_api_types.h:



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



Classes

- struct TEE_UUID
- struct TEE_Identity
- union TEE_Param
- struct TEE_ObjectInfo
- struct TEE_Attribute
- struct TEE_OperationInfo
- struct TEE_OperationInfoKey
- struct TEE_OperationInfoMultiple
- struct TEE_Time
- struct TEE_SEReaderProperties
- struct TEE_SEAID
- struct pollfd
- struct addrinfo

Macros

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

Typedefs

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

Enumerations

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

10.13.1 Macro Definition Documentation

10.13.1.1 DMREQ_FINISH `#define DMREQ_FINISH 0`

10.13.1.2 DMREQ_WRITE `#define DMREQ_WRITE 1`

10.13.1.3 TEE_MEM_INPUT `#define TEE_MEM_INPUT 0x00000001`

10.13.1.4 TEE_MEM_OUTPUT `#define TEE_MEM_OUTPUT 0x00000002`

10.13.1.5 TEE_MEMREF_0_USED

```
#define TEE_MEMREF_0_USED 0x00000001
```

10.13.1.6 TEE_MEMREF_1_USED

```
#define TEE_MEMREF_1_USED 0x00000002
```

10.13.1.7 TEE_MEMREF_2_USED

```
#define TEE_MEMREF_2_USED 0x00000004
```

10.13.1.8 TEE_MEMREF_3_USED

```
#define TEE_MEMREF_3_USED 0x00000008
```

10.13.1.9 TEE_SE_READER_NAME_MAX

```
#define TEE_SE_READER_NAME_MAX 20
```

10.13.2 Typedef Documentation

10.13.2.1 __aligned

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

10.13.2.2 nfds_t

```
typedef unsigned long int nfds_t
```

10.13.2.3 socklen_t

```
typedef unsigned int socklen_t
```

10.13.2.4 TEE_BigInt

```
typedef uint32_t TEE_BigInt
```

10.13.2.5 TEE_BigIntFMM

```
typedef uint32_t TEE_BigIntFMM
```

10.13.2.6 TEE_ErrorOrigin

```
typedef uint32_t TEE_ErrorOrigin
```

10.13.2.7 TEE_ObjectEnumHandle

```
typedef struct __TEE_ObjectEnumHandle* TEE_ObjectEnumHandle
```

10.13.2.8 TEE_ObjectHandle

```
typedef struct __TEE_ObjectHandle* TEE_ObjectHandle
```

10.13.2.9 TEE_ObjectType

```
typedef uint32_t TEE_ObjectType
```

10.13.2.10 TEE_OperationHandle

```
typedef struct __TEE_OperationHandle* TEE_OperationHandle
```

10.13.2.11 TEE_PropSetHandle

```
typedef struct __TEE_PropSetHandle* TEE_PropSetHandle
```

10.13.2.12 TEE_Result

```
typedef uint32_t TEE_Result
```

10.13.2.13 TEE_SEChannelHandle

```
typedef struct __TEE_SEChannelHandle* TEE_SEChannelHandle
```

10.13.2.14 TEE_SEReaderHandle

```
typedef struct __TEE_SEReaderHandle* TEE_SEReaderHandle
```

10.13.2.15 TEE_SEServiceHandle

```
typedef struct __TEE_SEServiceHandle* TEE_SEServiceHandle
```

10.13.2.16 TEE_SESessionHandle

```
typedef struct __TEE_SESessionHandle* TEE_SESessionHandle
```

10.13.2.17 TEE_Session

```
typedef void* TEE_Session
```

10.13.2.18 TEE_TASessionHandle

```
typedef struct __TEE_TASessionHandle* TEE_TASessionHandle
```

10.13.3 Enumeration Type Documentation

10.13.3.1 TEE_OperationMode

```
enum TEE_OperationMode
```

Enumerator

TEE_MODE_ENCRYPT	
TEE_MODE_DECRYPT	
TEE_MODE_SIGN	
TEE_MODE_VERIFY	
TEE_MODE_MAC	
TEE_MODE_DIGEST	
TEE_MODE_DERIVE	

10.13.3.2 TEE_Whence `enum TEE_Whence`

Enumerator

TEE_DATA_SEEK_SET	
TEE_DATA_SEEK_CUR	
TEE_DATA_SEEK_END	

10.14 tee_api_types.h

Go to the documentation of this file.

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

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

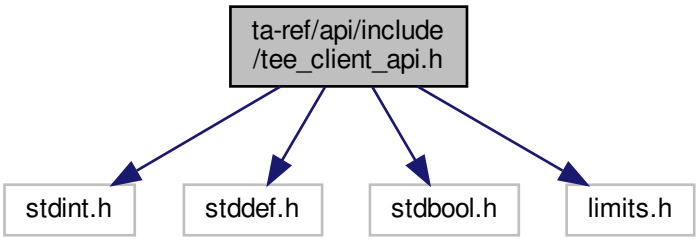
```
131         struct {
132             void *buffer;
133             uint32_t length;
134         } ref;
135         struct {
136             uint32_t a, b;
137         } value;
138     } content;
139 } TEE_Attribute;
140
141 #define DMREQ_FINISH 0
142 #define DMREQ_WRITE 1
143
144 /* Cryptographic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

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

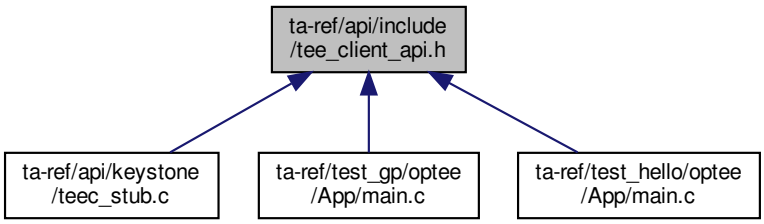
10.15 ta-ref/api/include/tee_client_api.h File Reference

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

Include dependency graph for tee_client_api.h:



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



Classes

- struct TEEC_Context
- struct TEEC_UUID
- struct TEEC_SharedMemory
- struct TEEC_TempMemoryReference
- struct TEEC_RegisteredMemoryReference
- struct TEEC_Value
- union TEEC_Parameter
- struct TEEC_Session
- struct TEEC_Operation

Macros

- #define TEEC_CONFIG_PAYLOAD_REF_COUNT 4
- #define TEEC_CONFIG_SHAREDMEM_MAX_SIZE ULONG_MAX
- #define TEEC_NONE 0x00000000
- #define TEEC_VALUE_INPUT 0x00000001
- #define TEEC_VALUE_OUTPUT 0x00000002
- #define TEEC_VALUE_INOUT 0x00000003
- #define TEEC_MEMREF_TEMP_INPUT 0x00000005
- #define TEEC_MEMREF_TEMP_OUTPUT 0x00000006
- #define TEEC_MEMREF_TEMP_INOUT 0x00000007
- #define TEEC_MEMREF_WHOLE 0x0000000C
- #define TEEC_MEMREF_PARTIAL_INPUT 0x0000000D
- #define TEEC_MEMREF_PARTIAL_OUTPUT 0x0000000E
- #define TEEC_MEMREF_PARTIAL_INOUT 0x0000000F
- #define TEEC_MEM_INPUT 0x00000001
- #define TEEC_MEM_OUTPUT 0x00000002
- #define TEEC_SUCCESS 0x00000000
- #define TEEC_ERROR_GENERIC 0xFFFF0000
- #define TEEC_ERROR_ACCESS_DENIED 0xFFFF0001
- #define TEEC_ERROR_CANCEL 0xFFFF0002
- #define TEEC_ERROR_ACCESS_CONFLICT 0xFFFF0003
- #define TEEC_ERROR_EXCESS_DATA 0xFFFF0004
- #define TEEC_ERROR_BAD_FORMAT 0xFFFF0005
- #define TEEC_ERROR_BAD_PARAMETERS 0xFFFF0006
- #define TEEC_ERROR_BAD_STATE 0xFFFF0007

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

Typedefs

- typedef uint32_t TEEC_Result

Functions

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

10.15.1 Macro Definition Documentation

10.15.1.1 TEEC_CONFIG_PAYLOAD_REF_COUNT `#define TEEC_CONFIG_PAYLOAD_REF_COUNT 4`

10.15.1.2 TEEC_CONFIG_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.

10.15.1.3 TEEC_ERROR_ACCESS_CONFLICT `#define TEEC_ERROR_ACCESS_CONFLICT 0xFFFF0003`

10.15.1.4 TEEC_ERROR_ACCESS_DENIED `#define TEEC_ERROR_ACCESS_DENIED 0xFFFF0001`

10.15.1.5 TEEC_ERROR_BAD_FORMAT `#define TEEC_ERROR_BAD_FORMAT 0xFFFF0005`

10.15.1.6 TEEC_ERROR_BAD_PARAMETERS `#define TEEC_ERROR_BAD_PARAMETERS 0xFFFF0006`

10.15.1.7 TEEC_ERROR_BAD_STATE `#define TEEC_ERROR_BAD_STATE 0xFFFF0007`

10.15.1.8 TEEC_ERROR_BUSY `#define TEEC_ERROR_BUSY 0xFFFF000D`

10.15.1.9 TEEC_ERROR_CANCEL `#define TEEC_ERROR_CANCEL 0xFFFF0002`

10.15.1.10 TEEC_ERROR_COMMUNICATION `#define TEEC_ERROR_COMMUNICATION 0xFFFF000E`

10.15.1.11 TEEC_ERROR_EXCESS_DATA `#define TEEC_ERROR_EXCESS_DATA 0xFFFF0004`

10.15.1.12 TEEC_ERROR_EXTERNAL_CANCEL `#define TEEC_ERROR_EXTERNAL_CANCEL 0xFFFF0011`

10.15.1.13 TEEC_ERROR_GENERIC

```
#define TEEC_ERROR_GENERIC 0xFFFF0000
```

10.15.1.14 TEEC_ERROR_ITEM_NOT_FOUND

```
#define TEEC_ERROR_ITEM_NOT_FOUND 0xFFFF0008
```

10.15.1.15 TEEC_ERROR_NO_DATA

```
#define TEEC_ERROR_NO_DATA 0xFFFF000B
```

10.15.1.16 TEEC_ERROR_NOT_IMPLEMENTED

```
#define TEEC_ERROR_NOT_IMPLEMENTED 0xFFFF0009
```

10.15.1.17 TEEC_ERROR_NOT_SUPPORTED

```
#define TEEC_ERROR_NOT_SUPPORTED 0xFFFF000A
```

10.15.1.18 TEEC_ERROR_OUT_OF_MEMORY

```
#define TEEC_ERROR_OUT_OF_MEMORY 0xFFFF000C
```

10.15.1.19 TEEC_ERROR_SECURITY

```
#define TEEC_ERROR_SECURITY 0xFFFF000F
```

10.15.1.20 TEEC_ERROR_SHORT_BUFFER

```
#define TEEC_ERROR_SHORT_BUFFER 0xFFFF0010
```

10.15.1.21 TEEC_ERROR_TARGET_DEAD

```
#define TEEC_ERROR_TARGET_DEAD 0xFFFF3024
```

10.15.1.22 TEEC_LOGIN_APPLICATION

```
#define TEEC_LOGIN_APPLICATION 0x00000004
```

10.15.1.23 TEEC_LOGIN_GROUP

```
#define TEEC_LOGIN_GROUP 0x00000002
```

10.15.1.24 TEEC_LOGIN_GROUP_APPLICATION `#define TEEC_LOGIN_GROUP_APPLICATION 0x00000006`

10.15.1.25 TEEC_LOGIN_PUBLIC `#define TEEC_LOGIN_PUBLIC 0x00000000`

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

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

10.15.1.26 TEEC_LOGIN_USER `#define TEEC_LOGIN_USER 0x00000001`

10.15.1.27 TEEC_LOGIN_USER_APPLICATION `#define TEEC_LOGIN_USER_APPLICATION 0x00000005`

10.15.1.28 TEEC_MEM_INPUT `#define TEEC_MEM_INPUT 0x00000001`

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

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

10.15.1.29 TEEC_MEM_OUTPUT `#define TEEC_MEM_OUTPUT 0x00000002`

10.15.1.30 TEEC_MEMREF_PARTIAL_INOUT `#define TEEC_MEMREF_PARTIAL_INOUT 0x0000000F`

10.15.1.31 TEEC_MEMREF_PARTIAL_INPUT `#define TEEC_MEMREF_PARTIAL_INPUT 0x0000000D`

10.15.1.32 TEEC_MEMREF_PARTIAL_OUTPUT `#define TEEC_MEMREF_PARTIAL_OUTPUT 0x0000000E`

10.15.1.33 TEEC_MEMREF_TEMP_INOUT `#define TEEC_MEMREF_TEMP_INOUT 0x00000007`

10.15.1.34 TEEC_MEMREF_TEMP_INPUT `#define TEEC_MEMREF_TEMP_INPUT 0x00000005`

10.15.1.35 TEEC_MEMREF_TEMP_OUTPUT `#define TEEC_MEMREF_TEMP_OUTPUT 0x00000006`

10.15.1.36 TEEC_MEMREF_WHOLE `#define TEEC_MEMREF_WHOLE 0x0000000C`

10.15.1.37 TEEC_NONE `#define TEEC_NONE 0x00000000`

Flag constants indicating the type of parameters encoded inside the operation payload (TEEC_Operation), Type is uint32_t.

TEEC_NONE The Parameter is not used

TEEC_VALUE_INPUT The Parameter is a TEEC_Value tagged as input.

TEEC_VALUE_OUTPUT The Parameter is a TEEC_Value tagged as output.

TEEC_VALUE_INOUT The Parameter is a TEEC_Value tagged as both as input and output, i.e., for which both the behaviors of TEEC_VALUE_INPUT and TEEC_VALUE_OUTPUT apply.

TEEC_MEMREF_TEMP_INPUT The Parameter is a TEEC_TempMemoryReference describing a region of memory which needs to be temporarily registered for the duration of the Operation and is tagged as input.

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

TEEC_MEMREF_TEMP_INOUT A Temporary Memory Reference tagged as both input and output, i.e., for which both the behaviors of TEEC_MEMREF_TEMP_INPUT and TEEC_MEMREF_TEMP_OUTPUT apply.

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

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

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

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

10.15.1.38 TEEC_ORIGIN_API `#define TEEC_ORIGIN_API 0x00000001`

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

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

10.15.1.39 TEEC_ORIGIN_COMMS `#define TEEC_ORIGIN_COMMS 0x00000002`

10.15.1.40 TEEC_ORIGIN_TEE `#define TEEC_ORIGIN_TEE 0x00000003`

10.15.1.41 TEEC_ORIGIN_TRUSTED_APP `#define TEEC_ORIGIN_TRUSTED_APP 0x00000004`

10.15.1.42 TEEC_PARAM_TYPE_GET

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

Get the i_th param type from the paramType.

Parameters

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

10.15.1.43 TEEC_PARAM_TYPES

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

Encode the paramTypes according to the supplied types.

Parameters

p0	The first param type.
p1	The second param type.
p2	The third param type.
p3	The fourth param type.

10.15.1.44 TEEC_SUCCESS `#define TEEC_SUCCESS 0x00000000`

Return values. Type is TEEC_Result

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

10.15.1.45 TEEC_VALUE_INOUT `#define TEEC_VALUE_INOUT 0x00000003`

10.15.1.46 TEEC_VALUE_INPUT `#define TEEC_VALUE_INPUT 0x00000001`

10.15.1.47 TEEC_VALUE_OUTPUT `#define TEEC_VALUE_OUTPUT 0x00000002`

10.15.2 Typedef Documentation

10.15.2.1 TEEC_Result `typedef uint32_t TEEC_Result`

10.15.3 Function Documentation

10.15.3.1 TEEC_AllocateSharedMemory()

```
TEEC_Result TEEC_AllocateSharedMemory (
        TEEC_Context * context,
        TEEC_SharedMemory * sharedMem )
```

TEEC_AllocateSharedMemory() - Allocate shared memory for TEE.

Parameters

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

Returns

TEEC_SUCCESS The registration was successful.

TEEC_ERROR_OUT_OF_MEMORY Memory exhaustion.

TEEC_Result Something failed.

10.15.3.2 TEEC_CloseSession() `void TEEC_CloseSession (`

`TEEC_Session * session )`

TEEC_CloseSession() - Closes the session which has been opened with the specific trusted application.

Parameters

session	The opened session to close.

10.15.3.3 TEEC_FinalizeContext() `void TEEC_FinalizeContext (`

`TEEC_Context * context )`

TEEC_FinalizeContext() - Destroys a context holding connection information on the specific TEE.

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

Parameters

context	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

context	The context to be finalized.

10.15.3.4 TEEC_InitializeContext()

```
TEEC_Result TEEC_InitializeContext (
    const char * name,
    TEEC_Context * context )
```

TEEC_InitializeContext() - Initializes a context holding connection information on the specific TEE, designated by the name string.

Parameters

name	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.
context	The context structure which is to be initialized.

Returns

TEEC_SUCCESS The initialization was successful.

TEEC_Result Something failed.

10.15.3.5 TEEC_InvokeCommand()

```
TEEC_Result TEEC_InvokeCommand (
    TEEC_Session * session,
    uint32_t commandID,
    TEEC_Operation * operation,
    uint32_t * returnOrigin )
```

TEEC_InvokeCommand() - Executes a command in the specified trusted application.

Parameters

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

Returns

TEEC_SUCCESS OpenSession successfully opened a new session.

TEEC_Result Something failed.

10.15.3.6 TEEC_OpenSession()

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

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

Parameters

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

Returns

TEEC_SUCCESS OpenSession successfully opened a new session.

TEEC_Result Something failed.

10.15.3.7 TEEC_RegisterSharedMemory()

```
TEEC_Result TEEC_RegisterSharedMemory (
    TEEC_Context * context,
    TEEC_SharedMemory * sharedMem )
```

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

Parameters

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

Returns

TEEC_SUCCESS The registration was successful.

TEEC_ERROR_OUT_OF_MEMORY Memory exhaustion.

TEEC_Result Something failed.

10.15.3.8 TEEC_ReleaseSharedMemory() `void TEEC_ReleaseSharedMemory ( TEEC_SharedMemory * sharedMemory )`

TEEC_ReleaseSharedMemory() - Free or deregister the shared memory.

Parameters

sharedMem	Pointer to the shared memory to be freed.

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

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

Parameters

operation	Pointer to an operation previously passed to open session or invoke.

10.16 tee_client_api.h

Go to the documentation of this file.

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

```
26  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
27  * POSSIBILITY OF SUCH DAMAGE.
28  */
29 #ifndef TEE_CLIENT_API_H
30 #define TEE_CLIENT_API_H
31
32 #ifdef __cplusplus
33 extern "C" {
34 #endif
35
36 #include <stdint.h>
37 #include <stddef.h>
38 #include <stdbool.h>
39 #include <limits.h>
40
41 /*
42  * Defines the number of available memory references in an open session or
43  * invoke command operation payload.
44  */
45 #define TEEC_CONFIG_PAYLOAD_REF_COUNT 4
46
53 #define TEEC_CONFIG_SHAREDMEM_MAX_SIZE ULONG_MAX
54
109 #define TEEC_NONE                    0x00000000
110 #define TEEC_VALUE_INPUT             0x00000001
111 #define TEEC_VALUE_OUTPUT            0x00000002
112 #define TEEC_VALUE_INOUT             0x00000003
113 #define TEEC_MEMREF_TEMP_INPUT       0x00000005
114 #define TEEC_MEMREF_TEMP_OUTPUT      0x00000006
115 #define TEEC_MEMREF_TEMP_INOUT       0x00000007
116 #define TEEC_MEMREF_WHOLE            0x0000000C
117 #define TEEC_MEMREF_PARTIAL_INPUT    0x0000000D
118 #define TEEC_MEMREF_PARTIAL_OUTPUT   0x0000000E
119 #define TEEC_MEMREF_PARTIAL_INOUT    0x0000000F
120
132 #define TEEC_MEM_INPUT    0x00000001
133 #define TEEC_MEM_OUTPUT   0x00000002
134
167 #define TEEC_SUCCESS                 0x00000000
168 #define TEEC_ERROR_GENERIC           0xFFFF0000
169 #define TEEC_ERROR_ACCESS_DENIED     0xFFFF0001
170 #define TEEC_ERROR_CANCEL            0xFFFF0002
171 #define TEEC_ERROR_ACCESS_CONFLICT   0xFFFF0003
172 #define TEEC_ERROR_EXCESS_DATA       0xFFFF0004
173 #define TEEC_ERROR_BAD_FORMAT        0xFFFF0005
174 #define TEEC_ERROR_BAD_PARAMETERS    0xFFFF0006
175 #define TEEC_ERROR_BAD_STATE         0xFFFF0007
176 #define TEEC_ERROR_ITEM_NOT_FOUND    0xFFFF0008
177 #define TEEC_ERROR_NOT_IMPLEMENTED   0xFFFF0009
178 #define TEEC_ERROR_NOT_SUPPORTED     0xFFFF000A
179 #define TEEC_ERROR_NO_DATA           0xFFFF000B
180 #define TEEC_ERROR_OUT_OF_MEMORY     0xFFFF000C
181 #define TEEC_ERROR_BUSY              0xFFFF000D
182 #define TEEC_ERROR_COMMUNICATION     0xFFFF000E
183 #define TEEC_ERROR_SECURITY          0xFFFF000F
184 #define TEEC_ERROR_SHORT_BUFFER      0xFFFF0010
185 #define TEEC_ERROR_EXTERNAL_CANCEL   0xFFFF0011
186 #define TEEC_ERROR_TARGET_DEAD       0xFFFF3024
187
201 #define TEEC_ORIGIN_API           0x00000001
202 #define TEEC_ORIGIN_COMMS         0x00000002
203 #define TEEC_ORIGIN_TEE           0x00000003
204 #define TEEC_ORIGIN_TRUSTED_APP   0x00000004
205
222 #define TEEC_LOGIN_PUBLIC         0x00000000
223 #define TEEC_LOGIN_USER           0x00000001
224 #define TEEC_LOGIN_GROUP          0x00000002
225 #define TEEC_LOGIN_APPLICATION    0x00000004
226 #define TEEC_LOGIN_USER_APPLICATION   0x00000005
227 #define TEEC_LOGIN_GROUP_APPLICATION  0x00000006
228
237 #define TEEC_PARAM_TYPES(p0, p1, p2, p3) \
238     ((p0) | ((p1) << 4) | ((p2) << 8) | ((p3) << 12))
239
246 #define TEEC_PARAM_TYPE_GET(p, i) (((p) >> (i * 4)) & 0xF)
247
248 typedef uint32_t TEEC_Result;
249
254 typedef struct {
255     /* Implementation defined */
256     int fd;
257     bool reg_mem;
258 } TEEC_Context;
259
265 typedef struct {
266     uint32_t timeLow;
267     uint16_t timeMid;
```

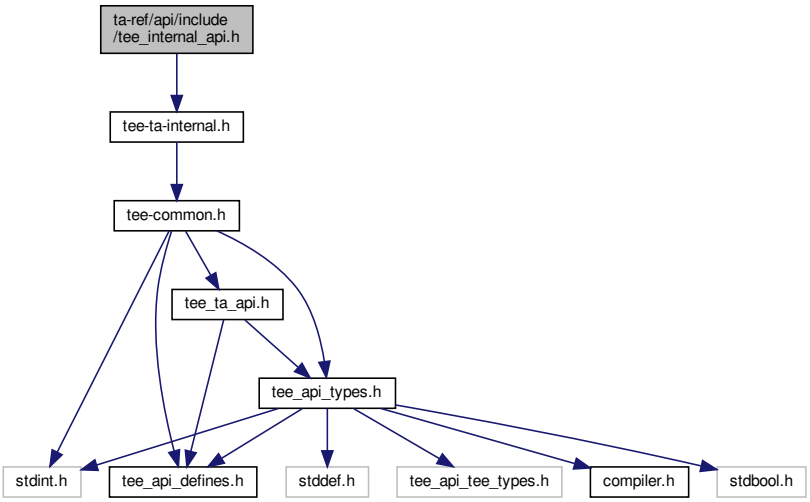
```
268     uint16_t timeHiAndVersion;
269     uint8_t clockSeqAndNode[8];
270 } TEEC_UUID;
271
288 typedef struct {
289     void *buffer;
290     size_t size;
291     uint32_t flags;
292     /*
293      * Implementation-Defined
294      */
295     int id;
296     size_t alloced_size;
297     void *shadow_buffer;
298     int registered_fd;
299     bool buffer_allocated;
300 } TEEC_SharedMemory;
301
314 typedef struct {
315     void *buffer;
316     size_t size;
317 } TEEC_TempMemoryReference;
318
334 typedef struct {
335     TEEC_SharedMemory *parent;
336     size_t size;
337     size_t offset;
338 } TEEC_RegisteredMemoryReference;
339
350 typedef struct {
351     uint32_t a;
352     uint32_t b;
353 } TEEC_Value;
354
369 typedef union {
370     TEEC_TempMemoryReference tmpref;
371     TEEC_RegisteredMemoryReference memref;
372     TEEC_Value value;
373 } TEEC_Parameter;
374
379 typedef struct {
380     /* Implementation defined */
381     TEEC_Context *ctx;
382     uint32_t session_id;
383 } TEEC_Session;
384
399 typedef struct {
400     uint32_t started;
401     uint32_t paramTypes;
402     TEEC_Parameter params[TEEC_CONFIG_PAYLOAD_REF_COUNT];
403     /* Implementation-Defined */
404     TEEC_Session *session;
405 } TEEC_Operation;
406
421 TEEC_Result TEEC_InitializeContext(const char *name, TEEC_Context *context);
422
434 void TEEC_FinalizeContext(TEEC_Context *context);
435
462 TEEC_Result TEEC_OpenSession(TEEC_Context *context,
463                 TEEC_Session *session,
464                 const TEEC_UUID *destination,
465                 uint32_t connectionMethod,
466                 const void *connectionData,
467                 TEEC_Operation *operation,
468                 uint32_t *returnOrigin);
469
476 void TEEC_CloseSession(TEEC_Session *session);
477
495 TEEC_Result TEEC_InvokeCommand(TEEC_Session *session,
496                   uint32_t commandID,
497                   TEEC_Operation *operation,
498                   uint32_t *returnOrigin);
499
512 TEEC_Result TEEC_RegisterSharedMemory(TEEC_Context *context,
513                       TEEC_SharedMemory *sharedMem);
514
526 TEEC_Result TEEC_AllocateSharedMemory(TEEC_Context *context,
527                       TEEC_SharedMemory *sharedMem);
528
534 void TEEC_ReleaseSharedMemory(TEEC_SharedMemory *sharedMemory);
535
543 void TEEC_RequestCancellation(TEEC_Operation *operation);
544
545 #ifdef __cplusplus
546 }
547 #endif
548
```

```
549 #endif
```

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

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

Include dependency graph for tee_internal_api.h:



10.18 tee_internal_api.h

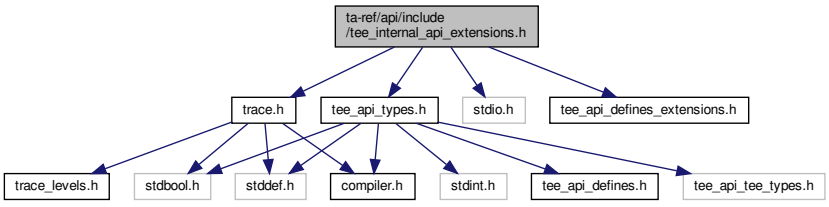
Go to the documentation of this file.

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

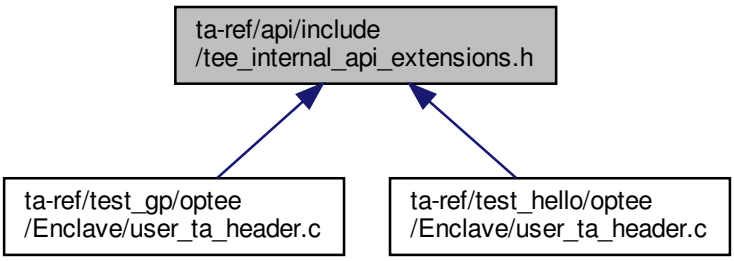
10.19 ta-ref/api/include/tee_internal_api_extensions.h File Reference

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

Include dependency graph for tee_internal_api_extensions.h:



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)

10.19.1 Macro Definition Documentation

10.19.1.1 TEE_USER_MEM_HINT_NO_FILL_ZERO `#define TEE_USER_MEM_HINT_NO_FILL_ZERO 0x80000000`

10.19.2 Function Documentation

10.19.2.1 TEE_CacheClean()

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

10.19.2.2 TEE_CacheFlush()

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

10.19.2.3 TEE_CacheInvalidate()

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

10.19.2.4 tee_map_zi()

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

10.19.2.5 tee_unmap()

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

10.19.2.6 tee_user_mem_check_heap()

```
size_t tee_user_mem_check_heap (
    void  )
```

10.19.2.7 tee_user_mem_mark_heap()

```
void tee_user_mem_mark_heap (
    void  )
```

10.19.2.8 tee_uuid_from_str()

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

10.20 tee_internal_api_extensions.h

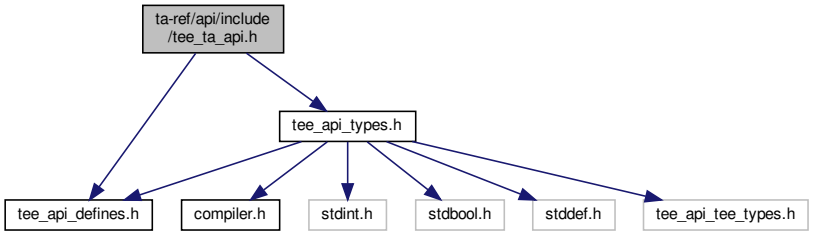
Go to the documentation of this file.

```
1 /* SPDX-License-Identifier: BSD-2-Clause */
2 /*
3  * Copyright (c) 2014, STMicroelectronics International N.V.
4  */
5
6 #ifndef TEE_INTERNAL_API_EXTENSIONS_H
7 #define TEE_INTERNAL_API_EXTENSIONS_H
8
9 /* trace support */
10 #include <trace.h>
11 #include <stdio.h>
12 #include <tee_api_defines_extensions.h>
13 #include <tee_api_types.h>
14
15 void tee_user_mem_mark_heap(void);
16 size_t tee_user_mem_check_heap(void);
17 /* Hint implementation defines */
18 #define TEE_USER_MEM_HINT_NO_FILL_ZERO       0x80000000
19
20 /*
21  * Cache maintenance support (TA requires the CACHE_MAINTENANCE property)
22  *
23  * TEE_CacheClean() Write back to memory any dirty data cache lines. The line
24  *                  is marked as not dirty. The valid bit is unchanged.
25  *
26  * TEE_CacheFlush() Purges any valid data cache lines. Any dirty cache lines
27  *                  are first written back to memory, then the cache line is
28  *                  invalidated.
29  *
30  * TEE_CacheInvalidate() Invalidate any valid data cache lines. Any dirty line
31  *                       are not written back to memory.
32  */
33 TEE_Result TEE_CacheClean(char *buf, size_t len);
34 TEE_Result TEE_CacheFlush(char *buf, size_t len);
35 TEE_Result TEE_CacheInvalidate(char *buf, size_t len);
36
37 /*
38  * tee_map_zi() - Map zero initialized memory
39  * @len:    Number of bytes
40  * @flags:  0 or TEE_MEMORY_ACCESS_ANY_OWNER to allow sharing with other TAs
41  *
42  * Returns valid pointer on success or NULL on error.
43  */
44 void *tee_map_zi(size_t len, uint32_t flags);
45
46 /*
47  * tee_unmap() - Unmap previously mapped memory
48  * @buf:    Buffer
49  * @len:    Number of bytes
50  *
51  * Note that supplied @buf and @len has to match exactly what has
52  * previously been returned by tee_map_zi().
53  *
54  * Return TEE_SUCCESS on success or TEE_ERRROR_* on failure.
55  */
56 TEE_Result tee_unmap(void *buf, size_t len);
57
58 /*
59  * Convert a UUID string @s into a TEE_UUID @uuid
60  * Expected format for @s is: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
61  * 'x' being any hexadecimal digit (0-9a-fA-F)
62  */
63 TEE_Result tee_uuid_from_str(TEE_UUID *uuid, const char *s);
64
65 #endif
```

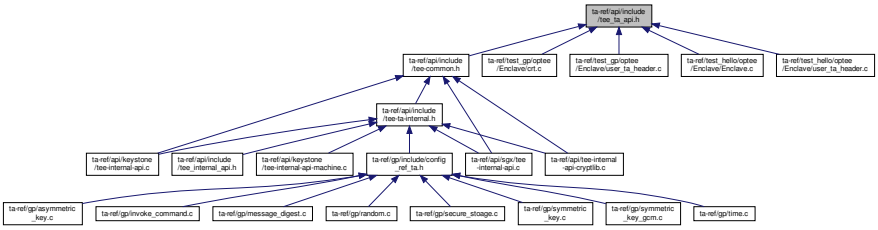
10.21 ta-ref/api/include/tee_ta_api.h File Reference

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

Include dependency graph for tee_ta_api.h:



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



Macros

- #define TA_EXPORT

Functions

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

10.21.1 Macro Definition Documentation

10.21.1.1 TA_EXPORT `#define TA_EXPORT`

10.21.2 Function Documentation

10.21.2.1 TA_CloseSessionEntryPoint()

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

10.21.2.2 TA_CreateEntryPoint()

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

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 occured.

10.21.2.3 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".

10.21.2.4 TA_InvokeCommandEntryPoint()

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

10.21.2.5 TA_OpenSessionEntryPoint()

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

10.22 tee_ta_api.h

Go to the documentation of this file.

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

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

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

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

Variables

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

10.23.1 Variable Documentation

10.23.1.1 _sanctum_dev_public_key `const unsigned char _sanctum_dev_public_key[ ] [static]`

Initial value:

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

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

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

Initial value:

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

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

10.24 test_dev_key.h

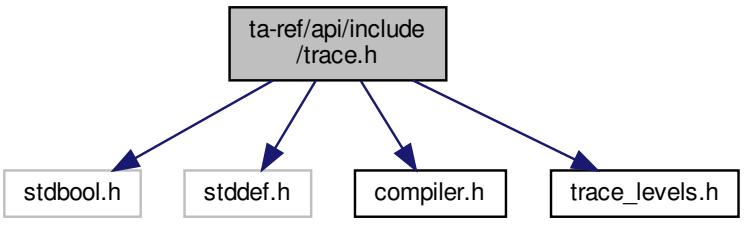
Go to the documentation of this file.

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

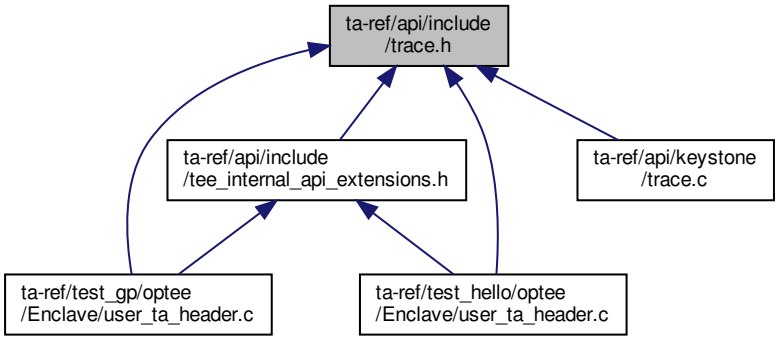
10.25 ta-ref/api/include/trace.h File Reference

```
#include <stdbool.h>
#include <stddef.h>
#include <compiler.h>
#include <trace_levels.h>
```

Include dependency graph for trace.h:



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



Macros

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

Functions

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

Variables

- int trace_level
- const char trace_ext_prefix [ ]

10.25.1 Macro Definition Documentation

10.25.1.1 DHEXDUMP

```
#define DHEXDUMP(
    buf,
    len )
```

Value:

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

10.25.1.2 DMSG

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

10.25.1.3 DMSG_RAW

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

10.25.1.4 DPRINT_STACK

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

10.25.1.5 EMSG

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

10.25.1.6 EMSG_RAW

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

10.25.1.7 EPRINT_STACK

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

10.25.1.8 FMSG

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

10.25.1.9 FMSG_RAW

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

10.25.1.10 FPRINT_STACK

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

10.25.1.11 IMSG

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

10.25.1.12 IMSG_RAW

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

10.25.1.13 INMSG

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

10.25.1.14 IPRINT_STACK

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

10.25.1.15 MAX_FUNC_PRINT_SIZE

```
#define MAX_FUNC_PRINT_SIZE 32
```

10.25.1.16 MAX_PRINT_SIZE

```
#define MAX_PRINT_SIZE 256
```

10.25.1.17 MSG

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

10.25.1.18 MSG_RAW #define MSG_RAW(

```
... ) (void)0
```

10.25.1.19 OUTMSG #define OUTMSG(

```
... ) FMSG("< " __VA_ARGS__)
```

10.25.1.20 OUTRMSG #define OUTRMSG(

```
r )
```

Value:

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

10.25.1.21 SMSG #define SMSG(

```
... ) (void)0
```

10.25.1.22 TRACE_LEVEL #define TRACE_LEVEL TRACE_MAX

10.25.1.23 trace_printf_helper #define trace_printf_helper(

```
level,
level_ok,
... )
```

Value:

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

10.25.1.24 trace_printf_helper_raw #define trace_printf_helper_raw(

```
level,
level_ok,
... )   trace_printf(NULL, 0, (level), (level_ok), __VA_ARGS__)
```

10.25.2 Function Documentation

10.25.2.1 dhex_dump()

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

10.25.2.2 trace_ext_get_thread_id()

```
int trace_ext_get_thread_id (
    void  )
```

10.25.2.3 trace_ext_puts()

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

10.25.2.4 trace_get_level()

```
int trace_get_level (
    void  )
```

10.25.2.5 trace_printf()

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

10.25.2.6 trace_set_level()

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

10.25.3 Variable Documentation

10.25.3.1 trace_ext_prefix

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

10.25.3.2 trace_level

```
int trace_level  [extern]
```

10.26 trace.h

Go to the documentation of this file.

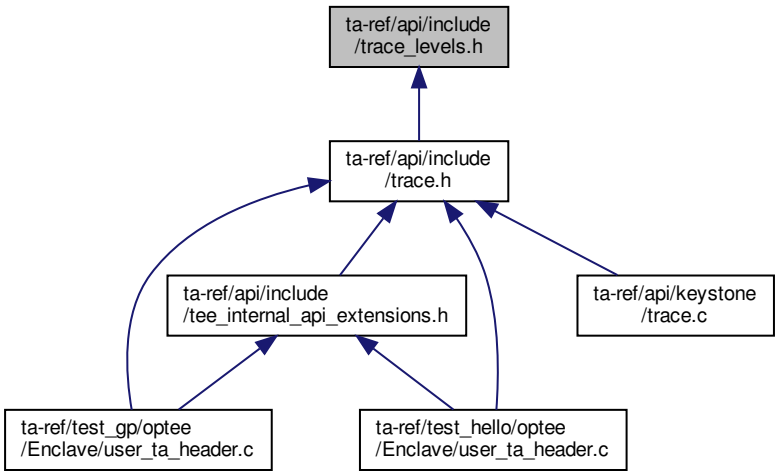
```
1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10  *
11  * 2. Redistributions in binary form must reproduce the above copyright notice,
12  * this list of conditions and the following disclaimer in the documentation
13  * and/or other materials provided with the distribution.
14  *
15  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25  * POSSIBILITY OF SUCH DAMAGE.
26  */
27 #ifndef TRACE_H
28 #define TRACE_H
29
30 #include <stdbool.h>
31 #include <stddef.h>
32 #include <compiler.h>
33 #include <trace_levels.h>
34
35 #define MAX_PRINT_SIZE      256
36 #define MAX_FUNC_PRINT_SIZE 32
37
38 #ifndef TRACE_LEVEL
39 #define TRACE_LEVEL TRACE_MAX
40 #endif
41
42 /*
43  * Symbols provided by the entity that uses this API.
44  */
45 extern int trace_level;
46 extern const char trace_ext_prefix[];
47 void trace_ext_puts(const char *str);
48 int trace_ext_get_thread_id(void);
49 void trace_set_level(int level);
50 int trace_get_level(void);
51
52 /* Internal functions used by the macros below */
53 void trace_printf(const char *func, int line, int level, bool level_ok,
54           const char *fmt, ...) __printf(5, 6);
55
56 #define trace_printf_helper(level, level_ok, ...) \
57     trace_printf(__func__, __LINE__, (level), (level_ok), \
58              __VA_ARGS__)
59
60 /* Formatted trace tagged with level independent */
61 #if (TRACE_LEVEL <= 0)
62 #define MSG(...)    (void)0
63 #else
64 #define MSG(...)    trace_printf_helper(0, false, __VA_ARGS__)
65 #endif
66
67 /* Formatted trace tagged with TRACE_ERROR level */
68 #if (TRACE_LEVEL < TRACE_ERROR)
69 #define EMSG(...)    (void)0
70 #else
71 #define EMSG(...)    trace_printf_helper(TRACE_ERROR, true, __VA_ARGS__)
72 #endif
73
74 /* Formatted trace tagged with TRACE_INFO level */
75 #if (TRACE_LEVEL < TRACE_INFO)
76 #define IMSG(...)    (void)0
77 #else
78 #define IMSG(...)    trace_printf_helper(TRACE_INFO, true, __VA_ARGS__)
79 #endif
80
81 /* Formatted trace tagged with TRACE_DEBUG level */
82 #if (TRACE_LEVEL < TRACE_DEBUG)
83 #define DMSG(...)    (void)0
```

