



Samsung Internal API reference

Date	2018.10.28	Version	2.0
Organizator	Mobile Security Technologies Group, IT & Mobile Communications		

Contents

1 Overview	33
1.1 Objective	33
1.2 Scope	33
1.3 Abbreviations and Acronyms	33
1.4 API type	33
1.4.1 TA properties	33
1.4.2 Samsung Internal API	36
1.4.3 Thread support library API	36
1.4.4 Math library API	36
1.4.5 Auxiliary API	36
2 Deprecated List	37
3 Data Structure Index	38
3.1 Data Structures	38
4 Module Documentation	39
4.1 Samsung Internal API	39
4.1.1 Detailed Description	40
4.1.2 Data Structure Documentation	40
4.1.2.1 struct rot_t	40
4.1.2.2 struct wrapped_wkth_rek_t	41
4.1.3 Macro Definition Documentation	41
4.1.3.1 KM_KW_MAX_AAD_LEN	41
4.1.3.2 KM_KW_MAX_INPUT_LEN	41
4.1.3.3 KM_KW_MAX_IV_LEN	42
4.1.3.4 KM_KW_MAX_KEY_LEN	42
4.1.3.5 KM_KW_MAX_SALT_LEN	42
4.1.3.6 KM_KW_MAX_TAG_LEN	42

4.1.3.7	SO_AC_LEN	42
4.1.3.8	SO_AUTH_ID_LEN	42
4.1.3.9	SO_HEADER_SIZE	42
4.1.3.10	SO_IV_LEN	43
4.1.3.11	SO_MAGIC_NUMBER_LEN	43
4.1.3.12	SO_OUT_BUF_SIZE	43
4.1.3.13	SO_TA_ID_LEN	43
4.1.3.14	SO_TAG_LEN	43
4.1.4	Function Documentation	43
4.1.4.1	errno_to_tee_error(int error_code)	43
4.1.4.2	TEES_CheckSecureObjectCreator(const unsigned char *in, uint32_t in_len, SO_AccessControllInfoType *ac)	44
4.1.4.3	TEES_DeriveKeyKDF(const void *label, uint32_t labelLen, const void *context, uint32_t contextLen, uint32_t outputKeyLen, TEE_ObjectHandle object)	44
4.1.4.4	TEES_DeriveKeySetKDF(const void *label, uint32_t labelLen, const void *context, uint32_t contextLen, uint32_t outputKeyLen, TEE_ObjectHandle object)	45
4.1.4.5	TEES_EnterCritical(void)	46
4.1.4.6	TEES_ExitCritical(void)	46
4.1.4.7	TEES_GetRoT(ROOT_OF_TRUST *rot)	46
4.1.4.8	TEES_LockHWCryptoBuf(void)	47
4.1.4.9	TEES_SECCAM_GetStatus(unsigned int *data)	47
4.1.4.10	TEES_UnlockHWCryptoBuf(void)	48
4.1.4.11	TEES_UnwrapSecureObject(const unsigned char *in, uint32_t in_len, unsigned char *out, uint32_t *out_len)	48
4.1.4.12	TEES_WrappedWithREK(WRAP_REK *data)	49
4.1.4.13	TEES_WrapSecureObject(const unsigned char *in, uint32_t in_len, unsigned char *out, uint32_t *out_len, SO_AccessControllInfoType *ac)	50
4.2	Loadable driver API	51
4.2.1	Detailed Description	52
4.2.2	Data Structure Documentation	52
4.2.2.1	struct fops	52

4.2.2.2	struct usr_drv_info	53
4.2.3	Typedef Documentation	53
4.2.3.1	TEES_DriverDestructor_t	53
4.2.3.2	TEES_InterruptHandle	53
4.2.4	Function Documentation	54
4.2.4.1	TEES_AcquireUserBuffer(struct drv_info *filp, uint64_t addr, const size_t size, int prot)	54
4.2.4.2	TEES_AllocateInterrupt(int nr, intruhandler handler, TEES_InterruptHandle *handle)	54
4.2.4.3	TEES_CompleteInterrupt(TEES_InterruptHandle handle)	55
4.2.4.4	TEES_CompleteRequest(struct drv_info *filp, long ret)	56
4.2.4.5	TEES_DcacheClean(const void *addr, size_t nbytes)	57
4.2.4.6	TEES_DcacheFlush(const void *addr, size_t nbytes)	57
4.2.4.7	TEES_FiniDriver(struct usr_drv_info *info)	58
4.2.4.8	TEES_GenerateInterrupt(TEES_InterruptHandle handle)	58
4.2.4.9	TEES_InitDriver(char *name, struct fops *fops, unsigned int drvid, struct usr_drv_info **info)	59
4.2.4.10	TEES_RegisterDriver(char *name, struct fops *fops, unsigned int drvid, struct usr_drv_info **info) _deprecated_	60
4.2.4.11	TEES_RegisterDriverDestructor(TEES_DriverDestructor_t destr)	61
4.2.4.12	TEES_ReleaseDriver(struct usr_drv_info **info) _deprecated_	61
4.2.4.13	TEES_ReleaseInterrupt(TEES_InterruptHandle handle)	62
4.2.4.14	TEES_ReleaseUserBuffer(const void *addr, const size_t size)	62
4.2.4.15	TEES_WaitForInterrupt(TEES_InterruptHandle handle, uint32_t timeout)	63
4.3	Custom handler API	64
4.4	Contiguous memory API	65
4.4.1	Detailed Description	65
4.4.2	Typedef Documentation	65
4.4.2.1	TEES_ContiguousMemoryHandle	65
4.4.3	Enumeration Type Documentation	66
4.4.3.1	TEES_ContiguousAllocateFlags	66
4.4.3.2	TEES_ContiguousMapFlags	66

4.4.4	Function Documentation	66
4.4.4.1	TEES_AllocateContiguousMemory(const char *region, size_t size, size_t align, uint32_t flags, TEES_ContiguousMemoryHandle *memory)	66
4.4.4.2	TEES_MapContiguousMemory(TEES_ContiguousMemoryHandle memory, void **addr, uint32_t flags)	67
4.4.4.3	TEES_ReleaseContiguousMemory(TEES_ContiguousMemoryHandle memory)	68
4.4.4.4	TEES_UnmapContiguousMemory(void *addr)	69
4.5	SPI API	70
4.5.1	Detailed Description	70
4.5.2	Data Structure Documentation	71
4.5.2.1	struct TEES_SPIConfig	71
4.5.2.2	struct TEES_SPITransfer	71
4.5.3	Typedef Documentation	71
4.5.3.1	TEES_SPIHandler	71
4.5.4	Function Documentation	71
4.5.4.1	TEES_SPIClockDisable(TEES_SPIHandler handler)	71
4.5.4.2	TEES_SPIClockEnable(TEES_SPIHandler handler)	72
4.5.4.3	TEES_SPIDMAInit(uintptr_t address)	72
4.5.4.4	TEES_SPIExit(TEES_SPIHandler handler)	72
4.5.4.5	TEES_SPIInit(uint32_t port, const TEES_SPIConfig *cfg, TEES_SPIHandler *handler)	73
4.5.4.6	TEES_SPIRead(TEES_SPIHandler handler, TEES_SPITransfer *rx)	74
4.5.4.7	TEES_SPISetBitsPerWord(TEES_SPIHandler handler, uint8_t bitsPerWord)	75
4.5.4.8	TEES_SPISetClockSpeed(TEES_SPIHandler handler, uint32_t speedHz)	75
4.5.4.9	TEES_SPISetConfig(TEES_SPIHandler handler, const TEES_SPIConfig *cfg)	76
4.5.4.10	TEES_SPISetCPHA(TEES_SPIHandler handler, uint8_t cpha)	76
4.5.4.11	TEES_SPISetCPOL(TEES_SPIHandler handler, uint8_t cpol)	77
4.5.4.12	TEES_SPISetDMAMode(TEES_SPIHandler handler, bool isEnabled)	77
4.5.4.13	TEES_SPIWrite(TEES_SPIHandler handler, TEES_SPITransfer *tx)	78
4.5.4.14	TEES_SPIWriteRead(TEES_SPIHandler handler, TEES_SPITransfer *tx, TEES_SPITransfer *rx)	78
4.6	I2C API	80

4.6.1	Detailed Description	80
4.6.2	Data Structure Documentation	80
4.6.2.1	struct TEES_I2CTransfer	80
4.6.3	Typedef Documentation	81
4.6.3.1	TEES_I2CHandler	81
4.6.4	Function Documentation	81
4.6.4.1	TEES_I2CExit(TEES_I2CHandler handler)	81
4.6.4.2	TEES_I2CInit(const char *name, uint32_t transac_len, TEES_I2CHandler *handler)	81
4.6.4.3	TEES_I2CRead(TEES_I2CHandler handler, uint8_t chip, TEES_I2CTransfer *rx)	82
4.6.4.4	TEES_I2CWrite(TEES_I2CHandler handler, uint8_t chip, TEES_I2CTransfer *tx)	82
4.6.4.5	TEES_I2CWriteRead(TEES_I2CHandler handler, uint8_t chip, TEES_I2CTransfer *tx, TEES_I2CTransfer *rx)	83
4.7	Trusted user interface	84
4.7.1	Detailed Description	86
4.7.2	Data Structure Documentation	86
4.7.2.1	struct TEES_TUIScreenInfo	86
4.7.2.2	struct TEES_TUITouchInfo	86
4.7.2.3	struct TEES_TUIImage	86
4.7.2.4	struct TEE_TUIImage	87
4.7.2.5	union TEE_TUIImage.__unnamed__	87
4.7.2.6	struct TEE_TUIImage.__unnamed__.ref	87
4.7.2.7	struct TEE_TUIImage.__unnamed__.object	87
4.7.2.8	struct TEE_TUIScreenLabel	88
4.7.2.9	struct TEE_TUIButton	88
4.7.2.10	struct TEE_TUIScreenConfiguration	88
4.7.2.11	struct TEE_TUIScreenButtonInfo	89
4.7.2.12	struct TEE_TUIScreenInfo	89
4.7.2.13	struct TEE_TUIEntryField	89
4.7.3	Macro Definition Documentation	90

4.7.3.1	MAX_FONT_SIZE	90
4.7.3.2	MIN_FONT_SIZE	90
4.7.3.3	TEE_TUI_NUMBER_BUTTON_TYPES	90
4.7.4	Enumeration Type Documentation	90
4.7.4.1	TEE_TUIButtonType	90
4.7.4.2	TEE_TUIEntryFieldMode	91
4.7.4.3	TEE_TUIEntryFieldType	91
4.7.4.4	TEE_TUIImageSource	91
4.7.4.5	TEE_TUIScreenOrientation	91
4.7.4.6	TEES_bool	92
4.7.4.7	TEES_Result	92
4.7.4.8	TEES_TUITouchTypes	92
4.7.5	Function Documentation	93
4.7.5.1	TEE_TUICheckTextFormat(char *text, uint32_t *width, uint32_t *height, uint32_t *lastIndex)	93
4.7.5.2	TEE_TUICloseSession(void)	93
4.7.5.3	TEE_TUIDisplayScreen(TEE_TUIScreenConfiguration *screenConfiguration, bool closeTUISession, TEE_TUIEntryField *entryFields, uint32_t entryFieldCount, TEE_TUIButtonType *selectedButton)	94
4.7.5.4	TEE_TUIGetScreenInfo(TEE_TUIScreenOrientation screenOrientation, uint32_t nbEntryFields, TEE_TUIScreenInfo *screenInfo)	95
4.7.5.5	TEE_TUIInitSession(void)	96
4.7.5.6	TEES_TUICheckTextFormat(char *inputText, uint32_t textSize, uint32_t *width, uint32_t *height, uint32_t *lastIndex)	96
4.7.5.7	TEES_TUICloseSession(void)	97
4.7.5.8	TEES_TUIDrawBuffer(uint32_t *buffer, uint32_t posX, uint32_t posY, uint32_t W, uint32_t H)	97
4.7.5.9	TEES_TUIDrawImage(TEES_TUIImage *image, uint32_t posX, uint32_t posY)	98
4.7.5.10	TEES_TUIGetBuffer(uint32_t *buffer, uint32_t posX, uint32_t posY, uint32_t W, uint32_t H)	99
4.7.5.11	TEES_TUIGetScreenInfo(TEES_TUIScreenInfo *screenInfo)	99
4.7.5.12	TEES_TUIGetTouchEvent(TEES_TUITouchInfo *touchInfo, uint32_t timeout)	100

4.7.5.13	TEES_TUIOpenSession(void)	100
4.7.5.14	TEES_TUIPrintString(char *inputText, uint32_t textSize, uint32_t posX, uint32_t posY, uint8_t redColorValue, uint8_t greenColorValue, uint8_t blueColorValue, bool rotation90)	101
4.7.5.15	TEES_TUIRefreshScreen(void)	101
4.8	Integrity Report System API	103
4.8.1	Detailed Description	104
4.8.2	Macro Definition Documentation	104
4.8.2.1	IRS_DENY_DELETE_FROM_SMC_TZ	104
4.8.2.2	IRS_DENY_READ_FROM_SMC_TZ	104
4.8.2.3	IRS_DENY_WRITE_FROM_SMC_TZ	104
4.8.2.4	IRS_FAIL_TZ	104
4.8.2.5	IRS_INCORRECT_CHECKSUM_TZ	104
4.8.2.6	IRS_INCORRECT_FLAG_TYPE_TZ	104
4.8.2.7	IRS_INCORRECT_RT_ID_TZ	104
4.8.2.8	IRS_MAX_VAL_COUNTER_RT_TZ	105
4.8.2.9	IRS_MAX_VAL_COUNTER_TZ	105
4.8.2.10	IRS_RT_FLAGS_EMPTY_TZ	105
4.8.2.11	IRS_RT_FLAGS_FULL_TZ	105
4.8.2.12	IRS_SUCCESS_TZ	105
4.8.2.13	IRS_UNKNOWN_ERROR_TZ	105
4.8.2.14	IRS_UNKNOWN_ID_TZ	105
4.8.2.15	IRS_UNKNOWN_INT_CMD_TZ	106
4.8.3	Enumeration Type Documentation	106
4.8.3.1	IRS_INTERNAL_CMD	106
4.8.3.2	IRS_PARAM	106
4.8.4	Function Documentation	107
4.8.4.1	TEES_AddIrsFlag(unsigned int *flag_id, unsigned int param)	107
4.8.4.2	TEES_DelIrsFlag(unsigned int *flag_id)	107
4.8.4.3	TEES_GetIrsFlagValue(unsigned int *flag_id, unsigned int *value)	108
4.8.4.4	TEES_IncIrsFlag(unsigned int *flag_id)	108