```
84 #else
85 #define DMSG(...)   trace_printf_helper(TRACE_DEBUG, true, __VA_ARGS__)
86 #endif
87
88 /* Formatted trace tagged with TRACE_FLOW level */
89 #if (TRACE_LEVEL < TRACE_FLOW)
90 #define FMSG(...)   (void)0
91 #else
92 #define FMSG(...)   trace_printf_helper(TRACE_FLOW, true, __VA_ARGS__)
93 #endif
94
95 /* Formatted trace tagged with TRACE_FLOW level and prefix with '> ' */
96 #define INMSG(...)     FMSG("> " __VA_ARGS__)
97 /* Formatted trace tagged with TRACE_FLOW level and prefix with '< ' */
98 #define OUTMSG(...)    FMSG("< " __VA_ARGS__)
99 /* Formatted trace tagged with TRACE_FLOW level and prefix with '< ' and print
100  * an error message if r != 0 */
101 #define OUTRMSG(r)          \
102     do {                    \
103         OUTMSG("r=[%x]", r);    \
104         return r;           \
105     } while (0)
106
107 void dhex_dump(const char *function, int line, int level,
108            const void *buf, int len);
109 #if (TRACE_LEVEL < TRACE_DEBUG)
110 #define DHEXDUMP(buf, len) (void)0
111 #else
112 #define DHEXDUMP(buf, len) dhex_dump(__func__, __LINE__, TRACE_DEBUG, \
113                      buf, len)
114 #endif
115
116
117 /* Trace api without trace formatting */
118
119 #define trace_printf_helper_raw(level, level_ok, ...) \
120     trace_printf(NULL, 0, (level), (level_ok), __VA_ARGS__)
121
122 /* No formatted trace tagged with level independent */
123 #if (TRACE_LEVEL <= 0)
124 #define MSG_RAW(...)    (void)0
125 #else
126 #define MSG_RAW(...)    trace_printf_helper_raw(0, false, __VA_ARGS__)
127 #endif
128
129 /* No formatted trace tagged with TRACE_ERROR level */
130 #if (TRACE_LEVEL < TRACE_ERROR)
131 #define EMSG_RAW(...)    (void)0
132 #else
133 #define EMSG_RAW(...)    trace_printf_helper_raw(TRACE_ERROR, true, __VA_ARGS__)
134 #endif
135
136 /* No formatted trace tagged with TRACE_INFO level */
137 #if (TRACE_LEVEL < TRACE_INFO)
138 #define IMSG_RAW(...)    (void)0
139 #else
140 #define IMSG_RAW(...)    trace_printf_helper_raw(TRACE_INFO, true, __VA_ARGS__)
141 #endif
142
143 /* No formatted trace tagged with TRACE_DEBUG level */
144 #if (TRACE_LEVEL < TRACE_DEBUG)
145 #define DMSG_RAW(...)    (void)0
146 #else
147 #define DMSG_RAW(...)    trace_printf_helper_raw(TRACE_DEBUG, true, __VA_ARGS__)
148 #endif
149
150 /* No formatted trace tagged with TRACE_FLOW level */
151 #if (TRACE_LEVEL < TRACE_FLOW)
152 #define FMSG_RAW(...)    (void)0
153 #else
154 #define FMSG_RAW(...)    trace_printf_helper_raw(TRACE_FLOW, true, __VA_ARGS__)
155 #endif
156
157 #if (TRACE_LEVEL <= 0)
158 #define SMSG(...)    (void)0
159 #else
160 /*
161  * Synchronised flushed trace, an Always message straight to HW trace IP.
162  * Current only supported inside OP-TEE kernel, will be just like an EMSG()
163  * in another context.
164  */
165 #define SMSG(...)    \
166     trace_printf(__func__, __LINE__, TRACE_ERROR, true, __VA_ARGS__)
167
168 #endif /* TRACE_LEVEL */
169
170 #if defined(__KERNEL__) && defined(CFG_UNWIND)
```

```
171 #include <kernel/unwind.h>
172 #define _PRINT_STACK
173 #endif
174
175 #if defined(_PRINT_STACK) && (TRACE_LEVEL >= TRACE_ERROR)
176 #define EPRINT_STACK() print_kernel_stack(TRACE_ERROR)
177 #else
178 #define EPRINT_STACK() (void)0
179 #endif
180
181 #if defined(_PRINT_STACK) && (TRACE_LEVEL >= TRACE_INFO)
182 #define IPRINT_STACK() print_kernel_stack(TRACE_INFO)
183 #else
184 #define IPRINT_STACK() (void)0
185 #endif
186
187 #if defined(_PRINT_STACK) && (TRACE_LEVEL >= TRACE_DEBUG)
188 #define DPRINT_STACK() print_kernel_stack(TRACE_DEBUG)
189 #else
190 #define DPRINT_STACK() (void)0
191 #endif
192
193 #if defined(_PRINT_STACK) && (TRACE_LEVEL >= TRACE_FLOW)
194 #define FPRINT_STACK() print_kernel_stack(TRACE_FLOW)
195 #else
196 #define FPRINT_STACK() (void)0
197 #endif
198
199 #if defined(__KERNEL__) && defined(CFG_UNWIND)
200 #undef _PRINT_STACK
201 #endif
202
203 #endif /* TRACE_H */
```

10.27 ta-ref/api/include/trace_levels.h File Reference

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



Macros

- #define TRACE_MIN 1
- #define TRACE_ERROR TRACE_MIN
- #define TRACE_INFO 2
- #define TRACE_DEBUG 3
- #define TRACE_FLOW 4
- #define TRACE_MAX TRACE_FLOW
- #define TRACE_PRINTF_LEVEL TRACE_ERROR

10.27.1 Macro Definition Documentation

10.27.1.1 TRACE_DEBUG `#define TRACE_DEBUG 3`

10.27.1.2 TRACE_ERROR `#define TRACE_ERROR TRACE_MIN`

10.27.1.3 TRACE_FLOW `#define TRACE_FLOW 4`

10.27.1.4 TRACE_INFO `#define TRACE_INFO 2`

10.27.1.5 TRACE_MAX `#define TRACE_MAX TRACE_FLOW`

10.27.1.6 TRACE_MIN `#define TRACE_MIN 1`

10.27.1.7 TRACE_PRINTF_LEVEL `#define TRACE_PRINTF_LEVEL TRACE_ERROR`

10.28 trace_levels.h

Go to the documentation of this file.

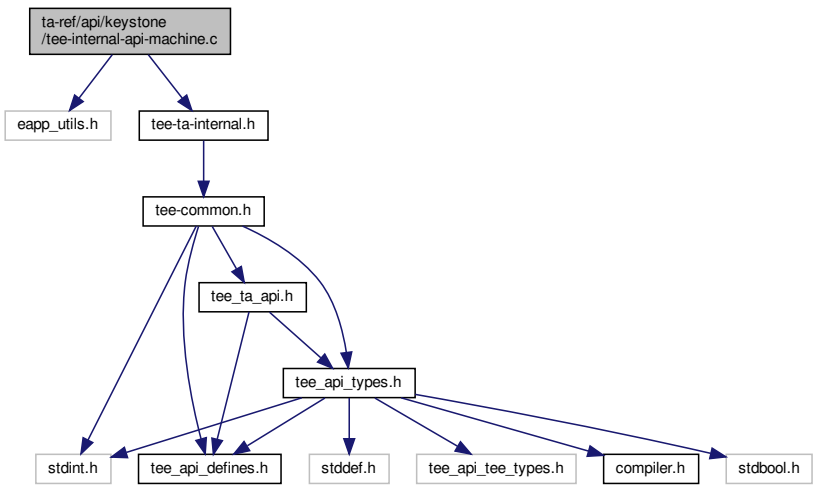
```
1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must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

```
23  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25  * POSSIBILITY OF SUCH DAMAGE.
26  */
27 #ifndef TRACE_LEVELS_H
28 #define TRACE_LEVELS_H
29
30 /*
31  * Trace levels.
32  *
33  * ALWAYS is used when you always want a print to be seen, but it is not always
34  * an error.
35  *
36  * ERROR is used when some kind of error has happened, this is most likely the
37  * print you will use most of the time when you report some kind of error.
38  *
39  * INFO is used when you want to print some 'normal' text to the user.
40  * This is the default level.
41  *
42  * DEBUG is used to print extra information to enter deeply in the module.
43  *
44  * FLOW is used to print the execution flox, typically the in/out of functions.
45  *
46  */
47
48 #define TRACE_MIN       1
49 #define TRACE_ERROR     TRACE_MIN
50 #define TRACE_INFO      2
51 #define TRACE_DEBUG     3
52 #define TRACE_FLOW      4
53 #define TRACE_MAX       TRACE_FLOW
54
55 /* Trace level of the casual printf */
56 #define TRACE_PRINTF_LEVEL TRACE_ERROR
57
58 #endif /*TRACE_LEVELS_H*/
```

10.29 ta-ref/api/keystone/tee-internal-api-machine.c File Reference

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

Include dependency graph for tee-internal-api-machine.c:



Functions

- void __attribute__ ((noreturn))

10.29.1 Function Documentation

10.29.1.1 __attribute__() void __attribute__ ((noreturn))

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

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

Parameters

code	An informative panic code defined by the TA.

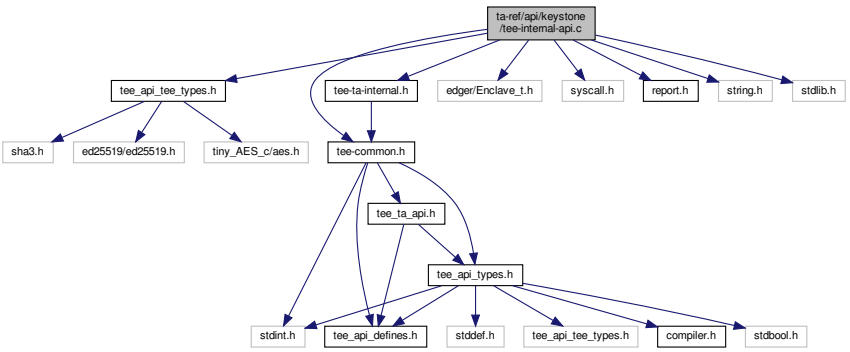
Returns

panic code will be returned.

10.30 ta-ref/api/keystone/tee-internal-api.c File Reference

```
#include "tee_api_tee_types.h"
#include "tee-common.h"
#include "tee-ta-internal.h"
#include "edger/Enclave_t.h"
#include "syscall.h"
#include "report.h"
#include <string.h>
#include <stdlib.h>
```

Include dependency graph for tee-internal-api.c:



Macros

- #define O_RDONLY 0
- #define O_WRONLY 00001
- #define O_RDWR 00002
- #define O_CREAT 00100
- #define O_EXCL 00200
- #define O_TRUNC 01000
- #define FPERMS 0600

Functions

- void * TEE_Malloc (uint32_t size, uint32_t hint)
- void * TEE_Realloc (void *buffer, uint32_t newSize)
- void TEE_Free (void *buffer)
- void TEE_GetREETime (TEE_Time *time)

  Core Functions, Time Functions.
- void TEE_GetSystemTime (TEE_Time *time)

  Core Functions, Time Functions.
- TEE_Result GetRelTimeStart (uint64_t start)

  Core Functions, Time Functions.
- TEE_Result GetRelTimeEnd (uint64_t end)

  Core Functions, Time Functions.
- static int flags2flags (int flags)
- static int set_object_key (void *id, unsigned int idlen, TEE_ObjectHandle object)
- static TEE_Result OpenPersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle *object, int ocreat)
- TEE_Result TEE_CreatePersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle attributes, const void *initialData, uint32_t initialDataLen, TEE_ObjectHandle *object)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- TEE_Result TEE_OpenPersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle *object)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- TEE_Result TEE_GetObjectInfo1 (TEE_ObjectHandle object, TEE_ObjectInfo *objectInfo)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- TEE_Result TEE_WriteObjectData (TEE_ObjectHandle object, const void *buffer, uint32_t size)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- TEE_Result TEE_ReadObjectData (TEE_ObjectHandle object, void *buffer, uint32_t size, uint32_t *count)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- void TEE_CloseObject (TEE_ObjectHandle object)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- void TEE_GenerateRandom (void *randomBuffer, uint32_t randomBufferLen)

  Crypto, common.

10.30.1 Macro Definition Documentation

10.30.1.1 FPERMS `#define FPERMS 0600`

10.30.1.2 O_CREAT `#define O_CREAT 00100`

10.30.1.3 O_EXCL `#define O_EXCL 00200`

10.30.1.4 O_RDONLY `#define O_RDONLY 0`

10.30.1.5 O_RDWR `#define O_RDWR 00002`

10.30.1.6 O_TRUNC `#define O_TRUNC 01000`

10.30.1.7 O_WRONLY `#define O_WRONLY 00001`

10.30.2 Function Documentation

10.30.2.1 flags2flags() `static int flags2flags ( int flags ) [inline], [static]`

flags2flags() - Checks the status for reading or writing of the file operational.

This function is used to check the status for reading or writing of the file operational.

Parameters

flags	Flags of the referencing node.

Returns

ret if success.

10.30.2.2 GetRelTimeEnd() TEE_Result GetRelTimeEnd (uint64_t *end*)

Core Functions, Time Functions.

GetRelTimeEnd() - finds the real time of the end timing.

This function prints the ending time.

Parameters

end	End timing

Returns

0 If success

10.30.2.3 GetRelTimeStart() TEE_Result GetRelTimeStart (uint64_t *start*)

Core Functions, Time Functions.

GetRelTimeStart() - Gets the real time of the start timing.

This function prints the starting time.

Parameters

start	Start timing

Returns

0 on success

10.30.2.4 OpenPersistentObject() static TEE_Result OpenPersistentObject (uint32_t *storageID*, const void * *objectID*, uint32_t *objectIDLen*, uint32_t *flags*,

```
TEE_ObjectHandle * object,
int ocreat )  [static]
```

OpenPersistentObject() - Opens a handle on an existing persistent object.

The flags parameter is a set of flags that controls the access rights and sharing permissions with which the object handle is opened. The value of the flags parameter is constructed by a bitwise-inclusive OR of flags TEE_DATA_↩ FLAG_ACCESS_READ, the object is opened with the read access right. This allows the Trusted Application to call the function TEE_ReadObjectData. TEE_DATA_FLAG_ACCESS_WRITE, the object is opened with the write access right. TEE_DATA_FLAG_ACCESS_WRITE_META, the object is opened with the write-meta access right.

Parameters

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

Returns

0 if success else error occured.

10.30.2.5 set_object_key()

```
static int set_object_key (
void * id,
unsigned int idlen,
TEE_ObjectHandle object )  [static]
```

set_object_key() - Initialize report and then attest enclave with file.

This function describes the intialization 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

id	id of the object.
idlen	length of the id.
object	TEE_ObjectHandle type handle.

Returns

ret if success.

10.30.2.6 TEE_CloseObject() `void TEE_CloseObject (`

```
TEE_ObjectHandle object )
```

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

TEE_CloseObject() - Closes an opened object handle.

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

Parameters

object	Handle of the object.

Returns

TEE_SUCCESS if success else error occured.

10.30.2.7 TEE_CreatePersistentObject() `TEE_Result TEE_CreatePersistentObject (`

```
uint32_t storageID,
const void * objectID,
uint32_t objectIDLen,
uint32_t flags,
TEE_ObjectHandle attributes,
const void * initialData,
uint32_t initialDataLen,
TEE_ObjectHandle * object )
```

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

TEE_CreatePersistentObject() - Creates a persistent object with initial attributes.

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

Parameters

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

Returns

0 if success else error occured.

10.30.2.8 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

buffer	The pointer to the memory block to be freed.

10.30.2.9 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

buf	character type buffer
len	size of the buffer
flags	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

randomBuffer	Reference to generated random data
randomBufferLen	Byte length of requested random data

Returns

ocall version random data

10.30.2.10 TEE_GetObjectInfo1() `TEE_Result TEE_GetObjectInfo1 ( TEE_ObjectHandle object, TEE_ObjectInfo * objectInfo )`

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

TEE_GetObjectInfo1() - Returns the characteristics of an object.

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

Parameters

objectInfo	Pointer to a structure filled with the object information
object	Handle of the object

Returns

0 if success else error occured.

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

Core Functions, Time Functions.

TEE_GetREETime() - Retrieves the current REE system time.

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

Parameters

time	Filled with the number of seconds and milliseconds

10.30.2.12 TEE_GetSystemTime() void TEE_GetSystemTime (TEE_Time * time)

Core Functions, Time Functions.

TEE_GetSystemTime() - Retrieves the current system time.

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

Parameters

time	Filled with the number of seconds and milliseconds

10.30.2.13 TEE_Malloc() void * TEE_Malloc (uint32_t size, uint32_t hint)

TEE_Malloc() - Allocates space for an object whose size in bytes is specified in the parameter size.

This function describes the pointer returned is guaranteed to be aligned
such that it may be assigned as a pointer to any basic C type.The valid hint values are a bitmask and can be independently set. This parameter allows Trusted Applications to refer to various pools of memory or to
request special characteristics for the allocated memory by using an
implementation-defined hint. Future versions of this specification may introduce additional standard hints.

Parameters

size	The size of the buffer to be allocated.
hint	A hint to the allocator.

Returns

Upon successful completion, with size not equal to zero, the function returns a pointer to the allocated space.

10.30.2.14 TEE_OpenPersistentObject() TEE_Result TEE_OpenPersistentObject (uint32_t storageID, const void * objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle * object)

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

TEE_OpenPersistentObject() - Opens a handle on an existing persistent object.

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

Parameters

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

Returns

0 if success else error occured.

10.30.2.15 TEE_ReadObjectData()

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

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

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

The bytes are read starting at the position in the data stream currently stored in the object handle. The handle's position is incremented by the number of bytes actually read. On completion of TEE_ReadObjectData sets the number of bytes actually read in the "uint32_t" pointed to by count. The value written to ∗count may be less than size if the number of bytes until the end-of3067 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

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

Returns

TEE_SUCCESS if success else error occured.

10.30.2.16 TEE_Realloc()

```
void * TEE_Realloc (
    void * buffer,
    uint32_t newSize )
```

TEE_Realloc() - Changes the size of the memory object pointed to by buffer to the size specified by new size.

This function describes the content of the object remains unchanged up to the lesser of the new and old sizes. Space in excess of the old size contains unspecified content. If the new size of the memory object requires movement of the object, the space for the previous instantiation of the object is deallocated. If the space cannot be allocated, the original object remains allocated, and this function returns a NULL pointer.

Parameters

buffer	The pointer to the object to be reallocated.
newSize	The new size required for the object

Returns

Upon successful completion, TEE_Realloc returns a pointer to the (possibly moved) allocated space. If there is not enough available memory, TEE_Realloc returns a NULL pointer and the original buffer is still allocated and unchanged.

10.30.2.17 TEE_WriteObjectData()

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

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

TEE_WriteObjectData() - Writes the buffer data in to persistent objects.

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

Parameters

object	Handle of the object
buffer	The buffer containing the data to be written
size	The number of bytes to write

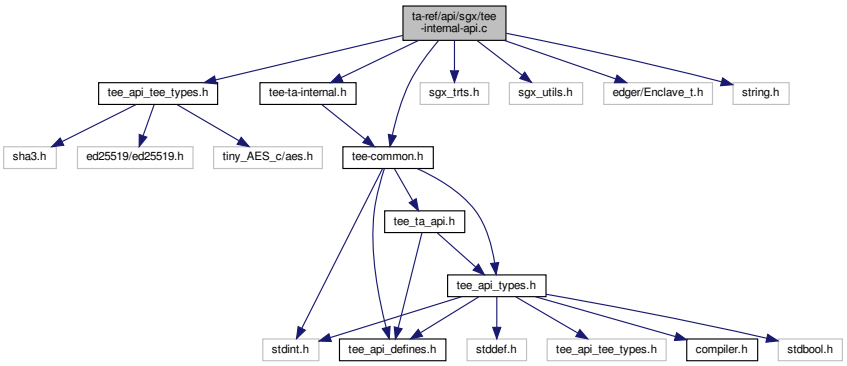
Returns

TEE_SUCCESS if success else error occured.

10.31 ta-ref/api/sgx/tee-internal-api.c File Reference

```
#include "tee_api_tee_types.h"
#include "tee-common.h"
#include "tee-ta-internal.h"
#include "sgx_trts.h"
#include "sgx_utils.h"
#include "edger/Enclave_t.h"
#include <string.h>
```

Include dependency graph for tee-internal-api.c:



Macros

- #define O_RDONLY 0
- #define O_WRONLY 00001
- #define O_RDWR 00002
- #define O_CREAT 00100
- #define O_EXCL 00200
- #define O_TRUNC 01000
- #define FPERMS 0600

Functions

- void __attribute__ ((noreturn))
- void TEE_GetREETime (TEE_Time *time)

  Core Functions, Time Functions.
- void TEE_GetSystemTime (TEE_Time *time)

  Core Functions, Time Functions.
- TEE_Result GetRelTimeStart (uint64_t start)

  Core Functions, Time Functions.
- TEE_Result GetRelTimeEnd (uint64_t end)

  Core Functions, Time Functions.
- static int flags2flags (int flags)

- static int set_object_key (const void *id, unsigned int idlen, TEE_ObjectHandle object)
- static TEE_Result OpenPersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle *object, int ocreat)
- TEE_Result TEE_CreatePersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle attributes, const void *initialData, uint32_t initialDataLen, TEE_ObjectHandle *object)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- TEE_Result TEE_OpenPersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle *object)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- TEE_Result TEE_GetObjectInfo1 (TEE_ObjectHandle object, TEE_ObjectInfo *objectInfo)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- TEE_Result TEE_WriteObjectData (TEE_ObjectHandle object, const void *buffer, uint32_t size)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- TEE_Result TEE_ReadObjectData (TEE_ObjectHandle object, void *buffer, uint32_t size, uint32_t *count)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- void TEE_CloseObject (TEE_ObjectHandle object)

  Core Functions, Secure Storage Functions (data is isolated for each TA)
- void TEE_GenerateRandom (void *randomBuffer, uint32_t randomBufferLen)

  Crypto, common.

10.31.1 Macro Definition Documentation

10.31.1.1 FPERMS `#define FPERMS 0600`

10.31.1.2 O_CREAT `#define O_CREAT 00100`

10.31.1.3 O_EXCL `#define O_EXCL 00200`

10.31.1.4 O_RDONLY `#define O_RDONLY 0`

10.31.1.5 O_RDWR `#define O_RDWR 00002`

10.31.1.6 O_TRUNC `#define O_TRUNC 01000`

10.31.1.7 O_WRONLY `#define O_WRONLY 00001`

10.31.2 Function Documentation

10.31.2.1 __attribute__() `void __attribute__ ( (noreturn) )`

TEE_Panic() - Raises a Panic in the Trusted Application instance

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

Parameters

ec	An informative panic code defined by the TA. May be displayed in traces if traces are available.

10.31.2.2 flags2flags() `static int flags2flags ( int flags ) [inline], [static]`

flags2flags() - Checks the status for reading or writing of the file operational.

This function is to check the status for reading or writing of the file operational.

Parameters

flags	Flags of the referencing node.

Returns

0 if success else error occured.

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

Core Functions, Time Functions.

GetRelTimeStart() - find the real time of the end timing.

This function prints the End timing.

Parameters

end	End timing

Returns

0 if success else error occured

10.31.2.4 GetRelTimeStart()

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

Core Functions, Time Functions.

GetRelTimeStart() - Gets the real time of the start timing.

Ths function prints the start timing.

Parameters

start	start timing

Returns

0 if success else error occured.

10.31.2.5 OpenPersistentObject()

```
static TEE_Result OpenPersistentObject (
    uint32_t storageID,
    const void * objectID,
    uint32_t objectIDLen,
    uint32_t flags,
    TEE_ObjectHandle * object,
    int ocreat )  [static]
```

OpenPersistentObject() - Opens a handle on an existing persistent object.

The flags parameter is a set of flags that controls the access rights and sharing permissions with which the object handle is opened. The value of the flags parameter is constructed by a bitwise-inclusive OR of flags TEE_DATA_↩ FLAG_ACCESS_READ, the object is opened with the read access right. This allows the Trusted Application to call the function TEE_ReadObjectData. TEE_DATA_FLAG_ACCESS_WRITE, the object is opened with the write access right. TEE_DATA_FLAG_ACCESS_WRITE_META, the object is opened with the write-meta access right.

Parameters

storageID	The storage to use.

Paramter list continued on next page

objectID	The object identifier
objectIDLen	length of the identifier
flags	The flags which determine the settings under which the object is opened.
object	A pointer to the handle, which contains the opened handle upon successful completion.

Returns

0 if success else error occured.

10.31.2.6 set_object_key()

```
static int set_object_key (
    const void * id,
    unsigned int idlen,
    TEE_ObjectHandle object )  [static]
```

set_object_key - To initalize 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

id	id of the object.
idlen	length of the id.
object	TEE_ObjectHandle type handle.

Returns

0 if success else error occured.

10.31.2.7 TEE_CloseObject()

```
void TEE_CloseObject (
    TEE_ObjectHandle object )
```

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

TEE_CloseObject() - Function closes an opened object handle.

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

Parameters

object	Handle of the object

Returns

TEE_SUCCESS if success else error occured.

10.31.2.8 TEE_CreatePersistentObject()

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

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

TEE_CreatePersistentObject() - Creates a persistent object with initial attributes.

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

Parameters

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

Returns

0 if success, else error occured.

10.31.2.9 TEE_GenerateRandom()

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

Crypto, common.

TEE_GenerateRandom() - Generates random data.

This function generates random data of random bufferlength and is stored in to randomBuffer by calling sgx_read↩ _rand().

Parameters

randomBuffer	Reference to generated random data
randomBufferLen	Byte length of requested random data

10.31.2.10 TEE_GetObjectInfo1()

```
TEE_Result TEE_GetObjectInfo1 (
    TEE_ObjectHandle object,
    TEE_ObjectInfo * objectInfo )
```

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

TEE_GetObjectInfo1() - Function returns the characteristics of an object.

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

Parameters

objectInfo	Pointer to a structure filled with the object information
object	Handle of the object

Returns

0 if success else error occured.

10.31.2.11 TEE_GetREETime()

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

Core Functions, Time Functions.

TEE_GetREETime() - Function retrieves the current REE system time.

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

Parameters

time	Filled with the number of seconds and milliseconds.

10.31.2.12 TEE_GetSystemTime() void TEE_GetSystemTime (TEE_Time * time)

Core Functions, Time Functions.

TEE_GetSystemTime() - Retrieves the current system time.

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

Parameters

time	Filled with the number of seconds and milliseconds.

10.31.2.13 TEE_OpenPersistentObject() TEE_Result TEE_OpenPersistentObject (uint32_t storageID, const void * objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle * object)

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

TEE_OpenPersistentObject() - Opens a handle on an existing persistent object.

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

Parameters

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

Returns

0 if success, else error occured.

10.31.2.14 TEE_ReadObjectData() TEE_Result TEE_ReadObjectData (TEE_ObjectHandle object, void * buffer,

```
uint32_t size,
uint32_t * count )
```

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

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

The bytes are read starting at the position in the data stream currently stored in the object handle. The handle's position is incremented by the number of bytes actually read. On completion TEE_ReadObjectData sets the number of bytes actually read in the uint32_t pointed to by count. The value written to ∗count may be less than size if the number of bytes until the end-of3067 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

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

Returns

TEE_SUCCESS if success, else error occured.

10.31.2.15 TEE_WriteObjectData()

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

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

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

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

Parameters

object	Handle of the object
buffer	The buffer containing the data to be written
size	The number of bytes to write

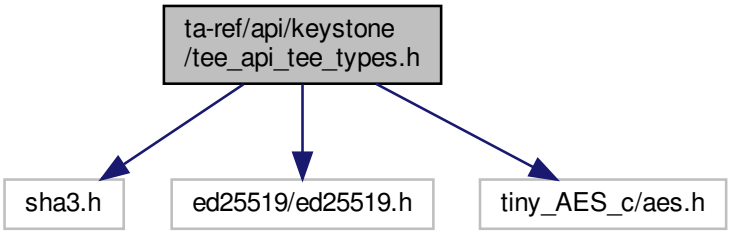
Returns

TEE_SUCCESS if success else error occured.

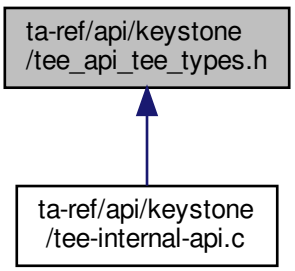
10.32 ta-ref/api/keystone/tee_api_tee_types.h File Reference

```
#include "sha3.h"
#include "ed25519/ed25519.h"
#include "tiny_AES_c/aes.h"
```

Include dependency graph for tee_api_tee_types.h:



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



Classes

- struct __TEE_OperationHandle
- struct __TEE_ObjectHandle

Macros

- #define MBEDCRYPT 1
- #define WOLFCRYPT 2
- #define AES256 1
- #define SHA_LENGTH (256/8)
- #define TEE_OBJECT_NONCE_SIZE 16
- #define TEE_OBJECT_KEY_SIZE 32
- #define TEE_OBJECT_SKEY_SIZE 64
- #define TEE_OBJECT_AAD_SIZE 16
- #define TEE_OBJECT_TAG_SIZE 16

10.32.1 Macro Definition Documentation

10.32.1.1 AES256 `#define AES256 1`

10.32.1.2 MBEDCRYPT `#define MBEDCRYPT 1`

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

10.32.1.4 TEE_OBJECT_AAD_SIZE `#define TEE_OBJECT_AAD_SIZE 16`

10.32.1.5 TEE_OBJECT_KEY_SIZE `#define TEE_OBJECT_KEY_SIZE 32`

10.32.1.6 TEE_OBJECT_NONCE_SIZE `#define TEE_OBJECT_NONCE_SIZE 16`

10.32.1.7 TEE_OBJECT_SKEY_SIZE `#define TEE_OBJECT_SKEY_SIZE 64`

10.32.1.8 TEE_OBJECT_TAG_SIZE `#define TEE_OBJECT_TAG_SIZE 16`

10.32.1.9 WOLFCRYPT #define WOLFCRYPT 2

10.33 tee_api_tee_types.h

Go to the documentation of this file.

```
1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                          and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10  *
11  * 1. Redistributions of source code must retain the above copyright notice,
12  * this list of conditions and the following disclaimer.
13  *
14  * 2. Redistributions in binary form must reproduce the above copyright notice,
15  * this list of conditions and the following disclaimer in the documentation
16  * and/or other materials provided with the distribution.
17  *
18  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28  * POSSIBILITY OF SUCH DAMAGE.
29  */
30
31 #ifndef TEE_API_TYPES_KEYSTONE_H
32 #define TEE_API_TYPES_KEYSTONE_H
33
34 #define MBEDCRYPT 1
35 #define WOLFCRYPT 2
36
37 #if CRYPTLIB==MBEDCRYPT
38 # define MBEDTLS_CONFIG_FILE "mbed-crypto-config.h"
39 # include "mbedtls/gcm.h"
40 # include "mbedtls/aes.h"
41 # include "sha3.h"
42 # include "ed25519/ed25519.h"
43 #define AES256 1
44 #elif CRYPTLIB==WOLFCRYPT
45 # define HAVE_AESGCM 1
46 # define HAVE_AES_CBC 1
47 # define HAVE_AES_DECRYPT 1
48 # define HAVE_FIPS 1
49 # define HAVE_FIPS_VERSION 2
50 # define HAVE_ED25519 1
51 # define HAVE_ED25519_SIGN 1
52 # define HAVE_ED25519_VERIFY 1
53 # define WOLFSSL_SHA512 1
54 # define WOLFSSL_SHA3 1
55 # define WOLFSSL_SHA3_SMALL 1
56 # define WOLFCRYPT_ONLY 1
57 # define WOLF_CRYPT_PORT_H
58 # include "wolfssl/wolfcrypt/sha3.h"
59 # include "wolfssl/wolfcrypt/aes.h"
60 # include "wolfssl/wolfcrypt/sha512.h"
61 # include "wolfssl/wolfcrypt/ed25519.h"
62 #else
63 #include "sha3.h"
64 #include "ed25519/ed25519.h"
65 #define AES256 1
66 # include "tiny_AES_c/aes.h"
67 #endif
68
69 #define SHA_LENGTH (256/8)
70 #define TEE_OBJECT_NONCE_SIZE 16
71 #define TEE_OBJECT_KEY_SIZE 32
72 #define TEE_OBJECT_SKEY_SIZE 64
73 #define TEE_OBJECT_AAD_SIZE 16
74 #define TEE_OBJECT_TAG_SIZE 16
75
76 struct __TEE_OperationHandle
77 {
```

```
78   int mode;
79   int flags;
80   int alg;
81 #if CRYPTLIB==MBEDCRYPT
82   sha3_ctx_t ctx;
83   mbedtls_aes_context aectx;
84   mbedtls_gcm_context aegcmctx;
85 #elif CRYPTLIB==WOLFCRYPT
86   wc_Sha3 ctx;
87   Aes aectx;
88   Aes aegcmctx;
89   unsigned int aegcm_aadsz;
90   unsigned char aegcm_aad[TEE_OBJECT_AAD_SIZE];
91   ed25519_key key;
92 #else
93   sha3_ctx_t ctx;
94   struct AES_ctx aectx;
95 #endif
96   int aegcm_state;
97   unsigned char aeiv[TEE_OBJECT_NONCE_SIZE];
98   unsigned char aekey[32];
99   unsigned char pubkey[TEE_OBJECT_KEY_SIZE];
100   unsigned char prikey[TEE_OBJECT_SKEY_SIZE];
101 };
102
103 struct __TEE_ObjectHandle
104 {
105   unsigned int type;
106   int flags;
107   int desc;
108 #if CRYPTLIB==MBEDCRYPT
109   mbedtls_aes_context persist_ctx;
110   unsigned char persist_iv[TEE_OBJECT_NONCE_SIZE];
111 #elif CRYPTLIB==WOLFCRYPT
112   Aes persist_ctx;
113   unsigned char persist_iv[TEE_OBJECT_NONCE_SIZE];
114   ed25519_key key;
115 #else
116   struct AES_ctx persist_ctx;
117 #endif
118   unsigned char public_key[TEE_OBJECT_KEY_SIZE];
119   unsigned char private_key[TEE_OBJECT_SKEY_SIZE];
120 };
121
122 // defined in tee_api_defines.h
123 // enum Data_Flag_Constants {
124 //   TEE_DATA_FLAG_ACCESS_READ = 0x00000001,
125 //   TEE_DATA_FLAG_ACCESS_WRITE = 0x00000002,
126 //   //TEE_DATA_FLAG_ACCESS_WRITE_META = 0x00000004,
127 //   //TEE_DATA_FLAG_SHARE_READ = 0x00000010,
128 //   //TEE_DATA_FLAG_SHARE_WRITE = 0x00000020,
129 //   TEE_DATA_FLAG_OVERWRITE = 0x00000400
130 // };
131 // enum Data_Flag_Constants {
132 //   TEE_DATA_FLAG_ACCESS_READ = 0x00000001,
133 //   TEE_DATA_FLAG_ACCESS_WRITE = 0x00000002,
134 //   //TEE_DATA_FLAG_ACCESS_WRITE_META = 0x00000004,
135 //   //TEE_DATA_FLAG_SHARE_READ = 0x00000010,
136 //   //TEE_DATA_FLAG_SHARE_WRITE = 0x00000020,
137 //   TEE_DATA_FLAG_OVERWRITE = 0x00000400
138 // };
139 #endif
```

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

10.35 tee_api_tee_types.h

Go to the documentation of this file.

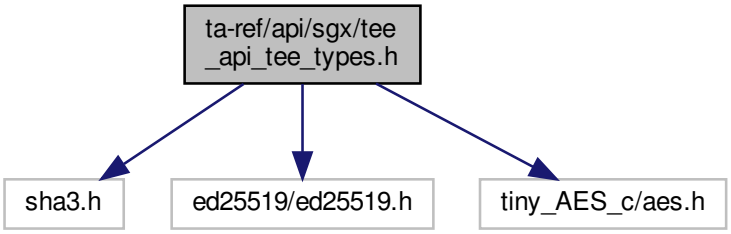
```
1 // empty
```

10.36 ta-ref/api/sgx/tee_api_tee_types.h File Reference

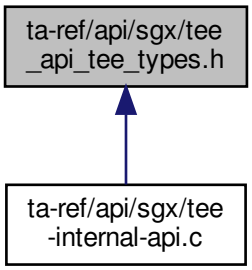
```
#include "sha3.h"
#include "ed25519/ed25519.h"
```

```
#include "tiny_AES_c/aes.h"
```

Include dependency graph for tee_api_tee_types.h:



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



Classes

- struct __TEE_OperationHandle
- struct __TEE_ObjectHandle

Macros

- #define MBEDCRYPT 1
- #define WOLFCRYPT 2
- #define SHA_LENGTH (256/8)
- #define AES256 1
- #define TEE_OBJECT_NONCE_SIZE 16
- #define TEE_OBJECT_KEY_SIZE 32
- #define TEE_OBJECT_SKEY_SIZE 64
- #define TEE_OBJECT_AAD_SIZE 16
- #define TEE_OBJECT_TAG_SIZE 16
- #define TEE_HANDLE_NULL 0

10.36.1 Macro Definition Documentation

10.36.1.1 AES256 `#define AES256 1`

10.36.1.2 MBEDCRYPT `#define MBEDCRYPT 1`

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

10.36.1.4 TEE_HANDLE_NULL `#define TEE_HANDLE_NULL 0`

10.36.1.5 TEE_OBJECT_AAD_SIZE `#define TEE_OBJECT_AAD_SIZE 16`

10.36.1.6 TEE_OBJECT_KEY_SIZE `#define TEE_OBJECT_KEY_SIZE 32`

10.36.1.7 TEE_OBJECT_NONCE_SIZE `#define TEE_OBJECT_NONCE_SIZE 16`

10.36.1.8 TEE_OBJECT_SKEY_SIZE `#define TEE_OBJECT_SKEY_SIZE 64`

10.36.1.9 TEE_OBJECT_TAG_SIZE `#define TEE_OBJECT_TAG_SIZE 16`

10.36.1.10 WOLFCRYPT `#define WOLFCRYPT 2`

10.37 tee_api_tee_types.h

Go to the documentation of this file.

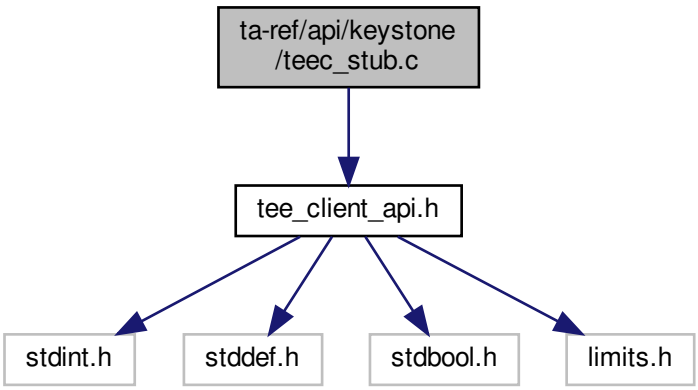
```
1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                          and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10  *
11  * 1. Redistributions of source code must retain the above copyright notice,
12  * this list of conditions and the following disclaimer.
13  *
14  * 2. Redistributions in binary form must reproduce the above copyright notice,
15  * this list of conditions and the following disclaimer in the documentation
16  * and/or other materials provided with the distribution.
17  *
18  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28  * POSSIBILITY OF SUCH DAMAGE.
29  */
30
31 #ifndef TEE_API_TYPES_KEYSTONE_H
32 #define TEE_API_TYPES_KEYSTONE_H
33
34 #define MBEDCRYPT 1
35 #define WOLFCRYPT 2
36 #define SHA_LENGTH (256/8)
37
38 #include "sha3.h"
39 #include "ed25519/ed25519.h"
40 #define AES256 1
41 #if CRYPTLIB==MBEDCRYPT
42 # define MBEDTLS_CONFIG_FILE "mbed-crypto-config.h"
43 # include "mbedtls/gcm.h"
44 # include "mbedtls/aes.h"
45 #elif CRYPTLIB==WOLFCRYPT
46 # define HAVE_AESGCM 1
47 # define HAVE_AES_CBC 1
48 # define HAVE_AES_DECRYPT 1
49 # define HAVE_FIPS 1
50 # define HAVE_FIPS_VERSION 2
51 # define HAVE_ED25519 1
52 # define HAVE_ED25519_SIGN 1
53 # define HAVE_ED25519_VERIFY 1
54 # define WOLFSSL_SHA3 1
55 # define WOLF_CRYPT_PORT_H
56 # include "wolfssl/wolfcrypt/sha3.h"
57 # include "wolfssl/wolfcrypt/aes.h"
58 # include "wolfssl/wolfcrypt/sha512.h"
59 # include "wolfssl/wolfcrypt/ed25519.h"
60 #else
61 # include "tiny_AES_c/aes.h"
62 #endif
63
64 #define TEE_OBJECT_NONCE_SIZE 16
65 #define TEE_OBJECT_KEY_SIZE 32
66 #define TEE_OBJECT_SKEY_SIZE 64
67 #define TEE_OBJECT_AAD_SIZE 16
68 #define TEE_OBJECT_TAG_SIZE 16
69
70 struct __TEE_OperationHandle
71 {
72   int mode;
73   int flags;
74   int alg;
75 #if CRYPTLIB==MBEDCRYPT
76   sha3_ctx_t ctx;
77   mbedtls_aes_context aectx;
78   mbedtls_gcm_context aegcmctx;
79 #elif CRYPTLIB==WOLFCRYPT
80   wc_Sha3 ctx;
81   Aes aectx;
82   Aes aegcmctx;
83   unsigned int aegcm_aadsz;
```

```
84   unsigned char aegcm_aad[TEE_OBJECT_AAD_SIZE];
85   ed25519_key key;
86 #else
87   sha3_ctx_t ctx;
88   struct AES_ctx aectx;
89 #endif
90   int aegcm_state;
91   unsigned char aeiv[TEE_OBJECT_NONCE_SIZE];
92   unsigned char aekey[32];
93   unsigned char pubkey[TEE_OBJECT_KEY_SIZE];
94   unsigned char prikey[TEE_OBJECT_SKEY_SIZE];
95 };
96
97 struct _TEE_ObjectHandle
98 {
99   unsigned int type;
100   int flags;
101   int desc;
102 #if CRYPTLIB==MBEDCRYPT
103   mbedtls_aes_context persist_ctx;
104   unsigned char persist_iv[TEE_OBJECT_NONCE_SIZE];
105 #elif CRYPTLIB==WOLFCRYPT
106   Aes persist_ctx;
107   unsigned char persist_iv[TEE_OBJECT_NONCE_SIZE];
108   ed25519_key key;
109 #else
110   struct AES_ctx persist_ctx;
111 #endif
112   unsigned char public_key[TEE_OBJECT_KEY_SIZE];
113   unsigned char private_key[TEE_OBJECT_SKEY_SIZE];
114 };
115
116 // Minimal constant definitions
117
118 #define TEE_HANDLE_NULL 0
119 #endif
```

10.38 ta-ref/api/keystone/teec_stub.c File Reference

`#include <tee_client_api.h>`

Include dependency graph for teec_stub.c:



Functions

- TEEC_Result TEEC_InitializeContext (const char *name, TEEC_Context *context)
- void TEEC_FinalizeContext (TEEC_Context *context)

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

10.38.1 Function Documentation

10.38.1.1 TEEC_AllocateSharedMemory()

```
TEEC_Result TEEC_AllocateSharedMemory (
    TEEC_Context * context,
    TEEC_SharedMemory * sharedMem )
```

TEEC_AllocateSharedMemory() - Allocate shared memory for TEE.

Parameters

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

Returns

TEEC_SUCCESS The registration was successful.

TEEC_ERROR_OUT_OF_MEMORY Memory exhaustion.

TEEC_Result Something failed.

10.38.1.2 TEEC_CloseSession()

```
void TEEC_CloseSession (
    TEEC_Session * session )
```

TEEC_CloseSession() - Closes the session which has been opened with the specific trusted application.

Parameters

session	The opened session to close.

10.38.1.3 TEEC_FinalizeContext()

```
void TEEC_FinalizeContext (
    TEEC_Context * context )
```

TEEC_FinalizeContext() - Destroys a context holding connection information on the specific TEE.

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

Parameters

context	The context to be finalized.

10.38.1.4 TEEC_InitializeContext()

```
TEEC_Result TEEC_InitializeContext (
    const char * name,
    TEEC_Context * context )
```

TEEC_InitializeContext() - Initializes a context holding connection information on the specific TEE, designated by the name string.

Parameters

name	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.
context	The context structure which is to be initialized.

Returns

TEEC_SUCCESS The initialization was successful.

TEEC_Result Something failed.

10.38.1.5 TEEC_OpenSession()

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

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

Parameters

context	The initialized TEE context structure in which scope to open the session.
session	The session to initialize.
destination	A structure identifying the trusted application with which to open a session.

Paramter list continued on next page

connectionMethod	The connection method to use.
connectionData	Any data necessary to connect with the chosen connection method. Not supported, should be set to NULL.
operation	An operation structure to use in the session. May be set to NULL to signify no operation structure needed.
returnOrigin	A parameter which will hold the error origin if this function returns any value other than TEEC_SUCCESS.