4.8.4.5	TEES_SetIrsFlag(unsigned int *flag_id)	108
4.8.4.6	TEES_SetIrsFlagValue(unsigned int *flag_id, unsigned int value)	109
4.9	Miscellaneous extensions	110
4.9.1	Detailed Description	110
4.9.2	Function Documentation	110
4.9.2.1	TEES_DuplWshmem(void *address, uint32_t size)	110
4.9.2.2	TEES_GetTaskDataSize(size_t *data_size)	111
4.9.2.3	TEES_IsPhysMemoryProtected(uint32_t flags, uint64_t base, size_t size)	111
4.9.2.4	TEES_IsREESharedMemory(uint32_t accessFlags, const void *buffer, size_t size)	112
4.10	RPMB API	113
4.10.1	Detailed Description	113
4.10.2	Function Documentation	113
4.10.2.1	TEES_RPMBCheckEnable(void)	113
4.10.2.2	TEES_RPMBRead(uint32_t partition, uint32_t offset, uint8_t *data, uint32_t data_size, uint8_t type_flag)	114
4.10.2.3	TEES_RPMBWrite(uint32_t partition, uint32_t offset, uint8_t *data, uint32_t data_size, uint8_t type_flag)	114
4.11	Thread support library API	115
4.11.1	Detailed Description	117
4.11.2	Data Structure Documentation	117
4.11.2.1	struct __pthread_once_t	117
4.11.2.2	struct __pthread_attr_t	117
4.11.2.3	struct __pthread_mutex_t	117
4.11.2.4	struct __pthread_cond_t	118
4.11.2.5	struct __pthread_condattr_t	118
4.11.3	Macro Definition Documentation	118
4.11.3.1	MUTEX_STATE_LOCKED	118
4.11.3.2	MUTEX_STATE_UNLOCKED	118
4.11.3.3	PTHREAD_ERRORCHECK_MUTEX_INITIALIZER_NP	118
4.11.3.4	PTHREAD_MUTEX_INITIALIZER	118
4.11.3.5	PTHREAD_ONCE_INIT	118

4.11.3.6	PTHREAD_RECURSIVE_MUTEX_INITIALIZER_NP	119
4.11.3.7	pthread_sigmask	119
4.11.3.8	PTHREAD_STACK_MIN	119
4.11.4	Typedef Documentation	119
4.11.4.1	pthread_attr_t	119
4.11.4.2	pthread_cond_t	120
4.11.4.3	pthread_condattr_t	120
4.11.4.4	pthread_impl_t	120
4.11.4.5	pthread_key_t	120
4.11.4.6	pthread_mutex_t	120
4.11.4.7	pthread_mutexattr_t	120
4.11.4.8	pthread_once_t	120
4.11.4.9	pthread_t	121
4.11.5	Enumeration Type Documentation	121
4.11.5.1	anonymous enum	121
4.11.6	Function Documentation	121
4.11.6.1	pthread_attr_destroy(pthread_attr_t *attr)	121
4.11.6.2	pthread_attr_init(pthread_attr_t *attr)	122
4.11.6.3	pthread_attr_setstacksize(pthread_attr_t *attr, size_t size)	122
4.11.6.4	pthread_cond_broadcast(pthread_cond_t *cond)	122
4.11.6.5	pthread_cond_destroy(pthread_cond_t *cond)	123
4.11.6.6	pthread_cond_init(pthread_cond_t *cond, const pthread_condattr_t *attr)	124
4.11.6.7	pthread_cond_signal(pthread_cond_t *cond)	125
4.11.6.8	pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex)	125
4.11.6.9	pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(start_routine)(void *), void *arg)	126
4.11.6.10	pthread_getspecific(pthread_key_t key)	127
4.11.6.11	pthread_join(pthread_t thread, void **retval)	128
4.11.6.12	pthread_key_create(pthread_key_t *key, void(*destructor)(void *))	128
4.11.6.13	pthread_key_delete(pthread_key_t key)	129
4.11.6.14	pthread_kill(pthread_t thread, int sig)	130

4.11.6.15 pthread_mutex_destroy(pthread_mutex_t *mutex)	131
4.11.6.16 pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr)	131
4.11.6.17 pthread_mutex_lock(pthread_mutex_t *mutex)	132
4.11.6.18 pthread_mutex_trylock(pthread_mutex_t *mutex)	133
4.11.6.19 pthread_mutex_unlock(pthread_mutex_t *mutex)	133
4.11.6.20 pthread_mutexattr_destroy(pthread_mutexattr_t *attr)	134
4.11.6.21 pthread_mutexattr_gettype(const pthread_mutexattr_t *attr, int *type)	134
4.11.6.22 pthread_mutexattr_init(pthread_mutexattr_t *attr)	135
4.11.6.23 pthread_mutexattr_settype(pthread_mutexattr_t *attr, int type)	136
4.11.6.24 pthread_once(pthread_once_t *once_control, void(*init_routine)(void))	136
4.11.6.25 pthread_self(void)	137
4.11.6.26 pthread_setspecific(pthread_key_t key, const void *value)	138
4.12 Math library API	139
4.12.1 Detailed Description	140
4.12.2 Macro Definition Documentation	140
4.12.2.1 isnan	140
4.12.2.2 M_E	141
4.12.2.3 M_PI	141
4.12.2.4 M_PI_2	141
4.12.2.5 M_PI_4	141
4.12.3 Function Documentation	141
4.12.3.1 atan(double x)	141
4.12.3.2 atan2(double y, double x)	142
4.12.3.3 atan2f(float y, float x)	142
4.12.3.4 atanf(float x)	142
4.12.3.5 ceil(double x)	143
4.12.3.6 ceilf(float x)	143
4.12.3.7 copysign(double x, double y)	143
4.12.3.8 copysignf(float x, float y)	144
4.12.3.9 cos(double x)	144

4.12.3.10 cosf(float x)	144
4.12.3.11 exp(double x)	145
4.12.3.12 expf(float x)	145
4.12.3.13 fabs(double x)	145
4.12.3.14 fabsf(float x)	146
4.12.3.15 floor(double x)	146
4.12.3.16 floorf(float x)	146
4.12.3.17 fmax(double x, double y)	147
4.12.3.18 fmaxf(float x, float y)	147
4.12.3.19 fmin(double x, double y)	147
4.12.3.20 fminf(float x, float y)	148
4.12.3.21 hypot(double x, double y)	148
4.12.3.22 hypotf(float x, float y)	148
4.12.3.23 log(double x)	149
4.12.3.24 logb(double x)	149
4.12.3.25 logbf(float x)	149
4.12.3.26 logf(float x)	150
4.12.3.27 modf(double x, double *iptr)	150
4.12.3.28 modff(float x, float *iptr)	150
4.12.3.29 pow(double base, double exp)	151
4.12.3.30 powf(float base, float exp)	151
4.12.3.31 round(double x)	151
4.12.3.32 roundf(float x)	152
4.12.3.33 scalbn(double x, int exp)	152
4.12.3.34 scalbnf(float x, int exp)	152
4.12.3.35 sin(double x)	153
4.12.3.36 sinf(float x)	153
4.12.3.37 sqrt(double x)	153
4.12.3.38 sqrtf(float x)	154
4.13 Message queue library API	155

4.13.1 Detailed Description	155
4.13.2 Macro Definition Documentation	155
4.13.2.1 MQ_MAX_NAME	155
4.13.3 Typedef Documentation	155
4.13.3.1 mqd_t	155
4.13.4 Function Documentation	156
4.13.4.1 mq_close(mqd_t fd)	156
4.13.4.2 mq_open(const char *pathname, int flags,...)	156
4.13.4.3 mq_receive(mqd_t fd, char *msg_ptr, size_t msg_len, unsigned *msg_prio)	156
4.13.4.4 mq_send(mqd_t fd, const char *msg_ptr, size_t msg_len, unsigned msg_prio)	157
4.13.4.5 mq_unlink(const char *pathname)	157
4.14 Socket library API	158
4.14.1 Detailed Description	158
4.14.2 Function Documentation	158
4.14.2.1 accept(int sockfd, struct sockaddr *addr, int *addrlen)	158
4.14.2.2 bind(int sockfd, const struct sockaddr *addr, int addrlen)	159
4.14.2.3 connect(int sockfd, const struct sockaddr *addr, int addrlen)	159
4.14.2.4 getsockopt(int sockfd, int level, int optname, void *optval, int *optlen)	159
4.14.2.5 listen(int sockfd, int backlog)	160
4.14.2.6 recv(int sockfd, void *buf, size_t len, int flags)	160
4.14.2.7 recvmsg(int sockfd, struct msghdr *msg, int flags)	160
4.14.2.8 send(int sockfd, const void *buf, size_t len, int flags)	161
4.14.2.9 sendmsg(int sockfd, struct msghdr *msg, int flags)	161
4.14.2.10 setsockopt(int sockfd, int level, int optname, void *optval, int optlen)	161
4.14.2.11 socket(int socket_family, int socket_type, int protocol)	162
4.14.2.12 socketpair(int socket_family, int socket_type, int protocol, int sv[2])	162
4.15 Auxiliary API	163
4.15.1 Detailed Description	173
4.15.2 Data Structure Documentation	173
4.15.2.1 struct cpu_set_t	173

4.15.2.2 struct tm	173
4.15.2.3 struct __uuid_t	174
4.15.3 Macro Definition Documentation	174
4.15.3.1 __CPUMASK	174
4.15.3.2 _POSIX_THREAD_KEYS_MAX	174
4.15.3.3 assert	174
4.15.3.4 AUTO_BUFFER_SIZE	175
4.15.3.5 BIT_PER_CPU	175
4.15.3.6 BITMAP_ELT	175
4.15.3.7 BITS_TO_CPU_MASK	175
4.15.3.8 CHAR_BIT	175
4.15.3.9 CPU_CLR	175
4.15.3.10 CPU_ISSET	176
4.15.3.11 CPU_SET	176
4.15.3.12 CPU_ZERO	176
4.15.3.13 DECLARE_BITMAP	176
4.15.3.14 E2BIG	177
4.15.3.15 EACCES	177
4.15.3.16 EADDRINUSE	177
4.15.3.17 EADDRNOTAVAIL	177
4.15.3.18 EADV	177
4.15.3.19 EAFNOSUPPORT	177
4.15.3.20EAGAIN	177
4.15.3.21 EALREADY	178
4.15.3.22EBADE	178
4.15.3.23EBADF	178
4.15.3.24EBADFD	178
4.15.3.25EBADMSG	178
4.15.3.26EBADR	178
4.15.3.27EBADRQC	178

4.15.3.28EBADSLT	179
4.15.3.29EBFONT	179
4.15.3.30EBUSY	179
4.15.3.31 ECANCELED	179
4.15.3.32ECHILD	179
4.15.3.33ECHRNG	179
4.15.3.34ECOMM	179
4.15.3.35ECONNABORTED	180
4.15.3.36ECONNREFUSED	180
4.15.3.37ECONNRESET	180
4.15.3.38EDEADLK	180
4.15.3.39EDEADLOCK	180
4.15.3.40EDESTADDRREQ	180
4.15.3.41 EDOM	180
4.15.3.42EDOTDOT	181
4.15.3.43EDQUOT	181
4.15.3.44EEXIST	181
4.15.3.45EFAULT	181
4.15.3.46EFBIG	181
4.15.3.47EHOSTDOWN	181
4.15.3.48EHOSTUNREACH	181
4.15.3.49EIDRM	182
4.15.3.50EILSEQ	182
4.15.3.51 EINPROGRESS	182
4.15.3.52EINTR	182
4.15.3.53EINVAL	182
4.15.3.54EIO	182
4.15.3.55EISCONN	182
4.15.3.56EISDIR	183
4.15.3.57EISNAM	183

4.15.3.58EL2HLT	183
4.15.3.59EL2NSYNC	183
4.15.3.60EL3HLT	183
4.15.3.61 EL3RST	183
4.15.3.62ELIBACC	183
4.15.3.63ELIBBAD	184
4.15.3.64ELIBEXEC	184
4.15.3.65ELIBMAX	184
4.15.3.66ELIBSCN	184
4.15.3.67ELNRNG	184
4.15.3.68ELOOP	184
4.15.3.69EMFILE	184
4.15.3.70EMLINK	185
4.15.3.71 EMSGSIZE	185
4.15.3.72EMULTIHOP	185
4.15.3.73ENAMETOOLONG	185
4.15.3.74ENAVAIL	185
4.15.3.75ENETDOWN	185
4.15.3.76 ENETRESET	185
4.15.3.77 ENETUNREACH	186
4.15.3.78 ENFILE	186
4.15.3.79 ENOANO	186
4.15.3.80ENOBUFS	186
4.15.3.81 ENOCSI	186
4.15.3.82ENODATA	186
4.15.3.83ENODEV	186
4.15.3.84ENOENT	187
4.15.3.85ENOEXEC	187
4.15.3.86ENOKEY	187
4.15.3.87ENOLCK	187

4.15.3.88ENOLINK	187
4.15.3.89ENOMEM	187
4.15.3.90ENOMSG	187
4.15.3.91ENONET	188
4.15.3.92ENOPKG	188
4.15.3.93ENOPROTOOPT	188
4.15.3.94ENOSPC	188
4.15.3.95ENOSR	188
4.15.3.96ENOSTR	188
4.15.3.97ENOSYS	188
4.15.3.98ENOTBLK	189
4.15.3.99ENOTCONN	189
4.15.3.100ENOTDIR	189
4.15.3.101ENOTEMPTY	189
4.15.3.102ENOTNAM	189
4.15.3.103ENOTSOCK	189
4.15.3.104ENOTTY	189
4.15.3.105ENOTUNIQ	190
4.15.3.106ENXIO	190
4.15.3.107EOF	190
4.15.3.108EOPNOTSUPP	190
4.15.3.109EOVERFLOW	190
4.15.3.110EPERM	190
4.15.3.111EPFNOSUPPORT	190
4.15.3.112EPIPE	191
4.15.3.113EPROTO	191
4.15.3.114EPROTONOSUPPORT	191
4.15.3.115EPROTOTYPE	191
4.15.3.116ERANGE	191
4.15.3.117EREMCHG	191

4.15.3.11	EREMOTE	191
4.15.3.11	EREMOTEIO	192
4.15.3.12	ERESTART	192
4.15.3.12	EROFS	192
4.15.3.12	Errno	192
4.15.3.12	ESHUTDOWN	192
4.15.3.12	ESOCKTNOSUPPORT	192
4.15.3.12	ESPIPE	192
4.15.3.12	ESRCH	193
4.15.3.12	ESRMNT	193
4.15.3.12	ESTALE	193
4.15.3.12	ESTRPIPE	193
4.15.3.13	ETIME	193
4.15.3.13	ETIMEDOUT	193
4.15.3.13	ETOOMANYREFS	193
4.15.3.13	ETXTBSY	194
4.15.3.13	EUCLEAN	194
4.15.3.13	EUNATCH	194
4.15.3.13	EUSERS	194
4.15.3.13	EWOULDBLOCK	194
4.15.3.13	EXDEV	194
4.15.3.13	EXFULL	194
4.15.3.14	0NT_MAX	195
4.15.3.14	1NT_MIN	195
4.15.3.14	2LONG_MAX	195
4.15.3.14	3LONG_MIN	195
4.15.3.14	4LONG_MAX	195
4.15.3.14	5LONG_MIN	195
4.15.3.14	6M_CACHE_PAGES	195
4.15.3.14	7MAP_ANONYMOUS	196

4.15.3.148	MAP_FAILED	196
4.15.3.149	MAP_FIXED	196
4.15.3.150	MAP_PHYS_NON_CACHED	196
4.15.3.151	MAP_PHYS_NON_SECURE	196
4.15.3.152	MAP_POPULATE	196
4.15.3.153	MAP_PRIVATE	196
4.15.3.154	MAP_SHARED	197
4.15.3.155	MAX_CPUS	197
4.15.3.156	NUM_MILLIS_IN_SEC	197
4.15.3.157	NUM_NANOS_IN_MILLI	197
4.15.3.158	NUM_NANOS_IN_SEC	197
4.15.3.159	NUM_NANOS_IN_USEC	197
4.15.3.160	NUM_SECONDS_IN_MIN	197
4.15.3.161	PGOFF_SHIFT	198
4.15.3.162	PHYS_DEV_NAME	198
4.15.3.163	PHYS_DEV_NAME_LEN	198
4.15.3.164	PRId16	198
4.15.3.165	PRId32	198
4.15.3.166	PRId64	198
4.15.3.167	PRi16	198
4.15.3.168	PRi32	199
4.15.3.169	PRi64	199
4.15.3.170	PRi16	199
4.15.3.171	PRi32	199
4.15.3.172	PRi64	199
4.15.3.173	PROT_EXEC	199
4.15.3.174	PROT_NONE	199
4.15.3.175	PROT_READ	200
4.15.3.176	PROT_WRITE	200
4.15.3.177	PTHREAD_KEYS_MAX	200

4.15.3.17	CHAR_MAX	200
4.15.3.17	CHAR_MIN	200
4.15.3.18	CNd16	200
4.15.3.18	CNd32	200
4.15.3.18	CNd64	201
4.15.3.18	CNu16	201
4.15.3.18	CNu32	201
4.15.3.18	CNu64	201
4.15.3.18	CNx16	201
4.15.3.18	CNx32	201
4.15.3.18	CNx64	201
4.15.3.18	SHRT_MAX	202
4.15.3.19	SHRT_MIN	202
4.15.3.19	TEMP_FAILURE_RETRY	202
4.15.3.19	UCHAR_MAX	202
4.15.3.19	UJINT_MAX	202
4.15.3.19	ULLONG_MAX	203
4.15.3.19	ULONG_MAX	203
4.15.3.19	USHRT_MAX	203
4.15.3.19	uid_generate_time	203
4.15.3.19	UID_STRING_LEN	203
4.15.3.19	uid_unparse_lower	203
4.15.4	Typedef Documentation	203
4.15.4.1	__uid_t	203
4.15.4.2	constraint_handler_t	204
4.15.4.3	errno_t	204
4.15.4.4	gid_t	204
4.15.4.5	mode_t	204
4.15.4.6	off_t	204
4.15.4.7	pid_t	204

4.15.4.8	ssize_t	204
4.15.4.9	time_t	205
4.15.4.10	uid_t	205
4.15.4.11	uuid_t	205
4.15.5	Function Documentation	205
4.15.5.1	_exit(int status)	205
4.15.5.2	_exit_thread(unsigned long status)	205
4.15.5.3	abort_handler_s(const char *restrict msg, void *restrict ptr, errno_t error)	206
4.15.5.4	abs(int j)	206
4.15.5.5	alarm(unsigned int seconds)	206
4.15.5.6	arm_timer_get_frequency(void)	207
4.15.5.7	asprintf(char **strp, const char *fmt,...)	207
4.15.5.8	atexit(void(*func)(void))	207
4.15.5.9	calloc(size_t nmemb, size_t size)	208
4.15.5.10	clock_gettime(clockid_t clk_id, struct timespec *ts)	208
4.15.5.11	close(int fd)	209
4.15.5.12	exit(int status)	209
4.15.5.13	fflush(FILE *stream)	209
4.15.5.14	fprintf(FILE *_restrict_stream, const char *_restrict_fmt,...)	210
4.15.5.15	free(void *ptr)	210
4.15.5.16	fstat(int fd, struct stat *buf)	211
4.15.5.17	ftruncate(int fd, int size)	211
4.15.5.18	get_errno_addr(void)	211
4.15.5.19	getcluster(void)	212
4.15.5.20	getcpu(void)	212
4.15.5.21	getitimer(int which, struct itimerval *curr_value)	212
4.15.5.22	getpid(void)	213
4.15.5.23	gettid(void)	213
4.15.5.24	ignore_handler_s(const char *restrict msg, void *restrict ptr, errno_t error)	213
4.15.5.25	invoke_constraint_handler_s(const char *msg, const char *file, const char *function, uint32_t line, errno_t error)	213

4.15.5.26	ioctl(int fd, int request, unsigned long data)	214
4.15.5.27	isalnum(int c)	214
4.15.5.28	isalpha(int c)	215
4.15.5.29	isascii(int c)	215
4.15.5.30	isblank(int c)	215
4.15.5.31	iscntrl(int c)	216
4.15.5.32	isdigit(int c)	216
4.15.5.33	isgraph(int c)	216
4.15.5.34	islower(int c)	217
4.15.5.35	isprint(int c)	217
4.15.5.36	ispunct(int c)	217
4.15.5.37	isspace(int c)	218
4.15.5.38	isupper(int c)	218
4.15.5.39	isxdigit(int c)	218
4.15.5.40	lseek(int fd, int offset, int whence)	219
4.15.5.41	malloc(size_t size)	219
4.15.5.42	memchr(const void *s, int c, size_t n)	220
4.15.5.43	memcmp(const void *s1, const void *s2, size_t n)	220
4.15.5.44	memcpy(void *dest, const void *src, size_t n)	220
4.15.5.45	memcpy_s(void *restrict dest, rsize_t dest_max, const void *restrict src, rsize_t n)	221
4.15.5.46	memmove(void *dest, const void *src, size_t n)	221
4.15.5.47	memmove_s(void *dest, rsize_t dest_max, const void *src, rsize_t n)	222
4.15.5.48	memset(void *block, int c, size_t size)	222
4.15.5.49	memset_s(void *block, rsize_t block_max, int c, rsize_t n)	222
4.15.5.50	mmap(void *addr, size_t len, int prot, int flags, int fd, off_t offset)	223
4.15.5.51	ms_to_timespec(int64_t t, struct timespec *a)	225
4.15.5.52	munmap(void *addr, size_t length)	225
4.15.5.53	nanosleep(struct timespec *req, struct timespec *rem)	226
4.15.5.54	open(const char *pathname, int flags,...)	226
4.15.5.55	printf(const char *fmt,...)	227

4.15.5.56	printf_no_alloc(const char *fmt,...)	228
4.15.5.57	printf_s(const char *restrict fmt,...)	228
4.15.5.58	profil(unsigned short *buf, size_t bufsiz, size_t offset, unsigned int scale)	229
4.15.5.59	putchar(int ch)	229
4.15.5.60	puts(const char *s)	230
4.15.5.61	qsort(void *base, size_t nmemb, size_t size, int(*compar)(const void *, const void *))	230
4.15.5.62	qsort_r(void *base, size_t nmemb, size_t size, int(*compar)(const void *, const void *, void *), void *arg)	230
4.15.5.63	read(int fd, void *buf, size_t count)	231
4.15.5.64	realloc(void *ptr, size_t size)	231
4.15.5.65	sched_getaffinity(pid_t pid, size_t cpusetsize, cpu_set_t *mask)	232
4.15.5.66	sched_setaffinity(pid_t pid, size_t cpusetsize, cpu_set_t *mask)	232
4.15.5.67	sched_yield(void)	232
4.15.5.68	set_constraint_handler_s(constraint_handler_t handler)	233
4.15.5.69	setitimer(int which, const struct itimerval *new_value, struct itimerval *old_value)	233
4.15.5.70	snprintf(char *s, size_t count, const char *fmt,...)	233
4.15.5.71	snprintf_s(char *restrict s, rsize_t n, const char *restrict format,...)	234
4.15.5.72	sprintf(char *s, const char *fmt,...)	235
4.15.5.73	sprintf_s(char *restrict s, rsize_t n, const char *restrict format,...)	235
4.15.5.74	sscanf(const char *buf, const char *fmt,...)	236
4.15.5.75	sscanf_s(const char *restrict s, const char *restrict format,...)	236
4.15.5.76	stat(const char *pathname, struct stat *buf)	237
4.15.5.77	strcat(char *dest, const char *src)	237
4.15.5.78	strcat_s(char *restrict dest, rsize_t dest_max, const char *restrict src)	238
4.15.5.79	strchr(const char *s, int c)	238
4.15.5.80	strchrnul(const char *s, int c)	239
4.15.5.81	strcmp(const char *s1, const char *s2)	239
4.15.5.82	strcpy(char *dest, const char *src)	240
4.15.5.83	strcpy_s(char *restrict dest, rsize_t dest_max, const char *restrict src)	240
4.15.5.84	strcspn(const char *s, const char *reject)	241

4.15.5.85	<code>strdup(const char *s)</code>	241
4.15.5.86	<code>strerror(int errnum)</code>	241
4.15.5.87	<code>strerror_r(int errnum, char *buf, size_t buflen)</code>	242
4.15.5.88	<code>strerror_s(char *buf, rsize_t bufmax, errno_t errnum)</code>	242
4.15.5.89	<code>strlcat(char *dest, const char *src, size_t n)</code>	242
4.15.5.90	<code>strlen(const char *s)</code>	243
4.15.5.91	<code>strncat(char *dest, const char *src, size_t n)</code>	243
4.15.5.92	<code>strncat_s(char *restrict dest, rsize_t dest_max, const char *restrict src, rsize_t n)</code>	244
4.15.5.93	<code>strncmp(const char *s1, const char *s2, size_t n)</code>	244
4.15.5.94	<code>strncpy(char *dest, const char *src, size_t n)</code>	245
4.15.5.95	<code>strncpy_s(char *restrict dest, rsize_t dest_max, const char *restrict src, rsize_t n)</code>	246
4.15.5.96	<code>strnlen(const char *s, size_t n)</code>	246
4.15.5.97	<code>strnlen_s(const char *s, size_t s_max)</code>	247
4.15.5.98	<code>strrchr(const char *s, int c)</code>	247
4.15.5.99	<code>strspn(const char *s, const char *accept)</code>	248
4.15.5.100	<code>strstr(const char *text, const char *pattern)</code>	248
4.15.5.101	<code>strtod(const char *nptr, char **endptr)</code>	249
4.15.5.102	<code>strtod(const char *nptr, char **endptr)</code>	250
4.15.5.103	<code>strtok(char *str, const char *delim)</code>	250
4.15.5.104	<code>strtok_r(char *_restrict_s, const char *_restrict_sep, char **_restrict_p)</code>	251
4.15.5.105	<code>strtol(const char *nptr, char **endptr, int base)</code>	251
4.15.5.106	<code>strtold(const char *nptr, char **endptr)</code>	252
4.15.5.107	<code>strtoll(const char *nptr, char **endptr, int base)</code>	252
4.15.5.108	<code>strtoul(const char *cp, char **endp, int base)</code>	253
4.15.5.109	<code>strtoull(const char *cp, char **endp, int base)</code>	254
4.15.5.110	<code>time(time_t *time)</code>	254
4.15.5.111	<code>timeadd(const struct timespec *a, const struct timespec *b, struct timespec *res)</code>	255
4.15.5.112	<code>timespec_to_ms(const struct timespec *a)</code>	255
4.15.5.113	<code>timespec_to_nsec(const struct timespec *a)</code>	255

4.15.5.114	timesub(const struct timespec *a, const struct timespec *b, struct time- spec *res)	256
4.15.5.115	tolower(int c)	256
4.15.5.116	toupper(int c)	256
4.15.5.117	unlink(const char *pathname)	257
4.15.5.118	uuid_clear(uuid_t *uu)	257
4.15.5.119	uuid_compare(const uuid_t *uu1, const uuid_t *uu2)	257
4.15.5.120	uuid_generate(uuid_t *out)	258
4.15.5.121	uuid_is_null(const uuid_t *uu)	258
4.15.5.122	uuid_parse(const char *in, uuid_t *uu)	258
4.15.5.123	uuid_unparse(const uuid_t *uu, char *out)	259
4.15.5.124	uuid_unparse_upper(const uuid_t *uu, char *out)	259
4.15.5.125	vasprintf(char **strp, const char *fmt, va_list args)	259
4.15.5.126	vasprintf_s(char **strp, const char *fmt, va_list args)	260
4.15.5.127	vfprintf(FILE *_restrict_ stream, const char *_restrict_ fmt, va_list ap)	260
4.15.5.128	vsprintf(char *buffer, size_t size, const char *format, va_list args)	261
4.15.5.129	vsprintf_s(char *restrict s, rsize_t n, const char *restrict format, va_list arg)	261
4.15.5.130	wsprintf(char *buffer, const char *format, va_list args)	262
4.15.5.131	wsprintf_s(char *restrict s, rsize_t n, const char *restrict format, va_list arg)	263
4.15.5.132	wscanf(const char *buffer, const char *fmt, va_list args)	263
4.15.5.133	wscanf_s(const char *restrict s, const char *restrict format, va_list arg)	264
4.15.5.134	write(int fd, const void *buf, size_t count)	264
4.15.6	Variable Documentation	265
4.15.6.1	stdin	265
4.16	TeesLsmc	266
4.16.1	Detailed Description	266
4.16.2	Function Documentation	266
4.16.2.1	TEES_SMCCCommand(TEES_SMCHandle handle, TEES_SMCData *data)	266
4.16.2.2	TEES_SMCFini(TEES_SMCHandle handle)	267
4.16.2.3	TEES_SMCInit(TEES_SMCHandle *handle)	267