Returns

TEEC_SUCCESS OpenSession successfully opened a new session.

TEEC_Result Something failed.

10.38.1.6 TEEC_RegisterSharedMemory()

```
TEEC_Result TEEC_RegisterSharedMemory (
    TEEC_Context * context,
    TEEC_SharedMemory * sharedMem )
```

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

Parameters

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

Returns

TEEC_SUCCESS The registration was successful.

TEEC_ERROR_OUT_OF_MEMORY Memory exhaustion.

TEEC_Result Something failed.

10.38.1.7 TEEC_ReleaseSharedMemory()

```
void TEEC_ReleaseSharedMemory (
    TEEC_SharedMemory * sharedMemory )
```

TEEC_ReleaseSharedMemory() - Free or deregister the shared memory.

Parameters

sharedMem	Pointer to the shared memory to be freed.

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

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

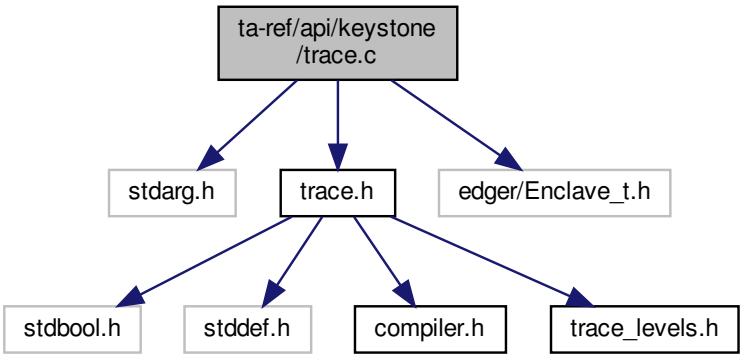
Parameters

operation	Pointer to an operation previously passed to open session or invoke.

10.39 ta-ref/api/keystone/trace.c File Reference

```
#include <stdarg.h>
#include "trace.h"
#include "edger/Enclave_t.h"
```

Include dependency graph for trace.c:



Functions

- void trace_vprintf (const char *func, int line, int level, bool level_ok, const char *fmt, va_list ap)
- void trace_printf (const char *func, int line, int level, bool level_ok, const char *fmt,...)

10.39.1 Function Documentation

10.39.1.1 trace_printf()

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

trace_printf() - Prints the formatted data to stdout.

This function returns the value of ap by calling va_end().

Parameters

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

Returns

Total number of characters is returned.

10.39.1.2 trace_vprintf()

```
void trace_vprintf (
    const char * func,
    int line,
    int level,
    bool level_ok,
    const char * fmt,
    va_list ap )
```

trace_vprintf() - Writes the formatted data from variable argument list to sized buffer.

This function returns the buffer character by calling ocall_print_string()

Parameters

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

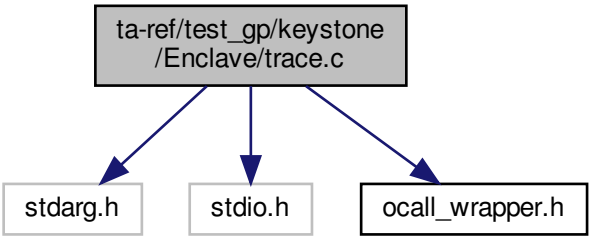
Returns

buf The total number of characters written is returned.

10.40 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,...)

10.40.1 Function Documentation

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

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

Parameters

str	str is argument of type pointer.

Returns

string string length.

10.40.1.2 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

fmt	fmt is constant character argument of type pointer.

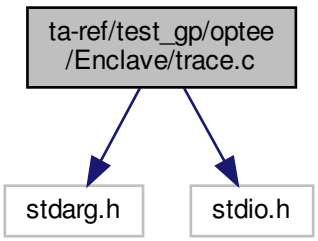
Returns

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

10.41 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,...)

10.41.1 Function Documentation

10.41.1.1 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

fmt	A string that specifies the format of the output.

Returns

result If success, else error occured.

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

fmt	fmt is constant character argument of type pointer.

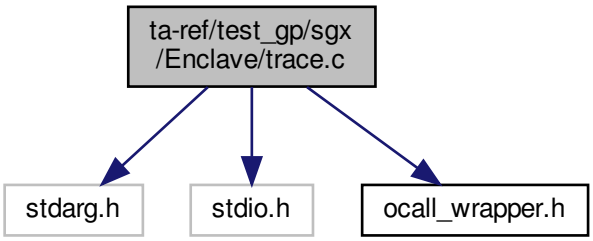
Returns

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

10.42 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,...)

10.42.1 Function Documentation

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

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

Parameters

str	str is an argument of type pointer.

Returns

string length on success.

10.42.1.2 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

fmt	fmt is a constant character argument of type pointer.

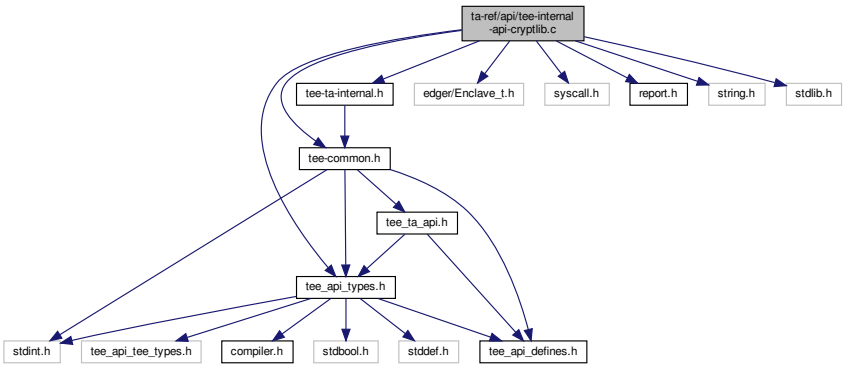
Returns

buffer If successfully executed, else error occured.

10.43 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.

10.43.1 Macro Definition Documentation

10.43.1.1 GCM_ST_AAD `#define GCM_ST_AAD 2`

10.43.1.2 GCM_ST_ACTIVE `#define GCM_ST_ACTIVE 3`

10.43.1.3 GCM_ST_FINAL `#define GCM_ST_FINAL 4`

10.43.1.4 GCM_ST_INIT `#define GCM_ST_INIT 1`

10.43.1.5 SIG_LENGTH `#define SIG_LENGTH 64`

10.43.2 Function Documentation

10.43.2.1 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

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

10.43.2.2 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

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

Returns

0 on success.

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

10.43.2.3 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

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

Returns

0 on success.

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

10.43.2.4 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

operation	Handle of a running AE operation.
srcData	Input data buffer to be encrypted or decrypted
srcLen	length of the input buffer.
destData	Output buffer
destLen	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.

10.43.2.5 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

operation	Handle on the AE operation
AADdata	Input buffer containing the chunk of AAD
AADdataLen	length of the chunk of AAD.

10.43.2.6 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

operation	reference to generated operation handle.
algorithm	One of the cipher algorithms.
mode	The operation mode.
maxKeySize	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.

10.43.2.7 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

objectType	Type of uninitialized object container to be created
maxKeySize	Key Size of the object.
object	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.

10.43.2.8 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

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

Returns

0 on sccess

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

10.43.2.9 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

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

Returns

TEE_SUCCESS on success

TEE_ERROR_SIGNATURE_INVALID if the signature is invalid.

10.43.2.10 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

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

Returns

0 on success

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

10.43.2.11 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

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

10.43.2.12 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

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

10.43.2.13 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

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

10.43.2.14 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

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

10.43.2.15 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

operation	Reference to operation handle.

Returns

nothing after the operation free.

10.43.2.16 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

object	Handle on the object to free.

10.43.2.17 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

object	Handle on an uninitialized transient key to populate with the generated key.
keySize	Requested key size shall be less than or equal to the maximum key size specified when the object container was created
params	Parameters for the key generation.
paramCount	The values of all parameters are copied nto the object so that the params array and all the memory buffers it points to may be freed after this routine returns without affecting the object.

Returns

0 on succes

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

10.43.2.18 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

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

10.43.2.19 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

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

10.43.2.20 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

operation	Operation handle.
key	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.

10.43.2.21 wolfSSL_Free() `void wolfSSL_Free ( void * p )`

wolfSSL_Free() - Deallocates the memory which allocated previously.

Parameters

p	This is the pointer to a memory block.

10.43.2.22 wolfSSL_Malloc() `void * wolfSSL_Malloc ( size_t n )`

wolfSSL_Malloc() - Allocates the requested memory and returns a pointer to it.

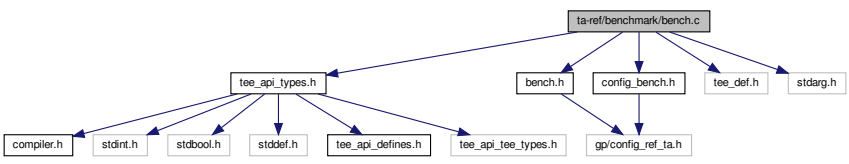
Parameters

n	size of the memory block.

10.44 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]

10.44.1 Function Documentation

10.44.1.1 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 condtion 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

type	The integer type argument for switch case.
unit	The integer type argument for loop.

10.44.1.2 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]"

10.44.1.3 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_Get↩ REETime(end). It iterates and records the start and end time of the benchmark execution, and test_printf() prints the values using for_loop.

Parameters

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

10.44.1.4 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

type	The integer type for switch case
time	The pointer type of TEE_Time
size	The maximum size to be stored.

10.44.1.5 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

t1	The pointer type argument of TEE_Time
t2	The pointer type argument of TEE_Time

Returns

It will return the difference time between t1, t2.

10.44.1.6 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

type	The character type argument for switch case
time	The pointer type of TEE_Time
idx	The integer type of time_t

10.44.1.7 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

time	The pointer type argument of TEE_Time.

Returns

It will return time value as a milliseconds.

10.44.2 Variable Documentation

10.44.2.1 labels

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

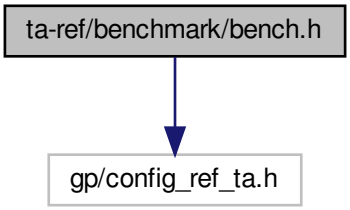
Initial value:

```
= {
    "TEE_GetREETime",
    "TEE_GetSystemTime",
    "cpu sensitive",
    "memory sensitive",
    "io sensitive",
}
```

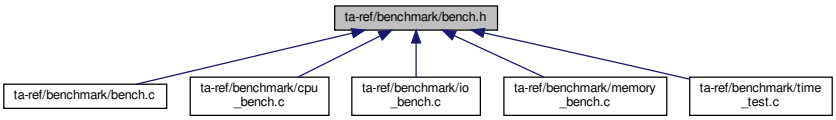
10.45 ta-ref/benchmark/bench.h File Reference

```
#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)

10.45.1 Macro Definition Documentation

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

10.45.2 Function Documentation

10.45.2.1 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 condtion gets satisfied.

10.45.2.2 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 condtion gets satisfied.

10.45.2.3 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

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

10.45.2.4 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

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

10.45.2.5 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

buf	A pointer to the buffer in the process of read and write
size	The size of the buffer.

10.45.2.6 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.

10.45.2.7 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

buf	A pointer to the buffer in the process of read and write
size	The size of the buffer.

10.45.2.8 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.

10.46 bench.h

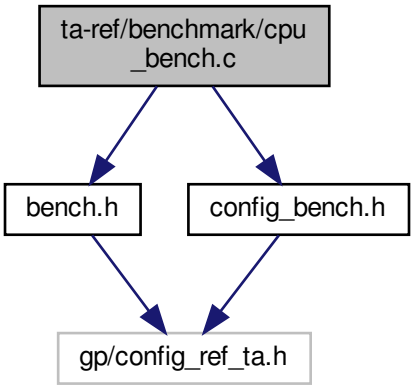
Go to the documentation of this file.

```
1 #pragma once
2 #include "gp/config_ref_ta.h"
3
4 #define NO_PERF __attribute__((no_instrument_function))
5 void NO_PERF ree_time_test(void);
6 void NO_PERF system_time_test(void);
7 void NO_PERF cpu_int_benchmark(void);
8 void NO_PERF cpu_double_benchmark(void);
9 void NO_PERF io_read_benchmark(char *buf, char *fname, int size);
10 void NO_PERF io_write_benchmark(char *buf, char *fname, int size);
11 void NO_PERF random_memory_benchmark(char *buf, int size);
12 void NO_PERF sequential_memory_benchmark(char *buf, int size);
```

10.47 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)

10.47.1 Function Documentation

10.47.1.1 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 condtion gets satisfied.

10.47.1.2 cpu_int_benchmark() `void NO_PERF cpu_int_benchmark ( void )`

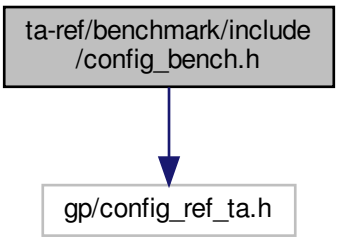
cpu_int_benchmark() - TO check the processing of cpu integer benchmark

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 condtion gets satisfied.

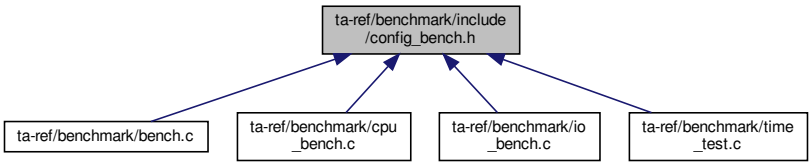
10.48 ta-ref/benchmark/include/config_bench.h File Reference

```
#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)

10.48.1 Macro Definition Documentation

10.48.1.1 BUF_SIZE `#define BUF_SIZE 1048576`

10.48.1.2 DOUBLE_OFFSET `#define DOUBLE_OFFSET (double)1234567890.123456789`

10.48.1.3 MULT_SIZE `#define MULT_SIZE 5000`

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

10.48.2 Enumeration Type Documentation

10.48.2.1 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	

10.48.3 Function Documentation

10.48.3.1 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_Get↩ REETime(end). It iterates and records the start and end time of the benchmark execution, and test_printf() prints the values using for_loop.

Parameters

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

10.49 config_bench.h

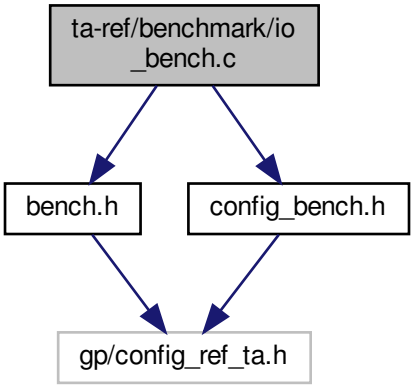
Go to the documentation of this file.

```
1 #pragma once
2 #include "gp/config_ref_ta.h" // for TEE_** API
3
4 enum BENCH_TYPE {
5     REE_TIME_TEST,
6     SYSTEM_TIME_TEST,
7     CPU_INT_SENSITIVE,
8     CPU_DOUBLE_SENSITIVE,
9     SEQUENTIAL_MEMORY_SENSITIVE,
10     RANDOM_MEMORY_SENSITIVE,
11     IO_WRITE_SENSITIVE,
12     IO_READ_SENSITIVE,
13 };
14
15 void record(int type, TEE_Time *start, TEE_Time *end, int size, int unit);
16 #define OFFSET (uint64_t)0x0102030405060708
17 #define DOUBLE_OFFSET (double)1234567890.123456789
18 #define MULT_SIZE 5000
19 #define BUF_SIZE 1048576
```

10.50 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)

10.50.1 Macro Definition Documentation

10.50.1.1 SPLITS `#define SPLITS 32`

10.50.2 Function Documentation

10.50.2.1 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 object. Finally it will close the object.

Parameters

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

10.50.2.2 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.

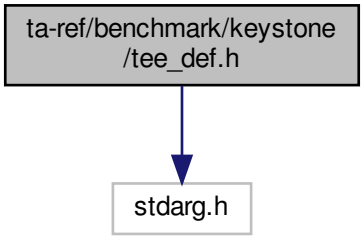
Parameters

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

10.51 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

10.51.1 Function Documentation

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

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

10.51.2 Variable Documentation

10.51.2.1 buf `char* buf [static]`

10.51.2.2 buf_flag `int buf_flag = 1 [static]`

10.52 tee_def.h

Go to the documentation of this file.

```
1 #include <stdarg.h>
2 static int buf_flag = 1;
3 static char *buf;
4
5 static void NO_PERF tee_init() {
6     if(buf_flag) {
7         buf = malloc(BUF_SIZE);
8         buf_flag = 0;
9     }
10 }
11 static int NO_PERF test_printf(const char* fmt, ...)
12 {
13   char buf[BUFSIZ] = { '\0' };
14   va_list ap;
15   va_start(ap, fmt);
16   vsnprintf(buf, BUFSIZ, fmt, ap);
17   va_end(ap);
18   ocall_print_string_wrapper(buf);
19   return 0;
20 }
```

10.53 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

10.53.1 Macro Definition Documentation

10.53.1.1 test_printf `#define test_printf tee_printf`

10.53.2 Function Documentation

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

10.53.3 Variable Documentation

10.53.3.1 buf `char buf[BUF_SIZE] [static]`

10.53.3.2 buf_flag `int buf_flag = 1 [static]`

10.54 tee_def.h

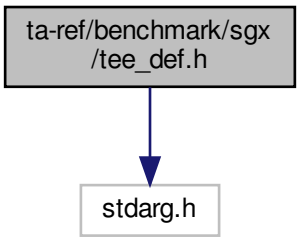
Go to the documentation of this file.

```
1 static char buf[BUF_SIZE];
2 static int buf_flag = 1;
3 static void NO_PERF tee_init(void) {}
4
5
6 #define test_printf tee_printf
```

10.55 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

10.55.1 Function Documentation

10.55.1.1 tee_init()

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

10.55.1.2 test_printf()

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

10.55.2 Variable Documentation

10.55.2.1 buf `char buf[BUF_SIZE] [static]`

10.55.2.2 buf_flag `int buf_flag = 1 [static]`

10.56 tee_def.h

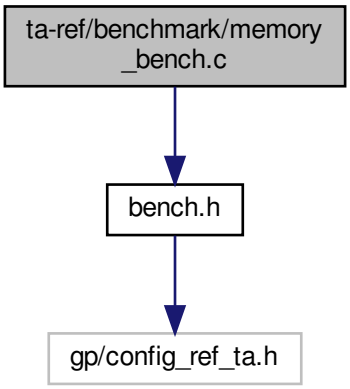
Go to the documentation of this file.

```
1 #include <stdarg.h>
2 static char buf[BUF_SIZE];
3 static int buf_flag = 1;
4
5 static void NO_PERF tee_init(void) {}
6 static int NO_PERF test_printf(const char* fmt, ...)
7 {
8   char buf[BUFSIZ] = { '\0' };
9   va_list ap;
10   va_start(ap, fmt);
11   vsnprintf(buf, BUFSIZ, fmt, ap);
12   va_end(ap);
13   ocall_print_string_wrapper(buf);
14   return 0;
15 }
```

10.57 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)

10.57.1 Macro Definition Documentation

10.57.1.1 INC `#define INC 390625`

10.57.2 Function Documentation

10.57.2.1 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

buf	A pointer to the buffer in the process of read and write
size	The size of the buffer.

10.57.2.2 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]”

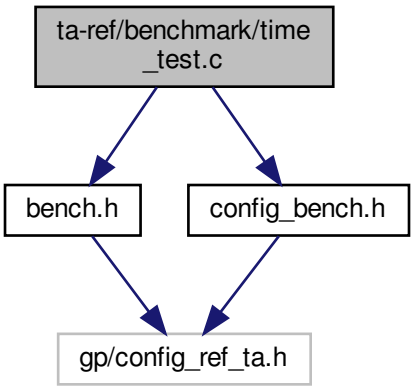
Parameters

buf	A pointer to the buffer in the process of read and write
size	The size of the buffer.

10.58 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)

10.58.1 Function Documentation

10.58.1.1 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.

10.58.1.2 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.

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

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

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

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

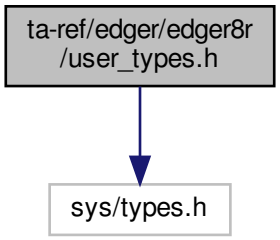
10.63 ta-ref/docs/preparation.md File Reference

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

10.65 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]

10.65.1 Macro Definition Documentation

10.65.1.1 LOOPS_PER_THREAD `#define LOOPS_PER_THREAD 500`

10.65.2 Typedef Documentation

10.65.2.1 array_t `typedef int array_t[10]`

10.65.2.2 buffer_t `typedef void* buffer_t`

10.66 user_types.h

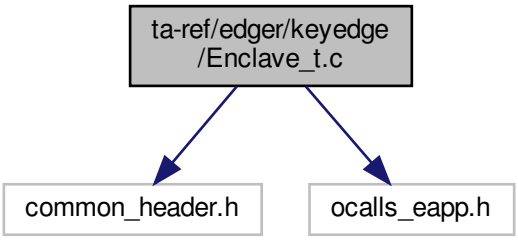
Go to the documentation of this file.

```
1 /*
2  * Copyright (C) 2011-2019 Intel Corporation. All rights reserved.
3  *
4  * Redistribution and use in source and binary forms, with or without
5  * modification, are permitted provided that the following conditions
6  * are met:
7  *
8  *   * Redistributions of source code must retain the above copyright
9  *     notice, this list of conditions and the following disclaimer.
10 *   * Redistributions in binary form must reproduce the above copyright
11 *     notice, this list of conditions and the following disclaimer in
12 *     the documentation and/or other materials provided with the
13 *     distribution.
14 *   * Neither the name of Intel Corporation nor the names of its
15 *     contributors may be used to endorse or promote products derived
16 *     from this software without specific prior written permission.
17 *
18 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
19 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
20 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
21 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
22 * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
23 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
24 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
25 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
26 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
27 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 *
30 */
31
32
33 /* User defined types */
34
35
36 #define LOOPS_PER_THREAD 500
37 #include <sys/types.h>
38
39 typedef void *buffer_t;
40 typedef int array_t[10];
```

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

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

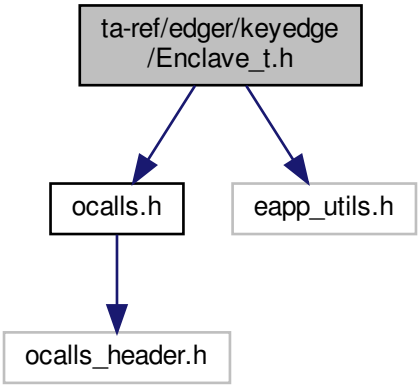
Include dependency graph for Enclave_t.c:



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

```
#include "ocalls.h"
#include "eapp_utils.h"
```

Include dependency graph for Enclave_t.h:



10.69 Enclave_t.h

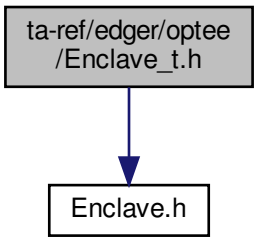
Go to the documentation of this file.

```
1 #pragma once
2 // ocall_** functions
3 #include "ocalls.h"
4 #include "eapp_utils.h"
```

10.70 ta-ref/edger/optee/Enclave_t.h File Reference

```
#include "Enclave.h"
```

Include dependency graph for Enclave_t.h:



10.71 Enclave_t.h

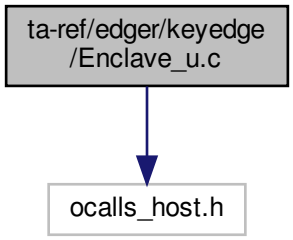
Go to the documentation of this file.

```
1 #pragma once
2 #include "Enclave.h"
```

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

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

Include dependency graph for Enclave_u.c:

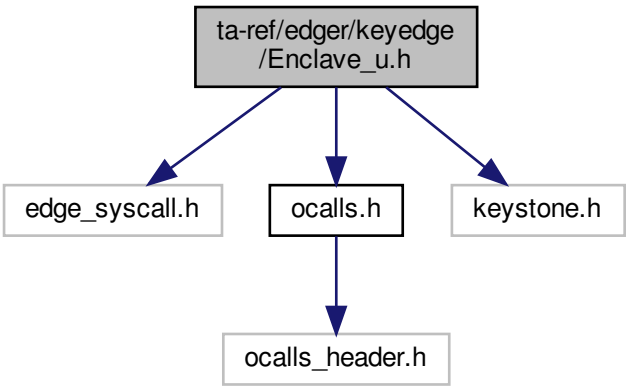


10.73 ta-ref/edger/keyedge/Enclave_u.h File Reference

```
#include "edge_syscall.h"
#include "ocalls.h"
```

```
#include "keystone.h"
```

Include dependency graph for Enclave_u.h:



Macros

- #define EDGE_EXTERNC_BEGIN
- #define EDGE_EXTERNC_END

Functions

- void register_functions ()
- void __wrapper_ocall_close_file (void *buffer)

10.73.1 Macro Definition Documentation

10.73.1.1 EDGE_EXTERNC_BEGIN `#define EDGE_EXTERNC_BEGIN`

10.73.1.2 EDGE_EXTERNC_END `#define EDGE_EXTERNC_END`

10.73.2 Function Documentation

10.73.2.1 __wrapper_ocall_close_file() void __wrapper_ocall_close_file (void * buffer)

10.73.2.2 register_functions() void register_functions ( )

10.74 Enclave_u.h

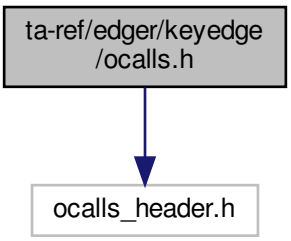
Go to the documentation of this file.

```
1 #pragma once
2 // from ocalls_host.h
3 void register_functions();
4 void __wrapper_ocall_close_file(void* buffer);
5 #include "edge_syscall.h"
6 #include "ocalls.h"
7 #include "keystone.h"
8 #define EDGE_EXTERNC_BEGIN
9 #define EDGE_EXTERNC_END
```

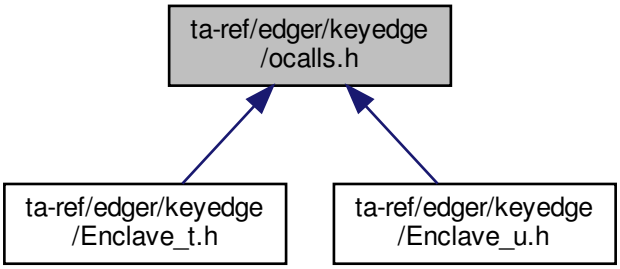
10.75 ta-ref/edger/keyedge/ocalls.h File Reference

`#include "ocalls_header.h"`

Include dependency graph for ocalls.h:



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



Classes

- struct ob256_t
- struct ree_time_t
- struct ob16_t
- struct ob196_t
- struct invoke_command_t
- struct nonce_t

Macros

- #define EDGE_OUT_WITH_STRUCTURE
- #define NONCE_SIZE 32

Typedefs

- typedef struct ob256_t ob256_t
- typedef struct ree_time_t ree_time_t
- typedef struct ob16_t ob16_t
- typedef struct ob196_t ob196_t
- typedef struct invoke_command_t invoke_command_t
- typedef struct nonce_t nonce_t

Functions

- unsigned int ocall_print_string (keyedge_str const char *str)
- int ocall_open_file (keyedge_str const char *str, int flags, int perm)
- int ocall_close_file (int desc)
- int ocall_write_file (int desc, keyedge_vla const char *buf, keyedge_size unsigned int len)
- int ocall_invoke_command_callback_write (keyedge_str const char *str, keyedge_vla const char *buf, keyedge_size unsigned int len)
- ob256_t ocall_read_file256 (int desc)
- 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_invoke_command_polling (int dummy)
- int ocall_invoke_command_callback (invoke_command_t cb_cmd)
- nonce_t ocall_import_nonce (void)

10.75.1 Macro Definition Documentation

10.75.1.1 EDGE_OUT_WITH_STRUCTURE `#define EDGE_OUT_WITH_STRUCTURE`

10.75.1.2 NONCE_SIZE `#define NONCE_SIZE 32`

10.75.2 Typedef Documentation

10.75.2.1 invoke_command_t `typedef struct invoke_command_t invoke_command_t`

10.75.2.2 nonce_t `typedef struct nonce_t nonce_t`

10.75.2.3 ob16_t `typedef struct ob16_t ob16_t`

10.75.2.4 ob196_t `typedef struct ob196_t ob196_t`

10.75.2.5 ob256_t `typedef struct ob256_t ob256_t`

10.75.2.6 ree_time_t `typedef struct ree_time_t ree_time_t`

10.75.3 Function Documentation

10.75.3.1 ocall_close_file() `int ocall_close_file ( int desc )`

ocall_close_file() - To close a file.

Parameters

fdesc	file descriptor.

Returns

integer value If success

ocall_close_file() - To close a file.

Parameters

desc	file descriptor.

Returns

integer value If success

ocall_close_file() - Frees the file descriptor in the process.

Parameters

fdesc	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

desc	File descriptor.

Returns

file descripto If success, else error occured.

10.75.3.2 ocall_getrandom16()

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

10.75.3.3 ocall_getrandom196()

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

10.75.3.4 ocall_import_nonce()

```
nonce_t ocall_import_nonce (
    void  )
```

10.75.3.5 ocall_invoke_command_callback()

```
int ocall_invoke_command_callback (
    invoke_command_t cb_cmd )
```

10.75.3.6 ocall_invoke_command_callback_write()

```
int ocall_invoke_command_callback_write (
    keyedge_str const char * str,
    keyedge_vla const char * buf,
    keyedge_size unsigned int len )
```

10.75.3.7 ocall_invoke_command_polling()

```
invoke_command_t ocall_invoke_command_polling (
    int dummy )
```

10.75.3.8 ocall_open_file()

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

10.75.3.9 ocall_print_string()

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

10.75.3.10 ocall_read_file256()

```
ob256_t ocall_read_file256 (
    int desc )
```

10.75.3.11 ocall_ree_time()

```
ree_time_t ocall_ree_time (
    void  )
```

10.75.3.12 ocall_write_file()

```
int ocall_write_file (
    int desc,
    keyedge_vla const char * buf,
    keyedge_size unsigned int len )
```

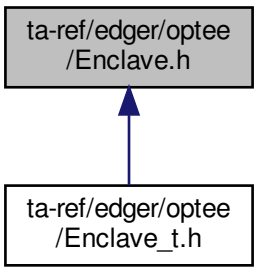
10.76 ocalls.h

Go to the documentation of this file.

```
1 #pragma once
2 // keyedge_(vla|size|str)
3 #include "ocalls_header.h"
4
5 // Add edge call function declarations here.
6
7 unsigned int ocall_print_string(keyedge_str const char* str);
8
9 int ocall_open_file(keyedge_str const char* str, int flags, int perm);
10 int ocall_close_file(int desc);
11 int ocall_write_file(int desc, keyedge_vla const char *buf, keyedge_size unsigned int len);
12 int ocall_invoke_command_callback_write(keyedge_str const char* str, keyedge_vla const char *buf,
       keyedge_size unsigned int len);
13
14 // keyedge has no [out] type buf i.e. all data from host to eapp should be
15 // returned as the return value which might be a fixed size structure.
16 #define EDGE_OUT_WITH_STRUCTURE
17
18 typedef struct ob256_t { int ret; unsigned char b[256]; } ob256_t;
19
20 ob256_t ocall_read_file256(int desc);
21
22 typedef struct ree_time_t {
23   unsigned int seconds;
24   unsigned int millis;
25 } ree_time_t;
26
27 ree_time_t ocall_ree_time(void);
28
29 typedef struct ob16_t { int ret; unsigned char b[16]; } ob16_t;
30
31 ob16_t ocall_getrandom16(unsigned int flags);
32
33 typedef struct ob196_t { int ret; unsigned char b[196]; } ob196_t;
34
35 ob196_t ocall_getrandom196(unsigned int flags);
36
37 // for TEE InvokeCommand
38 typedef struct invoke_command_t {
39   unsigned int commandID;
40   char params0_buffer[256];
41   unsigned int params0_size;
42   int param1_fd;
43   char params1_buffer[256];
44   unsigned int params1_size;
45 //  char params2[256];
46 //  char params3[256];
47 } invoke_command_t;
48 invoke_command_t ocall_invoke_command_polling(int dummy);
49
50 int ocall_invoke_command_callback(invoke_command_t cb_cmd);
51
52 #define NONCE_SIZE 32
53 typedef struct nonce_t {
54     unsigned char nonce[NONCE_SIZE];
55 } nonce_t;
56
57 nonce_t ocall_import_nonce(void);
```

10.77 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

10.77.1 Macro Definition Documentation

10.77.1.1 TA_REF_RUN_ALL `#define TA_REF_RUN_ALL 0`

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

10.78 Enclave.h

Go to the documentation of this file.

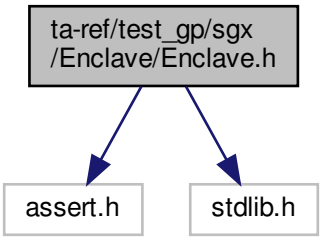
```
1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                    and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are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

```
17  *
18  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28  * POSSIBILITY OF SUCH DAMAGE.
29  */
30
31 #ifndef _ENCLAVE_H
32 #define _ENCLAVE_H
33
34 /*
35  * This UUID is generated with uuidgen
36  * the ITU-T UUID generator at http://www.itu.int/ITU-T/asn1/uuid.html
37  */
38 #define TA_REF_UUID \
39   { 0xa6f77c1e, 0x96fe, 0x4a0e, { 0x9e, 0x74, 0x26, 0x25, 0x82, 0xa4, 0xc8, 0xf1}}
40
41 /* The function IDs implemented in this TA */
42 #define TA_REF_RUN_ALL      0
43
44 #endif /*_ENCLAVE_H*/
```

10.79 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)

10.79.1 Function Documentation

10.79.1.1 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 anf allocate handle for a Sign hashed data with the generated keys and allocates 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.

10.79.1.2 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.

10.79.1.3 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.

10.79.1.4 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”.

10.79.1.5 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,finallyclose 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 varify fails.

10.79.1.6 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.

10.79.1.7 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.

10.79.1.8 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”.

10.80 Enclave.h

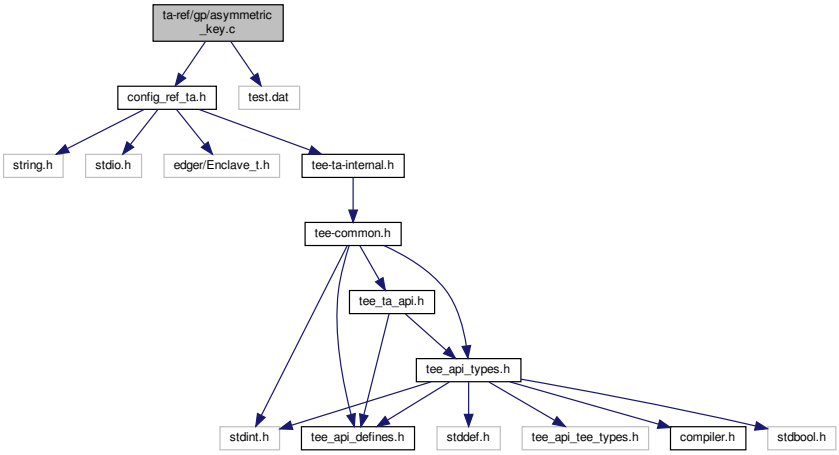
Go to the documentation of this file.

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

10.81 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)

10.81.1 Macro Definition Documentation

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

10.81.1.2 SIG_LENGTH `#define SIG_LENGTH 64`

10.81.2 Function Documentation

10.81.2.1 gp_asymmetric_key_sign_test() `void gp_asymmetric_key_sign_test ( void )`

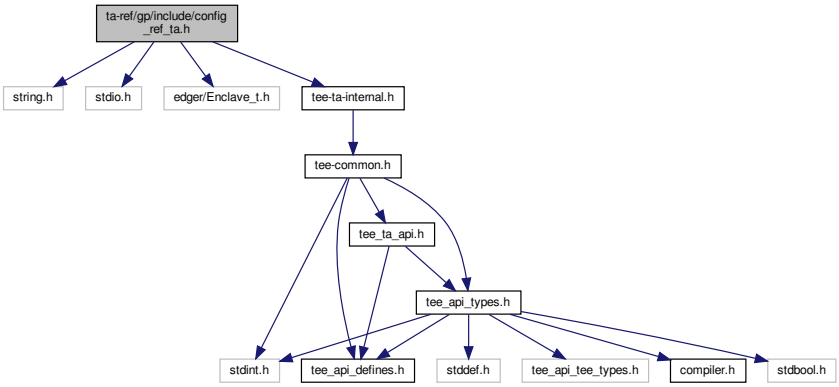
gp_asymmetric_key_sign_test() - Cryptographic Operations for API Message Digest Functions.

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 anf allocate handle for a Sign hashed data with the generated keys and allocates 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.

10.82 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,...)

10.82.1 Macro Definition Documentation

10.82.1.1 GP_ASSERT

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

10.82.2 Function Documentation

10.82.2.1 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

fmt	A string that specifies the format of the output.

Returns

result If success, else error occured.

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

fmt	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

fmt	fmt is a constant character argument of type pointer.

Returns

buffer If successfully executed, else error occured.

10.83 config_ref_ta.h

Go to the documentation of this file.

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

10.84 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)

10.84.1 Function Documentation

10.84.1.1 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 anf allocate handle for a Sign hashed data with the generated keys and allocates 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.

10.84.1.2 gp_invokecommand_test() void gp_invokecommand_test (void)

10.84.1.3 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.

10.84.1.4 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.

10.84.1.5 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”.

10.84.1.6 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,finallyclose 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 varify fails.

10.84.1.7 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.

10.84.1.8 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.

10.84.1.9 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”.

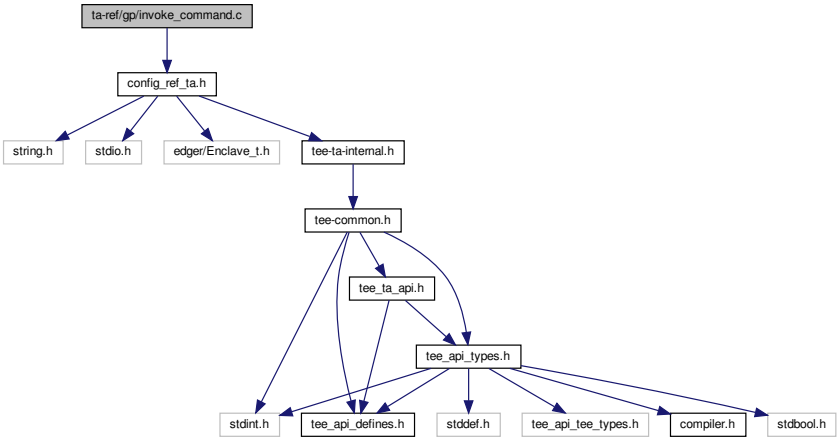
10.85 gp_test.h

Go to the documentation of this file.

```
1 #if defined(__cplusplus)
2 extern "C" {
3 #endif
4
5 void gp_random_test(void);
6 void gp_ree_time_test(void);
7 void gp_trusted_time_test(void);
8 void gp_secure_storage_test(void);
9 void gp_message_digest_test(void);
10 void gp_symmetric_key_enc_verify_test(void);
11 void gp_symmetric_key_gcm_verify_test(void);
12 void gp_asymmetric_key_sign_test(void);
13 void gp_invokecommand_test(void);
14
15 #if defined(__cplusplus)
16 }
17 #endif
```

10.86 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

10.86.1 Macro Definition Documentation

10.86.1.1 TA_MAX_SIZE `#define TA_MAX_SIZE 32768`

10.86.1.2 TEEP_AGENT_TA_DELETE `#define TEEP_AGENT_TA_DELETE 3`

10.86.1.3 TEEP_AGENT_TA_EXIT `#define TEEP_AGENT_TA_EXIT 999`

10.86.1.4 TEEP_AGENT_TA_INSTALL `#define TEEP_AGENT_TA_INSTALL 2`

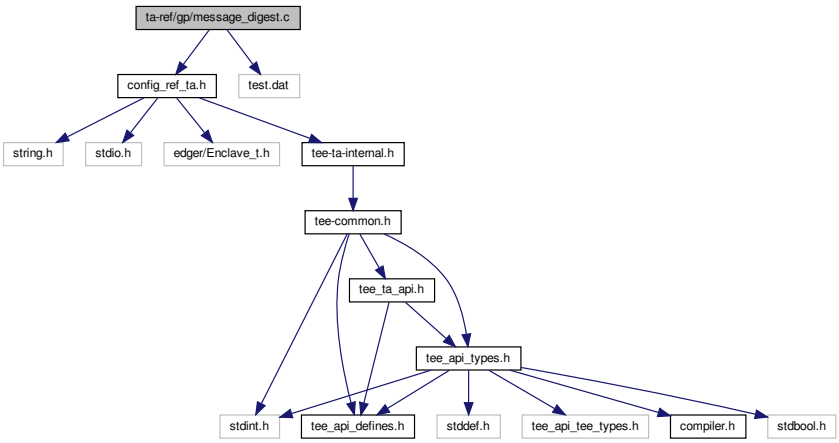
10.86.1.5 TEEP_AGENT_TA_LOAD `#define TEEP_AGENT_TA_LOAD 1`

10.86.1.6 TEEP_AGENT_TA_NONE `#define TEEP_AGENT_TA_NONE 0`

10.87 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)

10.87.1 Macro Definition Documentation

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

10.87.1.2 SIG_LENGTH `#define SIG_LENGTH 64`

10.87.2 Function Documentation

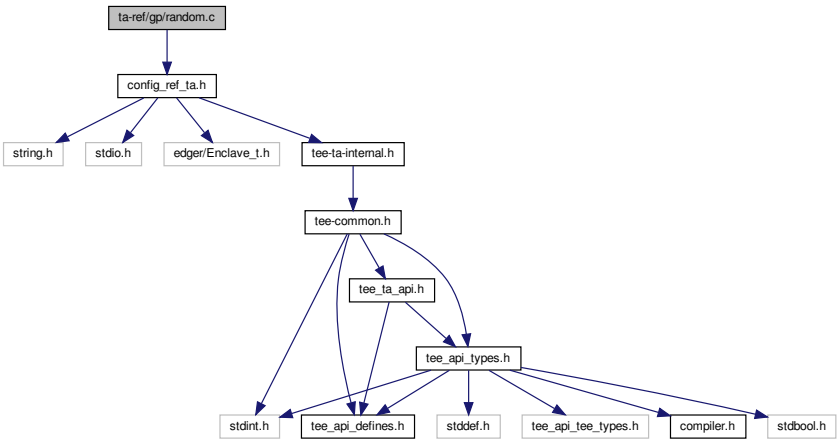
10.87.2.1 gp_message_digest_test() `void gp_message_digest_test ( void )`

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.

10.88 ta-ref/gp/random.c File Reference

`#include "config_ref_ta.h"`
Include dependency graph for random.c:



Functions

- void gp_random_test (void)

10.88.1 Function Documentation

10.88.1.1 gp_random_test() `void gp_random_test ( void )`

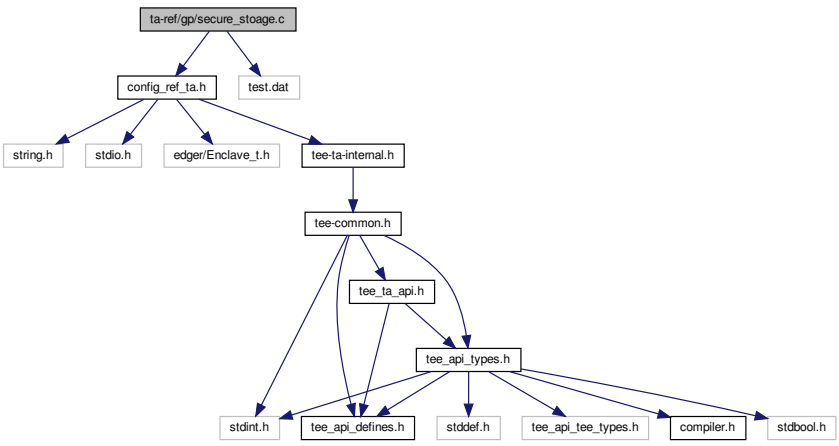
gp_random_test() - Generates the random data from the method.

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

10.89 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)

10.89.1 Macro Definition Documentation

10.89.1.1 DATA_LENGTH `#define DATA_LENGTH 256`

10.89.2 Function Documentation

10.89.2.1 gp_secure_storage_test() `void gp_secure_storage_test ( void )`

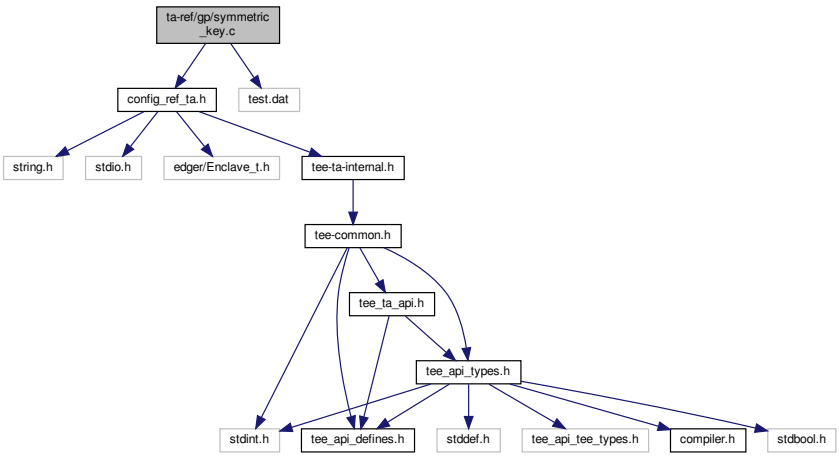
gp_secure_storage_test() - Create persistent object for read and write the object data.

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,finallyclose 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 varify fails.

10.90 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)

10.90.1 Macro Definition Documentation

10.90.1.1 CIPHER_LENGTH `#define CIPHER_LENGTH 256`

10.90.2 Function Documentation

10.90.2.1 gp_symmetric_key_enc_verify_test() `void gp_symmetric_key_enc_verify_test ( void )`

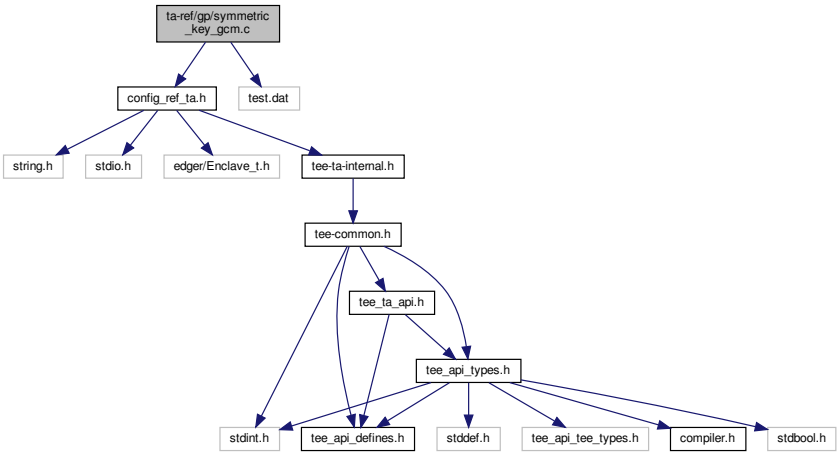
gp_symmetric_key_enc_verify_test() - starts the symmetric cipher operation.

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.

10.91 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)

10.91.1 Macro Definition Documentation

10.91.1.1 CIPHER_LENGTH `#define CIPHER_LENGTH 256`

10.91.2 Function Documentation

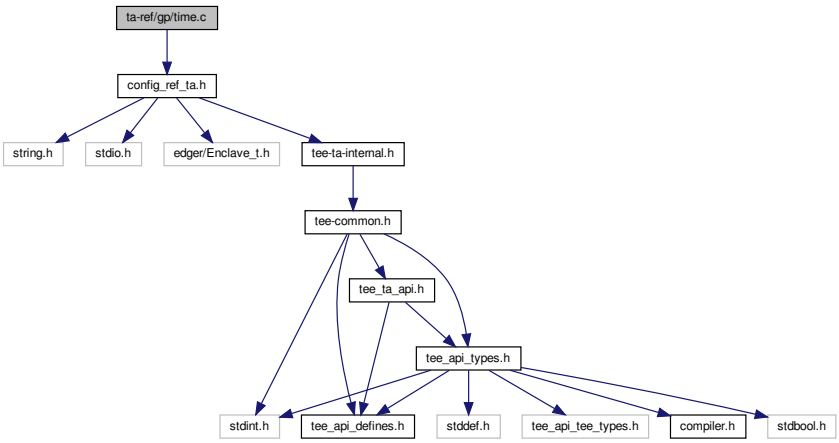
10.91.2.1 gp_symmetric_key_gcm_verify_test() `void gp_symmetric_key_gcm_verify_test ( void )`

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.

10.92 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)

10.92.1 Function Documentation

10.92.1.1 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”.

10.92.1.2 gp_trusted_time_test() `void gp_trusted_time_test ( void )`

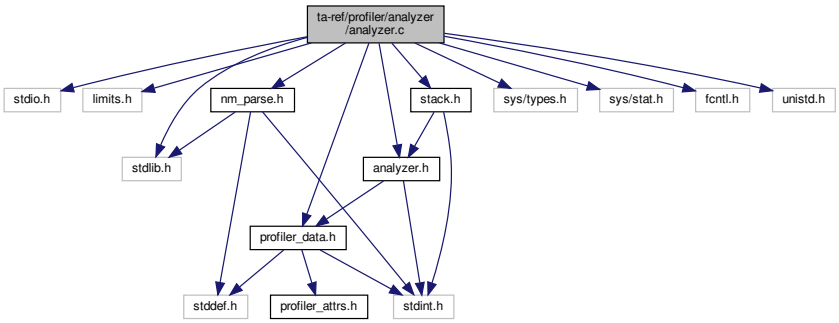
gp_trusted_time_test() - Retrieves the current system time.

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”.

10.93 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,%03ld,%d,%d,%ld,0x%08lx,%s,%ld,%ld,%ld\n"

Functions

- int main (int argc, char *argv[])

10.93.1 Macro Definition Documentation

10.93.1.1 BUF_MAX `#define BUF_MAX 65536`

10.93.1.2 COLS `#define COLS "id,idx,start_core_id,end_core_id,depth,addr,funcname,start[clocks],end,duration"`

10.93.1.3 FORMAT `#define FORMAT "%03d,%03ld,%d,%d,%ld,0x%08lx,%s,%ld,%ld,%ld\n"`

10.93.2 Function Documentation

10.93.2.1 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

argc	Argument Count is int and stores number of command-line arguments passed by the user including the name of the program.
argv	Argument Vector is array of character pointers listing all the arguments.

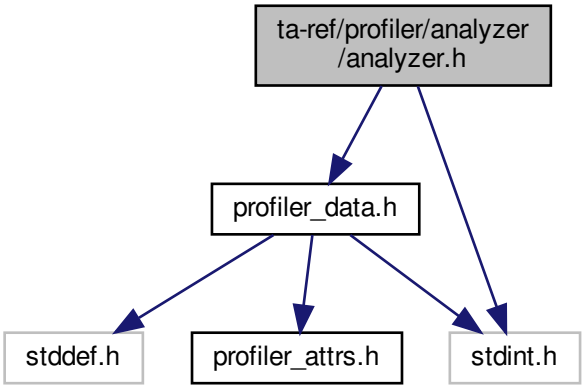
Returns

0 If success, else error occured.

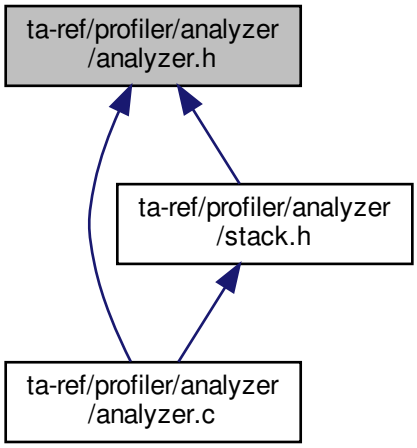
10.94 ta-ref/profiler/analyzer/analyzer.h File Reference

```
#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

10.95 analyzer.h

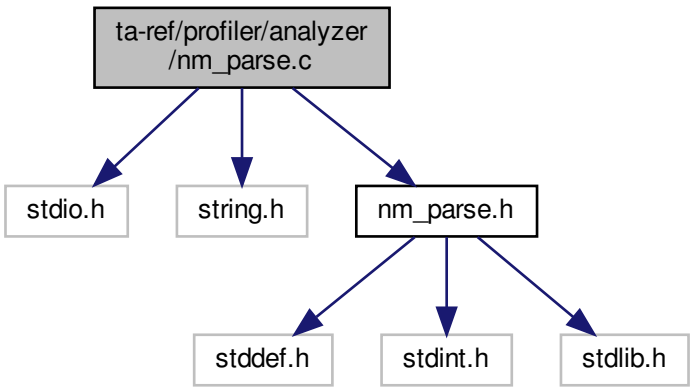
Go to the documentation of this file.

```
1 #pragma once
2 #include "profiler_data.h"
3 #include <stdint.h>
4
5 struct result {
6     size_t idx;
7     uintptr_t callee;
8     uint8_t start_hartid;
9     uint8_t end_hartid;
10     __profiler_nsec_t start;
11     __profiler_nsec_t end;
12     size_t depth;
13 };
```

10.96 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

10.96.1 Macro Definition Documentation

10.96.1.1 BUF_SIZE `#define BUF_SIZE 512`

10.96.1.2 MAX_ADDR `#define MAX_ADDR 0xFFFFFFFF`

10.96.1.3 POOL_SIZE `#define POOL_SIZE 30000`

10.96.2 Function Documentation

10.96.2.1 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 occured.

10.96.2.2 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

table	It's an object of struct list.
addr	Address to find the key value.

Returns

String length If success, else error occured.

10.96.2.3 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

addr	Address of the key value.

Returns

Address of the hash key If success, else error occured.

10.96.2.4 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

table	It's an object of struct list.
addr	Address of the key value.
nm	Name of the information of struct nm_info

10.96.2.5 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

fname	File name.

Returns

Updated structure list If success, else error occured.

10.96.3 Variable Documentation

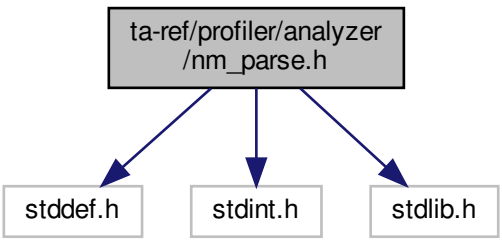
10.96.3.1 idx `int idx = 0 [static]`

10.96.3.2 nm_pool `struct nm_info nm_pool[POOL_SIZE] [static]`

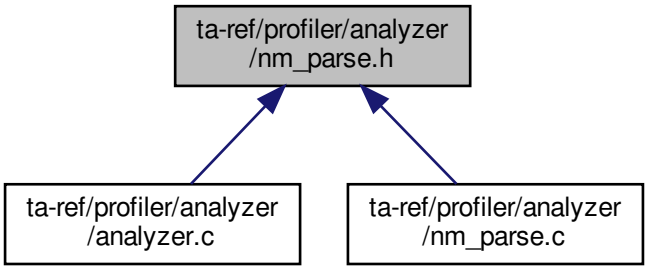
10.97 ta-ref/profiler/analyzer/nm_parse.h File Reference

```
#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)

10.97.1 Macro Definition Documentation

10.97.1.1 HASH_SIZE `#define HASH_SIZE 65536`

10.97.2 Function Documentation

10.97.2.1 get_func_name() `const char * get_func_name ( struct list * table, uintptr_t addr )`

10.97.2.2 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

fname	File name.

Returns

Updated structure list If success, else error occured.

10.98 nm_parse.h

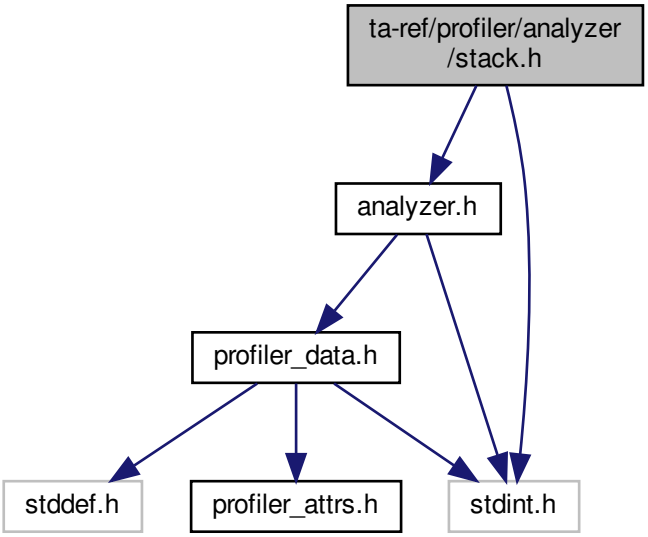
Go to the documentation of this file.

```
1 #pragma once
2 #include <stddef.h>
3 #include <stdint.h>
4 #include <stdlib.h>
5
6 #define HASH_SIZE 65536
7
8 struct nm_info {
9     char type;
10     char func_name[256];
11 };
12
13 struct list {
14     struct list *next;
15     uintptr_t addr;
16     struct nm_info* nm;
17 };
18
19 const char* get_func_name(struct list *table, uintptr_t addr);
20 struct list* parse_nm(const char *fname);
```

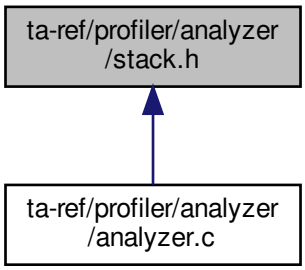
10.99 ta-ref/profiler/analyzer/stack.h File Reference

```
#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]

10.99.1 Macro Definition Documentation

10.99.1.1 STACK_SIZE `#define STACK_SIZE 100`

10.99.2 Function Documentation

10.99.2.1 is_empty() `char is_empty ( void )`

10.99.2.2 pop() `struct result pop ( void )`

10.99.2.3 push() `void push ( struct result data )`

10.99.3 Variable Documentation

10.99.3.1 pos `uint64_t pos = 0 [static]`

10.99.3.2 stack `struct result stack[STACK_SIZE] [static]`

10.100 stack.h

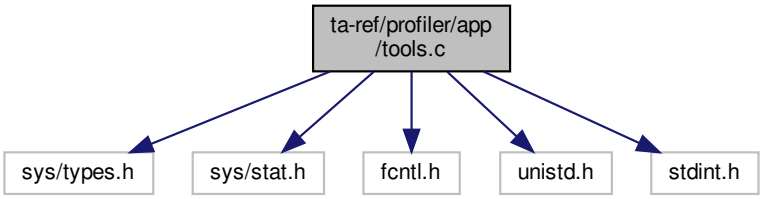
Go to the documentation of this file.

```
1 #pragma once
2 #include "analyzer.h"
3 #include <stdint.h>
4
5 #define STACK_SIZE 100
6 static uint64_t pos = 0;
7 static struct result stack[STACK_SIZE];
8
9 struct result pop(void) {
10     return stack[--pos];
11 }
12
13 void push(struct result data) {
14     data.depth = pos;
15     stack[pos++] = data;
16 }
17
18 char is_empty(void) {
19     return !pos;
20 }
```

10.101 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)

10.101.1 Function Documentation

10.101.1.1 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

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

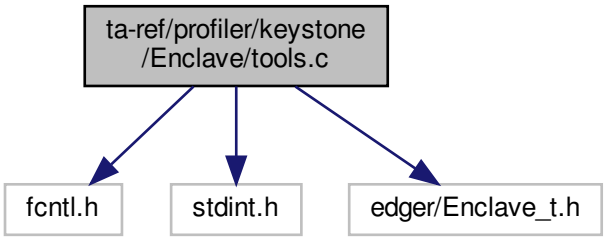
Returns

0 If success, else error occured.

10.102 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)

10.102.1 Function Documentation

10.102.1.1 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

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

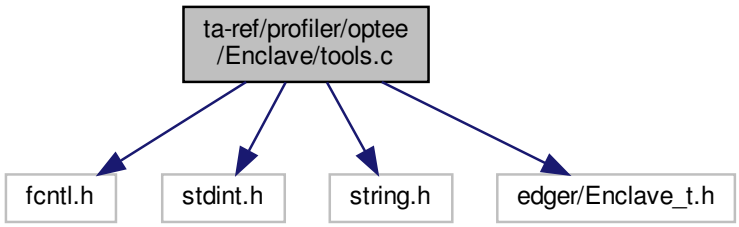
Returns

0 If success, else error occured.

10.103 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)

10.103.1 Function Documentation

10.103.1.1 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

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

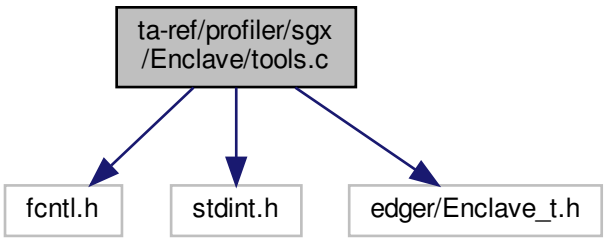
Returns

0 If success, else error occured.

10.104 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)

10.104.1 Function Documentation

10.104.1.1 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 perfomed.

Parameters

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

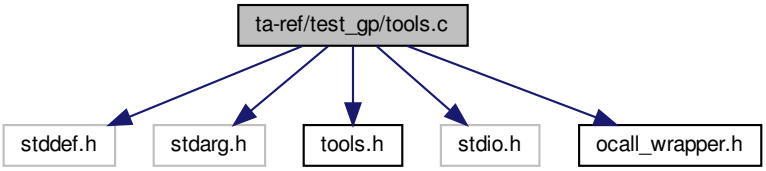
Returns

0 If success, else error occured.

10.105 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,...)

10.105.1 Function Documentation

10.105.1.1 _strlen()

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

10.105.1.2 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

fm	This is the string that contains the text to be written to stdout.

Returns

string length If success.

0 Error occured.

10.105.1.3 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

c	This is the character to be written. This is passed as its int promotion.

Returns

size If success.

0 Error occured.

10.105.1.4 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

s	This is the C string to be written

Returns

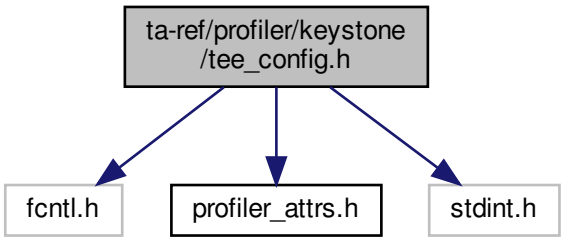
size If success.

0 Error occured.

10.106 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]

10.106.1 Function Documentation

10.106.1.1 tee_rdtscp() `static uint64_t NO_PERF tee_rdtscp ( uint8_t * id ) [inline], [static]`

10.106.2 Variable Documentation

10.106.2.1 __ImageBase `uintptr_t __ImageBase = 0 [static]`

10.106.2.2 perf_buffer `char PERF_SECTION perf_buffer[PERF_SIZE] [static]`

10.107 tee_config.h

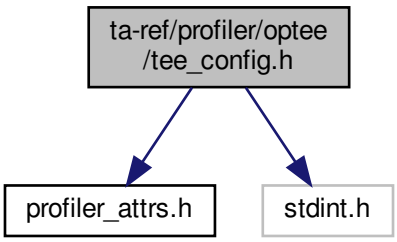
Go to the documentation of this file.

```
1 #include <fcntl.h>
2 #include "profiler_attrs.h"
3 #include <stdint.h>
4
5 static uintptr_t __ImageBase = 0;
6 static char PERF_SECTION perf_buffer[PERF_SIZE];
7
8 static inline uint64_t NO_PERF tee_rdtscp(uint8_t *id)
9 {
10     unsigned long cycles;
11     asm volatile ("rdcycle %0" : "=r" (cycles));
12     if(id) {
13         uint64_t x;
14         // eyrie OS put the hartid on tp register.
15         asm volatile("mv %0, tp" : "=r" (x) );
16         if(!(x & (1<<4))) {
17             x = 255;
18         } else {
19             x = x & ~(1<<4);
20         }
21         *id = (uint8_t)x;
22     }
23     return cycles;
24 }
```

10.108 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]

10.108.1 Macro Definition Documentation

10.108.1.1 COMMAND `#define COMMAND "mrs %0, cntpct_el0"`

10.108.2 Function Documentation

10.108.2.1 tee_rdtscp() `static uint64_t NO_PERF tee_rdtscp ( uint8_t * id ) [inline], [static]`

10.108.3 Variable Documentation

10.108.3.1 __ImageBase `uintptr_t __ImageBase[] [extern]`

10.108.3.2 perf_buffer `char perf_buffer[PERF_SIZE] [static]`

10.109 tee_config.h

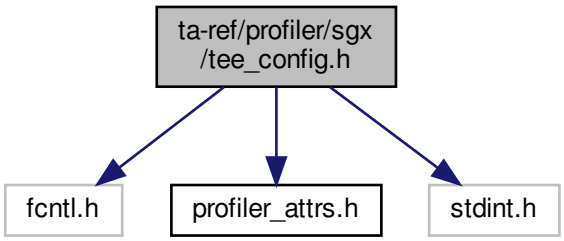
Go to the documentation of this file.

```
1 // #include <tee_internal_api.h>
2 // #include <tee_internal_api_extensions.h>
3 // this is defined in Enclave/ta.lds
4 #include "profiler_attrs.h"
5 #include <stdint.h>
6
7 extern uintptr_t __ImageBase[];
8 static char perf_buffer[PERF_SIZE];
9
10 #ifdef RPI3
11 #define COMMAND "mrs %0, cntvct_el0"
12 #else // qemu
13 #define COMMAND "mrs %0, cntpct_el0"
14 #endif
15
16 static inline uint64_t NO_PERF tee_rdtscp(uint8_t *id)
17 {
18     uint64_t cycles;
19     asm volatile(COMMAND : "=r"(cycles));
20     if(id) {
21         // uint32_t aff;
22         // asm volatile ("mrs %0, MPIDR_EL1"
23         //               : "=r"(aff));
24         // *id = aff & 255;
25         *id = 0;
26     }
27     return cycles;
28 }
```

10.110 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]

10.110.1 Function Documentation

10.110.1.1 tee_rdtscp() `static uint64_t tee_rdtscp ( uint8_t * id ) [inline], [static]`

10.110.2 Variable Documentation

10.110.2.1 __ImageBase `uintptr_t __ImageBase[] [extern]`

10.110.2.2 perf_buffer `char perf_buffer[PERF_SIZE] [static]`

10.111 tee_config.h

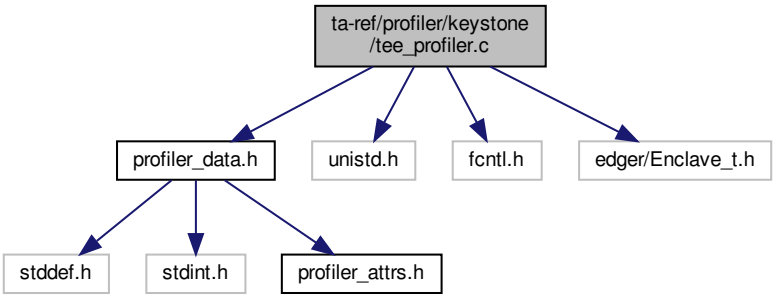
Go to the documentation of this file.

```
1 #include <fcntl.h>
2 #include "profiler_attrs.h"
3 #include <stdint.h>
4
5 extern uintptr_t __ImageBase[];
6 static char perf_buffer[PERF_SIZE];
7
8 static inline uint64_t tee_rdtscp(uint8_t *id)
9 {
10   uint32_t hi, lo, aux;
11   __asm__ volatile("rdtscp" : "=a" (lo), "=d" (hi), "=c" (aux));
12   if(id) *id = aux;
13   return ((uint64_t)hi << 32) | lo;
14 }
```

10.112 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

10.112.1 Function Documentation

10.112.1.1 __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.

10.112.1.2 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

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

Returns

0 If success, else error occured.

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

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

Returns

0 If success, else error occured.

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 perfomed.

Parameters

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

Returns

0 If success, else error occured.

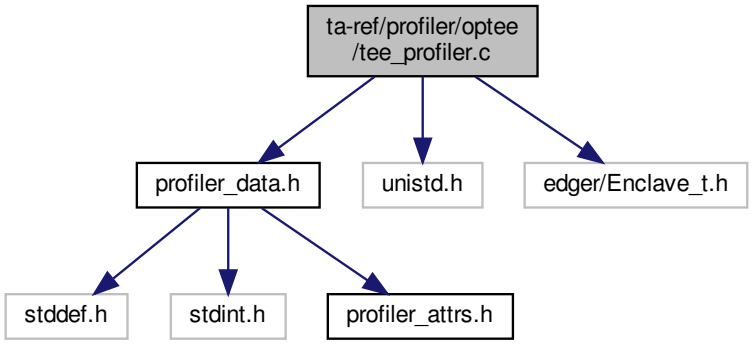
10.112.2 Variable Documentation

10.112.2.1 __profiler_head `struct __profiler_header* __profiler_head [extern]`

10.113 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

10.113.1 Function Documentation

10.113.1.1 __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

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

10.113.1.2 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

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

Returns

0 If success, else error occured.

10.113.2 Variable Documentation

10.113.2.1 __profiler_head

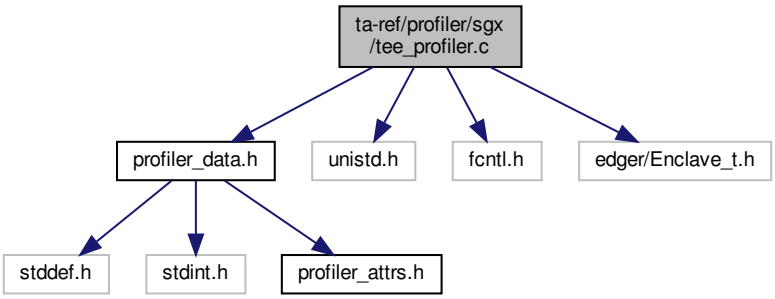
```
struct __profiler_header* __profiler_head [extern]
```

10.114 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

10.114.1 Function Documentation

10.114.1.1 __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.

10.114.1.2 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

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

Returns

0 If success, else error occured.

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

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

Returns

0 If success, else error occured.

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 perfomed.

Parameters

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

Returns

0 If success, else error occured.

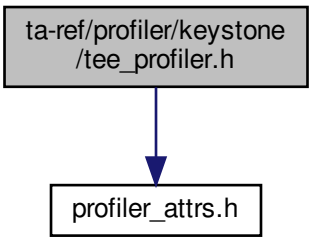
10.114.2 Variable Documentation

10.114.2.1 __profiler_head `struct __profiler_header* __profiler_head [extern]`

10.115 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)

10.115.1 Function Documentation

10.115.1.1 __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.

10.116 tee_profiler.h

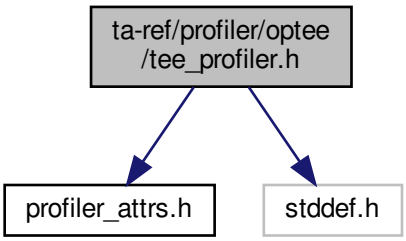
Go to the documentation of this file.

```
1 #pragma once
2 #include "profiler_attrs.h"
3
4 void NO_PERF __profiler_unmap_info(void);
```

10.117 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)

10.117.1 Function Documentation

10.117.1.1 __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

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

10.118 tee_profiler.h

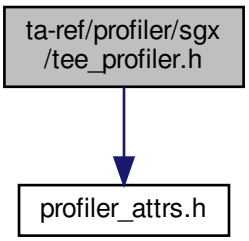
Go to the documentation of this file.

```
1 #pragma once
2 #include "profiler_attrs.h"
3 #include <stddef.h>
4
5 void NO_PERF __profiler_unmap_info(char *buf, size_t *size);
```

10.119 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)

10.119.1 Function Documentation

10.119.1.1 __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.

10.120 tee_profiler.h

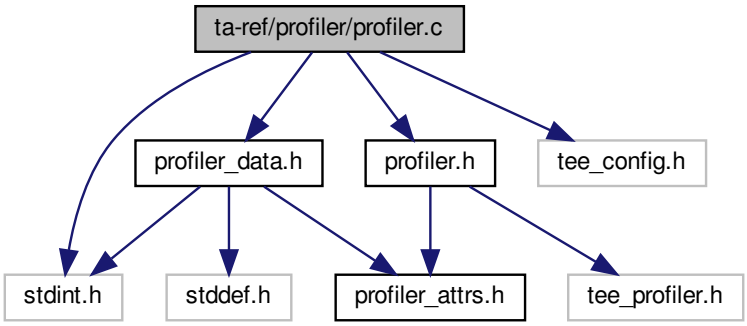
Go to the documentation of this file.

```
1 #pragma once
2 #include "profiler_attrs.h"
3
4 void NO_PERF __profiler_unmap_info(void);
```

10.121 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

10.121.1 Function Documentation

10.121.1.1 __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

this_fn	A keyword that refers to the current instance of the class.
dir	An enumeration constant.

10.121.1.2 __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

this_fn	A keyword that refers to the current instance of the class.
call_site	It means which operation perfoms for calling, start etc.

10.121.1.3 __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

this_fn	A keyword that refers to the current instance of the class.
call_site	It means which operation performs calling, stop etc.

10.121.1.4 __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.

10.121.1.5 __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.

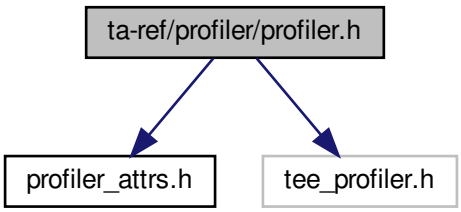
10.121.2 Variable Documentation

10.121.2.1 __profiler_head `struct __profiler_header* __profiler_head = NULL`

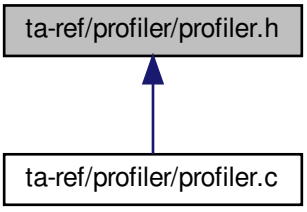
10.122 ta-ref/profiler/profiler.h File Reference

```
#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)

10.122.1 Function Documentation

10.122.1.1 __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.

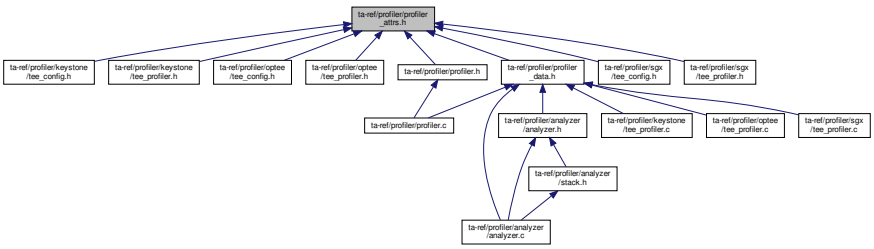
10.123 profiler.h

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifdef __cplusplus
4 extern "C" {
5 #endif
6
7 #include "profiler_attrs.h"
8 #include "tee_profiler.h"
9
10 void NO_PERF __profiler_map_info(void);
11
12 #ifdef __cplusplus
13 }
14 #endif /* __cplusplus */
```

10.124 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))

10.124.1 Macro Definition Documentation

10.124.1.1 NO_PERF `#define NO_PERF __attribute__((no_instrument_function,hot))`

10.124.1.2 PERF_SECTION `#define PERF_SECTION __attribute__((section(".perf_region")))`

10.124.1.3 USED `#define USED __attribute__((used))`

10.125 profiler_attrs.h

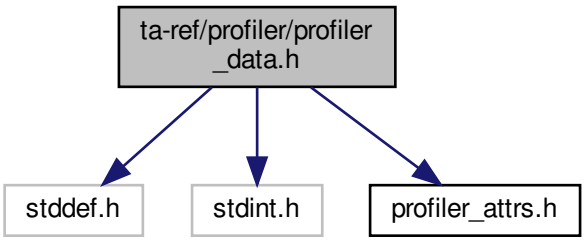
Go to the documentation of this file.

```
1 #define NO_PERF __attribute__((no_instrument_function,hot))
2 #define PERF_SECTION __attribute__((section(".perf_region")))
3 #define USED __attribute__((used))
```

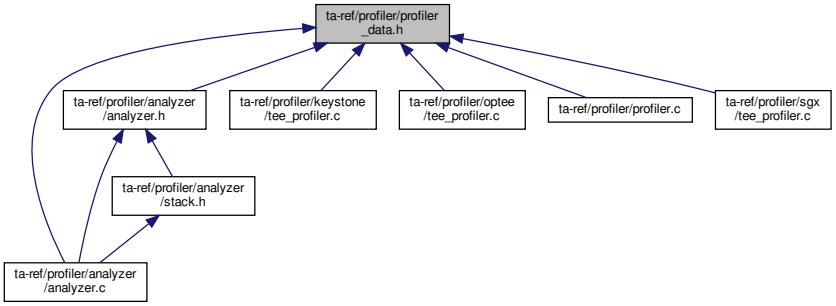
10.126 ta-ref/profiler/profiler_data.h File Reference

```
#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

10.126.1 Macro Definition Documentation

10.126.1.1 LOG_FILE `#define LOG_FILE "/root"`

10.126.1.2 PERF_SIZE `#define PERF_SIZE 8192`

10.126.2 Typedef Documentation

10.126.2.1 __profiler_nsec_t `typedef uint64_t __profiler_nsec_t`

10.126.3 Enumeration Type Documentation

10.126.3.1 direction_t `enum direction_t`

Enumerator

START	
CALL	
RET	

10.126.4 Function Documentation

10.126.4.1 __attribute__() `struct __profiler_header __attribute__ ( (packed, aligned(8)) )`

10.126.5 Variable Documentation

10.126.5.1 idx `uint64_t idx`

10.126.5.2 size `uint64_t size`

10.126.5.3 start uintptr_t start

10.127 profiler_data.h

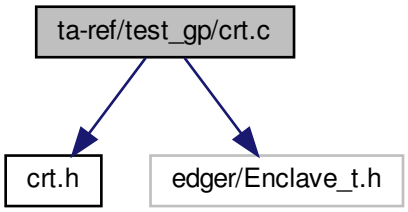
Go to the documentation of this file.

```
1 #pragma once
2 #ifndef LOG_FILE
3 #define LOG_FILE "/root"
4 #endif
5 #include <stddef.h>
6 #include <stdint.h>
7 #include "profiler_attrs.h"
8
9 #ifndef PERF_SIZE
10 #define PERF_SIZE 8192
11 #endif
12
13 enum direction_t {
14     START = 0,
15     CALL = 1,
16     RET  = 2,
17 };
18
19 typedef uint64_t __profiler_nsec_t;
20
21 struct __profiler_data {
22     uint8_t direction;
23     uint8_t hartid;
24     __profiler_nsec_t nsec;
25     uintptr_t       callee;
26 };
27
28 struct __profiler_header {
29     uint64_t size;
30     uint64_t idx;
31     uintptr_t start;
32 } __attribute__((packed,aligned(8)));
```

10.128 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)

10.128.1 Function Documentation

10.128.1.1 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.

10.128.2 Variable Documentation

10.128.2.1 __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.

10.128.2.2 aligned `void(*const aligned[])(sizeof(void *))) ( sizeof(void *) )`

Initial value:

```
= {
  }
```

10.128.2.3 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

fini_array[ ]	constant array.

10.128.2.4 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.

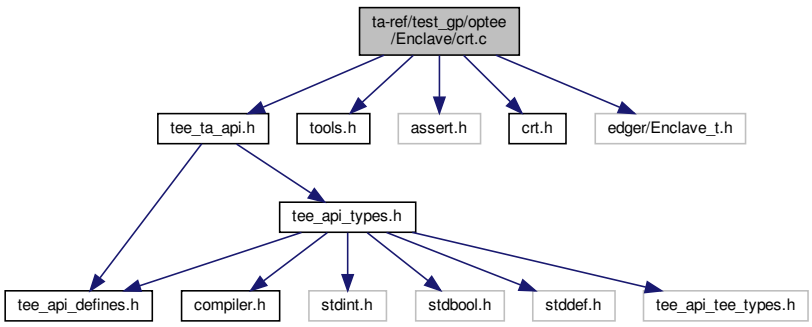
Parameters

init_array[]	constant array.

10.129 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 [ ]

10.129.1 Macro Definition Documentation

10.129.1.1 TEE_PARAM_TYPE0

```
#define TEE_PARAM_TYPE0 TEE_PARAM_TYPE_NONE
```

10.129.1.2 TEE_PARAM_TYPE1

```
#define TEE_PARAM_TYPE1 TEE_PARAM_TYPE_NONE
```

10.129.2 Function Documentation

10.129.2.1 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

param_types	The types of the four parameters.
params[4]	A pointer to an array of four parameters.
sess_ctx	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 occured in the function.

10.129.2.2 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.

10.129.2.3 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 occured.

10.129.2.4 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".

10.129.2.5 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

param_types	The types of the four parameters.
params[4]	A pointer to an array of four parameters.
sess_ctx	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 occured.

10.129.2.6 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

param_types	This denotes the types of the four parameters.
params[4]	A pointer to an array of four parameters.
sess_ctx	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 occured in the function.

10.129.2.7 tee_printf() `int tee_printf (`

```
const char * fmt,
...  )
```

tee_printf() - Printing the formatted output in to a character array.

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

Parameters

fmt	A string that specifies the format of the output.

Returns

result If success, else error occured.

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

fmt	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() - 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

fmt	A string that specifies the format of the output.

Returns

result If success, else error occured.

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

fmt	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

fmt	fmt is a constant character argument of type pointer.

Returns

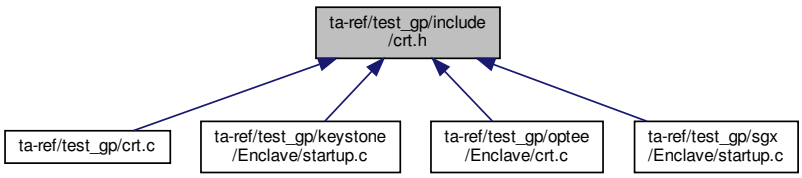
buffer If successfully executed, else error occured.

10.129.3 Variable Documentation

10.129.3.1 __ImageBase `uintptr_t __ImageBase[] [extern]`

10.130 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)

10.130.1 Function Documentation

10.130.1.1 crt_begin() `void crt_begin ( void )`

10.130.1.2 crt_end() `void crt_end ( void )`

crt_end() - Ends the certification.

It compares __fini_array_start and __fini_array_end; and then it the loops through the file pointer.

10.130.1.3 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.

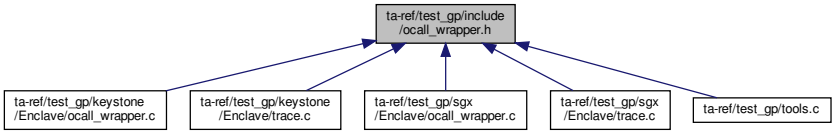
10.131 crt.h

Go to the documentation of this file.

```
1 void crt_begin(void);
2 void crt_end(void);
3 int main(void);
```

10.132 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)

10.132.1 Function Documentation

10.132.1.1 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

str	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

str	The string value for print.

Returns

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

10.133 ocall_wrapper.h

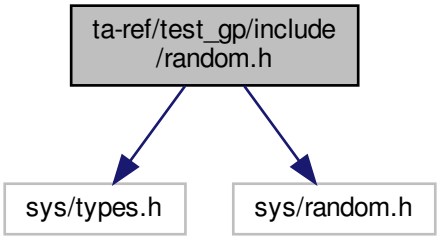
Go to the documentation of this file.

```
1 #pragma once
2 unsigned int ocall_print_string_wrapper(const char* str);
```

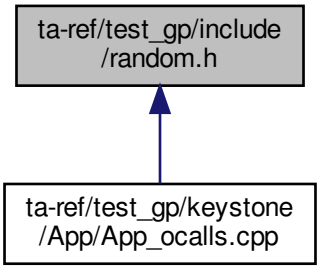
10.134 ta-ref/test_gp/include/random.h File Reference

```
#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:



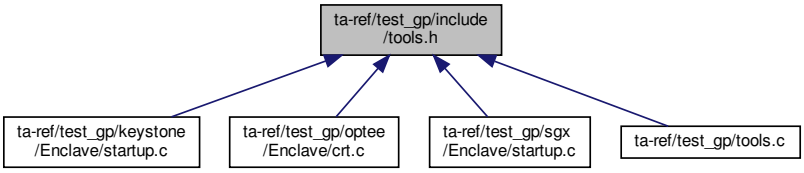
10.135 random.h

Go to the documentation of this file.

```
1 #include <sys/types.h>
2 // for keystone-enclave v0.4 we saw the getrandom(2) function freeze, so we use srandom/random instead
      when we set 'SEED' value.
3 #ifdef SEED
4 #include <stdlib.h>
5 #define getrandom seed_random
6 static ssize_t seed_random(void *buf, size_t buflen, unsigned int flags) {
7     (flags); // not used
8     const ssize_t ss = sizeof(unsigned int);
9     unsigned int retval;
10     unsigned int *b = (unsigned int*)buf;
11     ssize_t idx = 0;
12     srandom((unsigned int)SEED);
13     while(buflen) {
14         retval = random();
15         buflen -= ss;
16         b[idx++] = retval;
17     }
18     return idx*ss;
19 }
20 #else
21 #include <sys/random.h>
22 #endif
```

10.136 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,...)

10.136.1 Function Documentation

10.136.1.1 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

fm	This is the string that contains the text to be written to stdout.

Returns

string length If success.

0 Error occured.

10.136.1.2 putchar() `int putchar ( int c )`

putchar() - Function writes a character (an unsigned char) specified by the argument char to stdout.

This function returns the character written as an unsigned char cast to an int or EOF on error.

Parameters

c	This is the character to be written. This is passed as its int promotion.

Returns

size If success.

0 Error occured.

10.136.1.3 puts() `int puts ( const char * s )`

puts() - Function writes a string to stdout up to but not including the null character.

A newline character is appended to the output by calling putchar(). Compiler may replace simple printf to puts and putchar.

Parameters

s	This is the C string to be written

Returns

size If success.