4.17 Tee_sockets	268
4.17.1 Detailed Description	273
4.17.2 Data Structure Documentation	273
4.17.2.1 struct TEE_iSocket_s	273
4.17.2.2 struct TEE_tcpSocket_Setup_s	275
4.17.2.3 struct TEE_udpSocket_Setup_s	275
4.17.2.4 struct TEE_udpSocket_Change_s	276
4.17.2.5 struct TEE_tlsSocket_PSK_Info_s	276
4.17.2.6 struct TEE_tlsSocket_SRP_Info_s	276
4.17.2.7 struct TEE_tlsSocket_ClientPDC_s	276
4.17.2.8 struct TEE_tlsSocket_ServerPDC_s	277
4.17.2.9 struct TEE_tlsSocket_CertStorageCred_s	277
4.17.2.10 struct TEE_tlsSocket_Credentials_s	277
4.17.2.11 union TEE_tlsSocket_Credentials_s.__unnamed__	277
4.17.2.12 struct TEE_tlsSocket_CallbackInfo_s	278
4.17.2.13 struct TEE_tlsSocket_Setup_s	278
4.17.2.14 union TEE_tlsSocket_Setup_s.__unnamed__	280
4.17.2.15 struct TEE_tlsSocket_CB_Data_s	280
4.17.3 Typedef Documentation	280
4.17.3.1 TEE_iSocket	280
4.17.3.2 TEE_tlsSocket_CB_Data	281
4.17.4 Enumeration Type Documentation	281
4.17.4.1 anonymous enum	281
4.17.4.2 anonymous enum	281
4.17.4.3 anonymous enum	281
4.17.4.4 anonymous enum	282
4.17.4.5 anonymous enum	282
4.17.4.6 anonymous enum	282
4.17.4.7 anonymous enum	282
4.17.4.8 anonymous enum	282

4.17.4.9	anonymous enum	283
4.17.4.10	anonymous enum	283
4.17.4.11	anonymous enum	283
4.17.4.12	anonymous enum	283
4.17.4.13	protocol_error_code	284
4.17.4.14	TEE_ipSocket_ipVersion_e	286
4.17.4.15	TEE_tlsSocket_CallbackReasonType_e	286
4.17.4.16	TEE_tlsSocket_ClientCredentialType_e	286
4.17.4.17	TEE_tlsSocket_ExtensionFlags_e	286
4.17.4.18	TEE_tlsSocket_ServerCredentialType_e	287
4.17.4.19	TEE_tlsSocket_StatusRequestType_e	287
4.17.4.20	TEE_tlsSocket_tlsVersion_e	287
4.17.5	Function Documentation	287
4.17.5.1	TEES_lwtCloseChannel(TEES_lwtHandle iwtCtx)	287
4.17.5.2	TEES_lwtOpenChannel(const char *listenerName, TEES_lwtHandle *iwtCtx)	288
4.17.5.3	TEES_lwtRead(TEES_lwtHandle iwtCtx, void *buf, uint32_t *length)	288
4.17.5.4	TEES_lwtWrite(TEES_lwtHandle iwtCtx, const void *buf, uint32_t *length)	289
5	Data Structure Documentation	290
5.1	__TEE_SC_CardKeyRef	290
5.1.1	Detailed Description	290
5.1.2	Field Documentation	290
5.1.2.1	scKeyID	290
5.1.2.2	scKeyVersion	290
5.2	__TEE_SC_DeviceKeyRef	290
5.2.1	Detailed Description	290
5.2.2	Field Documentation	291
5.2.2.1	"@5	291
5.2.2.2	scKeyType	291
5.3	__TEE_SC_DeviceKeyRef.__unnamed__	291

5.3.1	Field Documentation	291
5.3.1.1	scBaseKeyHandle	291
5.3.1.2	scKeySetRef	291
5.4	__TEE_SC_KeySetRef	291
5.4.1	Detailed Description	291
5.4.2	Field Documentation	292
5.4.2.1	scKeyEncHandle	292
5.4.2.2	scKeyMacHandle	292
5.5	__TEE_SC_OID	292
5.5.1	Detailed Description	292
5.5.2	Field Documentation	292
5.5.2.1	buffer	292
5.5.2.2	bufferLen	292
5.6	__TEE_SC_Params	292
5.6.1	Detailed Description	293
5.6.2	Field Documentation	293
5.6.2.1	scCardKeyRef	293
5.6.2.2	scDeviceKeyRef	293
5.6.2.3	scOID	293
5.6.2.4	scSecurityLevel	293
5.6.2.5	scType	293
5.7	__TEE_SEAID	293
5.7.1	Detailed Description	293
5.7.2	Field Documentation	294
5.7.2.1	buffer	294
5.7.2.2	bufferLen	294
5.8	__TEE_SEReaderProperties	294
5.8.1	Detailed Description	294
5.8.2	Field Documentation	294
5.8.2.1	selectResponseEnable	294

5.8.2.2	sePresent	294
5.8.2.3	teeOnly	294
5.9	smc_data	294
5.9.1	Detailed Description	295
5.9.2	Field Documentation	295
5.9.2.1	arg_types	295
5.9.2.2	args	295
5.10	stat	295
5.10.1	Detailed Description	295
5.10.2	Field Documentation	295
5.10.2.1	st_gid	295
5.10.2.2	st_mode	295
5.10.2.3	st_size	295
5.10.2.4	st_uid	295
6	File Documentation	296
6.1	assert.h File Reference	296
6.1.1	Detailed Description	296
6.2	cache.h File Reference	296
6.2.1	Detailed Description	296
6.3	core/error.h File Reference	297
6.4	core/mman.h File Reference	299
6.5	sys/mman.h File Reference	299
6.5.1	Detailed Description	300
6.6	ctype.h File Reference	300
6.6.1	Detailed Description	301
6.7	driver.h File Reference	301
6.7.1	Detailed Description	301
6.8	driver/mem/phys.h File Reference	302
6.9	errno.h File Reference	302
6.9.1	Detailed Description	302

6.10	fcntl.h File Reference	302
6.10.1	Detailed Description	303
6.11	inttypes.h File Reference	303
6.11.1	Detailed Description	303
6.12	irs.h File Reference	304
6.12.1	Detailed Description	305
6.13	limits.h File Reference	305
6.13.1	Detailed Description	305
6.14	malloc.h File Reference	306
6.14.1	Detailed Description	306
6.15	math.h File Reference	306
6.15.1	Detailed Description	308
6.16	mqueue.h File Reference	308
6.16.1	Detailed Description	309
6.17	print_no_alloc.h File Reference	309
6.17.1	Detailed Description	309
6.18	protocol_errors.h File Reference	309
6.18.1	Detailed Description	310
6.19	pthread.h File Reference	311
6.19.1	Detailed Description	313
6.20	rpmb.h File Reference	313
6.20.1	Detailed Description	313
6.21	sched.h File Reference	314
6.21.1	Detailed Description	314
6.22	scma.h File Reference	314
6.22.1	Detailed Description	315
6.23	stdio.h File Reference	315
6.23.1	Detailed Description	317
6.24	stdlib.h File Reference	317
6.24.1	Detailed Description	318

6.25 string.h File Reference	318
6.25.1 Detailed Description	320
6.26 sys/credentials.h File Reference	320
6.26.1 Detailed Description	321
6.26.2 Function Documentation	321
6.26.2.1 cred_compare(const struct cred *as_is, const struct cred *to_be)	321
6.27 sys/ioctl.h File Reference	321
6.27.1 Detailed Description	321
6.28 sys/socket.h File Reference	321
6.28.1 Detailed Description	322
6.29 sys/types.h File Reference	322
6.29.1 Detailed Description	323
6.30 sys/un.h File Reference	323
6.30.1 Detailed Description	323
6.31 tee_error.h File Reference	323
6.31.1 Detailed Description	323
6.32 tee_i2c.h File Reference	323
6.32.1 Detailed Description	324
6.33 tee_interrupt.h File Reference	324
6.33.1 Detailed Description	325
6.34 tee_isocket.h File Reference	325
6.34.1 Detailed Description	326
6.35 tee_smc.h File Reference	326
6.35.1 Detailed Description	327
6.36 tee_spi.h File Reference	327
6.36.1 Detailed Description	328
6.37 tee_ta_destructor.h File Reference	328
6.37.1 Detailed Description	328
6.38 tee_tcpsocket.h File Reference	328
6.38.1 Detailed Description	329

6.39 tee_tlssocket.h File Reference	329
6.39.1 Detailed Description	332
6.40 tee_udpsocket.h File Reference	333
6.40.1 Detailed Description	333
6.41 tees_critical.h File Reference	334
6.41.1 Detailed Description	334
6.42 tees_extension.h File Reference	334
6.42.1 Detailed Description	334
6.43 tees_hwcrypto_buf.h File Reference	335
6.43.1 Detailed Description	335
6.44 tees_iwt.h File Reference	335
6.44.1 Detailed Description	336
6.45 tees_kdf.h File Reference	336
6.45.1 Detailed Description	336
6.46 tees_rot.h File Reference	336
6.46.1 Detailed Description	337
6.47 tees_seccam.h File Reference	337
6.47.1 Detailed Description	337
6.48 tees_secure_object.h File Reference	337
6.48.1 Detailed Description	338
6.49 tees_tui.h File Reference	339
6.49.1 Detailed Description	341
6.50 tees_wrapped_with_rek.h File Reference	341
6.50.1 Detailed Description	342
6.51 time.h File Reference	342
6.51.1 Detailed Description	343
6.52 tui.h File Reference	343
6.52.1 Detailed Description	344
6.53 unistd.h File Reference	345
6.53.1 Detailed Description	346
6.54 uuid/uuid.h File Reference	346
6.54.1 Detailed Description	346
Bibliography	347
Index	347

1 Overview

1.1 Objective

The goal of this document is to describe Secure OS and Service Library API categorized to Samsung Internal API. The intended audience is Trusted Application (TA) developers with permission to access Samsung Internal API.

1.2 Scope

The scope of this document is to describe Samsung Internal API to Trusted Application developers. Global Platform (GP) API is out of scope of this document, and they are explained TEE Internal API document at [Global Platform](#) portal.

1.3 Abbreviations and Acronyms

Terminology / Abbreviation	Description
CA	Client application
TA	Trusted application

1.4 API type

1.4.1 TA properties

Each TA has set of mandatory and custom properties which can help TA developers to setup TA features. Mandatory properties are implemented according to TEE specifications and custom one are Samsung extensions.

Table 1.1 Mandatory properties

Name	Type	Description
TA_PROP_DATASIZE	Integer	Maximum estimated amount of dynamic data in bytes configured for the Trusted Application. The memory blocks allocated through TEE_Malloc are drawn from this space, as well as the task stacks. How this value precisely relates to the exact number and sizes of blocks that can be allocated is implementation-dependent.
TA_PROP_GROUP_ID	String	Group identifier to which TA belongs to.

Name	Type	Description
TA_PROP_INSTANCE_KEEPLIVE	Boolean	Whether the Trusted Application instance context SHALL be preserved when there are no sessions connected to the instance. The instance context is defined as all writable data within the memory space of the Trusted Application instance, including the instance heap. This property is meaningful only when the TA_PROP_SINGLE_INSTANCE is set to true. When this property is set to false, then the TA instance MUST be created when one or more sessions are opened on the TA and it MUST be destroyed when there are no more sessions opened on the instance. When this property is set to true, then the TA instance is terminated only when the TEE shuts down, which includes when the device goes through a system-wide global power cycle. Note that the TEE MUST NOT shut down whenever the REE does not shut down and keeps a restorable state, including when it goes through transitions into lower power states (hibernation, suspend, etc.). The exact moment when a keep-alive single instance is created is implementation-defined but it MUST be no later than the first session opening.
TA_PROP_MULTISESSION	Boolean	Whether the Trusted Application instance supports multiple sessions.
TA_PROP_SINGLE_INSTANCE	Boolean	Whether the Implementation SHALL create a single TA instance for all the client sessions (if true) or SHALL create a separate instance for each client session (if false).
TA_PROP_STACKSIZE	Integer	Maximum stack size in bytes available to any task in the Trusted Application at any point in time. This corresponds to the stack size used by the TA code itself and does not include stack space possibly used by the Trusted Core Framework. For example, if this property is set to " 512 ", then the Framework MUST guarantee that, at any time, the TA code can consume up to 512 bytes of stack and still be able to call any functions in the API.
TA_PROP_UUID	UUID	16 bits identifier of the Trusted Application.
TA_PROP_VERSION	String	Version number of this Trusted Application.

Optional properties

GPD.TA.DBG_DLM.DATA_AVAILABLE - Debug Rules Property states what sort of DLM data can be retrieved according to the TA.

Table 1.2 Possible values

Name	Value	Description
TEE_BLOCKED	-64	As a TA rule value, indicates that while the TA with this value has active sessions, DLM events of any TA in the TEE shall be ignored and not reported through the DLM service. As a TEE rule value, shall be considered the same as BLOCKED.
BLOCKED	0	No DLM is available. As a TA rule value, indicates that while this TA may open a DLM session and use TEE_DLMPrintf(), there shall be no output. As a TEE rule value, indicates that while any TA may open a DLM session and use TEE_DLMPrintf(), there shall be no output.
ALL	64	All DLM data is available.