0 Error occured.

10.137 tools.h

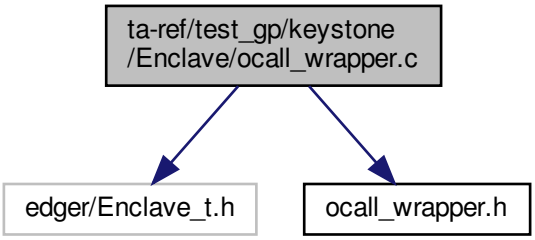
Go to the documentation of this file.

```
1 int puts(const char *s);
2 int putchar(int c);
3 int printf(const char* fmt, ...);
```

10.138 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)

10.138.1 Function Documentation

10.138.1.1 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

str	The string value for print.

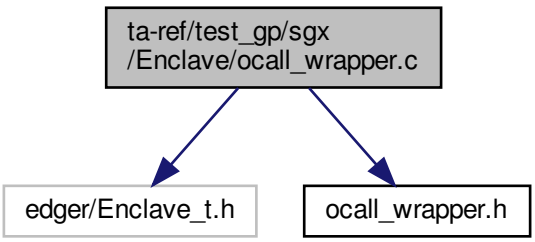
Returns

string It prints the value of str by calling ocall_print_string().

10.139 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)

10.139.1 Function Documentation

10.139.1.1 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

str	The string value for print.

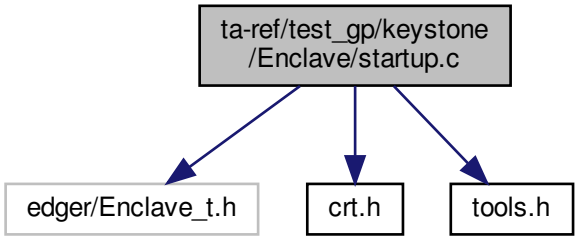
Returns

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

10.140 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 ()

10.140.1 Function Documentation

10.140.1.1 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().

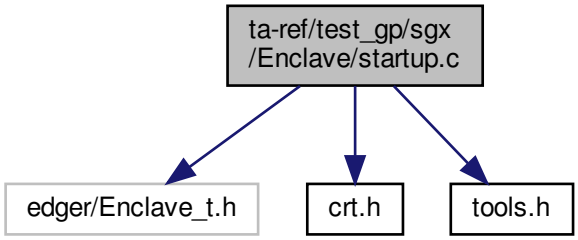
Returns

It will return EAPP_RETURN(0).

10.141 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)

10.141.1 Function Documentation

10.141.1.1 ecall_ta_main() `void ecall_ta_main ( 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().

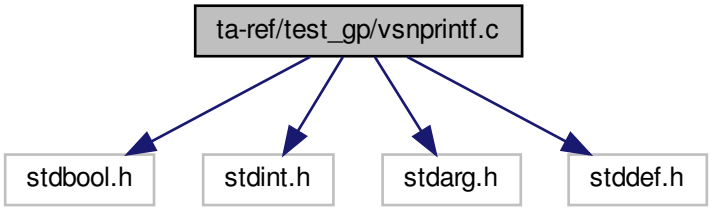
Returns

It will return EAPP_RETURN(0).

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

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

Include dependency graph for vsnprintf.c:



Classes

- struct out_fct_wrap_type

Macros

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

Typedefs

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

Functions

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

10.142.1 Macro Definition Documentation

10.142.1.1 _putchar `#define _putchar putchar`

10.142.1.2 FLAGS_CHAR `#define FLAGS_CHAR (1U << 6U)`

10.142.1.3 FLAGS_HASH `#define FLAGS_HASH (1U << 4U)`

10.142.1.4 FLAGS_LEFT `#define FLAGS_LEFT (1U << 1U)`

10.142.1.5 FLAGS_LONG `#define FLAGS_LONG (1U << 8U)`

10.142.1.6 FLAGS_LONG_LONG

```
#define FLAGS_LONG_LONG (1U << 9U)
```

10.142.1.7 FLAGS_PLUS

```
#define FLAGS_PLUS (1U << 2U)
```

10.142.1.8 FLAGS_PRECISION

```
#define FLAGS_PRECISION (1U << 10U)
```

10.142.1.9 FLAGS_SHORT

```
#define FLAGS_SHORT (1U << 7U)
```

10.142.1.10 FLAGS_SPACE

```
#define FLAGS_SPACE (1U << 3U)
```

10.142.1.11 FLAGS_UPPERCASE

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

10.142.1.12 FLAGS_ZEROPAD

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

10.142.1.13 PRINTF_FTOA_BUFFER_SIZE

```
#define PRINTF_FTOA_BUFFER_SIZE 32U
```

10.142.1.14 PRINTF_NTOA_BUFFER_SIZE

```
#define PRINTF_NTOA_BUFFER_SIZE 32U
```

10.142.1.15 PRINTF_SUPPORT_FLOAT

```
#define PRINTF_SUPPORT_FLOAT
```

10.142.1.16 PRINTF_SUPPORT_LONG_LONG

```
#define PRINTF_SUPPORT_LONG_LONG
```

10.142.1.17 PRINTF_SUPPORT_PTRDIFF_T

```
#define PRINTF_SUPPORT_PTRDIFF_T
```

10.142.2 Typedef Documentation

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

10.142.3 Function Documentation

10.142.3.1 _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

str	string representation of an integral number.

Returns

i unsigned integer value.

10.142.3.2 _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 condtion 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 precsion 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 lenghth. Finally using the conditional statement executes pad leading zeros, handling the hash value, padding spaces up to given width and reverses the string.

Parameters

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

Returns

non integer value if success else error occur

10.142.3.3 _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

ch	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

10.142.3.4 _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 condtion pad spaces up to given width what specifies in that. It reverse the string and again append pad spaces up to given width.

Parameters

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

Returns

idx non integer value if success else error occur.

10.142.3.5 _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

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

Returns

idx non integer value if success else error occur.

10.142.3.6 _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

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

Returns

idx non integer value if success else error occur.

10.142.3.7 _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

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

10.142.3.8 _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 condtion is success then putchar() writes a character into stdout.

Parameters

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

10.142.3.9 _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

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

10.142.3.10 _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

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

10.142.3.11 _strlen() `static unsigned int _strlen (`

```
const char * str )  [inline], [static]
```

_strlen() - calculates the length of the string.

Parameters

str	str is an argument of type pointer.

Returns

string length if successfully executed,else error occured.

10.142.3.12 _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 fucntion firstly initializes the varibles of format specifers like flags, width, precsion in this they evaluate all the specifiers invidually.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 evalucation take process using if condition.

Parameters

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

Returns

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

10.142.3.13 fctprintf()

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

fctprintf() - Function is using the libary macros of variable aruguments like vastart 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

out	An output function which takes one character and an argument pointer.
arg	An argument pointer for user data passed to output function.
format	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.

10.142.3.14 putchar()

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

10.142.3.15 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

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

Returns

ret returns the ret value as an integer type.

10.142.3.16 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

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

Returns

ret It returns the ret value as an integer type.

10.142.3.17 vsnprintf()

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

vsnprintf() - Invokes another function called _vsnprintf(). with some arguments.

Parameters

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

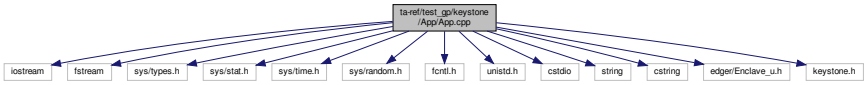
Returns

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

10.143 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"

10.143.1 Function Documentation

10.143.1.1 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 intialization by calling edge_call_init_internals() and then the enclave will run
and return 0.

Parameters

argc	Argument Count is int and stores number of command-line arguments passed by the user including the name of the program.
argv	Argument Vector is array of character pointers listing all the arguments.

Returns

0 If success, else error occurred.

10.143.2 Variable Documentation

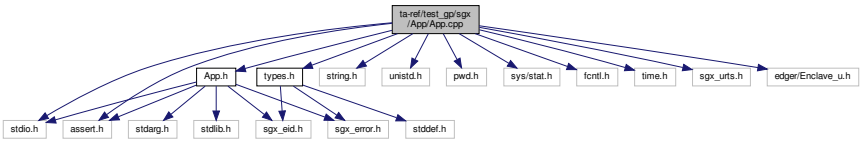
10.143.2.1 enc_path `const char* enc_path = "Enclave.eapp_riscv"`

10.143.2.2 runtime_path `const char* runtime_path = "eyrie-rt"`

10.144 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[ ])

10.144.1 Macro Definition Documentation

10.144.1.1 MAX_PATH `#define MAX_PATH FILENAME_MAX`

10.144.2 Function Documentation

10.144.2.1 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 occured.

10.144.2.2 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 intialized enclave's return value is less than zero then it destroys the enclave.

Parameters

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

Returns

0 If success, else error occured

10.144.2.3 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.

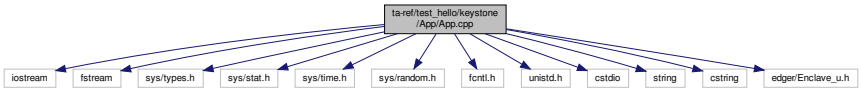
Parameters

ret	list containing all possible values of this data type.

10.145 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"

10.145.1 Function Documentation

10.145.1.1 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 intialization 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

argc	Argument count is int and stores number of command-line arguments passed by the user including the name of the program.
argv	Argument Vector is array of character pointers listing all the arguments.

Returns

0 If success, else error occurred.

10.145.2 Variable Documentation

10.145.2.1 enc_path `const char* enc_path = "Enclave.eapp_riscv"`

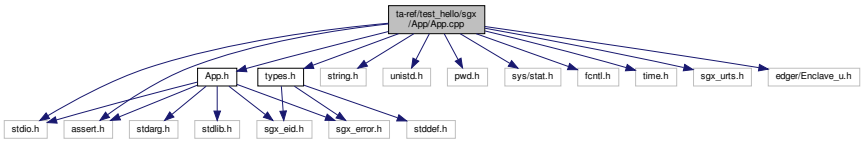
10.145.2.2 runtime_path `const char* runtime_path = "eyrie-rt"`

10.146 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[ ])

10.146.1 Macro Definition Documentation

10.146.1.1 MAX_PATH `#define MAX_PATH FILENAME_MAX`

10.146.2 Function Documentation

10.146.2.1 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 occured.

10.146.2.2 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

argc	Argument Count is int and stores number of command-line arguments passed by the user including the name of the program.
argv	Argument Vector is array of character pointers listing all the arguments.

Returns

0 If success, else error occured.

10.146.2.3 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.

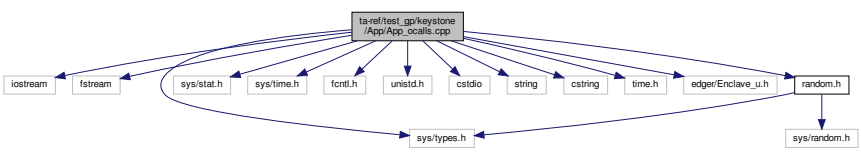
Parameters

ret	A list containing all possible values of sgx_status_t data type.

10.147 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 <cstdio>
#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

- bool load_invoke_command (invoke_command_t *ret)
- int store_invoke_callback_file (const char *name, const char *out, size_t out_len)
- EDGE_EXTERNC_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)

10.147.1 Macro Definition Documentation

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

10.147.2 Function Documentation

10.147.2.1 load_invoke_command() `EDGE_EXTERNC_END bool load_invoke_command ( invoke_command_t * ret )`

load_invoke_command() - To load the invoke command.

This function is to open and read the invoke csv file and remove file to reset, then load commandID and value and break a string into tokens.

Parameters

ret	pointer of invoke_command_t

@retrn true on success,else false appears.

10.147.2.2 ocall_close_file() `int ocall_close_file ( int fdesc )`

ocall_close_file() - Frees the file descriptor in the process.

Parameters

fdesc	fdesc is a file descriptor of the type integer.

Returns

rtn on success,-1 on failure.

10.147.2.3 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

buf	buf is a character datatype
len	len is a size_t datatype
flags	flags is a unsigned int datatype

Returns

the number of bytes stored in buf, -1 on failure.

10.147.2.4 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

str	str is a constant character data type.
buf	buf is a constant character data type.
len	len is a unsigned int type.

Returns

0 on success else, error occurred.

10.147.2.5 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

fname	file name is a constant character data type
flags	flags it is datatype of the integer
perm	permissions of the file if it is created

Returns

a nonnegative integer for success or -1 if an error occurred.

10.147.2.6 ocall_print_string()

```
EDGE_EXTERNC_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

str	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

str	Pointer of the string.

Returns

length If success, else error occured.

10.147.2.7 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

fdesc	an open file descriptor
buf	buffer of at least size bytes
len	number of bytes to be read.

Returns

number of bytes read on success, -1 on failure.

10.147.2.8 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

timep	timep is a structure type of ree_time_t

Returns

rtn value on success

10.147.2.9 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

fdesc	file descriptor
buf	buffer of at least size bytes
len	number of bytes to be write.

Returns

number of bytes written on success,-1 on failure.

10.147.2.10 store_invoke_callback_file()

```
int store_invoke_callback_file (
    const char * name,
    const char * out,
    size_t out_len )
```

store_invoke_callback_file() - To store the callback file.

In this function The opened file has been assigned to ”desc”.If desc is less than 0 it goes for file write mode else failed to open file error appears. It returns 0 on the successful execution of read,write and close of a file .If it fails error appears.

Parameters

name	name of the file.
out	buffer output.
out_len	Length of the file

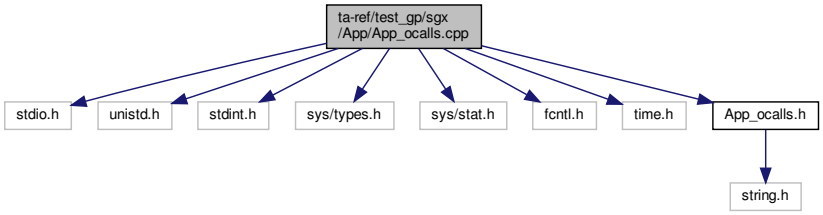
Returns

0 if success,else error appears.

10.148 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)

10.148.1 Macro Definition Documentation

10.148.1.1 MAX_PATH `#define MAX_PATH FILENAME_MAX`

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

10.148.2 Function Documentation

10.148.2.1 ocall_close_file() `int ocall_close_file ( int desc )`

ocall_close_file() - Used for closing a file

Parameters

desc	File descriptor.

Returns

file descripto If success, else error occured.

10.148.2.2 ocall_open_file()

```
int ocall_open_file (
    const char * fname,
    int flags,
    int perm )
```

ocall_open_file() - Used for opening a file.

Parameters

fname	File name
flags	Values for oflag are constructed by a bitwise-inclusive OR of flags from the following list.
perm	permision or mode

Returns

file descriptor If success, else error occured

10.148.2.3 ocall_print_string()

```
unsigned int NO_PERF ocall_print_string (
    const char * str )
```

ocall_print_string() - To print the argument string message.

Parameters

str	Pointer of the string.

Returns

length If success, else error occured.

10.148.2.4 ocall_read_file() int ocall_read_file (

```
int desc,
char * buf,
size_t len )
```

ocall_read_file() - Used to read from a file.

Parameters

desc	file descriptor
buf	pointer to a buffer
len	Size of elements

Returns

file descriptor If success, else error occured

10.148.2.5 ocall_ree_time() int ocall_ree_time (

```
struct ree_time_t * time )
```

ocall_ree_time() - Used to fetch the current time.

Parameters

time	Pointer to a current time.

Returns

current time If success, else error occurred

10.148.2.6 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

desc	file descriptor.
buf	pointer to a buffer.
len	Size of elements.

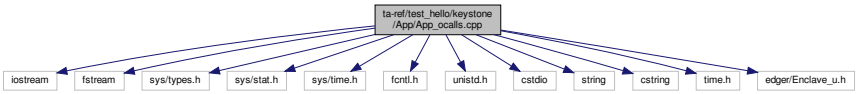
Returns

file descriptor If success, else error occured.

10.149 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_EXTERNC_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)

10.149.1 Function Documentation

10.149.1.1 ocall_close_file() `int ocall_close_file ( int fdesc )`

ocall_close_file() - To close a file.

Parameters

fdesc	file descriptor.

Returns

integer value If success

10.149.1.2 ocall_getrandom()

```
ssize_t ocall_getrandom (
    char * buf,
    size_t len,
    unsigned int flags )
```

ocall_getrandom() - To get random data.

Parameters

buf	Pointer of a buffer
len	length of buffer
flags	indicated permission.

Returns

integer value If success

10.149.1.3 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

str	pointer of a string.
buf	buffer to write data.
len	length of buffer.

Returns

integer value If success

10.149.1.4 ocall_open_file()

```
int ocall_open_file (
    const char * fname,
```

```
            int flags,
            int perm )
```

ocall_open_file() - To open a file.

Parameters

fname	name of the file.
flags	mode of the file.
perm	indicates permissions of a file.

Returns

integer If success

10.149.1.5 ocall_print_string()

```
EDGE_EXTERNC_BEGIN unsigned int ocall_print_string (
    const char * str )
```

ocall_print_string() - To print the string and returns the length of string.

Parameters

str	The string to print.

Returns

str length of the string.

10.149.1.6 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

fdesc	file descripter.
buf	buffer to write data.
len	length of buffer

Returns

integer value If success

10.149.1.7 ocall_ree_time() int ocall_ree_time (struct ree_time_t * timep)

ocall_ree_time() - gets the ree execution time.

Parameters

timep	pointer of time.

Returns

integer value If success

10.149.1.8 ocall_write_file() int ocall_write_file (int fdesc, const char * buf, unsigned int len)

ocall_write_file() - To write data in to a file.

Parameters

fdesc	file descripter.
buf	buffer to write data.
len	length of buffer.

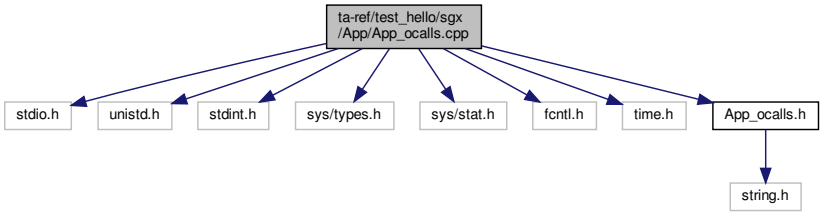
Returns

integer value If success

10.150 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)

10.150.1 Function Documentation

10.150.1.1 ocall_close_file()

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

ocall_close_file() - To close a file.

Parameters

desc	file descriptor.

Returns

integer value If success

10.150.1.2 ocall_open_file()

```
int ocall_open_file (
    const char * fname,
    int flags,
    int perm )
```

ocall_open_file() - To open a file.

Parameters

fname	name of the file.
flags	mode of the file.
perm	indicates permissions of a file.

Returns

integer value If success

10.150.1.3 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

str	Pointer of the string.

Returns

length If success, else error occured.

10.150.1.4 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

desc	file descripter.
buf	buffer to write data.
len	length of buffer

Returns

integer value If success

10.150.1.5 ocall_ree_time() `int ocall_ree_time ( struct ree_time_t * time )`

ocall_ree_time() - gets the ree execution time.

Parameters

time	pointer of time.

Returns

integer value If success

10.150.1.6 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

desc	file descripter.
buf	buffer to write data.
len	length of buffer.

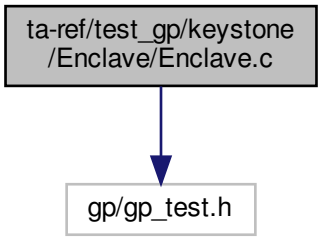
Returns

integer value If success

10.151 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)

10.151.1 Function Documentation

10.151.1.1 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.

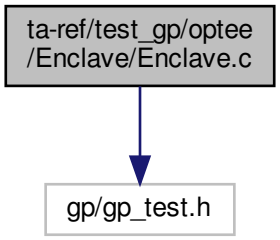
Returns

return 0 for success.

10.152 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)

10.152.1 Function Documentation

10.152.1.1 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.

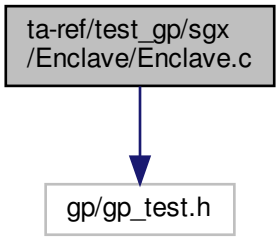
Returns

return 0 for success.

10.153 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)

10.153.1 Function Documentation

10.153.1.1 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.

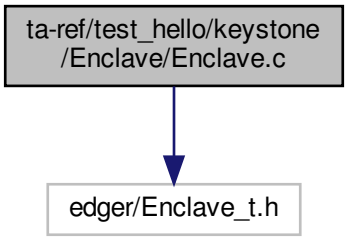
Returns

return 0 for success.

10.154 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 ()

10.154.1 Macro Definition Documentation

10.154.1.1 MESSAGE `#define MESSAGE "hello world!\n"`

10.154.2 Function Documentation

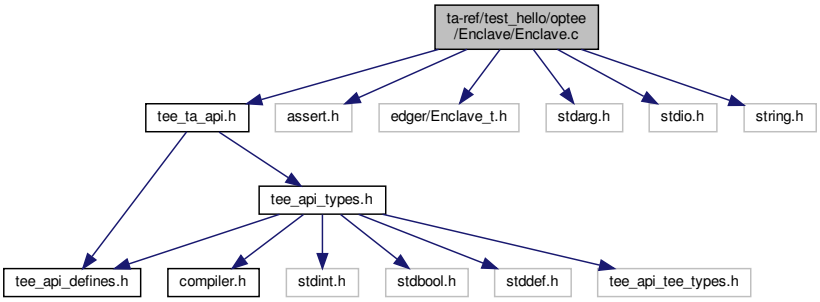
10.154.2.1 eapp_entry() `void EAPP_ENTRY eapp_entry ( )`

eapp_entry() - This function is used for printing the Message.

10.155 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

10.155.1 Macro Definition Documentation

10.155.1.1 BUF_SIZE `#define BUF_SIZE 8192`

10.155.1.2 MESSAGE `#define MESSAGE "hello world!\n"`

10.155.1.3 TEE_PARAM_TYPE1 `#define TEE_PARAM_TYPE1 TEE_PARAM_TYPE_MEMREF_OUTPUT`

10.155.2 Function Documentation

10.155.2.1 _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

str	This is the string whose length is to be found.

Returns

string length If success, else error occurred.

10.155.2.2 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

param_types	The types of the four parameters.
params[4]	A pointer to an array of four parameters.
sess_ctx	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.

10.155.2.3 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

sess_ctx	The value of the void* opaque data pointer set by the Trusted Application in this TA_OpenSessionEntryPoint() for this session.

10.155.2.4 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.

10.155.2.5 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.

10.155.2.6 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

sess_ctx	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.

10.155.2.7 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

param_types	The types of the four parameters.
params	A pointer to an array of four parameters.
sess_ctx	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.

10.155.2.8 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

fmt	A string that specifies the format of the output.

Returns

result If success, else error occured.

10.155.3 Variable Documentation

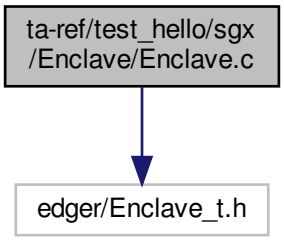
10.155.3.1 print_buf `char print_buf[BUF_SIZE]`

10.155.3.2 print_pos `size_t print_pos`

10.156 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)

10.156.1 Macro Definition Documentation

10.156.1.1 MESSAGE `#define MESSAGE "hello world!\n"`

10.156.2 Function Documentation

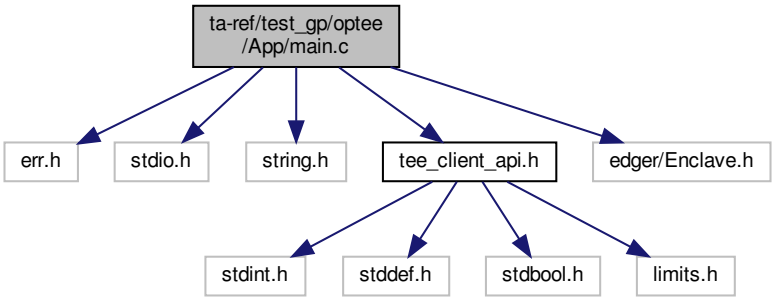
10.156.2.1 ecall_ta_main() `void ecall_ta_main ( void )`

ecall_ta_main() - Prints the string and returns the number of string.

10.157 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)

10.157.1 Macro Definition Documentation

10.157.1.1 BUF_SIZE `#define BUF_SIZE 65536`

10.157.1.2 PRINT_BUF_SIZE `#define PRINT_BUF_SIZE 16384`

10.157.1.3 TEEC_PARAM_TYPE0 `#define TEEC_PARAM_TYPE0 TEEC_NONE`

10.157.1.4 TEEC_PARAM_TYPE1 `#define TEEC_PARAM_TYPE1 TEEC_NONE`

10.157.2 Function Documentation

10.157.2.1 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.

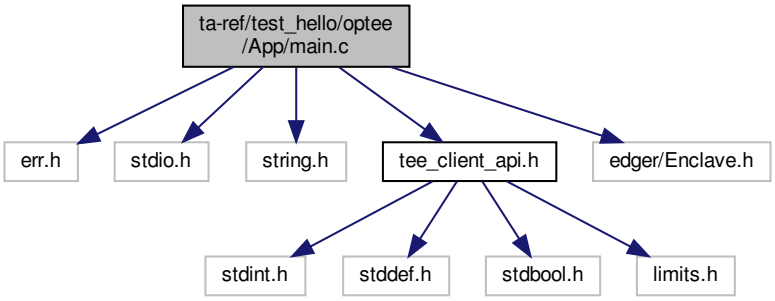
Returns

0 If the function is a success.

10.158 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]

10.158.1 Macro Definition Documentation

10.158.1.1 PRINT_BUF_SIZE `#define PRINT_BUF_SIZE 16384`

10.158.1.2 TEEC_PARAM_TYPE1 `#define TEEC_PARAM_TYPE1 TEEC_MEMREF_TEMP_OUTPUT`

10.158.2 Function Documentation

10.158.2.1 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.

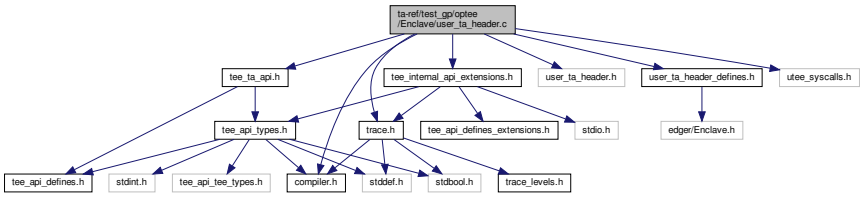
10.158.3 Variable Documentation

10.158.3.1 print_buf `char print_buf[PRINT_BUF_SIZE] [static]`

10.159 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])

10.159.1 Macro Definition Documentation

10.159.1.1 _C_FUNCTION `#define _C_FUNCTION( name ) name`

10.159.1.2 TA_DESCRIPTION `#define TA_DESCRIPTION "Undefined description"`

10.159.1.3 TA_FRAMEWORK_STACK_SIZE `#define TA_FRAMEWORK_STACK_SIZE 2048`

10.159.1.4 TA_VERSION `#define TA_VERSION "Undefined version"`

10.159.2 Function Documentation

10.159.2.1 __section() `const struct ta_head ta_head __section ( ".ta_head" )`

10.159.2.2 __ta_entry()

```
void __noreturn _C_FUNCTION() __ta_entry (
    unsigned long func,
    unsigned long session_id,
    struct utee_params * up,
    unsigned long cmd_id )
```

10.159.2.3 __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

func	func is the unsigned long data type.
session_id	session_id is the unsigned long data type.
up	up is the structure type of the utee_params.
cmd_id	cmd_id is the unsigned long data type.

Returns

TEE_SUCCESS If the command is successfully executed.

10.159.2.4 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 occured.

10.159.3 Variable Documentation

10.159.3.1 ta_heap `uint8_t ta_heap[TA_DATA_SIZE]`

10.159.3.2 ta_heap_size `const size_t ta_heap_size = sizeof(ta_heap)`

10.159.3.3 ta_num_props `const size_t ta_num_props = sizeof(ta_props) / sizeof(ta_props[0])`

10.159.3.4 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},
}
```

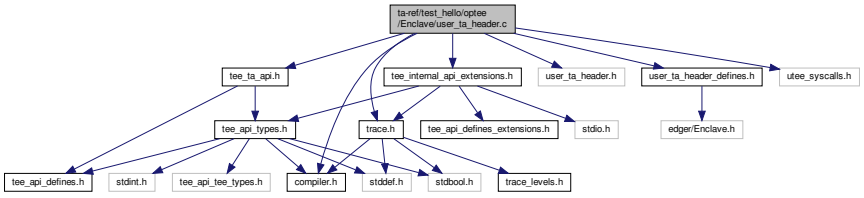
10.159.3.5 trace_ext_prefix `const char trace_ext_prefix[] = "TA"`

10.159.3.6 trace_level `int trace_level = TRACE_LEVEL`

10.160 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])

10.160.1 Macro Definition Documentation

10.160.1.1 _C_FUNCTION `#define _C_FUNCTION( name ) name`

10.160.1.2 TA_DESCRIPTION `#define TA_DESCRIPTION "Undefined description"`

10.160.1.3 TA_FRAMEWORK_STACK_SIZE `#define TA_FRAMEWORK_STACK_SIZE 2048`

10.160.1.4 TA_VERSION `#define TA_VERSION "Undefined version"`

10.160.2 Function Documentation

10.160.2.1 __section() const struct ta_head ta_head __section (".ta_head")

10.160.2.2 __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

func	Function
session_id	Session id
up	object of struct utee_params
cmd_id	command input id

10.160.2.3 __utee_entry() TEE_Result __utee_entry (unsigned long *func*, unsigned long *session_id*, struct utee_params * *up*, unsigned long *cmd_id*)

10.160.2.4 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.

10.160.3 Variable Documentation

10.160.3.1 ta_heap uint8_t ta_heap[TA_DATA_SIZE]

10.160.3.2 ta_heap_size

```
const size_t ta_heap_size = sizeof(ta_heap)
```

10.160.3.3 ta_num_props

```
const size_t ta_num_props = sizeof(ta_props) / sizeof(ta_props[0])
```

10.160.3.4 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},
}
```

10.160.3.5 trace_ext_prefix

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

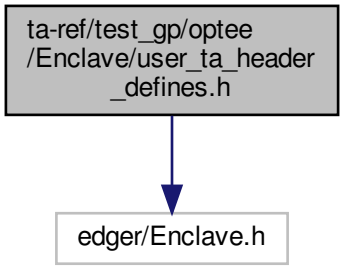
10.160.3.6 trace_level

```
int trace_level = TRACE_LEVEL
```

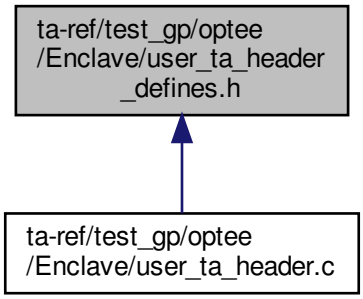
10.161 ta-ref/test_gp/optee/Enclave/user_ta_header_defines.h File Reference

```
#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

10.161.1 Macro Definition Documentation

10.161.1.1 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 } }
```

10.161.1.2 TA_DATA_SIZE `#define TA_DATA_SIZE (32 * 1024)`

10.161.1.3 TA_DESCRIPTION `#define TA_DESCRIPTION "Example of OP-TEE TEST Trusted Application"`

10.161.1.4 TA_FLAGS `#define TA_FLAGS TA_FLAG_EXEC_DDR`

10.161.1.5 TA_STACK_SIZE `#define TA_STACK_SIZE (2 * 1024)`

10.161.1.6 TA_UUID `#define TA_UUID TA_REF_UUID`

10.161.1.7 TA_VERSION `#define TA_VERSION "1.0"`

10.162 user_ta_header_defines.h

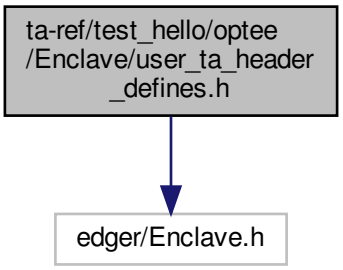
Go to the documentation of this file.

```
1 /*
2  * Copyright (c) 2016-2017, Linaro Limited
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10  *
11  * 2. Redistributions in binary form must reproduce the above copyright notice,
12  * this list of conditions and the following disclaimer in the documentation
13  * and/or other materials provided with the distribution.
14  *
15  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25  * POSSIBILITY OF SUCH DAMAGE.
26  */
27
28 /*
29  * The name of this file must not be modified
30  */
31
32 #ifndef USER_TA_HEADER_DEFINES_H
33 #define USER_TA_HEADER_DEFINES_H
34
35 /* To get the TA UUID definition */
36 #include "edger/Enclave.h"
37
38 #define TA_UUID             TA_REF_UUID
39
40 /*
41  * TA properties: multi-instance TA, no specific attribute
42  * TA_FLAG_EXEC_DDR is meaningless but mandated.
43  */
44 #define TA_FLAGS            TA_FLAG_EXEC_DDR
45
46 /* Provisioned stack size */
47 #define TA_STACK_SIZE           (2 * 1024)
48
49 /* Provisioned heap size for TEE_Malloc() and friends */
50 #define TA_DATA_SIZE            (32 * 1024)
51
52 /* The gpd.ta.version property */
53 #define TA_VERSION  "1.0"
54
```

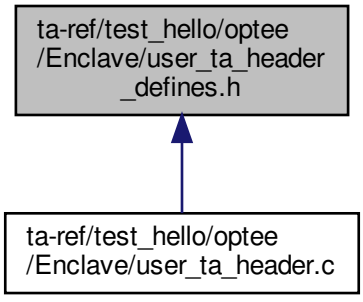
```
55 /* The gpd.ta.description property */
56 #define TA_DESCRIPTION  "Example of OP-TEE TEST Trusted Application"
57
58 /* Extra properties */
59 #define TA_CURRENT_TA_EXT_PROPERTIES \
60     { "org.linaro.optee.examples.test.property1", \
61     USER_TA_PROP_TYPE_STRING, \
62         "Some string" }, \
63     { "org.linaro.optee.examples.test.property2", \
64     USER_TA_PROP_TYPE_U32, &(const uint32_t){ 0x0010 } }
65
66 #endif /* USER_TA_HEADER_DEFINES_H */
```

10.163 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

10.163.1 Macro Definition Documentation

10.163.1.1 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 } }
```

10.163.1.2 TA_DATA_SIZE #define TA_DATA_SIZE (32 * 1024)

10.163.1.3 TA_DESCRIPTION #define TA_DESCRIPTION "Example of OP-TEE TEST Trusted Application"

10.163.1.4 TA_FLAGS #define TA_FLAGS TA_FLAG_EXEC_DDR

10.163.1.5 TA_STACK_SIZE #define TA_STACK_SIZE (2 * 1024)

10.163.1.6 TA_UUID #define TA_UUID TA_REF_UUID

10.163.1.7 TA_VERSION #define TA_VERSION "1.0"

10.164 user_ta_header_defines.h

Go to the documentation of this file.

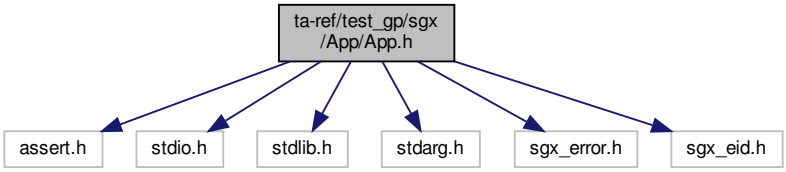
```
1 /*
2  * Copyright (c) 2016-2017, Linaro Limited
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10  *
11  * 2. Redistributions in binary form must reproduce the above copyright notice,
12  * this list of conditions and the following disclaimer in the documentation
13  * and/or other materials provided with the distribution.
14  *
15  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25  * POSSIBILITY OF SUCH DAMAGE.
26  */
27
28 /*
29  * The name of this file must not be modified
30  */
31
32 #ifndef USER_TA_HEADER_DEFINES_H
33 #define USER_TA_HEADER_DEFINES_H
34
35 /* To get the TA UUID definition */
36 #include "edger/Enclave.h"
37
38 #define TA_UUID             TA_REF_UUID
39
40 /*
41  * TA properties: multi-instance TA, no specific attribute
42  * TA_FLAG_EXEC_DDR is meaningless but mandated.
43  */
44 #define TA_FLAGS            TA_FLAG_EXEC_DDR
45
46 /* Provisioned stack size */
47 #define TA_STACK_SIZE           (2 * 1024)
48
49 /* Provisioned heap size for TEE_Malloc() and friends */
50 #define TA_DATA_SIZE            (32 * 1024)
51
52 /* The gpd.ta.version property */
53 #define TA_VERSION  "1.0"
54
55 /* The gpd.ta.description property */
56 #define TA_DESCRIPTION  "Example of OP-TEE TEST Trusted Application"
57
58 /* Extra properties */
59 #define TA_CURRENT_TA_EXT_PROPERTIES \
60     { "org.linaro.optee.examples.test.property1", \
61     USER_TA_PROP_TYPE_STRING, \
62         "Some string" }, \
63     { "org.linaro.optee.examples.test.property2", \
64     USER_TA_PROP_TYPE_U32, &(const uint32_t){ 0x0010 } }
65
66 #endif /* USER_TA_HEADER_DEFINES_H */
```

10.165 ta-ref/test_gp/sgx/App/App.h File Reference

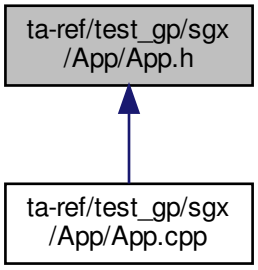
```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "sgx_error.h"
```

```
#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

10.165.1 Macro Definition Documentation

10.165.1.1 ENCLAVE_FILENAME `#define ENCLAVE_FILENAME "enclave.signed.so"`

10.165.1.2 FALSE `#define FALSE 0`

10.165.1.3 TRUE `#define TRUE 1`

10.165.2 Variable Documentation

10.165.2.1 global_eid `sgx_enclave_id_t global_eid [extern]`

10.166 App.h

Go to the documentation of this file.

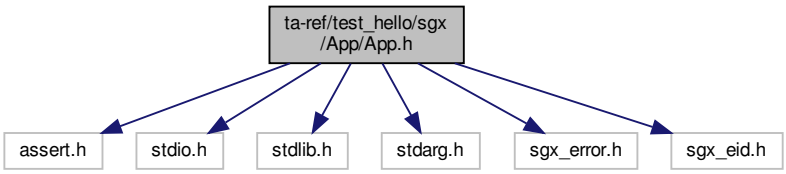
```
1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                          and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification,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

```
57 #endif
58
59 // sgx_status_t ecall_ta_main(sgx_enclave_id_t eid);
60
61 #if defined(__cplusplus)
62 }
63 #endif
64
65 #endif /* !_APP_H_ */
```

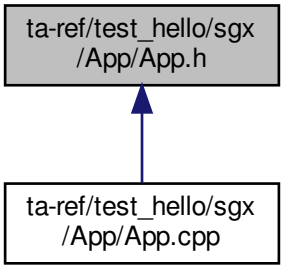
10.167 ta-ref/test_hello/sgx/App/App.h File Reference

```
#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

10.167.1 Macro Definition Documentation

10.167.1.1 ENCLAVE_FILENAME `#define ENCLAVE_FILENAME "enclave.signed.so"`

10.167.1.2 FALSE `#define FALSE 0`

10.167.1.3 TRUE `#define TRUE 1`

10.167.2 Variable Documentation

10.167.2.1 global_eid `sgx_enclave_id_t global_eid [extern]`

10.168 App.h

Go to the documentation of this file.

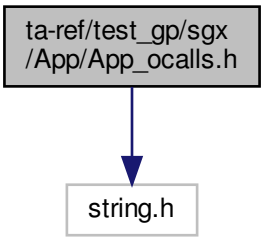
```
1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                          and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10  *
11  * 1. Redistributions of source code must retain the above copyright notice,
12  * this list of conditions and the following disclaimer.
13  *
14  * 2. Redistributions in binary form must reproduce the above copyright notice,
15  * this list of conditions and the following disclaimer in the documentation
16  * and/or other materials provided with the distribution.
17  *
18  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28  * POSSIBILITY OF SUCH DAMAGE.
29  */
30
31 #ifndef _APP_H_
32 #define _APP_H_
```

```
33
34 #include <assert.h>
35 #include <stdio.h>
36 #include <stdlib.h>
37 #include <stdarg.h>
38
39 #include "sgx_error.h"       /* sgx_status_t */
40 #include "sgx_eid.h"     /* sgx_enclave_id_t */
41
42 #ifndef TRUE
43 # define TRUE 1
44 #endif
45
46 #ifndef FALSE
47 # define FALSE 0
48 #endif
49
50 //# define TOKEN_FILENAME   "enclave.token"
51 # define ENCLAVE_FILENAME "enclave.signed.so"
52
53 extern sgx_enclave_id_t global_eid;    /* global enclave id */
54
55 #if defined(__cplusplus)
56 extern "C" {
57 #endif
58
59 // sgx_status_t ecall_ta_main(sgx_enclave_id_t eid);
60
61 #if defined(__cplusplus)
62 }
63 #endif
64
65 #endif /* !_APP_H_ */
```

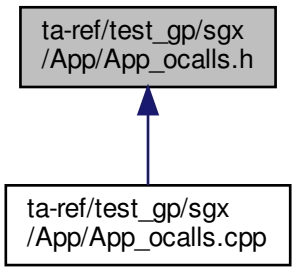
10.169 ta-ref/test_gp/sgx/App/App_ocalls.h File Reference

`#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)

10.169.1 Typedef Documentation

10.169.1.1 ree_time_t `typedef struct ree_time_t ree_time_t`

10.169.2 Function Documentation

10.169.2.1 ocall_close_file() `int ocall_close_file ( int desc )`

ocall_close_file() - To close a file.

Parameters

fdesc	file descriptor.

Returns

integer value If success

ocall_close_file() - To close a file.

Parameters

desc	file descriptor.

Returns

integer value If success

ocall_close_file() - Frees the file descriptor in the process.

Parameters

fdesc	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

desc	File descriptor.

Returns

file descripto If success, else error occured.

10.169.2.2 ocall_open_file()

```
int ocall_open_file (
    const char * fname,
    int flags,
    int perm )
```

ocall_open_file() - To open a file.

Parameters

fname	name of the file.
flags	mode of the file.
perm	indicates permissions of a file.

Returns

integer If success

ocall_open_file() - To open a file.

Parameters

fname	name of the file.
flags	mode of the file.
perm	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

fname	file name is a constant character data type
flags	flags it is datatype of the integer
perm	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

fname	File name
flags	Values for oflag are constructed by a bitwise-inclusive OR of flags from the following list.
perm	permision or mode

Returns

file descriptor If success, else error occured

10.169.2.3 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

str	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

str	Pointer of the string.

Returns

length If success, else error occured.

ocall_print_string() - To print the string and returns the length of string.

Parameters

str	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

str	Pointer of the string.

Returns

length If success, else error occured.

ocall_print_string() - To print the argument string message.

Parameters

str	Pointer of the string.

Returns

length If success, else error occured.

10.169.2.4 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

fdesc	file descripter.
buf	buffer to write data.
len	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

desc	file descripter.
buf	buffer to write data.
len	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

fdesc	an open file descriptor
buf	buffer of at least size bytes
len	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

desc	file descriptor
buf	pointer to a buffer
len	Size of elements

Returns

file descriptor If success, else error occured

10.169.2.5 ocall_ree_time() `int ocall_ree_time ( struct ree_time_t * time )`

ocall_ree_time() - gets the ree execution time.

Parameters

timep	pointer of time.

Returns

integer value If success

ocall_ree_time() - gets the ree execution time.

Parameters

time	pointer of time.

Returns

integer value If success

ocall_ree_time() - Function shall obtain the current time, expressed as seconds and microseconds.

Parameters

timep	timep is a structure type of ree_time_t

Returns

rtn value on success

ocall_ree_time() - Used to fetch the current time.

Parameters

time	Pointer to a current time.

Returns

current time If success, else error occurred

10.169.2.6 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

desc	file descripter.
buf	buffer to write data.
len	length of buffer.

Returns

integer value If success

ocall_write_file() - Used to write into a file.

Parameters

desc	file descriptor.
buf	pointer to a buffer.
len	Size of elements.

Returns

file descriptor If success, else error occured.

10.170 App_ocalls.h

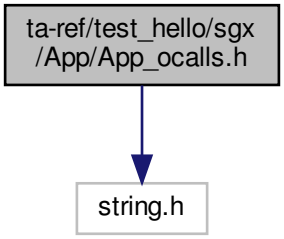
Go to the documentation of this file.