GPD.TA.DBG_PMR.DATA_AVAILABLE - Debug Rules Property states what sort of PMR data shall be retrievable, according to the TA

Table 1.3 Possible values

Name	Value	Description
TEE_BLOCKED	-64	As a TA rule value, indicates that while the TA with this value has active sessions, PMR events of any TA in the TEE shall be ignored and not reported through the PMR service.
BLOCKED (default)	0	As a TA rule value, indicates that PMR events of this TA shall be ignored and not reported through the PMR service.
CODE_ONLY	32	Only the following data about the panicked TA shall be available: spec-Number functionName markValue panicReasonCode
CODE_STATE	64	This adds the panicked TA's state to the CODE_ONLY set of data available. If sufficient resources are available, the state data shall be placed at the location defined by PMR_STATE_POINTER
HEAP_STACK	96	This adds the panicked TA's heap and stack to the CODE_STATE set of data available. If sufficient resources are available, the heap and stack data shall be placed at the locations defined by PMR_HEAP_POINTER and PMR_STACK_POINTER
ALL	112	All data in the TEE is available to be reported. In systems where TA specific state could not be presented due to the method of sharing TA stacks and heap between trusted applications, in this state such data may be presented.

gpd.ta.description - Trusted Application description. Default value is "descr. none".

samsung.ta.cacheHeapSize (old name is gpd.ta.cacheHeapSize) - Memory area that will be allocated while TA starting. This memory is not exempted with [free\(\)](#). Default value is 0.

samsung.ta.FIPS_mode_enable - Enable using FIPS. Default value is false.

samsung.ta.numInstances - Number of instances. It effects if TA_PROP_SINGLE_INSTANCE = false. Default value is 0.

samsung.ta.numSessions - Number of sessions. It effects if TA_PROP_MULTISESSION = true.

sys.ta.no_aslr - Allocation to special memory addresses. Default value is false.

sys.ta.rlim_cur - Priority of TA start. Default value is sys.ta.rlim_max / 2.

sys.ta.rlim_max - Maximum priority value. It is constant.

sys.ta.thread_count - Number of threads, default value is 2.

1.4.2 Samsung Internal API

[Samsung Internal API](#) is Samsung's own API to support any privileged features to internal TA developers such as driver API, interrupt API, custom SMC handler, and so on. Only TA developers with permission to the API can use it, and 3rd-party TA developers should be prevented from accessing the API for security reasons.

1.4.3 Thread support library API

[Thread support library API](#) A set of interfaces (functions, header files) for threaded programming commonly known as POSIX threads, or Pthreads. A single process can contain multiple threads, all of which are executing the same program. These threads share the same global memory (data and heap segments), but each thread has its own stack (automatic variables).

1.4.4 Math library API

[Math library API](#) Set of math functions for floating point calculation.

1.4.5 Auxiliary API

[Auxiliary API](#) is a set of API to provide POSIX-like interface to TA developers. The feature coverage is limited to the minimum set of functions rather than fully supporting POSIX standard in Secure World. Currently, only standard IO functions and string operations are supported by Auxiliary API.

2 Deprecated List

Global **TEES_RegisterDriver** (char *name, struct fops *fops, unsigned int drvid, struct **usr_drv_info** **info) **_deprecated_**

Use **TEES_InitDriver()** instead

Global **TEES_ReleaseDriver** (struct **usr_drv_info** **info) **_deprecated_**

Use **TEES_FiniDriver()** instead

3 Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

__TEE_SC_CardKeyRef	290
__TEE_SC_DeviceKeyRef	290
__TEE_SC_DeviceKeyRef.__unnamed__	291
__TEE_SC_KeySetRef	291
__TEE_SC_OID	292
__TEE_SC_Params	292
__TEE_SEAID	293
__TEE_SEReaderProperties	294
smc_data	
SMC command description	294
stat	295

4 Module Documentation

4.1 Samsung Internal API

Modules

- [Loadable driver API](#)
- [Custom handler API](#)
- [Contiguous memory API](#)
- [SPI API](#)
- [I2C API](#)
- [Trusted user interface](#)
- [Integrity Report System API](#)
- [Miscellaneous extensions](#)
- [RPMB API](#)

Data Structures

- struct [rot_t](#)
Structure to handle Root of Trust information. [More...](#)
- struct [wrapped_wkth_rek_t](#)
Structure for wrapping with REK. [More...](#)

Macros

- [#define SO_TAG_LEN](#) (16)
- [#define SO_IV_LEN](#) (16)
- [#define SO_AC_LEN](#) (4)
- [#define SO_MAGIC_NUMBER_LEN](#) (4)
- [#define SO_TA_ID_LEN](#) (16)
- [#define SO_AUTH_ID_LEN](#) (16)
- [#define SO_HEADER_SIZE](#)(delegated)
- [#define SO_OUT_BUF_SIZE](#)(in_len, delegated) ((in_len) + (SO_HEADER_SIZE(delegated)))
- [#define SHA256_DIGEST_LEN](#) 32
SHA256_DIGEST_LEN is defined to set size for verified_boot_key of ROOT_OF_TRUST.
- [#define KM_KW_MAX_SALT_LEN](#) 60
- [#define KM_KW_MAX_IV_LEN](#) 12
- [#define KM_KW_MAX_AAD_LEN](#) 32
- [#define KM_KW_MAX_KEY_LEN](#) 32
- [#define KM_KW_MAX_INPUT_LEN](#) 4096
- [#define KM_KW_MAX_TAG_LEN](#) 16

Typedefs

- typedef struct [rot_t](#) ROOT_OF_TRUST
Structure to handle Root of Trust information.
- typedef struct [wrapped_wkth_rek_t](#) WRAP_REK
Structure for wrapping with REK.

Enumerations

- enum `kw_mode` { **WRAP**, **UNWRAP** }
Wrapping mode. *WRAP* or *UNWRAP*.

Functions

- TEE_Result `errno_to_tee_error` (int error_code)
Translate *errno* to GP TEE errors code.
- TEE_Result `TEES_EnterCritical` (void)
Disable routing and handling of normal world interrupts.
- TEE_Result `TEES_ExitCritical` (void)
Enable routing and handling of normal world interrupts.
- TEE_Result `TEES_DeriveKeyKDF` (const void *label, uint32_t labelLen, const void *context, uint32_t contextLen, uint32_t outputKeyLen, TEE_ObjectHandle object)
Key Derivation Function(KDF) based on device key. Internal implementation of KDF depends on the chipset.
- TEE_Result `TEES_DeriveKeySetKDF` (const void *label, uint32_t labelLen, const void *context, uint32_t contextLen, uint32_t outputKeyLen, TEE_ObjectHandle object)
Key Derivation Function(KDF) based on device key. This function returns the same key for the set of TAs of the same authority. Internal implementation of KDF depends on the chipset.
- TEE_Result `TEES_LockHWCryptoBuf` (void)
Lock HW crypto buffer.
- TEE_Result `TEES_UnlockHWCryptoBuf` (void)
Unock HW crypto buffer.
- TEE_Result `TEES_WrapSecureObject` (const unsigned char *in, uint32_t in_len, unsigned char *out, uint32_t *out_len, SO_AccessControllInfoType *ac)
Encrypt and sign input data.
- TEE_Result `TEES_UnwrapSecureObject` (const unsigned char *in, uint32_t in_len, unsigned char *out, uint32_t *out_len)
Decrypt and verify wrapped data.
- TEE_Result `TEES_CheckSecureObjectCreator` (const unsigned char *in, uint32_t in_len, SO_AccessControllInfoType *ac)
Check UUID and AUTH_ID of creator on wrapped data.
- TEE_Result `TEES_GetRoT` (ROOT_OF_TRUST *rot)
Get RoT information.
- TEE_Result `TEES_WrappedWithREK` (WRAP_REK *data)
Wrapping with REK.
- TEE_Result `TEES_SECCAM_GetStatus` (unsigned int *data)
Get a status of secure camera.

4.1.1 Detailed Description

4.1.2 Data Structure Documentation

4.1.2.1 struct rot_t

Structure to handle Root of Trust information.

Data Fields

uint32_t	device_locked	
uint32_t	os_version	
uint32_t	patch_month_year	
uint64_t	reserved[4]	
uint8_t	verified_boot_key[32]	
uint32_t	verified_boot_state	

4.1.2.2 struct wrapped_wkth_rek_t

Structure for wrapping with REK.

Data Fields

uint8_t	aad[32]	
uint32_t	aad_len	
uint8_t	auth_tag[16]	
uint32_t	auth_tag_len	
uint8_t	encrypted_key[4096]	
uint32_t	encrypted_key_len	
uint8_t	iv[12]	
uint32_t	iv_len	
uint32_t	kw_mode	
uint8_t	plaintext_key[4096]	
uint32_t	plaintext_key_len	
uint8_t	salt[60]	
uint32_t	salt_len	

4.1.3 Macro Definition Documentation**4.1.3.1 #define KM_KW_MAX_AAD_LEN 32**

```
#include <tees_wrapped_with_rek.h>
```

Length in bytes of maximum Authenticated data for AES-GCM to wrapped with REK.

4.1.3.2 #define KM_KW_MAX_INPUT_LEN 4096

```
#include <tees_wrapped_with_rek.h>
```

Length in bytes of maximum input data which is wrapped with REK

4.1.3.3 #define KM_KW_MAX_IV_LEN 12

```
#include <tees_wrapped_with_rek.h>
```

Length in bytes of maximum Initial Vector field to wrapped with REK.

4.1.3.4 #define KM_KW_MAX_KEY_LEN 32

```
#include <tees_wrapped_with_rek.h>
```

Length in bytes of maximum key which wraps input data SW mode only

4.1.3.5 #define KM_KW_MAX_SALT_LEN 60

```
#include <tees_wrapped_with_rek.h>
```

Length in bytes of maximum Salt field to wrapped with REK.

4.1.3.6 #define KM_KW_MAX_TAG_LEN 16

```
#include <tees_wrapped_with_rek.h>
```

Length in bytes of maximum Tag field in wrapped with REK

4.1.3.7 #define SO_AC_LEN (4)

```
#include <tees_secure_object.h>
```

Length in bytes of Access Control field in wrapped object.

4.1.3.8 #define SO_AUTH_ID_LEN (16)

```
#include <tees_secure_object.h>
```

Length in bytes of Auth ID field in wrapped object. Present only in delegation case.

4.1.3.9 #define SO_HEADER_SIZE(*delegated*)

```
#include <tees_secure_object.h>
```

Value:

```
((SO_TAG_LEN) + (SO_IV_LEN) + (SO_AC_LEN) + (
SO_MAGIC_NUMBER_LEN) + \
((delegated) ? (SO_TA_ID_LEN) + (SO_AUTH_ID_LEN) : 0))
```

Get the size of the Secure Object's Header.

4.1.3.10 #define SO_IV_LEN (16)

```
#include <tees_secure_object.h>
```

Length in bytes of Input Vector field in wrapped object.

4.1.3.11 #define SO_MAGIC_NUMBER_LEN (4)

```
#include <tees_secure_object.h>
```

Length in bytes of magic number.

4.1.3.12 #define SO_OUT_BUF_SIZE(in_len, delegated) ((in_len) + (SO_HEADER_SIZE(delegated)))

```
#include <tees_secure_object.h>
```

Get the size of output buffer for Secure Object, accounting Header size.

4.1.3.13 #define SO_TA_ID_LEN (16)

```
#include <tees_secure_object.h>
```

Length in bytes of TA UUID field in wrapped object. Present only in delegation case.

4.1.3.14 #define SO_TAG_LEN (16)

```
#include <tees_secure_object.h>
```

Length in bytes of TAG field in wrapped object.

4.1.4 Function Documentation

4.1.4.1 TEE_Result errno_to_tee_error (int error_code)

```
#include <tee_error.h>
```

Translate errno to GP TEE errors code.

Parameters

in	error_code	errno error code.
----	------------	-------------------

Returns

TEE errors code.

4.1.4.2 TEE_Result TEES_CheckSecureObjectCreator (const unsigned char * *in*, uint32_t *in_len*, SO_AccessControllInfoType * *ac*)

```
#include <tees_secure_object.h>
```

Check UUID and AUTH_ID of creator on wrapped data.

Function will take a buffer containing wrapped SO and check UUID and AUTH_ID on it.

Parameters

in	<i>in</i>	Pointer to input buffer.
in	<i>in_len</i>	Length of input buffer.
in	<i>ac</i>	Pointer to Access Control struct SO_AccessControllInfoType. This is a structure containing access control information.

Return values

<i>TEE_SUCCESS</i>	successfully checked.
<i>TEE_ERROR_XXXX</i>	- unsuccessfully checked : <ul style="list-style-type: none"> • <i>TEE_ERROR_BAD_PARAMETERS</i> - <i>ac</i> is NULL or object size less than SO header length; • <i>TEE_ERROR_BAD_FORMAT</i> - SO magic number does not match or Not supported access flag on wrapped data; • <i>TEE_ERROR_SECURITY</i> - TA_ID or AUTH_ID between wrapped data and <i>ac</i> does not match.

Example:

```
TEES_CheckSecureObjectCreator((const unsigned char *)key1_str,
                              DATA256K,
                              &ac_info);
```

4.1.4.3 TEE_Result TEES_DeriveKeyKDF (const void * *label*, uint32_t *labelLen*, const void * *context*, uint32_t *contextLen*, uint32_t *outputKeyLen*, TEE_ObjectHandle *object*)

```
#include <tees_kdf.h>
```

Key Derivation Function(KDF) based on device key. Internal implementation of KDF depends on the chipset.

Parameters

in	<i>label</i>	label (see KDF description at NIST SP 800-108).
in	<i>labelLen</i>	label length in bytes.
in	<i>context</i>	context (see KDF description at NIST SP 800-108).
in	<i>contextLen</i>	context length in bytes.
in	<i>outputKeyLen</i>	required derived key length in bytes.
out	<i>object</i>	handle on a cryptographic object of appropriate type and size to hold derived key.

Return values

<i>TEE_SUCCESS</i>	in case of success.
<i>TEE_ERROR_XXXX</i>	in case of failure.

4.1.4.4 TEE_Result TEES_DeriveKeySetKDF (const void * *label*, uint32_t *labelLen*, const void * *context*, uint32_t *contextLen*, uint32_t *outputKeyLen*, TEE_ObjectHandle *object*)

```
#include <tees_kdf.h>
```

Key Derivation Function(KDF) based on device key. This function returns the same key for the set of TAs of the same authority. Internal implementation of KDF depends on the chipset.

Parameters

in	<i>label</i>	label (see KDF description at NIST SP 800-108).
in	<i>labelLen</i>	label length in bytes.
in	<i>context</i>	context (see KDF description at NIST SP 800-108).
in	<i>contextLen</i>	context length in bytes.
in	<i>outputKeyLen</i>	required derived key length in bytes.
out	<i>object</i>	handle on a cryptographic object of appropriate type and size to hold derived key.

Return values

<i>TEE_SUCCESS</i>	in case of success.
<i>TEE_ERROR_XXXX</i>	in case of failure.

4.1.4.5 TEE_Result TEES_EnterCritical (void)

```
#include <tees_critical.h>
```

Disable routing and handling of normal world interrupts.