```
1 #pragma once
2 #include <string.h>
3
4 #ifdef __cplusplus
5 extern "C" {
6 #endif
7
8 typedef struct ree_time_t {
9     unsigned int seconds;
10     unsigned int millis;
11 } ree_time_t;
12 /* OCall functions */
13 unsigned int ocall_print_string(const char *str);
14 int ocall_open_file(const char *fname, int flags, int perm);
15
16 int ocall_read_file(int desc, char *buf, size_t len);
17 int ocall_write_file(int desc, const char *buf, size_t len);
18 int ocall_close_file(int desc);
19 int ocall_ree_time(struct ree_time_t *time);
20
21 #ifdef __cplusplus
22 }
23 #endif /* __cplusplus */
```

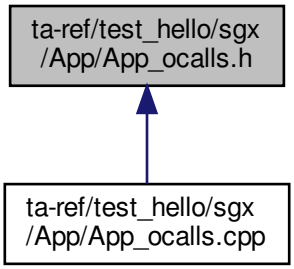
10.171 ta-ref/test_hello/sgx/App/App_ocalls.h File Reference

```
#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)

10.171.1 Typedef Documentation

10.171.1.1 ree_time_t `typedef struct ree_time_t ree_time_t`

10.171.2 Function Documentation

10.171.2.1 ocall_close_file() `int ocall_close_file ( int desc )`

ocall_close_file() - To close a file.

Parameters

fdesc	file descriptor.

Returns

integer value If success

ocall_close_file() - To close a file.

Parameters

desc	file descriptor.

Returns

integer value If success

ocall_close_file() - Frees the file descriptor in the process.

Parameters

fdesc	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

desc	File descriptor.

Returns

file descripto If success, else error occured.

10.171.2.2 ocall_open_file()

```
int ocall_open_file (
  const char * fname,
  int flags,
  int perm )
```

ocall_open_file() - To open a file.

Parameters

fname	name of the file.
flags	mode of the file.
perm	indicates permissions of a file.

Returns

integer If success

ocall_open_file() - To open a file.

Parameters

fname	name of the file.
flags	mode of the file.
perm	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

fname	file name is a constant character data type
flags	flags it is datatype of the integer
perm	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

fname	File name
flags	Values for oflag are constructed by a bitwise-inclusive OR of flags from the following list.
perm	permision or mode

Returns

file descriptor If success, else error occured

10.171.2.3 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

str	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

str	Pointer of the string.

Returns

length If success, else error occured.

ocall_print_string() - To print the argument string message.

Parameters

str	Pointer of the string.

Returns

length If success, else error occured.

10.171.2.4 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

fdesc	file descripter.
buf	buffer to write data.
len	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

desc	file descripter.
buf	buffer to write data.
len	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

fdesc	an open file descriptor
buf	buffer of at least size bytes
len	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

desc	file descriptor
buf	pointer to a buffer
len	Size of elements

Returns

file descriptor If success, else error occured

10.171.2.5 ocall_ree_time() `int ocall_ree_time ( struct ree_time_t * time )`

ocall_ree_time() - gets the ree execution time.

Parameters

timep	pointer of time.

Returns

integer value If success

ocall_ree_time() - gets the ree execution time.

Parameters

time	pointer of time.

Returns

integer value If success

ocall_ree_time() - Function shall obtain the current time, expressed as seconds and microseconds.

Parameters

timep	timep is a structure type of ree_time_t

Returns

rtn value on success

ocall_ree_time() - Used to fetch the current time.

Parameters

time	Pointer to a current time.

Returns

current time If success, else error occurred

10.171.2.6 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

desc	file descripter.
buf	buffer to write data.
len	length of buffer.

Returns

integer value If success

ocall_write_file() - Used to write into a file.

Parameters

desc	file descriptor.
buf	pointer to a buffer.
len	Size of elements.

Returns

file descriptor If success, else error occured.

10.172 App_ocalls.h

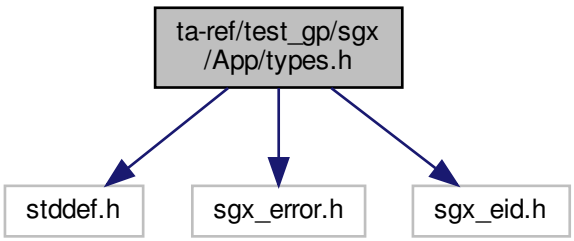
Go to the documentation of this file.

```
1 #pragma once
2 #include <string.h>
3
4 #ifdef __cplusplus
5 extern "C" {
6 #endif
7
8 typedef struct ree_time_t {
9     unsigned int seconds;
10     unsigned int millis;
11 } ree_time_t;
12 /* OCall functions */
13 unsigned int ocall_print_string(const char *str);
14 int ocall_open_file(const char *fname, int flags, int perm);
15
16 int ocall_read_file(int desc, char *buf, size_t len);
17 int ocall_write_file(int desc, const char *buf, size_t len);
18 int ocall_close_file(int desc);
19 int ocall_ree_time(struct ree_time_t *time);
20
21 #ifdef __cplusplus
22 }
23 #endif /* __cplusplus */
```

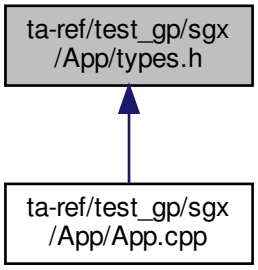
10.173 ta-ref/test_gp/sgx/App/types.h File Reference

```
#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 [ ]

10.173.1 Typedef Documentation

10.173.1.1 sgx_errlist_t `typedef struct _sgx_errlist_t sgx_errlist_t`

10.173.2 Variable Documentation

10.173.2.1 global_eid `sgx_enclave_id_t global_eid = 0`

10.173.2.2 sgx_errlist `sgx_errlist_t sgx_errlist[ ] [static]`

10.174 types.h

Go to the documentation of this file.

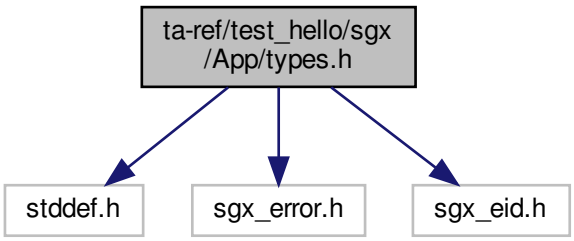
```
1 #pragma once
2 #include <stddef.h>
3 #include "sgx_error.h"       /* sgx_status_t */
4 #include "sgx_eid.h"     /* sgx_enclave_id_t */
5
6 /* Global EID shared by multiple threads */
7 sgx_enclave_id_t global_eid = 0;
8
9 typedef struct _sgx_errlist_t {
10     sgx_status_t err;
11     const char *msg;
12     const char *sug; /* Suggestion */
13 } sgx_errlist_t;
14
15 /* Error code returned by sgx_create_enclave */
16 static sgx_errlist_t sgx_errlist[] = {
17     {
18         SGX_ERROR_UNEXPECTED,
19         "Unexpected error occurred.",
20         NULL
21     },
22     {
23         SGX_ERROR_INVALID_PARAMETER,
24         "Invalid parameter.",
25         NULL
26     },
27     {
28         SGX_ERROR_OUT_OF_MEMORY,
29         "Out of memory.",
30         NULL
31     },
32     {
33         SGX_ERROR_ENCLAVE_LOST,
34         "Power transition occurred.",
35         "Please refer to the sample \"PowerTransition\" for details."
36     },
37     {
38         SGX_ERROR_INVALID_ENCLAVE,
39         "Invalid enclave image.",
40         NULL
41     },
42     {
43         SGX_ERROR_INVALID_ENCLAVE_ID,
44         "Invalid enclave identification.",
45         NULL
46     },
47     {
48         SGX_ERROR_INVALID_SIGNATURE,
49         "Invalid enclave signature.",
50         NULL
51     },
52     {
53         SGX_ERROR_OUT_OF_EPC,
54         "Out of EPC memory.",
55         NULL
56     },
57     {
58         SGX_ERROR_NO_DEVICE,
59         "Invalid SGX device.",
60         "Please make sure SGX module is enabled in the BIOS, and install SGX driver afterwards."
61     },
62     {
63         SGX_ERROR_MEMORY_MAP_CONFLICT,
64         "Memory map conflicted.",
65         NULL
66     },
67     {
68         SGX_ERROR_INVALID_METADATA,
69         "Invalid enclave metadata.",
70         NULL
71     },
72     {
73         SGX_ERROR_DEVICE_BUSY,
74         "SGX device was busy.",
75         NULL
76     },
77     {
78         SGX_ERROR_INVALID_VERSION,
79         "Enclave version was invalid.",
80         NULL
81     },
82     {
83         SGX_ERROR_INVALID_ATTRIBUTE,
```

```
84          "Enclave was not authorized.",
85          NULL
86      },
87      {
88          SGX_ERROR_ENCLAVE_FILE_ACCESS,
89          "Can't open enclave file.",
90          NULL
91      },
92  };
```

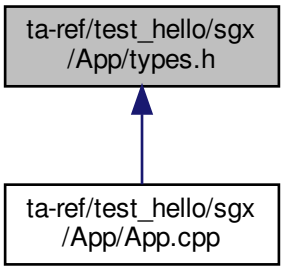
10.175 ta-ref/test_hello/sgx/App/types.h File Reference

```
#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 [ ]

10.175.1 Typedef Documentation

10.175.1.1 sgx_errlist_t typedef struct _sgx_errlist_t sgx_errlist_t

10.175.2 Variable Documentation

10.175.2.1 global_eid sgx_enclave_id_t global_eid = 0

10.175.2.2 sgx_errlist sgx_errlist_t sgx_errlist[ ] [static]

10.176 types.h

Go to the documentation of this file.

```
1 #pragma once
2 #include <stddef.h>
3 #include "sgx_error.h"       /* sgx_status_t */
4 #include "sgx_eid.h"     /* sgx_enclave_id_t */
5
6 /* Global EID shared by multiple threads */
7 sgx_enclave_id_t global_eid = 0;
8
9 typedef struct _sgx_errlist_t {
10     sgx_status_t err;
11     const char *msg;
12     const char *sug; /* Suggestion */
13 } sgx_errlist_t;
14
15 /* Error code returned by sgx_create_enclave */
16 static sgx_errlist_t sgx_errlist[] = {
17     {
18         SGX_ERROR_UNEXPECTED,
19         "Unexpected error occurred.",
20         NULL
21     },
22     {
23         SGX_ERROR_INVALID_PARAMETER,
24         "Invalid parameter.",
25         NULL
26     },
27     {
28         SGX_ERROR_OUT_OF_MEMORY,
29         "Out of memory.",
30         NULL
```

```
31      },
32      {
33          SGX_ERROR_ENCLAVE_LOST,
34          "Power transition occurred.",
35          "Please refer to the sample \"PowerTransition\" for details."
36      },
37      {
38          SGX_ERROR_INVALID_ENCLAVE,
39          "Invalid enclave image.",
40          NULL
41      },
42      {
43          SGX_ERROR_INVALID_ENCLAVE_ID,
44          "Invalid enclave identification.",
45          NULL
46      },
47      {
48          SGX_ERROR_INVALID_SIGNATURE,
49          "Invalid enclave signature.",
50          NULL
51      },
52      {
53          SGX_ERROR_OUT_OF_EPC,
54          "Out of EPC memory.",
55          NULL
56      },
57      {
58          SGX_ERROR_NO_DEVICE,
59          "Invalid SGX device.",
60          "Please make sure SGX module is enabled in the BIOS, and install SGX driver afterwards."
61      },
62      {
63          SGX_ERROR_MEMORY_MAP_CONFLICT,
64          "Memory map conflicted.",
65          NULL
66      },
67      {
68          SGX_ERROR_INVALID_METADATA,
69          "Invalid enclave metadata.",
70          NULL
71      },
72      {
73          SGX_ERROR_DEVICE_BUSY,
74          "SGX device was busy.",
75          NULL
76      },
77      {
78          SGX_ERROR_INVALID_VERSION,
79          "Enclave version was invalid.",
80          NULL
81      },
82      {
83          SGX_ERROR_INVALID_ATTRIBUTE,
84          "Enclave was not authorized.",
85          NULL
86      },
87      {
88          SGX_ERROR_ENCLAVE_FILE_ACCESS,
89          "Can't open enclave file.",
90          NULL
91      },
92  };
```

Index

_C_FUNCTION
    user_ta_header.c, 402, 405
__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, 344
    tee_config.h, 317, 319, 320
__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, 161
__attr_const
    compiler.h, 72
__attribute__
    profiler_data.h, 336
    tee-internal-api-machine.c, 201
    tee-internal-api.c, 214
    tee-ta-internal.h, 86
__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, 330
__cyg_profile_func_enter
    profiler.c, 331
__cyg_profile_func_exit
    profiler.c, 331
__data
    compiler.h, 73
__deprecated
    compiler.h, 73
__early_ta
    compiler.h, 74
__init_array_start
    crt.c, 338
__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, 331
__profiler_head
profiler.c, 332
tee_profiler.c, 323, 324, 326
__profiler_header, 36
idx, 36
size, 36
start, 36
__profiler_map_info
profiler.c, 331
profiler.h, 333
__profiler_nsec_t
profiler_data.h, 336
__profiler_unmap_info
tee_profiler.c, 321, 323, 325
tee_profiler.h, 327–329
__pure
compiler.h, 77
__rodata
compiler.h, 77
__rodata_unpaged
compiler.h, 77
__section
compiler.h, 77
user_ta_header.c, 402, 405
__ta_entry
user_ta_header.c, 402, 406
__unused
compiler.h, 77
__used
compiler.h, 77
__utee_entry
user_ta_header.c, 403, 406
__weak
compiler.h, 77
__wrapper_ocall_close_file
Enclave_u.h, 275
_atoi
vsnprintf.c, 357
_ftoa
vsnprintf.c, 357
_is_digit
vsnprintf.c, 358
_ntoa_format
vsnprintf.c, 358
_ntoa_long
vsnprintf.c, 359
_ntoa_long_long
vsnprintf.c, 360
_out_buffer
vsnprintf.c, 360
_out_char
vsnprintf.c, 361
_out_fct
vsnprintf.c, 361
_out_null
vsnprintf.c, 362
_putchar
vsnprintf.c, 355
_sanctum_dev_public_key
test_dev_key.h, 189
_sanctum_dev_public_key_len
test_dev_key.h, 189
_sanctum_dev_secret_key
test_dev_key.h, 190
_sanctum_dev_secret_key_len
test_dev_key.h, 190
_sgx_errlist_t, 39
err, 39
msg, 39
sug, 39
_strlen
Enclave.c, 394
tools.c, 315
trace.c, 234, 237
vsnprintf.c, 362
_vsnprintf
vsnprintf.c, 362

a
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, 222, 226
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, 54
TEE_OperationInfoMultiple, 56
aligned
crt.c, 338
alloced_size
TEEC_SharedMemory, 67
analyzer.c
BUF_MAX, 301
COLS, 301
FORMAT, 301
main, 301
App.cpp
enc_path, 367, 370
initialize_enclave, 368, 371
main, 366, 368, 370, 371
MAX_PATH, 368, 371
print_error_message, 369, 372
runtime_path, 367, 370
App.h
ENCLAVE_FILENAME, 413, 416
FALSE, 413, 416
global_eid, 414, 416
TRUE, 414, 416
App_ocalls.cpp
load_invoke_command, 373
MAX_PATH, 378
NO_PERF, 373, 378
ocall_close_file, 373, 378, 381, 386
ocall_getrandom, 374, 382
ocall_invoke_command_callback_write, 374, 382
ocall_open_file, 375, 379, 382, 386
ocall_print_string, 375, 379, 384, 387
ocall_read_file, 376, 379, 384, 387
ocall_ree_time, 376, 380, 384, 388
ocall_write_file, 376, 380, 385, 388
store_invoke_callback_file, 377
App_ocalls.h
ocall_close_file, 418, 427
ocall_open_file, 419, 428
ocall_print_string, 421, 429
ocall_read_file, 422, 430
ocall_ree_time, 423, 431
ocall_write_file, 424, 432
ree_time_t, 418, 427
arg
out_fct_wrap_type, 46
array_t
user_types.h, 272
asymmetric_key.c
gp_asymmetric_key_sign_test, 287
SHA_LENGTH, 287
SIG_LENGTH, 287
ATTEST_DATA_MAXLEN
report.h, 81
attributeID
TEE_Attribute, 51

b
ob16_t, 45
ob196_t, 45
ob256_t, 46
TEE_Attribute, 51
TEE_Param, 58
TEEC_Value, 70
bench.c
benchmark, 252
init, 252
labels, 254
record, 252
tee_time_tests, 253
time_diff, 253
time_test, 254
time_to_millis, 254
bench.h
cpu_double_benchmark, 256
cpu_int_benchmark, 256
io_read_benchmark, 256
io_write_benchmark, 256
NO_PERF, 256
random_memory_benchmark, 257
ree_time_test, 257
sequential_memory_benchmark, 257
system_time_test, 258
BENCH_TYPE
config_bench.h, 261
benchmark
bench.c, 252
buf
tee_def.h, 265–267
buf_flag
tee_def.h, 265, 266, 268
BUF_MAX
analyzer.c, 301
BUF_SIZE
config_bench.h, 261
Enclave.c, 393

main.c, 398
nm_parse.c, 304
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, 272
bufferLen
TEE_SEAID, 59

CALL
profiler_data.h, 336
callee
__profiler_data, 35
result, 49
CIPHER_LENGTH
symmetric_key.c, 298
symmetric_key_gcm.c, 299
clockSeqAndNode
TEE_UUID, 60
TEEC_UUID, 69
COLS
analyzer.c, 301
COMMAND
tee_config.h, 319
commandID
invoke_command_t, 42
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, 261
BUF_SIZE, 261
CPU_DOUBLE_SENSITIVE, 261
CPU_INT_SENSITIVE, 261
DOUBLE_OFFSET, 261
IO_READ_SENSITIVE, 261
IO_WRITE_SENSITIVE, 261
MULT_SIZE, 261
OFFSET, 261
RANDOM_MEMORY_SENSITIVE, 261
record, 261
REE_TIME_TEST, 261
SEQUENTIAL_MEMORY_SENSITIVE, 261
SYSTEM_TIME_TEST, 261
config_ref_ta.h
GP_ASSERT, 289
tee_printf, 289
content
TEE_Attribute, 51
cpu_bench.c
cpu_double_benchmark, 259
cpu_int_benchmark, 259
cpu_double_benchmark
bench.h, 256
cpu_bench.c, 259
CPU_DOUBLE_SENSITIVE
config_bench.h, 261
cpu_int_benchmark
bench.h, 256
cpu_bench.c, 259
CPU_INT_SENSITIVE
config_bench.h, 261
create_htable
nm_parse.c, 304
crt.c

__ImageBase, 344
__init_array_start, 338
aligned, 338
crt_end, 338
fini_array, 338
init_array, 339
run_all_test, 340
TA_CloseSessionEntryPoint, 341
TA_CreateEntryPoint, 341
TA_DestroyEntryPoint, 341
TA_InvokeCommandEntryPoint, 341
TA_OpenSessionEntryPoint, 342
TEE_PARAM_TYPE0, 340
TEE_PARAM_TYPE1, 340
tee_printf, 342
crt.h
crt_begin, 345
crt_end, 345
main, 345
crt_begin
crt.h, 345
crt_end
crt.c, 338
crt.h, 345
ctx
__TEE_OperationHandle, 38
TEEC_Session, 66

data
enclave_report, 41
data_len
enclave_report, 41
DATA_LENGTH
secure_stoage.c, 296
dataPosition
TEE_ObjectInfo, 53
dataSize
TEE_ObjectInfo, 53
depth
result, 49
desc
__TEE_ObjectHandle, 36
dev_public_key
report, 48
dhex_dump
trace.h, 194
DHEXDUMP
trace.h, 192
digestLength
TEE_OperationInfo, 55
TEE_OperationInfoMultiple, 57
direction
__profiler_data, 35
direction_t
profiler_data.h, 336
DMREQ_FINISH
tee_api_types.h, 160
DMREQ_WRITE
tee_api_types.h, 160
DMSG
trace.h, 192
DMSG_RAW
trace.h, 192
DOUBLE_OFFSET
config_bench.h, 261
DPRINT_STACK
trace.h, 192

eapp_entry
Enclave.c, 392
startup.c, 352
ecall_ta_main
Enclave.c, 398
startup.c, 353
EDGE_EXTERNC_BEGIN
Enclave_u.h, 275
EDGE_EXTERNC_END
Enclave_u.h, 275
EDGE_OUT_WITH_STRUCTURE
ocalls.h, 277
EMSG
trace.h, 192
EMSG_RAW
trace.h, 192
enc_path
App.cpp, 367, 370
enclave
report, 48
Enclave.c
_strlen, 394
BUF_SIZE, 393
eapp_entry, 392
ecall_ta_main, 398
main, 389–391
MESSAGE, 392, 394, 397
print_buf, 397
print_pos, 397
run_all_test, 394
TA_CloseSessionEntryPoint, 395
TA_CreateEntryPoint, 395
TA_DestroyEntryPoint, 395
TA_InvokeCommandEntryPoint, 395
TA_OpenSessionEntryPoint, 396
TEE_PARAM_TYPE1, 394
tee_printf, 396
Enclave.h
gp_asymmetric_key_sign_test, 284
gp_message_digest_test, 284
gp_random_test, 284
gp_ree_time_test, 284
gp_secure_storage_test, 284
gp_symmetric_key_enc_verify_test, 285
gp_symmetric_key_gcm_verify_test, 285
gp_trusted_time_test, 285
TA_REF_RUN_ALL, 282
TA_REF_UUID, 282
ENCLAVE_FILENAME
App.h, 413, 416

enclave_report, 41
    data, 41
    data_len, 41
    hash, 41
    signature, 41
Enclave_u.h
    __wrapper_ocall_close_file, 275
    EDGE_EXTERNC_BEGIN, 275
    EDGE_EXTERNC_END, 275
    register_functions, 276
end
    result, 49
end_hartid
    result, 49
EPRINT_STACK
    trace.h, 192
err
    _sgx_errlist_t, 39
events
    pollfd, 47

FALSE
    App.h, 413, 416
fct
    out_fct_wrap_type, 46
fctprintf
    vsnprintf.c, 363
fd
    pollfd, 47
    TEEC_Context, 61
fini_array
    crt.c, 338
flags
    __TEE_ObjectHandle, 37
    __TEE_OperationHandle, 38
    TEEC_SharedMemory, 67
flags2flags
    tee-internal-api.c, 203, 214
FLAGS_CHAR
    vsnprintf.c, 355
FLAGS_HASH
    vsnprintf.c, 355
FLAGS_LEFT
    vsnprintf.c, 355
FLAGS_LONG
    vsnprintf.c, 355
FLAGS_LONG_LONG
    vsnprintf.c, 355
FLAGS_PLUS
    vsnprintf.c, 356
FLAGS_PRECISION
    vsnprintf.c, 356
FLAGS_SHORT
    vsnprintf.c, 356
FLAGS_SPACE
    vsnprintf.c, 356
FLAGS_UPPERCASE
    vsnprintf.c, 356
FLAGS_ZEROPAD
    vsnprintf.c, 356
FMSG
    trace.h, 192
FMSG_RAW
    trace.h, 193
FORMAT
    analyzer.c, 301
FPERMS
    tee-internal-api.c, 202, 213
FPRINT_STACK
    trace.h, 193
func_name
    nm_info, 44

GCM_ST_AAD
    tee-internal-api-cryptlib.c, 239
GCM_ST_ACTIVE
    tee-internal-api-cryptlib.c, 239
GCM_ST_FINAL
    tee-internal-api-cryptlib.c, 239
GCM_ST_INIT
    tee-internal-api-cryptlib.c, 239
get_func_name
    nm_parse.c, 304
    nm_parse.h, 307
get_key
    nm_parse.c, 305
GetRelTimeEnd
    tee-internal-api.c, 204, 214
    tee-ta-internal.h, 86
GetRelTimeStart
    tee-internal-api.c, 204, 215
    tee-ta-internal.h, 87
global_eid
    App.h, 414, 416
    types.h, 434, 437
GP_ASSERT
    config_ref_ta.h, 289
gp_asymmetric_key_sign_test
    asymmetric_key.c, 287
    Enclave.h, 284
    gp_test.h, 291
gp_invokecommand_test
    gp_test.h, 291
gp_message_digest_test
    Enclave.h, 284
    gp_test.h, 291
    message_digest.c, 295
gp_random_test
    Enclave.h, 284
    gp_test.h, 291
    random.c, 296
gp_ree_time_test
    Enclave.h, 284
    gp_test.h, 292
    time.c, 300
gp_secure_storage_test
    Enclave.h, 284
    gp_test.h, 292

secure_stoage.c, 297
gp_symmetric_key_enc_verify_test
Enclave.h, 285
gp_test.h, 292
symmetric_key.c, 298
gp_symmetric_key_gcm_verify_test
Enclave.h, 285
gp_test.h, 292
symmetric_key_gcm.c, 299
gp_test.h
gp_asymmetric_key_sign_test, 291
gp_invokecommand_test, 291
gp_message_digest_test, 291
gp_random_test, 291
gp_ree_time_test, 292
gp_secure_storage_test, 292
gp_symmetric_key_enc_verify_test, 292
gp_symmetric_key_gcm_verify_test, 292
gp_trusted_time_test, 292
gp_trusted_time_test
Enclave.h, 285
gp_test.h, 292
time.c, 300

handleFlags
TEE_ObjectInfo, 53
handleState
TEE_OperationInfo, 55
TEE_OperationInfoMultiple, 57
hartid
__profiler_data, 35
hash
enclave_report, 41
sm_report, 50
HASH_SIZE
nm_parse.h, 307

id
TEEC_SharedMemory, 67
idx
__profiler_header, 36
nm_parse.c, 306
profiler_data.h, 336
result, 49
IMSG
trace.h, 193
IMSG_RAW
trace.h, 193
INC
memory_bench.c, 269
init
bench.c, 252
init_array
crt.c, 339
initialize_enclave
App.cpp, 368, 371
INMSG
trace.h, 193
insert_nm
nm_parse.c, 305
invoke_command.c
TA_MAX_SIZE, 294
TEEP_AGENT_TA_DELETE, 294
TEEP_AGENT_TA_EXIT, 294
TEEP_AGENT_TA_INSTALL, 294
TEEP_AGENT_TA_LOAD, 294
TEEP_AGENT_TA_NONE, 294
invoke_command_t, 41
commandID, 42
ocalls.h, 278
param1_fd, 42
params0_buffer, 42
params0_size, 42
params1_buffer, 42
params1_size, 42
io_bench.c
io_read_benchmark, 263
io_write_benchmark, 264
SPLITS, 263
io_read_benchmark
bench.h, 256
io_bench.c, 263
IO_READ_SENSITIVE
config_bench.h, 261
io_write_benchmark
bench.h, 256
io_bench.c, 264
IO_WRITE_SENSITIVE
config_bench.h, 261
IPRINT_STACK
trace.h, 193
is_empty
stack.h, 310

keyInformation
TEE_OperationInfoMultiple, 57
keySize
TEE_ObjectInfo, 53
TEE_OperationInfo, 55
TEE_OperationInfoKey, 56

labels
bench.c, 254
length
TEE_Attribute, 51
list, 43
addr, 43
next, 43
nm, 43
load_invoke_command
App_ocalls.cpp, 373
LOG_FILE
profiler_data.h, 335
login
TEE_Identity, 52
LOOPS_PER_THREAD
user_types.h, 271

main
analyzer.c, 301
App.cpp, 366, 368, 370, 371
crt.h, 345
Enclave.c, 389–391
main.c, 399, 401
main.c
BUF_SIZE, 398
main, 399, 401
print_buf, 401
PRINT_BUF_SIZE, 399, 400
TEEC_PARAM_TYPE0, 399
TEEC_PARAM_TYPE1, 399, 400
MAX_ADDR
nm_parse.c, 304
MAX_FUNC_PRINT_SIZE
trace.h, 193
MAX_PATH
App.cpp, 368, 371
App_ocalls.cpp, 378
MAX_PRINT_SIZE
trace.h, 193
maxKeySize
TEE_ObjectInfo, 54
TEE_OperationInfo, 55
TEE_OperationInfoMultiple, 57
maxObjectSize
TEE_ObjectInfo, 54
MBEDCRYPT
tee_api_tee_types.h, 222, 226
MDSIZE
report.h, 81
memory_bench.c
INC, 269
random_memory_benchmark, 269
sequential_memory_benchmark, 269
memref
TEE_Param, 58
TEEC_Parameter, 64
MESSAGE
Enclave.c, 392, 394, 397
message_digest.c
gp_message_digest_test, 295
SHA_LENGTH, 295
SIG_LENGTH, 295
millis
ree_time_t, 47
TEE_Time, 60
mode
__TEE_OperationHandle, 38
TEE_OperationInfo, 55
TEE_OperationInfoMultiple, 57
MSG
trace.h, 193
msg
_sgx_errlist_t, 39
MSG_RAW
trace.h, 193
MULT_SIZE
config_bench.h, 261

next
list, 43
nfds_t
tee_api_types.h, 161
nm
list, 43
nm_info, 43
func_name, 44
type, 44
nm_parse.c
BUF_SIZE, 304
create_htable, 304
get_func_name, 304
get_key, 305
idx, 306
insert_nm, 305
MAX_ADDR, 304
nm_pool, 306
parse_nm, 305
POOL_SIZE, 304
nm_parse.h
get_func_name, 307
HASH_SIZE, 307
parse_nm, 307
nm_pool
nm_parse.c, 306
NO_PERF
App_ocalls.cpp, 373, 378
bench.h, 256
profiler_attrs.h, 334
nonce
nonce_t, 44
NONCE_SIZE
ocalls.h, 277
nonce_t, 44
nonce, 44
ocalls.h, 278
nsec
__profiler_data, 35
numberOfKeys
TEE_OperationInfoMultiple, 57

O_CREAT
tee-internal-api.c, 203, 213
O_EXCL
tee-internal-api.c, 203, 213
O_RDONLY
tee-internal-api.c, 203, 213
O_RDWR
tee-internal-api.c, 203, 213
O_TRUNC
tee-internal-api.c, 203, 213
O_WRONLY
tee-internal-api.c, 203, 213
ob16_t, 44
b, 45

ocalls.h, 278
ret, 45
ob196_t, 45
b, 45
ocalls.h, 278
ret, 45
ob256_t, 46
b, 46
ocalls.h, 278
ret, 46
objectSize
TEE_ObjectInfo, 54
objectType
TEE_ObjectInfo, 54
objectUsage
TEE_ObjectInfo, 54
ocall_close_file
App_ocalls.cpp, 373, 378, 381, 386
App_ocalls.h, 418, 427
ocalls.h, 278
ocall_getrandom
App_ocalls.cpp, 374, 382
ocall_getrandom16
ocalls.h, 279
ocall_getrandom196
ocalls.h, 279
ocall_import_nonce
ocalls.h, 279
ocall_invoke_command_callback
ocalls.h, 279
ocall_invoke_command_callback_write
App_ocalls.cpp, 374, 382
ocalls.h, 280
ocall_invoke_command_polling
ocalls.h, 280
ocall_open_file
App_ocalls.cpp, 375, 379, 382, 386
App_ocalls.h, 419, 428
ocalls.h, 280
ocall_print_string
App_ocalls.cpp, 375, 379, 384, 387
App_ocalls.h, 421, 429
ocalls.h, 280
ocall_print_string_wrapper
ocall_wrapper.c, 350, 351
ocall_wrapper.h, 346
ocall_read_file
App_ocalls.cpp, 376, 379, 384, 387
App_ocalls.h, 422, 430
ocall_read_file256
ocalls.h, 280
ocall_ree_time
App_ocalls.cpp, 376, 380, 384, 388
App_ocalls.h, 423, 431
ocalls.h, 280
ocall_wrapper.c
ocall_print_string_wrapper, 350, 351
ocall_wrapper.h
ocall_print_string_wrapper, 346
ocall_write_file
App_ocalls.cpp, 376, 380, 385, 388
App_ocalls.h, 424, 432
ocalls.h, 280
ocalls.h
EDGE_OUT_WITH_STRUCTURE, 277
invoke_command_t, 278
NONCE_SIZE, 277
nonce_t, 278
ob16_t, 278
ob196_t, 278
ob256_t, 278
ocall_close_file, 278
ocall_getrandom16, 279
ocall_getrandom196, 279
ocall_import_nonce, 279
ocall_invoke_command_callback, 279
ocall_invoke_command_callback_write, 280
ocall_invoke_command_polling, 280
ocall_open_file, 280
ocall_print_string, 280
ocall_read_file256, 280
ocall_ree_time, 280
ocall_write_file, 280
ree_time_t, 278
OFFSET
config_bench.h, 261
offset
TEEC_RegisteredMemoryReference, 65
OpenPersistentObject
tee-internal-api.c, 204, 215
operationClass
TEE_OperationInfo, 55
TEE_OperationInfoMultiple, 57
operationState
TEE_OperationInfoMultiple, 57
out_fct_type
vsnprintf.c, 357
out_fct_wrap_type, 46
arg, 46
fct, 46
OUTMSG
trace.h, 194
OUTRMSG
trace.h, 194

param1_fd
invoke_command_t, 42
params
TEEC_Operation, 62
params0_buffer
invoke_command_t, 42
params0_size
invoke_command_t, 42
params1_buffer
invoke_command_t, 42
params1_size
invoke_command_t, 42

paramTypes
TEEC_Operation, 62
parent
TEEC_RegisteredMemoryReference, 65
parse_nm
nm_parse.c, 305
nm_parse.h, 307
perf_buffer
tee_config.h, 317, 319, 320
PERF_SECTION
profiler_attrs.h, 334
PERF_SIZE
profiler_data.h, 336
persist_ctx
__TEE_ObjectHandle, 37
pollfd, 47
events, 47
fd, 47
revents, 47
POOL_SIZE
nm_parse.c, 304
pop
stack.h, 310
pos
stack.h, 310
pr_deb
tee-common.h, 83
prikey
__TEE_OperationHandle, 38
print_buf
Enclave.c, 397
main.c, 401
PRINT_BUF_SIZE
main.c, 399, 400
print_error_message
App.cpp, 369, 372
print_pos
Enclave.c, 397
printf
tools.c, 315
tools.h, 348
PRINTF_FTOA_BUFFER_SIZE
vsnprintf.c, 356
PRINTF_NTOA_BUFFER_SIZE
vsnprintf.c, 356
PRINTF_SUPPORT_FLOAT
vsnprintf.c, 356
PRINTF_SUPPORT_LONG_LONG
vsnprintf.c, 356
PRINTF_SUPPORT_PTRDIFF_T
vsnprintf.c, 356
private_key
__TEE_ObjectHandle, 37
profiler.c
__cyg_profile_func, 330
__cyg_profile_func_enter, 331
__cyg_profile_func_exit, 331
__profiler_get_data_ptr, 331
__profiler_head, 332
__profiler_map_info, 331
profiler.h
__profiler_map_info, 333
profiler_attrs.h
NO_PERF, 334
PERF_SECTION, 334
USED, 334
profiler_data.h
__attribute__, 336
__profiler_nsec_t, 336
CALL, 336
direction_t, 336
idx, 336
LOG_FILE, 335
PERF_SIZE, 336
RET, 336
size, 336
START, 336
start, 336
profiler_write
tee_profiler.c, 322, 324, 325
tools.c, 311–314
pubkey
__TEE_OperationHandle, 38
public_key
__TEE_ObjectHandle, 37
sm_report, 50
PUBLIC_KEY_SIZE
report.h, 81
push
stack.h, 310
putchar
tools.c, 316
tools.h, 349
vsnprintf.c, 363
puts
tools.c, 316
tools.h, 349

random.c
gp_random_test, 296
random_memory_benchmark
bench.h, 257
memory_bench.c, 269
RANDOM_MEMORY_SENSITIVE
config_bench.h, 261
record
bench.c, 252
config_bench.h, 261
ree_time_t, 47
App_ocalls.h, 418, 427
millis, 47
ocalls.h, 278
seconds, 47
REE_TIME_TEST
config_bench.h, 261
ree_time_test
bench.h, 257

time_test.c, 270
ref
TEE_Attribute, 51
reg_mem
TEEC_Context, 61
register_functions
Enclave_u.h, 276
registered_fd
TEEC_SharedMemory, 68
report, 48
dev_public_key, 48
enclave, 48
sm, 48
report.h
ATTEST_DATA_MAXLEN, 81
MDSIZE, 81
PUBLIC_KEY_SIZE, 81
SIGNATURE_SIZE, 81
requiredKeyUsage
TEE_OperationInfo, 55
TEE_OperationInfoKey, 56
result, 49
callee, 49
depth, 49
end, 49
end_hartid, 49
idx, 49
start, 49
start_hartid, 50
RET
profiler_data.h, 336
ret
ob16_t, 45
ob196_t, 45
ob256_t, 46
revents
pollfd, 47
run_all_test
crt.c, 340
Enclave.c, 394
runtime_path
App.cpp, 367, 370

seconds
ree_time_t, 47
TEE_Time, 60
secure_stoage.c
DATA_LENGTH, 296
gp_secure_storage_test, 297
selectResponseEnable
TEE_SEReaderProperties, 59
sePresent
TEE_SEReaderProperties, 59
sequential_memory_benchmark
bench.h, 257
memory_bench.c, 269
SEQUENTIAL_MEMORY_SENSITIVE
config_bench.h, 261
session
TEEC_Operation, 63
session_id
TEEC_Session, 66
set_object_key
tee-internal-api.c, 205, 216
sgx_errlist
types.h, 434, 437
sgx_errlist_t
types.h, 434, 437
SHA_LENGTH
asymmetric_key.c, 287
message_digest.c, 295
tee_api_tee_types.h, 222, 226
shadow_buffer
TEEC_SharedMemory, 68
SIG_LENGTH
asymmetric_key.c, 287
message_digest.c, 295
tee-internal-api-cryptlib.c, 239
signature
enclave_report, 41
sm_report, 50
SIGNATURE_SIZE
report.h, 81
size
__profiler_header, 36
profiler_data.h, 336
TEE_Param, 58
TEEC_RegisteredMemoryReference, 65
TEEC_SharedMemory, 68
TEEC_TempMemoryReference, 69
sm
report, 48
sm_report, 50
hash, 50
public_key, 50
signature, 50
SMSG
trace.h, 194
snprintf
vsnprintf.c, 363
socklen_t
tee_api_types.h, 161
SPLITS
io_bench.c, 263
sprintf
vsnprintf.c, 365
stack
stack.h, 310
stack.h
is_empty, 310
pop, 310
pos, 310
push, 310
stack, 310
STACK_SIZE, 310
STACK_SIZE
stack.h, 310

START
profiler_data.h, 336
start
__profiler_header, 36
profiler_data.h, 336
result, 49
start_hartid
result, 50
started
TEEC_Operation, 63
startup.c
eapp_entry, 352
ecall_ta_main, 353
store_invoke_callback_file
App_ocalls.cpp, 377
sug
_sgx_errlist_t, 39
symmetric_key.c
CIPHER_LENGTH, 298
gp_symmetric_key_enc_verify_test, 298
symmetric_key_gcm.c
CIPHER_LENGTH, 299
gp_symmetric_key_gcm_verify_test, 299
SYSTEM_TIME_TEST
config_bench.h, 261
system_time_test
bench.h, 258
time_test.c, 270

ta-ref/api/include/compiler.h, 71, 78
ta-ref/api/include/report.h, 80, 81
ta-ref/api/include/tee-common.h, 82, 83
ta-ref/api/include/tee-ta-internal.h, 83, 108
ta-ref/api/include/tee_api_defines.h, 110, 148
ta-ref/api/include/tee_api_defines_extensions.h, 153, 157
ta-ref/api/include/tee_api_types.h, 158, 163
ta-ref/api/include/tee_client_api.h, 166, 178
ta-ref/api/include/tee_internal_api.h, 181
ta-ref/api/include/tee_internal_api_extensions.h, 181, 184
ta-ref/api/include/tee_ta_api.h, 184, 187
ta-ref/api/include/test_dev_key.h, 189, 190
ta-ref/api/include/trace.h, 190, 196
ta-ref/api/include/trace_levels.h, 198, 199
ta-ref/api/keystone/tee-internal-api-machine.c, 200
ta-ref/api/keystone/tee-internal-api.c, 201
ta-ref/api/keystone/tee_api_tee_types.h, 221, 223
ta-ref/api/keystone/teec_stub.c, 228
ta-ref/api/keystone/trace.c, 232
ta-ref/api/optee/tee_api_tee_types.h, 224
ta-ref/api/sgx/tee-internal-api.c, 212
ta-ref/api/sgx/tee_api_tee_types.h, 224, 227
ta-ref/api/tee-internal-api-cryptlib.c, 237
ta-ref/benchmark/bench.c, 251
ta-ref/benchmark/bench.h, 255, 258
ta-ref/benchmark/cpu_bench.c, 258
ta-ref/benchmark/include/config_bench.h, 260, 262
ta-ref/benchmark/io_bench.c, 262
ta-ref/benchmark/keystone/tee_def.h, 264, 265
ta-ref/benchmark/memory_bench.c, 268
ta-ref/benchmark/optee/tee_def.h, 266
ta-ref/benchmark/sgx/tee_def.h, 267, 268
ta-ref/benchmark/time_test.c, 270
ta-ref/docs/building.md, 271
ta-ref/docs/gp_api.md, 271
ta-ref/docs/how_to_program_on_ta-ref.md, 271
ta-ref/docs/overview_of_ta-ref.md, 271
ta-ref/docs/preparation.md, 271
ta-ref/docs/running_on_dev_boards.md, 271
ta-ref/edger/edger8r/user_types.h, 271, 272
ta-ref/edger/keyedge/Enclave_t.c, 272
ta-ref/edger/keyedge/Enclave_t.h, 273
ta-ref/edger/keyedge/Enclave_u.c, 274
ta-ref/edger/keyedge/Enclave_u.h, 274, 276
ta-ref/edger/keyedge/ocalls.h, 276, 281
ta-ref/edger/optee/Enclave.h, 282
ta-ref/edger/optee/Enclave_t.h, 274
ta-ref/gp/asymmetric_key.c, 286
ta-ref/gp/include/config_ref_ta.h, 288, 290
ta-ref/gp/include/gp_test.h, 291, 293
ta-ref/gp/invoke_command.c, 293
ta-ref/gp/message_digest.c, 294
ta-ref/gp/random.c, 295
ta-ref/gp/secure_stoage.c, 296
ta-ref/gp/symmetric_key.c, 297
ta-ref/gp/symmetric_key_gcm.c, 298
ta-ref/gp/time.c, 299
ta-ref/profiler/analyzer/analyzer.c, 300
ta-ref/profiler/analyzer/analyzer.h, 302, 303
ta-ref/profiler/analyzer/nm_parse.c, 303
ta-ref/profiler/analyzer/nm_parse.h, 306, 308
ta-ref/profiler/analyzer/stack.h, 308, 311
ta-ref/profiler/app/tools.c, 311
ta-ref/profiler/keystone/Enclave/tools.c, 312
ta-ref/profiler/keystone/tee_config.h, 317, 318
ta-ref/profiler/keystone/tee_profiler.c, 321
ta-ref/profiler/keystone/tee_profiler.h, 327
ta-ref/profiler/optee/Enclave/tools.c, 313
ta-ref/profiler/optee/tee_config.h, 318, 319
ta-ref/profiler/optee/tee_profiler.c, 323
ta-ref/profiler/optee/tee_profiler.h, 328
ta-ref/profiler/profiler.c, 330
ta-ref/profiler/profiler.h, 332, 333
ta-ref/profiler/profiler_attrs.h, 333, 334
ta-ref/profiler/profiler_data.h, 334, 337
ta-ref/profiler/sgx/Enclave/tools.c, 314
ta-ref/profiler/sgx/tee_config.h, 320, 321
ta-ref/profiler/sgx/tee_profiler.c, 324
ta-ref/profiler/sgx/tee_profiler.h, 329
ta-ref/test_gp/crt.c, 337
ta-ref/test_gp/include/crt.h, 344, 346
ta-ref/test_gp/include/ocall_wrapper.h, 346, 347
ta-ref/test_gp/include/random.h, 347, 348
ta-ref/test_gp/include/tools.h, 348, 350
ta-ref/test_gp/keystone/App/App.cpp, 366
ta-ref/test_gp/keystone/App/App_ocalls.cpp, 372
ta-ref/test_gp/keystone/Enclave/Enclave.c, 388
ta-ref/test_gp/keystone/Enclave/ocall_wrapper.c, 350

ta-ref/test_gp/keystone/Enclave/startup.c, 352
ta-ref/test_gp/keystone/Enclave/trace.c, 234
ta-ref/test_gp/optee/App/main.c, 398
ta-ref/test_gp/optee/Enclave/crt.c, 339
ta-ref/test_gp/optee/Enclave/Enclave.c, 389
ta-ref/test_gp/optee/Enclave/trace.c, 235
ta-ref/test_gp/optee/Enclave/user_ta_header.c, 401
ta-ref/test_gp/optee/Enclave/user_ta_header_defines.h, 407, 409
ta-ref/test_gp/sgx/App/App.cpp, 367
ta-ref/test_gp/sgx/App/App.h, 412, 414
ta-ref/test_gp/sgx/App/App_ocalls.cpp, 377
ta-ref/test_gp/sgx/App/App_ocalls.h, 417, 425
ta-ref/test_gp/sgx/App/types.h, 433, 435
ta-ref/test_gp/sgx/Enclave/Enclave.c, 390
ta-ref/test_gp/sgx/Enclave/Enclave.h, 283, 286
ta-ref/test_gp/sgx/Enclave/ocall_wrapper.c, 351
ta-ref/test_gp/sgx/Enclave/startup.c, 353
ta-ref/test_gp/sgx/Enclave/trace.c, 236
ta-ref/test_gp/tools.c, 315
ta-ref/test_gp/vsnprintf.c, 354
ta-ref/test_hello/keystone/App/App.cpp, 369
ta-ref/test_hello/keystone/App/App_ocalls.cpp, 381
ta-ref/test_hello/keystone/Enclave/Enclave.c, 391
ta-ref/test_hello/optee/App/main.c, 400
ta-ref/test_hello/optee/Enclave/Enclave.c, 393
ta-ref/test_hello/optee/Enclave/user_ta_header.c, 404
ta-ref/test_hello/optee/Enclave/user_ta_header_defines.h, 410, 412
ta-ref/test_hello/sgx/App/App.cpp, 370
ta-ref/test_hello/sgx/App/App.h, 415, 416
ta-ref/test_hello/sgx/App/App_ocalls.cpp, 385
ta-ref/test_hello/sgx/App/App_ocalls.h, 426, 433
ta-ref/test_hello/sgx/App/types.h, 436, 437
ta-ref/test_hello/sgx/Enclave/Enclave.c, 397
TA_CloseSessionEntryPoint
    crt.c, 341
    Enclave.c, 395
    tee_ta_api.h, 185
TA_CreateEntryPoint
    crt.c, 341
    Enclave.c, 395
    tee_ta_api.h, 186
TA_CURRENT_TA_EXT_PROPERTIES
    user_ta_header_defines.h, 408, 411
TA_DATA_SIZE
    user_ta_header_defines.h, 408, 411
TA_DESCRIPTION
    user_ta_header.c, 402, 405
    user_ta_header_defines.h, 408, 411
TA_DestroyEntryPoint
    crt.c, 341
    Enclave.c, 395
    tee_ta_api.h, 186
TA_EXPORT
    tee_ta_api.h, 185
TA_FLAGS
    user_ta_header_defines.h, 408, 411
TA_FRAMEWORK_STACK_SIZE
    user_ta_header.c, 402, 405
ta_heap
    user_ta_header.c, 403, 406
ta_heap_size
    user_ta_header.c, 404, 406
TA_InvokeCommandEntryPoint
    crt.c, 341
    Enclave.c, 395
    tee_ta_api.h, 186
TA_MAX_SIZE
    invoke_command.c, 294
ta_num_props
    user_ta_header.c, 404, 407
TA_OpenSessionEntryPoint
    crt.c, 342
    Enclave.c, 396
    tee_ta_api.h, 186
ta_props
    user_ta_header.c, 404, 407
TA_REF_RUN_ALL
    Enclave.h, 282
TA_REF_UUID
    Enclave.h, 282
TA_STACK_SIZE
    user_ta_header_defines.h, 409, 411
TA_UUID
    user_ta_header_defines.h, 409, 411
TA_VERSION
    user_ta_header.c, 402, 405
    user_ta_header_defines.h, 409, 411
tahead_get_trace_level
    user_ta_header.c, 403, 406
tee-common.h
    pr_deb, 83
tee-internal-api-cryptlib.c
    GCM_ST_AAD, 239
    GCM_ST_ACTIVE, 239
    GCM_ST_FINAL, 239
    GCM_ST_INIT, 239
    SIG_LENGTH, 239
    TEE_AEDecryptFinal, 240
    TEE_AEEncryptFinal, 240
    TEE_AEInit, 241
    TEE_AEUpdate, 242
    TEE_AEUpdateAAD, 242
    TEE_AllocateOperation, 243
    TEE_AllocateTransientObject, 243
    TEE_AsymmetricSignDigest, 244
    TEE_AsymmetricVerifyDigest, 245
    TEE_CipherDoFinal, 245
    TEE_CipherInit, 246
    TEE_CipherUpdate, 246
    TEE_DigestDoFinal, 247
    TEE_DigestUpdate, 247
    TEE_FreeOperation, 248
    TEE_FreeTransientObject, 248
    TEE_GenerateKey, 249

TEE_InitRefAttribute, 249
TEE_InitValueAttribute, 250
TEE_SetOperationKey, 250
wolfSSL_Free, 251
wolfSSL_Malloc, 251
tee-internal-api-machine.c
__attribute__, 201
tee-internal-api.c
__attribute__, 214
flags2flags, 203, 214
FPERMS, 202, 213
GetRelTimeEnd, 204, 214
GetRelTimeStart, 204, 215
O_CREAT, 203, 213
O_EXCL, 203, 213
O_RDONLY, 203, 213
O_RDWR, 203, 213
O_TRUNC, 203, 213
O_WRONLY, 203, 213
OpenPersistentObject, 204, 215
set_object_key, 205, 216
TEE_CloseObject, 205, 216
TEE_CreatePersistentObject, 206, 217
TEE_Free, 207
TEE_GenerateRandom, 207, 217
TEE_GetObjectInfo1, 208, 218
TEE_GetREETime, 208, 218
TEE_GetSystemTime, 209, 219
TEE_Malloc, 209
TEE_OpenPersistentObject, 209, 219
TEE_ReadObjectData, 210, 219
TEE_Realloc, 210
TEE_WriteObjectData, 211, 220
tee-ta-internal.h
__attribute__, 86
GetRelTimeEnd, 86
GetRelTimeStart, 87
TEE_AEDecryptFinal, 88
TEE_AEEncryptFinal, 89
TEE_AEInit, 89
TEE_AEUpdate, 90
TEE_AEUpdateAAD, 91
TEE_AllocateOperation, 91
TEE_AllocateTransientObject, 92
TEE_AsymmetricSignDigest, 92
TEE_AsymmetricVerifyDigest, 93
TEE_CipherInit, 94
TEE_CipherUpdate, 94
TEE_CloseObject, 95
TEE_CreatePersistentObject, 96
TEE_DigestDoFinal, 98
TEE_DigestUpdate, 98
TEE_FreeOperation, 99
TEE_FreeTransientObject, 99
TEE_GenerateKey, 100
TEE_GenerateRandom, 100
TEE_GetObjectInfo1, 102
TEE_GetREETime, 102
TEE_GetSystemTime, 103
TEE_InitRefAttribute, 103
TEE_InitValueAttribute, 104
TEE_OpenPersistentObject, 104
TEE_ReadObjectData, 105
TEE_SetOperationKey, 106
TEE_WriteObjectData, 107
TEE_AEDecryptFinal
tee-internal-api-cryptlib.c, 240
tee-ta-internal.h, 88
TEE_AEEncryptFinal
tee-internal-api-cryptlib.c, 240
tee-ta-internal.h, 89
TEE_AEInit
tee-internal-api-cryptlib.c, 241
tee-ta-internal.h, 89
TEE_AEUpdate
tee-internal-api-cryptlib.c, 242
tee-ta-internal.h, 90
TEE_AEUpdateAAD
tee-internal-api-cryptlib.c, 242
tee-ta-internal.h, 91
TEE_ALG_AES_CBC_MAC_NOPAD
tee_api_defines.h, 116
TEE_ALG_AES_CBC_MAC_PKCS5
tee_api_defines.h, 116
TEE_ALG_AES_CBC_NOPAD
tee_api_defines.h, 117
TEE_ALG_AES_CCM
tee_api_defines.h, 117
TEE_ALG_AES_CMAC
tee_api_defines.h, 117
TEE_ALG_AES_CTR
tee_api_defines.h, 117
TEE_ALG_AES_CTS
tee_api_defines.h, 117
TEE_ALG_AES_ECB_NOPAD
tee_api_defines.h, 117
TEE_ALG_AES_GCM
tee_api_defines.h, 117
TEE_ALG_AES_XTS
tee_api_defines.h, 117
TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY
tee_api_defines_extensions.h, 154
TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY
tee_api_defines_extensions.h, 154
TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY
tee_api_defines_extensions.h, 154
TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY
tee_api_defines_extensions.h, 154
TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY
tee_api_defines_extensions.h, 155
TEE_ALG_DES3_CBC_MAC_NOPAD
tee_api_defines.h, 117
TEE_ALG_DES3_CBC_MAC_PKCS5
tee_api_defines.h, 117
TEE_ALG_DES3_CBC_NOPAD
tee_api_defines.h, 117

TEE_ALG_DES3_ECB_NOPAD
tee_api_defines.h, 118
TEE_ALG_DES_CBC_MAC_NOPAD
tee_api_defines.h, 118
TEE_ALG_DES_CBC_MAC_PKCS5
tee_api_defines.h, 118
TEE_ALG_DES_CBC_NOPAD
tee_api_defines.h, 118
TEE_ALG_DES_ECB_NOPAD
tee_api_defines.h, 118
TEE_ALG_DH_DERIVE_SHARED_SECRET
tee_api_defines.h, 118
TEE_ALG_DSA_SHA1
tee_api_defines.h, 118
TEE_ALG_DSA_SHA224
tee_api_defines.h, 118
TEE_ALG_DSA_SHA256
tee_api_defines.h, 118
TEE_ALG_ECDH_P192
tee_api_defines.h, 118
TEE_ALG_ECDH_P224
tee_api_defines.h, 118
TEE_ALG_ECDH_P256
tee_api_defines.h, 119
TEE_ALG_ECDH_P384
tee_api_defines.h, 119
TEE_ALG_ECDH_P521
tee_api_defines.h, 119
TEE_ALG_ECDSA_P192
tee_api_defines.h, 119
TEE_ALG_ECDSA_P224
tee_api_defines.h, 119
TEE_ALG_ECDSA_P256
tee_api_defines.h, 119
TEE_ALG_ECDSA_P384
tee_api_defines.h, 119
TEE_ALG_ECDSA_P521
tee_api_defines.h, 119
TEE_ALG_HKDF_MD5_DERIVE_KEY
tee_api_defines_extensions.h, 155
TEE_ALG_HKDF_SHA1_DERIVE_KEY
tee_api_defines_extensions.h, 155
TEE_ALG_HKDF_SHA224_DERIVE_KEY
tee_api_defines_extensions.h, 155
TEE_ALG_HKDF_SHA256_DERIVE_KEY
tee_api_defines_extensions.h, 155
TEE_ALG_HKDF_SHA384_DERIVE_KEY
tee_api_defines_extensions.h, 155
TEE_ALG_HKDF_SHA512_DERIVE_KEY
tee_api_defines_extensions.h, 155
TEE_ALG_HMAC_MD5
tee_api_defines.h, 119
TEE_ALG_HMAC_SHA1
tee_api_defines.h, 119
TEE_ALG_HMAC_SHA224
tee_api_defines.h, 119
TEE_ALG_HMAC_SHA256
tee_api_defines.h, 120
TEE_ALG_HMAC_SHA384
tee_api_defines.h, 120
TEE_ALG_HMAC_SHA512
tee_api_defines.h, 120
TEE_ALG_MD5
tee_api_defines.h, 120
TEE_ALG_MD5SHA1
tee_api_defines.h, 120
TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY
tee_api_defines_extensions.h, 155
TEE_ALG_RSA_NOPAD
tee_api_defines.h, 120
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1
tee_api_defines.h, 120
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224
tee_api_defines.h, 120
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256
tee_api_defines.h, 120
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384
tee_api_defines.h, 120
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512
tee_api_defines.h, 120
TEE_ALG_RSAES_PKCS1_V1_5
tee_api_defines.h, 121
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1
tee_api_defines.h, 121
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224
tee_api_defines.h, 121
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256
tee_api_defines.h, 121
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384
tee_api_defines.h, 121
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512
tee_api_defines.h, 121
TEE_ALG_RSASSA_PKCS1_V1_5_MD5
tee_api_defines.h, 121
TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1
tee_api_defines.h, 121
TEE_ALG_RSASSA_PKCS1_V1_5_SHA1
tee_api_defines.h, 121
TEE_ALG_RSASSA_PKCS1_V1_5_SHA224
tee_api_defines.h, 121
TEE_ALG_RSASSA_PKCS1_V1_5_SHA256
tee_api_defines.h, 122
TEE_ALG_RSASSA_PKCS1_V1_5_SHA384
tee_api_defines.h, 122
TEE_ALG_RSASSA_PKCS1_V1_5_SHA512
tee_api_defines.h, 122
TEE_ALG_SHA1
tee_api_defines.h, 122
TEE_ALG_SHA224
tee_api_defines.h, 122
TEE_ALG_SHA256
tee_api_defines.h, 122
TEE_ALG_SHA384
tee_api_defines.h, 122
TEE_ALG_SHA512
tee_api_defines.h, 122

TEE_AllocateOperation
tee-internal-api-cryptlib.c, 243
tee-ta-internal.h, 91
TEE_AllocateTransientObject
tee-internal-api-cryptlib.c, 243
tee-ta-internal.h, 92
tee_api_defines.h
TEE_ALG_AES_CBC_MAC_NOPAD, 116
TEE_ALG_AES_CBC_MAC_PKCS5, 116
TEE_ALG_AES_CBC_NOPAD, 117
TEE_ALG_AES_CCM, 117
TEE_ALG_AES_CMAC, 117
TEE_ALG_AES_CTR, 117
TEE_ALG_AES_CTS, 117
TEE_ALG_AES_ECB_NOPAD, 117
TEE_ALG_AES_GCM, 117
TEE_ALG_AES_XTS, 117
TEE_ALG_DES3_CBC_MAC_NOPAD, 117
TEE_ALG_DES3_CBC_MAC_PKCS5, 117
TEE_ALG_DES3_CBC_NOPAD, 117
TEE_ALG_DES3_ECB_NOPAD, 118
TEE_ALG_DES_CBC_MAC_NOPAD, 118
TEE_ALG_DES_CBC_MAC_PKCS5, 118
TEE_ALG_DES_CBC_NOPAD, 118
TEE_ALG_DES_ECB_NOPAD, 118
TEE_ALG_DH_DERIVE_SHARED_SECRET, 118
TEE_ALG_DSA_SHA1, 118
TEE_ALG_DSA_SHA224, 118
TEE_ALG_DSA_SHA256, 118
TEE_ALG_ECDH_P192, 118
TEE_ALG_ECDH_P224, 118
TEE_ALG_ECDH_P256, 119
TEE_ALG_ECDH_P384, 119
TEE_ALG_ECDH_P521, 119
TEE_ALG_ECDSA_P192, 119
TEE_ALG_ECDSA_P224, 119
TEE_ALG_ECDSA_P256, 119
TEE_ALG_ECDSA_P384, 119
TEE_ALG_ECDSA_P521, 119
TEE_ALG_HMAC_MD5, 119
TEE_ALG_HMAC_SHA1, 119
TEE_ALG_HMAC_SHA224, 119
TEE_ALG_HMAC_SHA256, 120
TEE_ALG_HMAC_SHA384, 120
TEE_ALG_HMAC_SHA512, 120
TEE_ALG_MD5, 120
TEE_ALG_MD5SHA1, 120
TEE_ALG_RSA_NOPAD, 120
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1, 120
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224, 120
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256, 120
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384, 120
TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512, 120
TEE_ALG_RSAES_PKCS1_V1_5, 121
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1, 121
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224, 121
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256, 121
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384, 121
TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512, 121
TEE_ALG_RSASSA_PKCS1_V1_5_MD5, 121
TEE_ALG_RSASSA_PKCS1_V1_5_MD5SHA1, 121
TEE_ALG_RSASSA_PKCS1_V1_5_SHA1, 121
TEE_ALG_RSASSA_PKCS1_V1_5_SHA224, 121
TEE_ALG_RSASSA_PKCS1_V1_5_SHA256, 122
TEE_ALG_RSASSA_PKCS1_V1_5_SHA384, 122
TEE_ALG_RSASSA_PKCS1_V1_5_SHA512, 122
TEE_ALG_SHA1, 122
TEE_ALG_SHA224, 122
TEE_ALG_SHA256, 122
TEE_ALG_SHA384, 122
TEE_ALG_SHA512, 122
TEE_ATTR_BIT_PROTECTED, 122
TEE_ATTR_BIT_VALUE, 122
TEE_ATTR_DH_BASE, 122
TEE_ATTR_DH_PRIME, 123
TEE_ATTR_DH_PRIVATE_VALUE, 123
TEE_ATTR_DH_PUBLIC_VALUE, 123
TEE_ATTR_DH_SUBPRIME, 123
TEE_ATTR_DH_X_BITS, 123
TEE_ATTR_DSA_BASE, 123
TEE_ATTR_DSA_PRIME, 123
TEE_ATTR_DSA_PRIVATE_VALUE, 123
TEE_ATTR_DSA_PUBLIC_VALUE, 123
TEE_ATTR_DSA_SUBPRIME, 123
TEE_ATTR_ECC_CURVE, 123
TEE_ATTR_ECC_PRIVATE_VALUE, 124
TEE_ATTR_ECC_PUBLIC_VALUE_X, 124
TEE_ATTR_ECC_PUBLIC_VALUE_Y, 124
TEE_ATTR_RSA_COEFFICIENT, 124
TEE_ATTR_RSA_EXPONENT1, 124
TEE_ATTR_RSA_EXPONENT2, 124
TEE_ATTR_RSA_MODULUS, 124
TEE_ATTR_RSA_OAEP_LABEL, 124
TEE_ATTR_RSA_PRIME1, 124
TEE_ATTR_RSA_PRIME2, 124
TEE_ATTR_RSA_PRIVATE_EXPONENT, 124
TEE_ATTR_RSA_PSS_SALT_LENGTH, 125
TEE_ATTR_RSA_PUBLIC_EXPONENT, 125
TEE_ATTR_SECRET_VALUE, 125
TEE_BigIntSizeInU32, 125
TEE_DATA_FLAG_ACCESS_READ, 125
TEE_DATA_FLAG_ACCESS_WRITE, 125
TEE_DATA_FLAG_ACCESS_WRITE_META, 125
TEE_DATA_FLAG_OVERWRITE, 125
TEE_DATA_FLAG_SHARE_READ, 125
TEE_DATA_FLAG_SHARE_WRITE, 125

TEE_DATA_MAX_POSITION, 125
TEE_ECC_CURVE_NIST_P192, 126
TEE_ECC_CURVE_NIST_P224, 126
TEE_ECC_CURVE_NIST_P256, 126
TEE_ECC_CURVE_NIST_P384, 126
TEE_ECC_CURVE_NIST_P521, 126
TEE_ERROR_ACCESS_CONFLICT, 126
TEE_ERROR_ACCESS_DENIED, 126
TEE_ERROR_BAD_FORMAT, 126
TEE_ERROR_BAD_PARAMETERS, 126
TEE_ERROR_BAD_STATE, 126
TEE_ERROR_BUSY, 126
TEE_ERROR_CANCEL, 127
TEE_ERROR_COMMUNICATION, 127
TEE_ERROR_CORRUPT_OBJECT, 127
TEE_ERROR_CORRUPT_OBJECT_2, 127
TEE_ERROR_EXCESS_DATA, 127
TEE_ERROR_EXTERNAL_CANCEL, 127
TEE_ERROR_GENERIC, 127
TEE_ERROR_ITEM_NOT_FOUND, 127
TEE_ERROR_MAC_INVALID, 127
TEE_ERROR_NO_DATA, 127
TEE_ERROR_NOT_IMPLEMENTED, 127
TEE_ERROR_NOT_SUPPORTED, 128
TEE_ERROR_OUT_OF_MEMORY, 128
TEE_ERROR_OVERFLOW, 128
TEE_ERROR_SECURITY, 128
TEE_ERROR_SHORT_BUFFER, 128
TEE_ERROR_SIGNATURE_INVALID, 128
TEE_ERROR_STORAGE_NO_SPACE, 128
TEE_ERROR_STORAGE_NOT_AVAILABLE, 128
TEE_ERROR_STORAGE_NOT_AVAILABLE_2, 128
TEE_ERROR_TARGET_DEAD, 128
TEE_ERROR_TIME_NEEDS_RESET, 128
TEE_ERROR_TIME_NOT_SET, 129
TEE_HANDLE_FLAG_EXPECT_TWO_KEYS, 129
TEE_HANDLE_FLAG_INITIALIZED, 129
TEE_HANDLE_FLAG_KEY_SET, 129
TEE_HANDLE_FLAG_PERSISTENT, 129
TEE_HANDLE_NULL, 129
TEE_INT_CORE_API_SPEC_VERSION, 129
TEE_LOGIN_APPLICATION, 129
TEE_LOGIN_APPLICATION_GROUP, 129
TEE_LOGIN_APPLICATION_USER, 129
TEE_LOGIN_GROUP, 129
TEE_LOGIN_PUBLIC, 130
TEE_LOGIN_TRUSTED_APP, 130
TEE_LOGIN_USER, 130
TEE_MALLOC_FILL_ZERO, 130
TEE_MEMORY_ACCESS_ANY_OWNER, 130
TEE_MEMORY_ACCESS_READ, 130
TEE_MEMORY_ACCESS_WRITE, 130
TEE_NUM_PARAMS, 130
TEE_OBJECT_ID_MAX_LEN, 130
TEE_OPERATION_AE, 130
TEE_OPERATION_ASYMMETRIC_CIPHER, 130
TEE_OPERATION_ASYMMETRIC_SIGNATURE, 131
TEE_OPERATION_CIPHER, 131
TEE_OPERATION_DIGEST, 131
TEE_OPERATION_KEY_DERIVATION, 131
TEE_OPERATION_MAC, 131
TEE_OPERATION_STATE_ACTIVE, 131
TEE_OPERATION_STATE_INITIAL, 131
TEE_ORIGIN_API, 131
TEE_ORIGIN_COMMS, 131
TEE_ORIGIN_TEE, 131
TEE_ORIGIN_TRUSTED_APP, 131
TEE_PANIC_ID_TA_CLOSESESSIONENTRYPOINT, 132
TEE_PANIC_ID_TA_CREATEENTRYPOINT, 132
TEE_PANIC_ID_TA_DESTROYENTRYPOINT, 132
TEE_PANIC_ID_TA_INVOKECOMMANDENTRYPOINT, 132
TEE_PANIC_ID_TA_OPENSESSIONENTRYPOINT, 132
TEE_PANIC_ID_TEE_AEDECRYPTFINAL, 132
TEE_PANIC_ID_TEE_AEENCRYPTFINAL, 132
TEE_PANIC_ID_TEE_AEINIT, 132
TEE_PANIC_ID_TEE_AEUPDATE, 132
TEE_PANIC_ID_TEE_AEUPDATEAAD, 132
TEE_PANIC_ID_TEE_ALLOCATEOPERATION, 132
TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATO 133
TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERATOR, 133
TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT, 133
TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT, 133
TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT, 133
TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST, 133
TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST, 133
TEE_PANIC_ID_TEE_BIGINTADD, 133
TEE_PANIC_ID_TEE_BIGINTADDMOD, 133
TEE_PANIC_ID_TEE_BIGINTCMP, 133
TEE_PANIC_ID_TEE_BIGINTCMPS32, 133
TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD, 134
TEE_PANIC_ID_TEE_BIGINTCOMPUTEFMM, 134
TEE_PANIC_ID_TEE_BIGINTCONVERTFROMFMM, 134
TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING, 134
TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32, 134
TEE_PANIC_ID_TEE_BIGINTCONVERTTOFMM, 134
TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING, 134
TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32, 134
TEE_PANIC_ID_TEE_BIGINTDIV, 134

TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32, 134
TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32, 135
TEE_PANIC_ID_TEE_BIGINTGETBIT, 135
TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT, 135
TEE_PANIC_ID_TEE_BIGINTINIT, 135
TEE_PANIC_ID_TEE_BIGINTINITFMM, 135
TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT, 135
TEE_PANIC_ID_TEE_BIGINTINVMOD, 135
TEE_PANIC_ID_TEE_BIGINTISPROBABLEPRIME, 135
TEE_PANIC_ID_TEE_BIGINTMOD, 135
TEE_PANIC_ID_TEE_BIGINTMUL, 135
TEE_PANIC_ID_TEE_BIGINTMULMOD, 135
TEE_PANIC_ID_TEE_BIGINTNEG, 136
TEE_PANIC_ID_TEE_BIGINTRELATIVEPRIME, 136
TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT, 136
TEE_PANIC_ID_TEE_BIGINTSQUARE, 136
TEE_PANIC_ID_TEE_BIGINTSQUAREMOD, 136
TEE_PANIC_ID_TEE_BIGINTSUB, 136
TEE_PANIC_ID_TEE_BIGINTSUBMOD, 136
TEE_PANIC_ID_TEE_CHECKMEMORYACCESSRIGHTS, 136
TEE_PANIC_ID_TEE_CIPHERDOFINAL, 136
TEE_PANIC_ID_TEE_CIPHERINIT, 136
TEE_PANIC_ID_TEE_CIPHERUPDATE, 136
TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT, 137
TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT1, 137
TEE_PANIC_ID_TEE_CLOSEOBJECT, 137
TEE_PANIC_ID_TEE_CLOSETASESSION, 137
TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES, 137
TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES1, 137
TEE_PANIC_ID_TEE_COPYOPERATION, 137
TEE_PANIC_ID_TEE_CREATEPERSISTENTOBJECT, 137
TEE_PANIC_ID_TEE_DERIVEKEY, 137
TEE_PANIC_ID_TEE_DIGESTDOFINAL, 137
TEE_PANIC_ID_TEE_DIGESTUPDATE, 137
TEE_PANIC_ID_TEE_FREE, 138
TEE_PANIC_ID_TEE_FREEOPERATION, 138
TEE_PANIC_ID_TEE_FREEPERSISTENTOBJECTENUMERATOR, 138
TEE_PANIC_ID_TEE_FREEPROPERTYENUMERATOR, 138
TEE_PANIC_ID_TEE_FREETRANSIENTOBJECT, 138
TEE_PANIC_ID_TEE_GENERATEKEY, 138
TEE_PANIC_ID_TEE_GENERATERANDOM, 138
TEE_PANIC_ID_TEE_GETCANCELLATIONFLAG, 138
TEE_PANIC_ID_TEE_GETINSTANCEDATA, 138
TEE_PANIC_ID_TEE_GETNEXTPERSISTENTOBJECT, 138
TEE_PANIC_ID_TEE_GETNEXTPROPERTY, 138
TEE_PANIC_ID_TEE_GETOBJECTBUFFERATTRIBUTE, 139
TEE_PANIC_ID_TEE_GETOBJECTINFO, 139
TEE_PANIC_ID_TEE_GETOBJECTINFO1, 139
TEE_PANIC_ID_TEE_GETOBJECTVALUEATTRIBUTE, 139
TEE_PANIC_ID_TEE_GETOPERATIONINFO, 139
TEE_PANIC_ID_TEE_GETOPERATIONINFOMULTIPLE, 139
TEE_PANIC_ID_TEE_GETPROPERTYASBINARYBLOCK, 139
TEE_PANIC_ID_TEE_GETPROPERTYASBOOL, 139
TEE_PANIC_ID_TEE_GETPROPERTYASIDENTITY, 139
TEE_PANIC_ID_TEE_GETPROPERTYASSTRING, 139
TEE_PANIC_ID_TEE_GETPROPERTYASU32, 139
TEE_PANIC_ID_TEE_GETPROPERTYASUUID, 140
TEE_PANIC_ID_TEE_GETPROPERTYNAME, 140
TEE_PANIC_ID_TEE_GETREETIME, 140
TEE_PANIC_ID_TEE_GETSYSTEMTIME, 140
TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME, 140
TEE_PANIC_ID_TEE_INITREFATTRIBUTE, 140
TEE_PANIC_ID_TEE_INITVALUEATTRIBUTE, 140
TEE_PANIC_ID_TEE_INVOKETACOMMAND, 140
TEE_PANIC_ID_TEE_MACCOMPAREFINAL, 140
TEE_PANIC_ID_TEE_MACCOMPUTEFINAL, 140
TEE_PANIC_ID_TEE_MACINIT, 140
TEE_PANIC_ID_TEE_MACUPDATE, 141
TEE_PANIC_ID_TEE_MALLOC, 141
TEE_PANIC_ID_TEE_MASKCANCELLATION, 141
TEE_PANIC_ID_TEE_MEMCOMPARE, 141
TEE_PANIC_ID_TEE_MEMFILL, 141
TEE_PANIC_ID_TEE_MEMMOVE, 141
TEE_PANIC_ID_TEE_OPENPERSISTENTOBJECT, 141
TEE_PANIC_ID_TEE_OPENTASESSION, 141
TEE_PANIC_ID_TEE_PANIC, 141
TEE_PANIC_ID_TEE_POPULATETRANSIENTOBJECT, 141
TEE_PANIC_ID_TEE_READOBJECTDATA, 141
TEE_PANIC_ID_TEE_REALLOC, 142
TEE_PANIC_ID_TEE_RENAMEPERSISTENTOBJECT, 142
TEE_PANIC_ID_TEE_RESETOPERATION, 142
TEE_PANIC_ID_TEE_RESETPERSISTENTOBJECTENUMERATOR, 142
TEE_PANIC_ID_TEE_RESETPROPERTYENUMERATOR, 142
TEE_PANIC_ID_TEE_RESETTRANSIENTOBJECT, 142
TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE, 142

TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE1, 142
TEE_PANIC_ID_TEE_SEEKOBJECTDATA, 142
TEE_PANIC_ID_TEE_SETINSTANCEDATA, 142
TEE_PANIC_ID_TEE_SETOPERATIONKEY, 142
TEE_PANIC_ID_TEE_SETOPERATIONKEY2, 143
TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME, 143
TEE_PANIC_ID_TEE_STARTPERSISTENTOBJECTENUMERATOR, 143
TEE_PANIC_ID_TEE_STARTPROPERTYENUMERATOR, 143
TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA, 143
TEE_PANIC_ID_TEE_UNMASKCANCELLATION, 143
TEE_PANIC_ID_TEE_WAIT, 143
TEE_PANIC_ID_TEE_WRITEOBJECTDATA, 143
TEE_PARAM_TYPE_GET, 143
TEE_PARAM_TYPE_MEMREF_INOUT, 143
TEE_PARAM_TYPE_MEMREF_INPUT, 144
TEE_PARAM_TYPE_MEMREF_OUTPUT, 144
TEE_PARAM_TYPE_NONE, 144
TEE_PARAM_TYPE_SET, 144
TEE_PARAM_TYPE_VALUE_INOUT, 144
TEE_PARAM_TYPE_VALUE_INPUT, 144
TEE_PARAM_TYPE_VALUE_OUTPUT, 144
TEE_PARAM_TYPES, 144
TEE_PROPSET_CURRENT_CLIENT, 144
TEE_PROPSET_CURRENT_TA, 144
TEE_PROPSET_TEE_IMPLEMENTATION, 145
TEE_STORAGE_PRIVATE, 145
TEE_SUCCESS, 145
TEE_TIMEOUT_INFINITE, 145
TEE_TYPE_AES, 145
TEE_TYPE_CORRUPTED_OBJECT, 145
TEE_TYPE_DATA, 145
TEE_TYPE_DES, 145
TEE_TYPE_DES3, 145
TEE_TYPE_DH_KEYPAIR, 145
TEE_TYPE_DSA_KEYPAIR, 145
TEE_TYPE_DSA_PUBLIC_KEY, 146
TEE_TYPE_ECDH_KEYPAIR, 146
TEE_TYPE_ECDH_PUBLIC_KEY, 146
TEE_TYPE_ECDSA_KEYPAIR, 146
TEE_TYPE_ECDSA_PUBLIC_KEY, 146
TEE_TYPE_GENERIC_SECRET, 146
TEE_TYPE_HMAC_MD5, 146
TEE_TYPE_HMAC_SHA1, 146
TEE_TYPE_HMAC_SHA224, 146
TEE_TYPE_HMAC_SHA256, 146
TEE_TYPE_HMAC_SHA384, 146
TEE_TYPE_HMAC_SHA512, 147
TEE_TYPE_RSA_KEYPAIR, 147
TEE_TYPE_RSA_PUBLIC_KEY, 147
TEE_USAGE_DECRYPT, 147
TEE_USAGE_DERIVE, 147
TEE_USAGE_ENCRYPT, 147
TEE_USAGE_EXTRACTABLE, 147
TEE_USAGE_MAC, 147
TEE_USAGE_SIGN, 147
TEE_USAGE_VERIFY, 147
tee_api_defines_extensions.h
TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY, 154
TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY, 154
TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY, 154
TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY, 154
TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY, 155
TEE_ALG_HKDF_MD5_DERIVE_KEY, 155
TEE_ALG_HKDF_SHA1_DERIVE_KEY, 155
TEE_ALG_HKDF_SHA224_DERIVE_KEY, 155
TEE_ALG_HKDF_SHA256_DERIVE_KEY, 155
TEE_ALG_HKDF_SHA384_DERIVE_KEY, 155
TEE_ALG_HKDF_SHA512_DERIVE_KEY, 155
TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY, 155
TEE_ATTR_CONCAT_KDF_DKM_LENGTH, 155
TEE_ATTR_CONCAT_KDF_OTHER_INFO, 155
TEE_ATTR_CONCAT_KDF_Z, 155
TEE_ATTR_HKDF_IKM, 156
TEE_ATTR_HKDF_INFO, 156
TEE_ATTR_HKDF_OKM_LENGTH, 156
TEE_ATTR_HKDF_SALT, 156
TEE_ATTR_PBKDF2_DKM_LENGTH, 156
TEE_ATTR_PBKDF2_ITERATION_COUNT, 156
TEE_ATTR_PBKDF2_PASSWORD, 156
TEE_ATTR_PBKDF2_SALT, 156
TEE_MEMORY_ACCESS_NONSECURE, 156
TEE_MEMORY_ACCESS_SECURE, 156
TEE_STORAGE_PRIVATE_REE, 156
TEE_STORAGE_PRIVATE_RPMB, 157
TEE_STORAGE_PRIVATE_SQL_RESERVED, 157
TEE_TYPE_CONCAT_KDF_Z, 157
TEE_TYPE_HKDF_IKM, 157
TEE_TYPE_PBKDF2_PASSWORD, 157
tee_api_tee_types.h
AES256, 222, 226
MBEDCRYPT, 222, 226
SHA_LENGTH, 222, 226
TEE_HANDLE_NULL, 226
TEE_OBJECT_AAD_SIZE, 222, 226
TEE_OBJECT_KEY_SIZE, 222, 226
TEE_OBJECT_NONCE_SIZE, 222, 226
TEE_OBJECT_SKEY_SIZE, 222, 226
TEE_OBJECT_TAG_SIZE, 222, 226
WOLFCRYPT, 222, 226
tee_api_types.h
__aligned, 161
DMREQ_FINISH, 160
DMREQ_WRITE, 160
nfds_t, 161

socklen_t, 161
TEE_BigInt, 161
TEE_BigIntFMM, 161
TEE_DATA_SEEK_CUR, 163
TEE_DATA_SEEK_END, 163
TEE_DATA_SEEK_SET, 163
TEE_ErrorOrigin, 161
TEE_MEM_INPUT, 160
TEE_MEM_OUTPUT, 160
TEE_MEMREF_0_USED, 160
TEE_MEMREF_1_USED, 161
TEE_MEMREF_2_USED, 161
TEE_MEMREF_3_USED, 161
TEE_MODE_DECRYPT, 163
TEE_MODE_DERIVE, 163
TEE_MODE_DIGEST, 163
TEE_MODE_ENCRYPT, 163
TEE_MODE_MAC, 163
TEE_MODE_SIGN, 163
TEE_MODE_VERIFY, 163
TEE_ObjectEnumHandle, 161
TEE_ObjectHandle, 162
TEE_ObjectType, 162
TEE_OperationHandle, 162
TEE_OperationMode, 162
TEE_PropSetHandle, 162
TEE_Result, 162
TEE_SE_READER_NAME_MAX, 161
TEE_SEChannelHandle, 162
TEE_SEReaderHandle, 162
TEE_SEServiceHandle, 162
TEE_SESessionHandle, 162
TEE_Session, 162
TEE_TASessionHandle, 162
TEE_Whence, 163
TEE_AsymmetricSignDigest
tee-internal-api-cryptlib.c, 244
tee-ta-internal.h, 92
TEE_AsymmetricVerifyDigest
tee-internal-api-cryptlib.c, 245
tee-ta-internal.h, 93
TEE_ATTR_BIT_PROTECTED
tee_api_defines.h, 122
TEE_ATTR_BIT_VALUE
tee_api_defines.h, 122
TEE_ATTR_CONCAT_KDF_DKM_LENGTH
tee_api_defines_extensions.h, 155
TEE_ATTR_CONCAT_KDF_OTHER_INFO
tee_api_defines_extensions.h, 155
TEE_ATTR_CONCAT_KDF_Z
tee_api_defines_extensions.h, 155
TEE_ATTR_DH_BASE
tee_api_defines.h, 122
TEE_ATTR_DH_PRIME
tee_api_defines.h, 123
TEE_ATTR_DH_PRIVATE_VALUE
tee_api_defines.h, 123
TEE_ATTR_DH_PUBLIC_VALUE
tee_api_defines.h, 123
TEE_ATTR_DH_SUBPRIME
tee_api_defines.h, 123
TEE_ATTR_DH_X_BITS
tee_api_defines.h, 123
TEE_ATTR_DSA_BASE
tee_api_defines.h, 123
TEE_ATTR_DSA_PRIME
tee_api_defines.h, 123
TEE_ATTR_DSA_PRIVATE_VALUE
tee_api_defines.h, 123
TEE_ATTR_DSA_PUBLIC_VALUE
tee_api_defines.h, 123
TEE_ATTR_DSA_SUBPRIME
tee_api_defines.h, 123
TEE_ATTR_ECC_CURVE
tee_api_defines.h, 123
TEE_ATTR_ECC_PRIVATE_VALUE
tee_api_defines.h, 124
TEE_ATTR_ECC_PUBLIC_VALUE_X
tee_api_defines.h, 124
TEE_ATTR_ECC_PUBLIC_VALUE_Y
tee_api_defines.h, 124
TEE_ATTR_HKDF_IKM
tee_api_defines_extensions.h, 156
TEE_ATTR_HKDF_INFO
tee_api_defines_extensions.h, 156
TEE_ATTR_HKDF_OKM_LENGTH
tee_api_defines_extensions.h, 156
TEE_ATTR_HKDF_SALT
tee_api_defines_extensions.h, 156
TEE_ATTR_PBKDF2_DKM_LENGTH
tee_api_defines_extensions.h, 156
TEE_ATTR_PBKDF2_ITERATION_COUNT
tee_api_defines_extensions.h, 156
TEE_ATTR_PBKDF2_PASSWORD
tee_api_defines_extensions.h, 156
TEE_ATTR_PBKDF2_SALT
tee_api_defines_extensions.h, 156
TEE_ATTR_RSA_COEFFICIENT
tee_api_defines.h, 124
TEE_ATTR_RSA_EXPONENT1
tee_api_defines.h, 124
TEE_ATTR_RSA_EXPONENT2
tee_api_defines.h, 124
TEE_ATTR_RSA_MODULUS
tee_api_defines.h, 124
TEE_ATTR_RSA_OAEP_LABEL
tee_api_defines.h, 124
TEE_ATTR_RSA_PRIME1
tee_api_defines.h, 124
TEE_ATTR_RSA_PRIME2
tee_api_defines.h, 124
TEE_ATTR_RSA_PRIVATE_EXPONENT
tee_api_defines.h, 124
TEE_ATTR_RSA_PSS_SALT_LENGTH
tee_api_defines.h, 125
TEE_ATTR_RSA_PUBLIC_EXPONENT