Return values

<i>TEE_SUCCESS</i>	on success or error otherwise.
--------------------	--------------------------------

Example:

```
TEES_EnterCritical();
// Do some short actions
TEES_ExitCritical();
```

4.1.4.6 TEE_Result TEES_ExitCritical (void)

```
#include <tees_critical.h>
```

Enable routing and handling of normal world interrupts.

Return values

<i>TEE_SUCCESS</i>	on success or error otherwise.
--------------------	--------------------------------

Example:

```
TEES_EnterCritical();
// Do some short actions
TEES_ExitCritical();
```

4.1.4.7 TEE_Result TEES_GetRoT (ROOT_OF_TRUST * rot)

```
#include <tees_rot.h>
```

Get RoT information.

Function will be used to get RoT information from special SMC.

Parameters

in, out	<i>rot</i>	Pointer to get RoT information
---------	------------	--------------------------------

Return values

<i>TEE_SUCCESS</i>	no error.
<i>TEE_ERROR_*</i>	on failure

Example:

```
TEES_GetRoT(&rot);
```

4.1.4.8 TEE_Result TEES_LockHWCryptoBuf (void)

```
#include <tees_hwcrypto_buf.h>
```

Lock HW crypto buffer.

Lock HW crypto buffer for special driver TA

Return values

<i>TEE_SUCCESS</i>	on success
<i>TEE_ERROR_*</i>	on error <ul style="list-style-type: none"> • <i>TEE_ERROR_*</i> is based on error number of <code>open()</code> or <code>ioctl()</code> for crypto driver in secure kernel

Example:

```
...
TEE_Result res = TEES_LockHWCryptoBuf();
if (res != TEE_SUCCESS) {
    printf("TEES_LockHWCryptBuf() is failed. res = %d\n", res);
    return res;
}
...
```

4.1.4.9 TEE_Result TEES_SECCAM_GetStatus (unsigned int * data)

```
#include <tees_seccam.h>
```

Get a status of secure camera.

This function is used to check whether the camera is operated with normal or secure mode.

Parameters

out	<i>data</i>	Pointer to data for secure camera
-----	-------------	-----------------------------------

Return values

<i>TEE_SUCCESS</i>	no error.
<i>TEE_ERROR_*</i>	on failure <ul style="list-style-type: none"> • <i>TEE_ERROR_GENERIC</i> - Crypto driver or SMC for secure camera is not supported; • <i>TEE_ERROR_COMMUNICATION</i> - Unable to open the crypto driver;

Example:

```

...
TEE_Result res = TEE_SUCCESS;
unsigned int data;

res = TEES_SECCAM_GetStatus( &data );
if( res != TEE_SUCCESS ){
    printf("ERROR : TEES_SECCAM_GetStatus = %#x\n", res);
    return res;
}
if( data == 0x00 ){
    printf("Normal mode\n");
}
else if( data == 0x01 ){
    printf("Secure mode\n");
}
else{
    printf("undefined mode\n");
}
...

```

4.1.4.10 TEE_Result TEES_UnlockHWCryptoBuf (void)

```
#include <tees_hwcrypto_buf.h>
```

Unock HW crypto buffer.

Unlock HW crypto buffer for special driver TA

Return values

<i>TEE_SUCCESS</i>	on success
<i>TEE_ERROR_*</i>	on error <ul style="list-style-type: none"> • <i>TEE_ERROR_*</i> is based on error number of open() or ioctl() for crypto driver in secure kernel

Example:

```

...
TEE_Result res = TEES_UnlockHWCryptoBuf();
if( res != TEE_SUCCESS) {
    printf("TEES_UnlockHWCryptBuf() is failed. res = %d\n", res);
    return res;
}
...

```

4.1.4.11 TEE_Result TEES_UnwrapSecureObject (const unsigned char * in, uint32_t in_len, unsigned char * out, uint32_t * out_len)

```
#include <tees_secure_object.h>
```

Decrypt and verify wrapped data.

Function will take a buffer containing wrapped SO and decrypt it to a format understandable by the caller.

Parameters

in	<i>in</i>	Pointer to input buffer.
in	<i>in_len</i>	Length of input buffer.
out	<i>out</i>	Pointer to outdata. Can be set to NULL in combination with *out_len = 0 for getting required output buffer size.
in,out	<i>out_len</i>	Maximum/actual size of out buffer.

Return values

<i>TEE_SUCCESS</i>	data was successfully unwrapped.
<i>TEE_ERROR_XXXX</i>	- if unsuccessfully unwrapped.

Example:

```
TEES_UnwrapSecureObject((const unsigned char *)key1_str,
                        DATA256K,
                        wrapout,
                        &wrapout_len);
```

4.1.4.12 TEE_Result TEES_WrappedWithREK (WRAP_REK * data)

```
#include <tees_wrapped_with_rek.h>
```

Wrapping with REK.

Function will be used to wrap a data with REK by special SMC.

Parameters

in,out	<i>data</i>	Pointer to wrap/unwrap data with REK
--------	-------------	--------------------------------------

Return values

<i>TEE_SUCCESS</i>	no error.
<i>TEE_ERROR_*</i>	on failure

Example:

```
TEES_WrappedWithREK(&data);
```

4.1.4.13 TEE_Result TEES_WrapSecureObject (const unsigned char * in, uint32_t in_len, unsigned char * out, uint32_t * out_len, SO_AccessControllInfoType * ac)

```
#include <tees_secure_object.h>
```

Encrypt and sign input data.

Function will be used to create an encrypted or wrapped secure object from an unprotected data.

Parameters

in	<i>in</i>	Pointer to input buffer.
in	<i>in_len</i>	Input buffer length.
out	<i>out</i>	Pointer to outdata. Can be set to NULL in combination with *out_len = 0 for getting required output buffer size.
in,out	<i>out_len</i>	Maximum/actual size of out buffer.
in	<i>ac</i>	Pointer to Access Control struct SO_AccessControllInfoType. This is a structure containing access control information.

Return values

<i>TEE_SUCCESS</i>	data was successfully wrapped.
<i>TEE_ERROR_XXXX</i>	error occurred during wapping.

Example:

```
TEES_WrapSecureObject((const unsigned char *)key1_str,
                      DATA256K,
                      wrapout,
                      &wrapout_len,
                      &ac_info);
```

4.2 Loadable driver API

Data Structures

- struct `fops`
Structure that contains file operations callbacks supported by the driver. If user wants the driver to carry out some additional action, then handlers should be assigned to the appropriate callbacks. [More...](#)
- struct `usr_drv_info`
Structure that contains information describing the driver. [More...](#)

Typedefs

- typedef int `TEES_InterruptHandle`
- typedef void(* `TEES_DriverDestructor_t`) (int)

Functions

- int `TEES_DcacheFlush` (const void *addr, size_t nbytes)
Perform data cache flush.
- int `TEES_DcacheClean` (const void *addr, size_t nbytes)
Perform data cache clean.
- int `TEES_InitDriver` (char *name, struct `fops` *fops, unsigned int drvid, struct `usr_drv_info` **info)
Register user driver within namespace system under *name*, using mask of file operations *fops* and *drvid* of served asset.
- int `TEES_FiniDriver` (struct `usr_drv_info` *info)
Allow to release driver that was registered by using struct `usr_drv_info`.
- int `TEES_RegisterDriver` (char *name, struct `fops` *fops, unsigned int drvid, struct `usr_drv_info` **info) `_deprecated_`
Register user driver within namespace system under *name*, using mask of file operations *fops* and *drvid* of served asset.
- int `TEES_ReleaseDriver` (struct `usr_drv_info` **info) `_deprecated_`
Allow to release driver that was registered by using struct `usr_drv_info`.
- int `TEES_CompleteRequest` (struct `drv_info` *filp, long ret)
Complete deferred request(read, write, etc.) to driver.
- void * `TEES_AcquireUserBuffer` (struct `drv_info` *filp, uint64_t addr, const size_t size, int prot)
Get shared mapped area to use as buffer.
- int `TEES_ReleaseUserBuffer` (const void *addr, const size_t size)
Deletes the shared mapped area for the specified address range.
- TEE_Result `TEES_AllocateInterrupt` (int nr, intruhandler handler, `TEES_InterruptHandle` *handle)
This function is used to allocate interrupt and register user handler.
- TEE_Result `TEES_ReleaseInterrupt` (`TEES_InterruptHandle` handle)
This function is used to release interrupt.
- TEE_Result `TEES_GenerateInterrupt` (`TEES_InterruptHandle` handle)
This function is used to generate interrupt.
- TEE_Result `TEES_WaitForInterrupt` (`TEES_InterruptHandle` handle, uint32_t timeout)
This function is used to wait for interrupt.
- TEE_Result `TEES_CompleteInterrupt` (`TEES_InterruptHandle` handle)
This function is used to notify about interrupt arrival.
- TEE_Result `TEES_RegisterDriverDestructor` (`TEES_DriverDestructor_t` destr)
Initializes driver destructor function pointer. The destructor will be called when secure kernel generates any interrupting signal.

4.2.1 Detailed Description

Provides set of function to interact with device drivers from userspace comonents.

4.2.2 Data Structure Documentation

4.2.2.1 struct fops

Structure that contains file operations callbacks supported by the driver. If user wants the driver to carry out some additional action, then handlers should be assigned to the appropriate callbacks.

Data Fields

- `int(* open)(struct drv_info *filp, const char *subpath,...)`
- `int(* close)(struct drv_info *filp)`
- `int(* truncate)(struct drv_info *filp, unsigned int size)`
- `ssize_t(* read)(struct drv_info *filp, void *buf, size_t len)`
- `ssize_t(* write)(struct drv_info *filp, void *buf, size_t len)`
- `int(* ioctl)(struct drv_info *filp, int cmd, unsigned long data)`
- `int(* stat)(struct drv_info *filp, struct stat *buf)`
- `void *(* mmap)(struct drv_info *filp, void *addr, size_t len, int prot, int flags, off_t offset)`
- `int(* unlink)(const char *subpath)`
- `int(* lseek)(struct drv_info *filp, int offset, int whence)`

Field Documentation

`int(* fops::close)(struct drv_info *filp)`

callback for close operation

`int(* fops::ioctl)(struct drv_info *filp, int cmd, unsigned long data)`

callback for ioctl operation

`int(* fops::lseek)(struct drv_info *filp, int offset, int whence)`

callback for seek operation

`void *(* fops::mmap)(struct drv_info *filp, void *addr, size_t len, int prot, int flags, off_t offset)`

callback for mmap operation

`int(* fops::open)(struct drv_info *filp, const char *subpath,...)`

callback for open operation

ssize_t(* fops::read) (struct drv_info *filp, void *buf, size_t len)

callback for read operation

int(* fops::stat) (struct drv_info *filp, struct stat *buf)

callback for stat operation

int(* fops::truncate) (struct drv_info *filp, unsigned int size)

callback for truncate operation

int(* fops::unlink) (const char *subpath)

callback for unlink operation

ssize_t(* fops::write) (struct drv_info *filp, void *buf, size_t len)

callback for write operation

4.2.2.2 struct usr_drv_info

Structure that contains information describing the driver.

Data Fields

struct fops *	fops	See also fops
int	handle	driver handle

4.2.3 Typedef Documentation

4.2.3.1 TEES_DriverDestructor_t

#include <[tee_ta_destructor.h](#)>

Pointer to driver destructor function. int parameter - is an interrupting signal value

4.2.3.2 TEES_InterruptHandle

#include <[tee_interrupt.h](#)>

interrupt handle abstraction

4.2.4 Function Documentation

4.2.4.1 void* TEES_AcquireUserBuffer (struct drv_info * filp, uint64_t addr, const size_t size, int prot)

```
#include <driver.h>
```

Get shared mapped area to use as buffer.

Parameters

in	<i>filp</i>	Pointer to struct drv_info related to driver: struct drv_info { int pid; int fd; int share_fd; struct uuid uuid; char profile_name[PROFILE_STR_LEN]; };
in	<i>addr</i>	Offset in the file (or other object) referred to by the file descriptor filp->share_fd
in	<i>size</i>	The length of the buffer
in	<i>prot</i>	Desired memory protection of the mapping

Returns

Pointer to userspace buffer

Example:

```
static int driver_read(struct drv_info *info, char *data, unsigned int data_len)
{
    char *vaddr = NULL;

    // Always return "Read"
    vaddr = TEES_AcquireUserBuffer(info, data, data_len,
        PROT_READ | PROT_WRITE);
    strncpy(vaddr, "Read", data_len);
    TEES_ReleaseUserBuffer(vaddr, data_len);

    return data_len;
}
```

4.2.4.2 TEE_Result TEES_AllocateInterrupt (int nr, intruhandler handler, TEES_InterruptHandle * handle)

```
#include <tee_interrupt.h>
```

This function is used to allocate interrupt and register user handler.

Parameters

in	<i>nr</i>	An interrupt id.
in	<i>handler</i>	An interrupt handler.
out	<i>handle</i>	An interrupt handle.

Return values

<code>TEE_SUCCESS</code>	on success.
<code>TEE_ERROR</code>	on fail.

Example:

```

#define NUM_IRQS_GENERATED (10)
static TEE_InterruptHandle handle;

void handler(struct intrinfo *info)
{
    const int error = errno;

    if (info->nr == IRQ) {
        interrupt_counter++;
        // printf_no_alloc() used here due to printf() can sleep in allocator
        // but sleeping functions are prohibited in signal handlers
        printf_no_alloc("IRQ: nr = %d, counter = %d\n", info->nr, interrupt_counter);
        TEE_CompleteInterrupt(handle);
    } else {
        printf_no_alloc("ERROR:UNKNOWN IRQ: nr = %d\n", info->nr);
        TEE_Panic(0);
    }
    errno = error;
}

static int test_interrupt_allocate(TEE_Param *param)
{
    (void)param;
    TEE_Result ret;
    ret = TEE_AllocateInterrupt(IRQ, &handle, handler);
    if (ret != TEE_SUCCESS) {
        return -1;
    }
    return 0;
}

```

4.2.4.3 TEE_Result TEE_CompleteInterrupt (TEE_InterruptHandle *handle*)

```
#include <tee_interrupt.h>
```

This function is used to notify about interrupt arrival.

Parameters

in	<i>handle</i>	An interrupt handle.
----	---------------	----------------------

Return values

<code>TEE_SUCCESS</code>	on success.
<code>TEE_ERROR</code>	on failure.