tee_api_defines.h, 125
TEE_ATTR_SECRET_VALUE
tee_api_defines.h, 125
TEE_Attribute, 50
a, 51
attributeID, 51
b, 51
buffer, 51
content, 51
length, 51
ref, 51
value, 51
TEE_BigInt
tee_api_types.h, 161
TEE_BigIntFMM
tee_api_types.h, 161
TEE_BigIntSizeInU32
tee_api_defines.h, 125
TEE_CacheClean
tee_internal_api_extensions.h, 182
TEE_CacheFlush
tee_internal_api_extensions.h, 182
TEE_CacheInvalidate
tee_internal_api_extensions.h, 183
TEE_CipherDoFinal
tee-internal-api-cryptlib.c, 245
TEE_CipherInit
tee-internal-api-cryptlib.c, 246
tee-ta-internal.h, 94
TEE_CipherUpdate
tee-internal-api-cryptlib.c, 246
tee-ta-internal.h, 94
tee_client_api.h
TEEC_AllocateSharedMemory, 174
TEEC_CloseSession, 175
TEEC_CONFIG_PAYLOAD_REF_COUNT, 168
TEEC_CONFIG_SHAREDMEM_MAX_SIZE, 168
TEEC_ERROR_ACCESS_CONFLICT, 169
TEEC_ERROR_ACCESS_DENIED, 169
TEEC_ERROR_BAD_FORMAT, 169
TEEC_ERROR_BAD_PARAMETERS, 169
TEEC_ERROR_BAD_STATE, 169
TEEC_ERROR_BUSY, 169
TEEC_ERROR_CANCEL, 169
TEEC_ERROR_COMMUNICATION, 169
TEEC_ERROR_EXCESS_DATA, 169
TEEC_ERROR_EXTERNAL_CANCEL, 169
TEEC_ERROR_GENERIC, 169
TEEC_ERROR_ITEM_NOT_FOUND, 170
TEEC_ERROR_NO_DATA, 170
TEEC_ERROR_NOT_IMPLEMENTED, 170
TEEC_ERROR_NOT_SUPPORTED, 170
TEEC_ERROR_OUT_OF_MEMORY, 170
TEEC_ERROR_SECURITY, 170
TEEC_ERROR_SHORT_BUFFER, 170
TEEC_ERROR_TARGET_DEAD, 170
TEEC_FinalizeContext, 175
TEEC_InitializeContext, 176
TEEC_InvokeCommand, 176
TEEC_LOGIN_APPLICATION, 170
TEEC_LOGIN_GROUP, 170
TEEC_LOGIN_GROUP_APPLICATION, 170
TEEC_LOGIN_PUBLIC, 171
TEEC_LOGIN_USER, 171
TEEC_LOGIN_USER_APPLICATION, 171
TEEC_MEM_INPUT, 171
TEEC_MEM_OUTPUT, 171
TEEC_MEMREF_PARTIAL_INOUT, 171
TEEC_MEMREF_PARTIAL_INPUT, 171
TEEC_MEMREF_PARTIAL_OUTPUT, 171
TEEC_MEMREF_TEMP_INOUT, 171
TEEC_MEMREF_TEMP_INPUT, 172
TEEC_MEMREF_TEMP_OUTPUT, 172
TEEC_MEMREF_WHOLE, 172
TEEC_NONE, 172
TEEC_OpenSession, 177
TEEC_ORIGIN_API, 172
TEEC_ORIGIN_COMMS, 173
TEEC_ORIGIN_TEE, 173
TEEC_ORIGIN_TRUSTED_APP, 173
TEEC_PARAM_TYPE_GET, 173
TEEC_PARAM_TYPES, 173
TEEC_RegisterSharedMemory, 177
TEEC_ReleaseSharedMemory, 178
TEEC_RequestCancellation, 178
TEEC_Result, 174
TEEC_SUCCESS, 174
TEEC_VALUE_INOUT, 174
TEEC_VALUE_INPUT, 174
TEEC_VALUE_OUTPUT, 174
TEE_CloseObject
tee-internal-api.c, 205, 216
tee-ta-internal.h, 95
tee_config.h
__ImageBase, 317, 319, 320
COMMAND, 319
perf_buffer, 317, 319, 320
tee_rdtscp, 317, 319, 320
TEE_CreatePersistentObject
tee-internal-api.c, 206, 217
tee-ta-internal.h, 96
TEE_DATA_FLAG_ACCESS_READ
tee_api_defines.h, 125
TEE_DATA_FLAG_ACCESS_WRITE
tee_api_defines.h, 125
TEE_DATA_FLAG_ACCESS_WRITE_META
tee_api_defines.h, 125
TEE_DATA_FLAG_OVERWRITE
tee_api_defines.h, 125
TEE_DATA_FLAG_SHARE_READ
tee_api_defines.h, 125
TEE_DATA_FLAG_SHARE_WRITE
tee_api_defines.h, 125
TEE_DATA_MAX_POSITION
tee_api_defines.h, 125
TEE_DATA_SEEK_CUR

tee_api_types.h, 163
TEE_DATA_SEEK_END
tee_api_types.h, 163
TEE_DATA_SEEK_SET
tee_api_types.h, 163
tee_def.h
buf, 265–267
buf_flag, 265, 266, 268
tee_init, 265–267
test_printf, 265–267
TEE_DigestDoFinal
tee-internal-api-cryptlib.c, 247
tee-ta-internal.h, 98
TEE_DigestUpdate
tee-internal-api-cryptlib.c, 247
tee-ta-internal.h, 98
TEE_ECC_CURVE_NIST_P192
tee_api_defines.h, 126
TEE_ECC_CURVE_NIST_P224
tee_api_defines.h, 126
TEE_ECC_CURVE_NIST_P256
tee_api_defines.h, 126
TEE_ECC_CURVE_NIST_P384
tee_api_defines.h, 126
TEE_ECC_CURVE_NIST_P521
tee_api_defines.h, 126
TEE_ERROR_ACCESS_CONFLICT
tee_api_defines.h, 126
TEE_ERROR_ACCESS_DENIED
tee_api_defines.h, 126
TEE_ERROR_BAD_FORMAT
tee_api_defines.h, 126
TEE_ERROR_BAD_PARAMETERS
tee_api_defines.h, 126
TEE_ERROR_BAD_STATE
tee_api_defines.h, 126
TEE_ERROR_BUSY
tee_api_defines.h, 126
TEE_ERROR_CANCEL
tee_api_defines.h, 127
TEE_ERROR_COMMUNICATION
tee_api_defines.h, 127
TEE_ERROR_CORRUPT_OBJECT
tee_api_defines.h, 127
TEE_ERROR_CORRUPT_OBJECT_2
tee_api_defines.h, 127
TEE_ERROR_EXCESS_DATA
tee_api_defines.h, 127
TEE_ERROR_EXTERNAL_CANCEL
tee_api_defines.h, 127
TEE_ERROR_GENERIC
tee_api_defines.h, 127
TEE_ERROR_ITEM_NOT_FOUND
tee_api_defines.h, 127
TEE_ERROR_MAC_INVALID
tee_api_defines.h, 127
TEE_ERROR_NO_DATA
tee_api_defines.h, 127
TEE_ERROR_NOT_IMPLEMENTED
tee_api_defines.h, 127
TEE_ERROR_NOT_SUPPORTED
tee_api_defines.h, 128
TEE_ERROR_OUT_OF_MEMORY
tee_api_defines.h, 128
TEE_ERROR_OVERFLOW
tee_api_defines.h, 128
TEE_ERROR_SECURITY
tee_api_defines.h, 128
TEE_ERROR_SHORT_BUFFER
tee_api_defines.h, 128
TEE_ERROR_SIGNATURE_INVALID
tee_api_defines.h, 128
TEE_ERROR_STORAGE_NO_SPACE
tee_api_defines.h, 128
TEE_ERROR_STORAGE_NOT_AVAILABLE
tee_api_defines.h, 128
TEE_ERROR_STORAGE_NOT_AVAILABLE_2
tee_api_defines.h, 128
TEE_ERROR_TARGET_DEAD
tee_api_defines.h, 128
TEE_ERROR_TIME_NEEDS_RESET
tee_api_defines.h, 128
TEE_ERROR_TIME_NOT_SET
tee_api_defines.h, 129
TEE_ErrorOrigin
tee_api_types.h, 161
TEE_Free
tee-internal-api.c, 207
TEE_FreeOperation
tee-internal-api-cryptlib.c, 248
tee-ta-internal.h, 99
TEE_FreeTransientObject
tee-internal-api-cryptlib.c, 248
tee-ta-internal.h, 99
TEE_GenerateKey
tee-internal-api-cryptlib.c, 249
tee-ta-internal.h, 100
TEE_GenerateRandom
tee-internal-api.c, 207, 217
tee-ta-internal.h, 100
TEE_GetObjectInfo1
tee-internal-api.c, 208, 218
tee-ta-internal.h, 102
TEE_GetREETime
tee-internal-api.c, 208, 218
tee-ta-internal.h, 102
TEE_GetSystemTime
tee-internal-api.c, 209, 219
tee-ta-internal.h, 103
TEE_HANDLE_FLAG_EXPECT_TWO_KEYS
tee_api_defines.h, 129
TEE_HANDLE_FLAG_INITIALIZED
tee_api_defines.h, 129
TEE_HANDLE_FLAG_KEY_SET
tee_api_defines.h, 129
TEE_HANDLE_FLAG_PERSISTENT

tee_api_defines.h, 129
TEE_HANDLE_NULL
tee_api_defines.h, 129
tee_api_tee_types.h, 226
TEE_Identity, 52
login, 52
uuid, 52
tee_init
tee_def.h, 265–267
TEE_InitRefAttribute
tee-internal-api-cryptlib.c, 249
tee-ta-internal.h, 103
TEE_InitValueAttribute
tee-internal-api-cryptlib.c, 250
tee-ta-internal.h, 104
TEE_INT_CORE_API_SPEC_VERSION
tee_api_defines.h, 129
tee_internal_api_extensions.h
TEE_CacheClean, 182
TEE_CacheFlush, 182
TEE_CacheInvalidate, 183
tee_map_zi, 183
tee_unmap, 183
tee_user_mem_check_heap, 183
TEE_USER_MEM_HINT_NO_FILL_ZERO, 182
tee_user_mem_mark_heap, 183
tee_uuid_from_str, 183
TEE_LOGIN_APPLICATION
tee_api_defines.h, 129
TEE_LOGIN_APPLICATION_GROUP
tee_api_defines.h, 129
TEE_LOGIN_APPLICATION_USER
tee_api_defines.h, 129
TEE_LOGIN_GROUP
tee_api_defines.h, 129
TEE_LOGIN_PUBLIC
tee_api_defines.h, 130
TEE_LOGIN_TRUSTED_APP
tee_api_defines.h, 130
TEE_LOGIN_USER
tee_api_defines.h, 130
TEE_Malloc
tee-internal-api.c, 209
TEE_MALLOC_FILL_ZERO
tee_api_defines.h, 130
tee_map_zi
tee_internal_api_extensions.h, 183
TEE_MEM_INPUT
tee_api_types.h, 160
TEE_MEM_OUTPUT
tee_api_types.h, 160
TEE_MEMORY_ACCESS_ANY_OWNER
tee_api_defines.h, 130
TEE_MEMORY_ACCESS_NONSECURE
tee_api_defines_extensions.h, 156
TEE_MEMORY_ACCESS_READ
tee_api_defines.h, 130
TEE_MEMORY_ACCESS_SECURE
tee_api_defines_extensions.h, 156
TEE_MEMORY_ACCESS_WRITE
tee_api_defines.h, 130
TEE_MEMREF_0_USED
tee_api_types.h, 160
TEE_MEMREF_1_USED
tee_api_types.h, 161
TEE_MEMREF_2_USED
tee_api_types.h, 161
TEE_MEMREF_3_USED
tee_api_types.h, 161
TEE_MODE_DECRYPT
tee_api_types.h, 163
TEE_MODE_DERIVE
tee_api_types.h, 163
TEE_MODE_DIGEST
tee_api_types.h, 163
TEE_MODE_ENCRYPT
tee_api_types.h, 163
TEE_MODE_MAC
tee_api_types.h, 163
TEE_MODE_SIGN
tee_api_types.h, 163
TEE_MODE_VERIFY
tee_api_types.h, 163
TEE_NUM_PARAMS
tee_api_defines.h, 130
TEE_OBJECT_AAD_SIZE
tee_api_tee_types.h, 222, 226
TEE_OBJECT_ID_MAX_LEN
tee_api_defines.h, 130
TEE_OBJECT_KEY_SIZE
tee_api_tee_types.h, 222, 226
TEE_OBJECT_NONCE_SIZE
tee_api_tee_types.h, 222, 226
TEE_OBJECT_SKEY_SIZE
tee_api_tee_types.h, 222, 226
TEE_OBJECT_TAG_SIZE
tee_api_tee_types.h, 222, 226
TEE_ObjectEnumHandle
tee_api_types.h, 161
TEE_ObjectHandle
tee_api_types.h, 162
TEE_ObjectInfo, 53
dataPosition, 53
dataSize, 53
handleFlags, 53
keySize, 53
maxKeySize, 54
maxObjectSize, 54
objectSize, 54
objectType, 54
objectUsage, 54
TEE_ObjectType
tee_api_types.h, 162
TEE_OpenPersistentObject
tee-internal-api.c, 209, 219
tee-ta-internal.h, 104

TEE_OPERATION_AE
tee_api_defines.h, 130
TEE_OPERATION_ASYMMETRIC_CIPHER
tee_api_defines.h, 130
TEE_OPERATION_ASYMMETRIC_SIGNATURE
tee_api_defines.h, 131
TEE_OPERATION_CIPHER
tee_api_defines.h, 131
TEE_OPERATION_DIGEST
tee_api_defines.h, 131
TEE_OPERATION_KEY_DERIVATION
tee_api_defines.h, 131
TEE_OPERATION_MAC
tee_api_defines.h, 131
TEE_OPERATION_STATE_ACTIVE
tee_api_defines.h, 131
TEE_OPERATION_STATE_INITIAL
tee_api_defines.h, 131
TEE_OperationHandle
tee_api_types.h, 162
TEE_OperationInfo, 54
algorithm, 54
digestLength, 55
handleState, 55
keySize, 55
maxKeySize, 55
mode, 55
operationClass, 55
requiredKeyUsage, 55
TEE_OperationInfoKey, 55
keySize, 56
requiredKeyUsage, 56
TEE_OperationInfoMultiple, 56
algorithm, 56
digestLength, 57
handleState, 57
keyInformation, 57
maxKeySize, 57
mode, 57
numberOfKeys, 57
operationClass, 57
operationState, 57
TEE_OperationMode
tee_api_types.h, 162
TEE_ORIGIN_API
tee_api_defines.h, 131
TEE_ORIGIN_COMMS
tee_api_defines.h, 131
TEE_ORIGIN_TEE
tee_api_defines.h, 131
TEE_ORIGIN_TRUSTED_APP
tee_api_defines.h, 131
TEE_PANIC_ID_TA_CLOSESESSIONENTRYPOINT
tee_api_defines.h, 132
TEE_PANIC_ID_TA_CREATEENTRYPOINT
tee_api_defines.h, 132
TEE_PANIC_ID_TA_DESTROYENTRYPOINT
tee_api_defines.h, 132
TEE_PANIC_ID_TA_INVOKECOMMANDENTRYPOINT
tee_api_defines.h, 132
TEE_PANIC_ID_TA_OPENSESSIONENTRYPOINT
tee_api_defines.h, 132
TEE_PANIC_ID_TEE_AEDECRYPTFINAL
tee_api_defines.h, 132
TEE_PANIC_ID_TEE_AEENCRYPTFINAL
tee_api_defines.h, 132
TEE_PANIC_ID_TEE_AEINIT
tee_api_defines.h, 132
TEE_PANIC_ID_TEE_AEUPDATE
tee_api_defines.h, 132
TEE_PANIC_ID_TEE_AEUPDATEAAD
tee_api_defines.h, 132
TEE_PANIC_ID_TEE_ALLOCATEOPERATION
tee_api_defines.h, 132
TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR
tee_api_defines.h, 133
TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERATOR
tee_api_defines.h, 133
TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT
tee_api_defines.h, 133
TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT
tee_api_defines.h, 133
TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT
tee_api_defines.h, 133
TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST
tee_api_defines.h, 133
TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST
tee_api_defines.h, 133
TEE_PANIC_ID_TEE_BIGINTADD
tee_api_defines.h, 133
TEE_PANIC_ID_TEE_BIGINTADDMOD
tee_api_defines.h, 133
TEE_PANIC_ID_TEE_BIGINTCMP
tee_api_defines.h, 133
TEE_PANIC_ID_TEE_BIGINTCMPS32
tee_api_defines.h, 133
TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD
tee_api_defines.h, 134
TEE_PANIC_ID_TEE_BIGINTCOMPUTEFMM
tee_api_defines.h, 134
TEE_PANIC_ID_TEE_BIGINTCONVERTFROMFMM
tee_api_defines.h, 134
TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING
tee_api_defines.h, 134
TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32
tee_api_defines.h, 134
TEE_PANIC_ID_TEE_BIGINTCONVERTTOFMM
tee_api_defines.h, 134
TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING
tee_api_defines.h, 134
TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32
tee_api_defines.h, 134
TEE_PANIC_ID_TEE_BIGINTDIV
tee_api_defines.h, 134
TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32
tee_api_defines.h, 134

TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32
tee_api_defines.h, 135
TEE_PANIC_ID_TEE_BIGINTGETBIT
tee_api_defines.h, 135
TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT
tee_api_defines.h, 135
TEE_PANIC_ID_TEE_BIGINTINIT
tee_api_defines.h, 135
TEE_PANIC_ID_TEE_BIGINTINITFMM
tee_api_defines.h, 135
TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT
tee_api_defines.h, 135
TEE_PANIC_ID_TEE_BIGINTINVMOD
tee_api_defines.h, 135
TEE_PANIC_ID_TEE_BIGINTISPROBABLEPRIME
tee_api_defines.h, 135
TEE_PANIC_ID_TEE_BIGINTMOD
tee_api_defines.h, 135
TEE_PANIC_ID_TEE_BIGINTMUL
tee_api_defines.h, 135
TEE_PANIC_ID_TEE_BIGINTMULMOD
tee_api_defines.h, 135
TEE_PANIC_ID_TEE_BIGINTNEG
tee_api_defines.h, 136
TEE_PANIC_ID_TEE_BIGINTRELATIVEPRIME
tee_api_defines.h, 136
TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT
tee_api_defines.h, 136
TEE_PANIC_ID_TEE_BIGINTSQUARE
tee_api_defines.h, 136
TEE_PANIC_ID_TEE_BIGINTSQUAREMOD
tee_api_defines.h, 136
TEE_PANIC_ID_TEE_BIGINTSUB
tee_api_defines.h, 136
TEE_PANIC_ID_TEE_BIGINTSUBMOD
tee_api_defines.h, 136
TEE_PANIC_ID_TEE_CHECKMEMORYACCESSRIGHTS
tee_api_defines.h, 136
TEE_PANIC_ID_TEE_CIPHERDOFINAL
tee_api_defines.h, 136
TEE_PANIC_ID_TEE_CIPHERINIT
tee_api_defines.h, 136
TEE_PANIC_ID_TEE_CIPHERUPDATE
tee_api_defines.h, 136
TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT
tee_api_defines.h, 137
TEE_PANIC_ID_TEE_CLOSEANDDELETEPERSISTENTOBJECT1
tee_api_defines.h, 137
TEE_PANIC_ID_TEE_CLOSEOBJECT
tee_api_defines.h, 137
TEE_PANIC_ID_TEE_CLOSETASESSION
tee_api_defines.h, 137
TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES
tee_api_defines.h, 137
TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES1
tee_api_defines.h, 137
TEE_PANIC_ID_TEE_COPYOPERATION
tee_api_defines.h, 137
TEE_PANIC_ID_TEE_CREATEPERSISTENTOBJECT
tee_api_defines.h, 137
TEE_PANIC_ID_TEE_DERIVEKEY
tee_api_defines.h, 137
TEE_PANIC_ID_TEE_DIGESTDOFINAL
tee_api_defines.h, 137
TEE_PANIC_ID_TEE_DIGESTUPDATE
tee_api_defines.h, 137
TEE_PANIC_ID_TEE_FREE
tee_api_defines.h, 138
TEE_PANIC_ID_TEE_FREEOPERATION
tee_api_defines.h, 138
TEE_PANIC_ID_TEE_FREEPERSISTENTOBJECTENUMERATOR
tee_api_defines.h, 138
TEE_PANIC_ID_TEE_FREEPROPERTYENUMERATOR
tee_api_defines.h, 138
TEE_PANIC_ID_TEE_FREETRANSIENTOBJECT
tee_api_defines.h, 138
TEE_PANIC_ID_TEE_GENERATEKEY
tee_api_defines.h, 138
TEE_PANIC_ID_TEE_GENERATERANDOM
tee_api_defines.h, 138
TEE_PANIC_ID_TEE_GETCANCELLATIONFLAG
tee_api_defines.h, 138
TEE_PANIC_ID_TEE_GETINSTANCEDATA
tee_api_defines.h, 138
TEE_PANIC_ID_TEE_GETNEXTPERSISTENTOBJECT
tee_api_defines.h, 138
TEE_PANIC_ID_TEE_GETNEXTPROPERTY
tee_api_defines.h, 138
TEE_PANIC_ID_TEE_GETOBJECTBUFFERATTRIBUTE
tee_api_defines.h, 139
TEE_PANIC_ID_TEE_GETOBJECTINFO
tee_api_defines.h, 139
TEE_PANIC_ID_TEE_GETOBJECTINFO1
tee_api_defines.h, 139
TEE_PANIC_ID_TEE_GETOBJECTVALUEATTRIBUTE
tee_api_defines.h, 139
TEE_PANIC_ID_TEE_GETOPERATIONINFO
tee_api_defines.h, 139
TEE_PANIC_ID_TEE_GETOPERATIONINFOMULTIPLE
tee_api_defines.h, 139
TEE_PANIC_ID_TEE_GETPROPERTYASBINARYBLOCK
tee_api_defines.h, 139
TEE_PANIC_ID_TEE_GETPROPERTYASBOOL
tee_api_defines.h, 139
TEE_PANIC_ID_TEE_GETPROPERTYASIDENTITY
tee_api_defines.h, 139
TEE_PANIC_ID_TEE_GETPROPERTYASSTRING
tee_api_defines.h, 139
TEE_PANIC_ID_TEE_GETPROPERTYASU32
tee_api_defines.h, 139
TEE_PANIC_ID_TEE_GETPROPERTYASUUID
tee_api_defines.h, 140
TEE_PANIC_ID_TEE_GETPROPERTYNAME
tee_api_defines.h, 140
TEE_PANIC_ID_TEE_GETREETIME
tee_api_defines.h, 140

TEE_PANIC_ID_TEE_GETSYSTEMTIME
tee_api_defines.h, 140
TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME
tee_api_defines.h, 140
TEE_PANIC_ID_TEE_INITREFATTRIBUTE
tee_api_defines.h, 140
TEE_PANIC_ID_TEE_INITVALUEATTRIBUTE
tee_api_defines.h, 140
TEE_PANIC_ID_TEE_INVOKETACOMMAND
tee_api_defines.h, 140
TEE_PANIC_ID_TEE_MACCOMPAREFINAL
tee_api_defines.h, 140
TEE_PANIC_ID_TEE_MACCOMPUTEFINAL
tee_api_defines.h, 140
TEE_PANIC_ID_TEE_MACINIT
tee_api_defines.h, 140
TEE_PANIC_ID_TEE_MACUPDATE
tee_api_defines.h, 141
TEE_PANIC_ID_TEE_MALLOC
tee_api_defines.h, 141
TEE_PANIC_ID_TEE_MASKCANCELLATION
tee_api_defines.h, 141
TEE_PANIC_ID_TEE_MEMCOMPARE
tee_api_defines.h, 141
TEE_PANIC_ID_TEE_MEMFILL
tee_api_defines.h, 141
TEE_PANIC_ID_TEE_MEMMOVE
tee_api_defines.h, 141
TEE_PANIC_ID_TEE_OPENPERSISTENTOBJECT
tee_api_defines.h, 141
TEE_PANIC_ID_TEE_OPENTASESSION
tee_api_defines.h, 141
TEE_PANIC_ID_TEE_PANIC
tee_api_defines.h, 141
TEE_PANIC_ID_TEE_POPULATETRANSIENTOBJECT
tee_api_defines.h, 141
TEE_PANIC_ID_TEE_READOBJECTDATA
tee_api_defines.h, 141
TEE_PANIC_ID_TEE_REALLOC
tee_api_defines.h, 142
TEE_PANIC_ID_TEE_RENAMEPERSISTENTOBJECT
tee_api_defines.h, 142
TEE_PANIC_ID_TEE_RESETOPERATION
tee_api_defines.h, 142
TEE_PANIC_ID_TEE_RESETPERSISTENTOBJECTENUMERATOR
tee_api_defines.h, 142
TEE_PANIC_ID_TEE_RESETPROPERTYENUMERATOR
tee_api_defines.h, 142
TEE_PANIC_ID_TEE_RESETTRANSIENTOBJECT
tee_api_defines.h, 142
TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE
tee_api_defines.h, 142
TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE1
tee_api_defines.h, 142
TEE_PANIC_ID_TEE_SEEKOBJECTDATA
tee_api_defines.h, 142
TEE_PANIC_ID_TEE_SETINSTANCEDATA
tee_api_defines.h, 142
TEE_PANIC_ID_TEE_SETOPERATIONKEY
tee_api_defines.h, 142
TEE_PANIC_ID_TEE_SETOPERATIONKEY2
tee_api_defines.h, 143
TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME
tee_api_defines.h, 143
TEE_PANIC_ID_TEE_STARTPERSISTENTOBJECTENUMERATOR
tee_api_defines.h, 143
TEE_PANIC_ID_TEE_STARTPROPERTYENUMERATOR
tee_api_defines.h, 143
TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA
tee_api_defines.h, 143
TEE_PANIC_ID_TEE_UNMASKCANCELLATION
tee_api_defines.h, 143
TEE_PANIC_ID_TEE_WAIT
tee_api_defines.h, 143
TEE_PANIC_ID_TEE_WRITEOBJECTDATA
tee_api_defines.h, 143
TEE_Param, 57
a, 58
b, 58
buffer, 58
memref, 58
size, 58
value, 58
TEE_PARAM_TYPE0
crt.c, 340
TEE_PARAM_TYPE1
crt.c, 340
Enclave.c, 394
TEE_PARAM_TYPE_GET
tee_api_defines.h, 143
TEE_PARAM_TYPE_MEMREF_INOUT
tee_api_defines.h, 143
TEE_PARAM_TYPE_MEMREF_INPUT
tee_api_defines.h, 144
TEE_PARAM_TYPE_MEMREF_OUTPUT
tee_api_defines.h, 144
TEE_PARAM_TYPE_NONE
tee_api_defines.h, 144
TEE_PARAM_TYPE_SET
tee_api_defines.h, 144
TEE_PARAM_TYPE_VALUE_INOUT
tee_api_defines.h, 144
TEE_PARAM_TYPE_VALUE_INPUT
tee_api_defines.h, 144
TEE_PARAM_TYPE_VALUE_OUTPUT
tee_api_defines.h, 144
TEE_PARAM_TYPES
tee_api_defines.h, 144
tee_printf
config_ref_ta.h, 289
crt.c, 342
Enclave.c, 396
trace.c, 234, 235, 237
tee_profiler.c
__profiler_head, 323, 324, 326
__profiler_unmap_info, 321, 323, 325

profiler_write, 322, 324, 325
tee_profiler.h
__profiler_unmap_info, 327–329
TEE_PROPSET_CURRENT_CLIENT
tee_api_defines.h, 144
TEE_PROPSET_CURRENT_TA
tee_api_defines.h, 144
TEE_PROPSET_TEE_IMPLEMENTATION
tee_api_defines.h, 145
TEE_PropSetHandle
tee_api_types.h, 162
tee_rdtscp
tee_config.h, 317, 319, 320
TEE_ReadObjectData
tee-internal-api.c, 210, 219
tee-ta-internal.h, 105
TEE_Realloc
tee-internal-api.c, 210
TEE_Result
tee_api_types.h, 162
TEE_SE_READER_NAME_MAX
tee_api_types.h, 161
TEE_SEAID, 58
buffer, 59
bufferLen, 59
TEE_SEChannelHandle
tee_api_types.h, 162
TEE_SEReaderHandle
tee_api_types.h, 162
TEE_SEReaderProperties, 59
selectResponseEnable, 59
sePresent, 59
teeOnly, 59
TEE_SEServiceHandle
tee_api_types.h, 162
TEE_SESessionHandle
tee_api_types.h, 162
TEE_Session
tee_api_types.h, 162
TEE_SetOperationKey
tee-internal-api-cryptlib.c, 250
tee-ta-internal.h, 106
TEE_STORAGE_PRIVATE
tee_api_defines.h, 145
TEE_STORAGE_PRIVATE_REE
tee_api_defines_extensions.h, 156
TEE_STORAGE_PRIVATE_RPMB
tee_api_defines_extensions.h, 157
TEE_STORAGE_PRIVATE_SQL_RESERVED
tee_api_defines_extensions.h, 157
TEE_SUCCESS
tee_api_defines.h, 145
tee_ta_api.h
TA_CloseSessionEntryPoint, 185
TA_CreateEntryPoint, 186
TA_DestroyEntryPoint, 186
TA_EXPORT, 185
TA_InvokeCommandEntryPoint, 186
TA_OpenSessionEntryPoint, 186
TEE_TASessionHandle
tee_api_types.h, 162
TEE_Time, 60
millis, 60
seconds, 60
tee_time_tests
bench.c, 253
TEE_TIMEOUT_INFINITE
tee_api_defines.h, 145
TEE_TYPE_AES
tee_api_defines.h, 145
TEE_TYPE_CONCAT_KDF_Z
tee_api_defines_extensions.h, 157
TEE_TYPE_CORRUPTED_OBJECT
tee_api_defines.h, 145
TEE_TYPE_DATA
tee_api_defines.h, 145
TEE_TYPE_DES
tee_api_defines.h, 145
TEE_TYPE_DES3
tee_api_defines.h, 145
TEE_TYPE_DH_KEYPAIR
tee_api_defines.h, 145
TEE_TYPE_DSA_KEYPAIR
tee_api_defines.h, 145
TEE_TYPE_DSA_PUBLIC_KEY
tee_api_defines.h, 146
TEE_TYPE_ECDH_KEYPAIR
tee_api_defines.h, 146
TEE_TYPE_ECDH_PUBLIC_KEY
tee_api_defines.h, 146
TEE_TYPE_ECDSA_KEYPAIR
tee_api_defines.h, 146
TEE_TYPE_ECDSA_PUBLIC_KEY
tee_api_defines.h, 146
TEE_TYPE_GENERIC_SECRET
tee_api_defines.h, 146
TEE_TYPE_HKDF_IKM
tee_api_defines_extensions.h, 157
TEE_TYPE_HMAC_MD5
tee_api_defines.h, 146
TEE_TYPE_HMAC_SHA1
tee_api_defines.h, 146
TEE_TYPE_HMAC_SHA224
tee_api_defines.h, 146
TEE_TYPE_HMAC_SHA256
tee_api_defines.h, 146
TEE_TYPE_HMAC_SHA384
tee_api_defines.h, 146
TEE_TYPE_HMAC_SHA512
tee_api_defines.h, 147
TEE_TYPE_PBKDF2_PASSWORD
tee_api_defines_extensions.h, 157
TEE_TYPE_RSA_KEYPAIR
tee_api_defines.h, 147
TEE_TYPE_RSA_PUBLIC_KEY
tee_api_defines.h, 147

tee_unmap
tee_internal_api_extensions.h, 183
TEE_USAGE_DECRYPT
tee_api_defines.h, 147
TEE_USAGE_DERIVE
tee_api_defines.h, 147
TEE_USAGE_ENCRYPT
tee_api_defines.h, 147
TEE_USAGE_EXTRACTABLE
tee_api_defines.h, 147
TEE_USAGE_MAC
tee_api_defines.h, 147
TEE_USAGE_SIGN
tee_api_defines.h, 147
TEE_USAGE_VERIFY
tee_api_defines.h, 147
tee_user_mem_check_heap
tee_internal_api_extensions.h, 183
TEE_USER_MEM_HINT_NO_FILL_ZERO
tee_internal_api_extensions.h, 182
tee_user_mem_mark_heap
tee_internal_api_extensions.h, 183
TEE_UUID, 60
clockSeqAndNode, 60
timeHiAndVersion, 60
timeLow, 61
timeMid, 61
tee_uuid_from_str
tee_internal_api_extensions.h, 183
TEE_Whence
tee_api_types.h, 163
TEE_WriteObjectData
tee-internal-api.c, 211, 220
tee-ta-internal.h, 107
TEEC_AllocateSharedMemory
tee_client_api.h, 174
teec_stub.c, 229
TEEC_CloseSession
tee_client_api.h, 175
teec_stub.c, 229
TEEC_CONFIG_PAYLOAD_REF_COUNT
tee_client_api.h, 168
TEEC_CONFIG_SHAREDMEM_MAX_SIZE
tee_client_api.h, 168
TEEC_Context, 61
fd, 61
reg_mem, 61
TEEC_ERROR_ACCESS_CONFLICT
tee_client_api.h, 169
TEEC_ERROR_ACCESS_DENIED
tee_client_api.h, 169
TEEC_ERROR_BAD_FORMAT
tee_client_api.h, 169
TEEC_ERROR_BAD_PARAMETERS
tee_client_api.h, 169
TEEC_ERROR_BAD_STATE
tee_client_api.h, 169
TEEC_ERROR_BUSY
tee_client_api.h, 169
TEEC_ERROR_CANCEL
tee_client_api.h, 169
TEEC_ERROR_COMMUNICATION
tee_client_api.h, 169
TEEC_ERROR_EXCESS_DATA
tee_client_api.h, 169
TEEC_ERROR_EXTERNAL_CANCEL
tee_client_api.h, 169
TEEC_ERROR_GENERIC
tee_client_api.h, 169
TEEC_ERROR_ITEM_NOT_FOUND
tee_client_api.h, 170
TEEC_ERROR_NO_DATA
tee_client_api.h, 170
TEEC_ERROR_NOT_IMPLEMENTED
tee_client_api.h, 170
TEEC_ERROR_NOT_SUPPORTED
tee_client_api.h, 170
TEEC_ERROR_OUT_OF_MEMORY
tee_client_api.h, 170
TEEC_ERROR_SECURITY
tee_client_api.h, 170
TEEC_ERROR_SHORT_BUFFER
tee_client_api.h, 170
TEEC_ERROR_TARGET_DEAD
tee_client_api.h, 170
TEEC_FinalizeContext
tee_client_api.h, 175
teec_stub.c, 229
TEEC_InitializeContext
tee_client_api.h, 176
teec_stub.c, 230
TEEC_InvokeCommand
tee_client_api.h, 176
TEEC_LOGIN_APPLICATION
tee_client_api.h, 170
TEEC_LOGIN_GROUP
tee_client_api.h, 170
TEEC_LOGIN_GROUP_APPLICATION
tee_client_api.h, 170
TEEC_LOGIN_PUBLIC
tee_client_api.h, 171
TEEC_LOGIN_USER
tee_client_api.h, 171
TEEC_LOGIN_USER_APPLICATION
tee_client_api.h, 171
TEEC_MEM_INPUT
tee_client_api.h, 171
TEEC_MEM_OUTPUT
tee_client_api.h, 171
TEEC_MEMREF_PARTIAL_INOUT
tee_client_api.h, 171
TEEC_MEMREF_PARTIAL_INPUT
tee_client_api.h, 171
TEEC_MEMREF_PARTIAL_OUTPUT
tee_client_api.h, 171
TEEC_MEMREF_TEMP_INOUT

tee_client_api.h, 171
TEEC_MEMREF_TEMP_INPUT
tee_client_api.h, 172
TEEC_MEMREF_TEMP_OUTPUT
tee_client_api.h, 172
TEEC_MEMREF_WHOLE
tee_client_api.h, 172
TEEC_NONE
tee_client_api.h, 172
TEEC_OpenSession
tee_client_api.h, 177
teec_stub.c, 230
TEEC_Operation, 62
params, 62
paramTypes, 62
session, 63
started, 63
TEEC_ORIGIN_API
tee_client_api.h, 172
TEEC_ORIGIN_COMMS
tee_client_api.h, 173
TEEC_ORIGIN_TEE
tee_client_api.h, 173
TEEC_ORIGIN_TRUSTED_APP
tee_client_api.h, 173
TEEC_PARAM_TYPE0
main.c, 399
TEEC_PARAM_TYPE1
main.c, 399, 400
TEEC_PARAM_TYPE_GET
tee_client_api.h, 173
TEEC_PARAM_TYPES
tee_client_api.h, 173
TEEC_Parameter, 63
memref, 64
tmpref, 64
value, 64
TEEC_RegisteredMemoryReference, 64
offset, 65
parent, 65
size, 65
TEEC_RegisterSharedMemory
tee_client_api.h, 177
teec_stub.c, 231
TEEC_ReleaseSharedMemory
tee_client_api.h, 178
teec_stub.c, 231
TEEC_RequestCancellation
tee_client_api.h, 178
teec_stub.c, 232
TEEC_Result
tee_client_api.h, 174
TEEC_Session, 66
ctx, 66
session_id, 66
TEEC_SharedMemory, 66
alloced_size, 67
buffer, 67
buffer_allocated, 67
flags, 67
id, 67
registered_fd, 68
shadow_buffer, 68
size, 68
teec_stub.c
TEEC_AllocateSharedMemory, 229
TEEC_CloseSession, 229
TEEC_FinalizeContext, 229
TEEC_InitializeContext, 230
TEEC_OpenSession, 230
TEEC_RegisterSharedMemory, 231
TEEC_ReleaseSharedMemory, 231
TEEC_RequestCancellation, 232
TEEC_SUCCESS
tee_client_api.h, 174
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, 174
TEEC_VALUE_INPUT
tee_client_api.h, 174
TEEC_VALUE_OUTPUT
tee_client_api.h, 174
teeOnly
TEE_SEReaderProperties, 59
TEEP_AGENT_TA_DELETE
invoke_command.c, 294
TEEP_AGENT_TA_EXIT
invoke_command.c, 294
TEEP_AGENT_TA_INSTALL
invoke_command.c, 294
TEEP_AGENT_TA_LOAD
invoke_command.c, 294
TEEP_AGENT_TA_NONE
invoke_command.c, 294
test_dev_key.h
_sanctum_dev_public_key, 189
_sanctum_dev_public_key_len, 189
_sanctum_dev_secret_key, 190
_sanctum_dev_secret_key_len, 190
test_printf
tee_def.h, 265–267
time.c
gp_ree_time_test, 300
gp_trusted_time_test, 300
time_diff
bench.c, 253

time_test
bench.c, 254
time_test.c
ree_time_test, 270
system_time_test, 270
time_to_millis
bench.c, 254
timeHiAndVersion
TEE_UUID, 60
TEEC_UUID, 69
timeLow
TEE_UUID, 61
TEEC_UUID, 69
timeMid
TEE_UUID, 61
TEEC_UUID, 69
tmpref
TEEC_Parameter, 64
tools.c
_strlen, 315
printf, 315
profiler_write, 311–314
putchar, 316
puts, 316
tools.h
printf, 348
putchar, 349
puts, 349
trace.c
_strlen, 234, 237
tee_printf, 234, 235, 237
trace_printf, 232
trace_vprintf, 233
trace.h
dhex_dump, 194
DHEXDUMP, 192
DMSG, 192
DMSG_RAW, 192
DPRINT_STACK, 192
EMSG, 192
EMSG_RAW, 192
EPRINT_STACK, 192
FMSG, 192
FMSG_RAW, 193
FPRINT_STACK, 193
IMSG, 193
IMSG_RAW, 193
INMSG, 193
IPRINT_STACK, 193
MAX_FUNC_PRINT_SIZE, 193
MAX_PRINT_SIZE, 193
MSG, 193
MSG_RAW, 193
OUTMSG, 194
OUTRMSG, 194
SMSG, 194
trace_ext_get_thread_id, 195
trace_ext_prefix, 195
trace_ext_puts, 195
trace_get_level, 195
TRACE_LEVEL, 194
trace_level, 195
trace_printf, 195
trace_printf_helper, 194
trace_printf_helper_raw, 194
trace_set_level, 195
TRACE_DEBUG
trace_levels.h, 199
TRACE_ERROR
trace_levels.h, 199
trace_ext_get_thread_id
trace.h, 195
trace_ext_prefix
trace.h, 195
user_ta_header.c, 404, 407
trace_ext_puts
trace.h, 195
TRACE_FLOW
trace_levels.h, 199
trace_get_level
trace.h, 195
TRACE_INFO
trace_levels.h, 199
TRACE_LEVEL
trace.h, 194
trace_level
trace.h, 195
user_ta_header.c, 404, 407
trace_levels.h
TRACE_DEBUG, 199
TRACE_ERROR, 199
TRACE_FLOW, 199
TRACE_INFO, 199
TRACE_MAX, 199
TRACE_MIN, 199
TRACE_PRINTF_LEVEL, 199
TRACE_MAX
trace_levels.h, 199
TRACE_MIN
trace_levels.h, 199
trace_printf
trace.c, 232
trace.h, 195
trace_printf_helper
trace.h, 194
trace_printf_helper_raw
trace.h, 194
TRACE_PRINTF_LEVEL
trace_levels.h, 199
trace_set_level
trace.h, 195
trace_vprintf
trace.c, 233
TRUE
App.h, 414, 416
type

__TEE_ObjectHandle, 37
nm_info, 44
types.h
global_eid, 434, 437
sgx_errlist, 434, 437
sgx_errlist_t, 434, 437

USED
profiler_attrs.h, 334
user_ta_header.c
_C_FUNCTION, 402, 405
__section, 402, 405
__ta_entry, 402, 406
__utee_entry, 403, 406
TA_DESCRIPTION, 402, 405
TA_FRAMEWORK_STACK_SIZE, 402, 405
ta_heap, 403, 406
ta_heap_size, 404, 406
ta_num_props, 404, 407
ta_props, 404, 407
TA_VERSION, 402, 405
tahead_get_trace_level, 403, 406
trace_ext_prefix, 404, 407
trace_level, 404, 407
user_ta_header_defines.h
TA_CURRENT_TA_EXT_PROPERTIES, 408, 411
TA_DATA_SIZE, 408, 411
TA_DESCRIPTION, 408, 411
TA_FLAGS, 408, 411
TA_STACK_SIZE, 409, 411
TA_UUID, 409, 411
TA_VERSION, 409, 411
user_types.h
array_t, 272
buffer_t, 272
LOOPS_PER_THREAD, 271
uuid
TEE_Identity, 52

value
TEE_Attribute, 51
TEE_Param, 58
TEEC_Parameter, 64
vsnprintf
vsnprintf.c, 365
vsnprintf.c
_atoi, 357
_ftoa, 357
_is_digit, 358
_ntoa_format, 358
_ntoa_long, 359
_ntoa_long_long, 360
_out_buffer, 360
_out_char, 361
_out_fct, 361
_out_null, 362
_putchar, 355
_strlen, 362
_vsnprintf, 362
fctprintf, 363
FLAGS_CHAR, 355
FLAGS_HASH, 355
FLAGS_LEFT, 355
FLAGS_LONG, 355
FLAGS_LONG_LONG, 355
FLAGS_PLUS, 356
FLAGS_PRECISION, 356
FLAGS_SHORT, 356
FLAGS_SPACE, 356
FLAGS_UPPERCASE, 356
FLAGS_ZEROPAD, 356
out_fct_type, 357
PRINTF_FTOA_BUFFER_SIZE, 356
PRINTF_NTOA_BUFFER_SIZE, 356
PRINTF_SUPPORT_FLOAT, 356
PRINTF_SUPPORT_LONG_LONG, 356
PRINTF_SUPPORT_PTRDIFF_T, 356
putchar, 363
snprintf, 363
sprintf, 365
vsnprintf, 365

WOLFCRYPT
tee_api_tee_types.h, 222, 226
wolfSSL_Free
tee-internal-api-cryptlib.c, 251
wolfSSL_Malloc
tee-internal-api-cryptlib.c, 251