Example:

```

#define NUM_IRQS_GENERATED (10)
static TEE_InterruptHandle handle;

void handler(struct intrinfo *info)
{
    const int error = errno;

    if (info->nr == IRQ) {
        interrupt_counter++;
        // printf_no_alloc() used here due to printf() can sleep in allocator
        // but sleeping functions are prohibited in signal handlers
        printf_no_alloc("IRQ: nr = %d, counter = %d\n", info->nr, interrupt_counter);
        TEE_CompleteInterrupt(handle);
    } else {
        printf_no_alloc("ERROR:UNKNOWN IRQ: nr = %d\n", info->nr);
        TEE_Panic(0);
    }
    errno = error;
}

static int test_interrupt_generate(TEE_Param *param)
{
    (void)param;
    int i = 0;

    while (i < NUM_IRQS_GENERATED) {
        TEE_GenerateInterrupt(handle);
        if (TEE_WaitForInterrupt(handle, 0)) {
            printf("Failed test TEE_WaitForInterrupt() call at line %d.\n"
                "errno = %d\n", __LINE__, errno);
            return -1;
        }
        i++;
    }
    return 0;
}

```

4.2.4.4 int TEE_CompleteRequest (struct drv_info * *filp*, long *ret*)

```
#include <driver.h>
```

Complete deferred request(read, write, etc.) to driver.

Parameters

in	<i>filp</i>	Pointer to structure related to driver: struct drv_info { int pid; int fd; int share_fd; struct uuid uuid; char profile_name[PROFILE_STR_LEN]; };
in	<i>ret</i>	Ret value that should be delivered to the driver's client.

Return values

0	no error.
-1	on failure. <code>errno</code> variable is set appropriately.

Example:

```

static struct drv_info filp_write;
static struct drv_info filp_read;
static bool pending_write = false;
static bool pending_read = false;
static int current_data;
static char *current_vaddr;

static void complete_read_write(void)
{
    // Wait to receive both read and write requests
    if (pending_read && pending_write) {
        pending_read = false;
        pending_write = false;

        *(int *)current_vaddr = current_data;
        TEES_ReleaseUserBuffer(current_vaddr, 4);

        TEES_CompleteRequest(&filp_read, 0);
        TEES_CompleteRequest(&filp_write, 0);
    }
}

```

4.2.4.5 int TEES_DcacheClean (const void * *addr*, size_t *nbytes*)

```
#include <cache.h>
```

Perform data cache clean.

Parameters

in	<i>addr</i>	Start address pointer.
in	<i>nbytes</i>	Size of the region.

Return values

0	no error.
-1	on failure. errno variable is set appropriately.

4.2.4.6 int TEES_DcacheFlush (const void * *addr*, size_t *nbytes*)

```
#include <cache.h>
```

Perform data cache flush.

Parameters

in	<i>addr</i>	Start address pointer.
in	<i>nbytes</i>	Size of the region.

Return values

0	no error.
-1	on failure. errno variable is set appropriately.

4.2.4.7 int TEES_FiniDriver (struct usr_drv_info * info)

```
#include <driver.h>
```

Allow to release driver that was registered by using struct `usr_drv_info`.

Parameters

in	<i>info</i>	A pointer to struct <code>usr_drv_info</code> that contains information describing the driver.
----	-------------	--

Return values

<i>non-negative</i>	driver handle.
-1	on failure. <code>errno</code> variable is set appropriately.

Example:

```
ret = TEES_FiniDriver(my_driver_info);
if (ret) {
    printf("release_driver failed, ret = %d\n", ret);
}
```

4.2.4.8 TEE_Result TEES_GenerateInterrupt (TEE_InterruptHandle handle)

```
#include <tee_interrupt.h>
```

This function is used to generate interrupt.

Parameters

in	<i>handle</i>	An interrupt handle.
----	---------------	----------------------

Return values

<i>TEE_SUCCESS</i>	on success.
<i>TEE_ERROR</i>	on failure.

Example:

```

#define NUM_IRQS_GENERATED (10)
static TEE_InterruptHandle handle;

void handler(struct intrinfo *info)
{
    const int error = errno;

    if (info->nr == IRQ) {
        interrupt_counter++;
        // printf_no_alloc() used here due to printf() can sleep in allocator
        // but sleeping functions are prohibited in signal handlers
        printf_no_alloc("IRQ: nr = %d, counter = %d\n", info->nr, interrupt_counter);
        TEE_CompleteInterrupt(handle);
    } else {
        printf_no_alloc("ERROR:UNKNOWN IRQ: nr = %d\n", info->nr);
        TEE_Panic(0);
    }
    errno = error;
}

static int test_interrupt_generate(TEE_Param *param)
{
    (void)param;
    int i = 0;

    while (i < NUM_IRQS_GENERATED) {
        TEE_GenerateInterrupt(handle);
        if (TEE_WaitForInterrupt(irq_fd, 0) != TEE_SUCCESS) {
            return -1;
        }
        i++;
    }
    return 0;
}

```

4.2.4.9 int TEE_InitDriver (char * *name*, struct fops * *fops*, unsigned int *drvid*, struct usr_drv_info ** *info*)

```
#include <driver.h>
```

Register user driver within namespace system under *name*, using mask of file operations *fops* and *drvid* of served asset.

Parameters

in	<i>name</i>	A pointer to an array storing driver name
in	<i>fops</i>	A pointer to struct fops containing file operations supported by driver
in	<i>drvid</i>	Identifier of driver asset, used by access control
out	<i>info</i>	Double pointer to struct usr_drv_info that contains information describing the driver

Return values

0	no error.
-1	on failure. errno variable is set appropriately.

Example:

```

int ret;
char my_driver_name[] = "dev://my_driver";
struct usr_drv_info *my_driver_info = NULL;
struct fops file_ops = {0};

file_ops.open = drv_open;
file_ops.close = drv_close;
file_ops.read = drv_read;
file_ops.write = drv_write;

ret = TEES_InitDriver(my_driver_name, &file_ops, ACC_PERM_I2C, &my_driver_info);
if (ret) {
    printf("register_driver failed, ret = %d\n", ret);
    return TEE_ERROR_GENERIC;
}

```

4.2.4.10 int TEES_RegisterDriver (char * name, struct fops * fops, unsigned int drvid, struct usr_drv_info ** info)

```
#include <driver.h>
```

Register user driver within namespace system under *name*, using mask of file operations *fops* and *drvid* of served asset.

Deprecated Use [TEES_InitDriver\(\)](#) instead

Parameters

in	<i>name</i>	A pointer to an array storing driver name
in	<i>fops</i>	A pointer to struct fops containing file operations supported by driver
in	<i>drvid</i>	Identificator of driver asset, used by access control
out	<i>info</i>	Double pointer to struct usr_drv_info that contains information describing the driver

Return values

0	no error.
-1	on failure. errno variable is set appropriately.

Example:

```

int ret;
char my_driver_name[] = "dev://my_driver";
struct usr_drv_info *my_driver_info = NULL;
struct fops file_ops = {0};

file_ops.open = drv_open;
file_ops.close = drv_close;
file_ops.read = drv_read;
file_ops.write = drv_write;

ret = TEES_RegisterDriver(my_driver_name, &file_ops, ACC_PERM_I2C, &my_driver_info);
if (ret) {
    printf("register_driver failed, ret = %d\n", ret);
    return TEE_ERROR_GENERIC;
}

```

4.2.4.11 TEE_Result TEES_RegisterDriverDestructor (TEES_DriverDestructor_t destr)

```
#include <tee_ta_destructor.h>
```

Initializes driver destructor function pointer. The destructor will be called when secure kernel generates any interrupting signal.

Parameters

in	<i>destr</i>	- pointer to driver destructor function.
----	--------------	--

Return values

<i>TEE_ERROR_BAD_PARAMETERS</i>	when <i>destr</i> is NULL
<i>TEE_SUCCESS</i>	all other cases

4.2.4.12 int TEES_ReleaseDriver (struct usr_drv_info ** info)

```
#include <driver.h>
```

Allow to release driver that was registered by using struct [usr_drv_info](#).

Deprecated Use [TEES_FiniDriver\(\)](#) instead

Parameters

in	<i>info</i>	Double pointer to struct usr_drv_info that contains information describing the driver.
----	-------------	--

Return values

<i>non-negative</i>	driver handle.
<i>-1</i>	on failure. errno variable is set appropriately.

Example:

```
ret = TEES_ReleaseDriver(my_driver_info);
if (ret) {
    printf("release_driver failed, ret = %d\n", ret);
}
```

4.2.4.13 TEE_Result TEES_ReleaseInterrupt (TEES_InterruptHandle *handle*)

```
#include <tee_interrupt.h>
```

This function is used to release interrupt.

Parameters

in	<i>handle</i>	An interrupt handle.
----	---------------	----------------------

Return values

<i>TEE_SUCCESS</i>	on success.
<i>TEE_ERROR</i>	on failure.

Example:

```
static TEES_InterruptHandle handle;

static int test_interrupt_release(TEE_Param *param)
{
    (void)param;
    if (TEES_ReleaseInterrupt(handle) != TEE_SUCCESS) {
        return -1;
    }
    return 0;
}
```

4.2.4.14 int TEES_ReleaseUserBuffer (const void * *addr*, const size_t *size*)

```
#include <driver.h>
```

Deletes the shared mapped area for the specified address range.

Parameters

in	<i>addr</i>	Starting address of releasing buffer
in	<i>size</i>	The length of the mapping

Return values

0	no error.
-1	on failure. errno variable is set appropriately.

Example:

```

static int driver_read(struct drv_info *info, char *data, unsigned int data_len)
{
    char *vaddr = NULL;

    // Always return "Read"
    vaddr = TEE_AcquireUserBuffer(info, data, data_len,
        PROT_READ | PROT_WRITE);
    strncpy(vaddr, "Read", data_len);
    TEE_ReleaseUserBuffer(vaddr, data_len);

    return data_len;
}

```

4.2.4.15 TEE_Result TEE_WaitForInterrupt (TEE_InterruptHandle *handle*, uint32_t *timeout*)

#include <tee_interrupt.h>

This function is used to wait for interrupt.

Parameters

in	<i>handle</i>	An interrupt handle.
in	<i>timeout</i>	waiting timeout.

Return values

<i>TEE_SUCCESS</i>	on success.
<i>TEE_ERROR</i>	on failure.

Example:

```

#define NUM_IRQS_GENERATED (10)
static TEE_InterruptHandle handle;

void handler(struct intrinfo *info)
{
    const int error = errno;

    if (info->nr == IRQ) {
        interrupt_counter++;
        // printf_no_alloc() used here due to printf() can sleep in allocator
        // but sleeping functions are prohibited in signal handlers
        printf_no_alloc("IRQ: nr = %d, counter = %d\n", info->nr, interrupt_counter);
        TEE_CompleteInterrupt(handle);
    } else {
        printf_no_alloc("ERROR:UNKNOWN IRQ: nr = %d\n", info->nr);
        TEE_Panic(0);
    }
    errno = error;
}

static int test_interrupt_generate(TEE_Param *param)
{
    (void)param;
    int i = 0;

    while (i < NUM_IRQS_GENERATED) {
        TEE_GenerateInterrupt(handle);
        if (TEE_WaitForInterrupt(handle, NULL) != TEE_SUCCESS) {
            printf("Failed test TEE_WaitForInterrupt() call at line %d.\n"
                "errno = %d\n", __LINE__, errno);
            return -1;
        }
        i++;
    }
    return 0;
}

```

4.3 Custom handler API

Provides set of function to handle requests from rich world OS kernel.

4.4 Contiguous memory API

Typedefs

- typedef struct __TEES_ContiguousMemoryHandle * [TEES_ContiguousMemoryHandle](#)

Enumerations

- enum [TEES_ContiguousAllocateFlags](#) { [TEES_CONTIGUOUS_FLAG_ZERO_ON_FREE](#) = (1U << 0), [TEES_CONTIGUOUS_FLAG_ZERO_ON_ALLOC](#) = (1U << 1) }
Determines the desired allocated memory handle properties.
- enum [TEES_ContiguousMapFlags](#) { [TEES_CONTIGUOUS_MAP_READ](#) = (1U << 0), [TEES_CONTIGUOUS_MAP_WRITE](#) = (1U << 1), [TEES_CONTIGUOUS_MAP_NON_CACHEABLE](#) = (1U << 2), [TEES_CONTIGUOUS_MAP_FIXED](#) = (1U << 3), [TEES_CONTIGUOUS_MAP_POPULATE](#) = (1U << 4) }
Determines the desired mapping properties.

Functions

- TEE_Result [TEES_AllocateContiguousMemory](#) (const char *region, size_t size, size_t align, uint32_t flags, [TEES_ContiguousMemoryHandle](#) *memory)
Allocates physically contiguous memory buffer handle of the specified region type.
- void [TEES_ReleaseContiguousMemory](#) ([TEES_ContiguousMemoryHandle](#) memory)
Releases physically contiguous memory buffer that was previously allocated by [TEES_AllocateContiguousMemory](#).
- TEE_Result [TEES_MapContiguousMemory](#) ([TEES_ContiguousMemoryHandle](#) memory, void **addr, uint32_t flags)
Maps physically contiguous memory buffer previously allocated by [TEES_AllocateContiguousMemory](#) into the address space of current process.
- void [TEES_UnmapContiguousMemory](#) (void *addr)
Unmaps physically contiguous memory buffer that was previously mapped by [TEES_MapContiguousMemory](#).

4.4.1 Detailed Description

Provides set of functions to manipulate secure contiguous memory allocator.

4.4.2 Typedef Documentation

4.4.2.1 [TEES_ContiguousMemoryHandle](#)

```
#include <scma.h>
```

SCMA handle abstraction

4.4.3 Enumeration Type Documentation

4.4.3.1 enum TEES_ContiguousAllocateFlags

```
#include <scma.h>
```

Determines the desired allocated memory handle properties.

Enumerator

TEES_CONTIGUOUS_FLAG_ZERO_ON_FREE Allocated memory will be zeroed on free.
TEES_CONTIGUOUS_FLAG_ZERO_ON_ALLOC Allocated memory will be zeroed on allocate.

4.4.3.2 enum TEES_ContiguousMapFlags

```
#include <scma.h>
```

Determines the desired mapping properties.

Enumerator

TEES_CONTIGUOUS_MAP_READ Map memory with read permission.
TEES_CONTIGUOUS_MAP_WRITE Map memory with write permission.
TEES_CONTIGUOUS_MAP_NON_CACHEABLE Map memory as non-cacheable.
TEES_CONTIGUOUS_MAP_FIXED Don't interpret addr as a hint: place the mapping at exactly that address. addr must be a multiple of alignment (maximum of alignment specified in region properties and allocate function). If addr is not aligned or addr + size (specified on alloc) overlaps existing mapping function fails. If this flag is passed and addr is NULL function fails.
TEES_CONTIGUOUS_MAP_POPULATE Do not use lazy mapping.

4.4.4 Function Documentation

4.4.4.1 TEE_Result TEES_AllocateContiguousMemory (const char * *region*, size_t *size*, size_t *align*, uint32_t *flags*, TEES_ContiguousMemoryHandle * *memory*)

```
#include <scma.h>
```

Allocates physically contiguous memory buffer handle of the specified region type.

Parameters

in	<i>region</i>	A pointer to the zero-terminated string containing region type name of the contiguous memory region to allocate buffer from.
in	<i>size</i>	The size in bytes of the buffer to be allocated.
in	<i>align</i>	The minimal alignment requirement of starting address and size in bytes. Must be the power of 2.
in	<i>flags</i>	Determines the desired allocated memory handle properties. Containing zero or more bitwise ORed flags from the following list: TEES_CONTIGUOUS_FLAG_ZERO_ON_FREE: allocated memory will be zeroed on free. TEES_CONTIGUOUS_FLAG_ZERO_ON_ALLOC: allocated memory will be zeroed on allocate.
out	<i>memory</i>	Filled with a handle on the allocated memory.

Return values

<i>TEE_SUCCESS</i>	in case of success
<i>TEE_ERROR_ACCESS_DENIED</i>	If this API is prohibited for use by current TA.
<i>TEE_ERROR_ITEM_NOT_FOUND</i>	If the region is not found or access to that region is denied for this TA.
<i>TEE_ERROR_OUT_OF_MEMORY</i>	If there are not enough contiguous memory of the requested region type.
<i>TEE_ERROR_ACCESS_CONFLICT</i>	If requested flags are incompatible with flags specified in region configuration.
<i>TEE_ERROR_BAD_PARAMETERS</i>	If requested size is not Page-aligned, if align is not power of 2 or if called with unknown flags.

Example:

```

static TEE_ContiguousMemoryHandle handle;

static int test_allocate_contiguous_memory(void)
{
    TEE_Result ret;
    const size_t size = 1024 * 1024;
    const size_t align = 0x1000;
    const uint32_t flags = TEE_CONTIGUOUS_FLAG_ZERO_ON_ALLOC;

    ret = TEE_AllocateContiguousMemory("region_name", size, align, flags, &
        handle);
    if (ret != TEE_SUCCESS) {
        return -1;
    }
    return 0;
}

```

4.4.4.2 TEE_Result TEE_MapContiguousMemory (TEE_ContiguousMemoryHandle memory, void ** addr, uint32_t flags)

```
#include <scma.h>
```

Maps physically contiguous memory buffer previously allocated by TEE_AllocateContiguousMemory into the address space of current process.

Parameters

in	<i>memory</i>	A handle on the previously allocated memory.
in,out	<i>addr</i>	On input filled with a hint address at which memory is to be mapped. If filled by NULL address is chosen automatically. If function success on output contains address of the mapped area.
in	<i>flags</i>	Determines the desired mapping properties. Containing at least TEEES_CONTIGUOUS_MAP_READ or TEEES_CONTIGUOUS_MAP_WRITE and the bitwise OR of zero or more of the following flags: TEEES_CONTIGUOUS_MAP_NON_CACHEABLE: map memory as non-cacheable. TEEES_CONTIGUOUS_MAP_READ: map memory with read permission. TEEES_CONTIGUOUS_MAP_WRITE: map memory with write permission. TEEES_CONTIGUOUS_MAP_FIXED: Don't interpret <i>addr</i> as a hint: place the mapping at exactly that address. <i>addr</i> must be a multiple of alignment (maximum of alignment specified in region properties and allocate function). If <i>addr</i> is not aligned or <i>addr</i> + size (specified on alloc) overlaps existing mapping function fails. If this flag is passed and <i>addr</i> is NULL function fails. TEEES_CONTIGUOUS_MAP_POPULATE: do not use lazy mapping.

Return values

<i>TEE_SUCCESS</i>	in case of success
<i>TEE_ERROR_OUT_OF_MEMORY</i>	If there are not enough virtual memory to make mapping or it can't be mapped at fixed hint.
<i>TEE_ERROR_ACCESS_CONFLICT</i>	If requested flags are incompatible with flags specified in region configuration or during alloc.
<i>TEE_ERROR_BAD_PARAMETERS</i>	If requested flags don't have TEEES_CONTIGUOUS_MAP_READ or TEEES_CONTIGUOUS_MAP_WRITE or contains unknown bits set.

Example:

```

static TEEES_ContiguousMemoryHandle handle;
static void *addr;

static int test_map_contiguous_memory(void)
{
    TEE_Result ret;
    const uint32_t flags = TEEES_CONTIGUOUS_MAP_READ;

    ret = TEEES_MapContiguousMemory(handle, &addr, flags);
    if (ret != TEE_SUCCESS) {
        return -1;
    }
    return 0;
}

```

4.4.4.3 void TEEES_ReleaseContiguousMemory (TEEES_ContiguousMemoryHandle *memory*)

```
#include <scma.h>
```

Releases physically contiguous memory buffer that was previously allocated by TEEES_AllocateContiguousMemory.

Parameters

in	<i>memory</i>	Handle of the allocated memory to release.
----	---------------	--

Example:

```
static TEES_ContiguousMemoryHandle handle;

static void test_release_contiguous_memory(void)
{
    TEES_ReleaseContiguousMemory(handle);
}
```

4.4.4.4 void TEES_UnmapContiguousMemory (void * *addr*)

```
#include <scma.h>
```

Unmaps physically contiguous memory buffer that was previously mapped by TEES_MapContiguousMemory.

Parameters

in	<i>addr</i>	Address at which contiguous memory buffer was mapped.
----	-------------	---

Example:

```
static void *addr;

static void test_unmap_contiguous_memory(void)
{
    TEES_UnmapContiguousMemory(addr);
}
```

4.5 SPI API

Data Structures

- struct [TEES_SPIConfig](#)
Configuration of SPI device. [More...](#)
- struct [TEES_SPITransfer](#)
Descriptor to transfer data over SPI. [More...](#)

Typedefs

- typedef struct [__TEES_SPIHandlerImpl](#) * [TEES_SPIHandler](#)

Functions

- TEE_Result [TEES_SPIDMAInit](#) (uintptr_t address)
Initialize DMA for working with SPI device.
- TEE_Result [TEES_SPIInit](#) (uint32_t port, const [TEES_SPIConfig](#) *cfg, [TEES_SPIHandler](#) *handler)
Initialize handler fields.
- TEE_Result [TEES_SPIExit](#) ([TEES_SPIHandler](#) handler)
Free handler resource.
- TEE_Result [TEES_SPISetConfig](#) ([TEES_SPIHandler](#) handler, const [TEES_SPIConfig](#) *cfg)
Initialize handler with configuration values.
- TEE_Result [TEES_SPISetClockSpeed](#) ([TEES_SPIHandler](#) handler, uint32_t speedHz)
Set clock frequency rate.
- TEE_Result [TEES_SPISetBitsPerWord](#) ([TEES_SPIHandler](#) handler, uint8_t bitsPerWord)
Set the number of bits that will be transferred per SPI rate.
- TEE_Result [TEES_SPISetDMAMode](#) ([TEES_SPIHandler](#) handler, bool isEnabled)
Enable or disable DMA mode.
- TEE_Result [TEES_SPISetCPOL](#) ([TEES_SPIHandler](#) handler, uint8_t cpol)
Set SPI clock polarity bit.
- TEE_Result [TEES_SPISetCPHA](#) ([TEES_SPIHandler](#) handler, uint8_t cpha)
Set SPI clock phase bit.
- TEE_Result [TEES_SPIClockEnable](#) ([TEES_SPIHandler](#) handler)
Not implemented.
- TEE_Result [TEES_SPIClockDisable](#) ([TEES_SPIHandler](#) handler)
Not implemented.
- TEE_Result [TEES_SPIWrite](#) ([TEES_SPIHandler](#) handler, [TEES_SPITransfer](#) *tx)
Transfer data from buffer bus to SPI.
- TEE_Result [TEES_SPIRead](#) ([TEES_SPIHandler](#) handler, [TEES_SPITransfer](#) *rx)
Transfer data from SPI bus to buffer.
- TEE_Result [TEES_SPIWriteRead](#) ([TEES_SPIHandler](#) handler, [TEES_SPITransfer](#) *tx, [TEES_SPITransfer](#) *rx)
Transfer data from buffer to SPI bus.

4.5.1 Detailed Description

Provides set of functions to manipulate device connected with SPI bus.

4.5.2 Data Structure Documentation

4.5.2.1 struct TEES_SPIConfig

Configuration of SPI device.

Data Fields

uint8_t	bitsPerWord	Word size used to transfer data.
uint8_t	CPHA	Clock phase.
uint8_t	CPOL	Clock polarity.
bool	isDMAMode	Use DMA with SPI device for each transfer.
bool	manualClockControl	SPI clocks are controlled manually. SPI driver shouldn't enable/disable clocks for transfers.
uint32_t	speedHz	Clock rate to be used with SPI device for each transfer.

4.5.2.2 struct TEES_SPITransfer

Descriptor to transfer data over SPI.

Data Fields

void *	bufAddr	Buffer to be transmitted or received over SPI.
size_t	bufLen	Size of buffer in bytes.
size_t	transferredLen	Actual transferred size in bytes (for client information).

4.5.3 Typedef Documentation

4.5.3.1 TEES_SPIHandler

```
#include <tee_spi.h>
```

SPI handler structure

4.5.4 Function Documentation

4.5.4.1 TEE_Result TEES_SPIClockDisable (TEES_SPIHandler *handler*)

```
#include <tee_spi.h>
```

Not implemented.

Parameters

in	<i>handler</i>	SPI handler which contains device descriptor.
----	----------------	---

Returns

TEE_SUCCESS - In case of success.

4.5.4.2 TEE_Result TEES_SPIClockEnable (TEES_SPIHandler *handler*)

```
#include <tee_spi.h>
```

Not implemented.

Parameters

in	<i>handler</i>	SPI handler which contains device descriptor.
----	----------------	---

Returns

TEE_SUCCESS - In case of success.

4.5.4.3 TEE_Result TEES_SPIDMAInit (uintptr_t *address*)

```
#include <tee_spi.h>
```

Initialize DMA for working with SPI device.

Parameters

in	<i>address</i>	DMA address base.
----	----------------	-------------------

Returns

TEE_SUCCESS - In case of success.

4.5.4.4 TEE_Result TEES_SPIExit (TEES_SPIHandler *handler*)

```
#include <tee_spi.h>
```

Free handler resource.

Parameters

in	<i>handler</i>	SPI handler which contains device descriptor.
----	----------------	---

Returns

TEE_SUCCESS - In case of success.

Example:

```

...
TEES_SPIHandler handler;
TEES_SPIConfig cfg = {
    .speedHz = 5000000,
    .CPOL = 0,
    .CPHA = 0,
    .bitsPerWord = SPI_TYPE_WORD,
    .isDMAMode = true,
};

TEES_SPIDMAInit(pa);

TEE_Result res = TEES_SPIInit(port, NULL, &handler);
if (res != TEE_SUCCESS) {
    printf("TEST: Failed to initialize SPI, tee_err: %#x\n", res);
    return res;
}

res = TEES_SPISetConfig(handler, &cfg);
if (res != TEE_SUCCESS) {
    printf("TEST: Failed to configure SPI, tee_err: %#x\n", res);
    TEES_SPIExit(handler);
    return res;
}
...

```

4.5.4.5 TEE_Result TEES_SPIInit (uint32_t port, const TEES_SPIConfig * cfg, TEES_SPIHandler * handler)

```
#include <tee_spi.h>
```

Initialize handler fields.

Parameters

in	<i>port</i>	Number of SPI controller.
in	<i>cfg</i>	Pointer to SPI configuration for port initialisation.
out	<i>handler</i>	Pointer to SPI handler which contains device descriptor.

Return values

<i>TEE_SUCCESS</i>	on success.
<i>TEE_ERROR_OUT_OF_MEMORY</i>	on failure.

Example:

```

TEES_SPIHandler handler;
TEES_SPIConfig cfg = {
    .speedHz = 5000000,
    .CPOL = 0,
    .CPHA = 0,
    .bitsPerWord = SPI_TYPE_WORD,
    .isDMAMode = true,
};
unsigned long pa = 0;
unsigned int port = 0;
TEES_SPIDMAInit(pa);

TEE_Result res = TEES_SPIInit(port, NULL, &handler);
if (res != TEE_SUCCESS) {
    printf("TEST: Failed to initialize SPI, tee_err: %#x\n", res);
    return res;
}

```

4.5.4.6 TEE_Result TEES_SPIRead (TEES_SPIHandler handler, TEES_SPITransfer * rx)

```
#include <tee_spi.h>
```

Transfer data from SPI bus to buffer.

Parameters

in	<i>handler</i>	SPI handler which contains device descriptor.
out	<i>rx</i>	Pointer to buffer to store data.

Return values

<i>TEE_SUCCESS</i>	on success.
<i>TEE_ERROR_GENERIC</i>	on failure.

Example:

```

TEES_SPIHandler handler;
TEES_SPIConfig cfg = {
    .speedHz = 5000000,
    .CPOL = 0,
    .CPHA = 0,
    .bitsPerWord = SPI_TYPE_WORD,
    .isDMAMode = true
};

TEES_SPIDMAInit(pa);

TEE_Result res = TEES_SPIInit(port, &cfg, &handler);
if (res != TEE_SUCCESS) {
    printf("TEST: Failed to initialize SPI, tee_err: %#x\n", res);
    return res;
}

TEES_SPITransfer rx = {
    .bufAddr = (void *) (pa + DMA_RX_OFFSET),
    .bufLen = buflen,
    .transferredLen = 0
};

res = TEES_SPIRead(handler, &rx);
if (TEES_SPIExit(handler)) {
    TEE_Panic(0);
}

if (rx.transferredLen == 0) {
    printf("TEST: nothing was transferred over SPI, err: %#x\n", res);
    return TEE_ERROR_GENERIC;
}

```

4.5.4.7 TEE_Result TEES_SPISetBitsPerWord (TEES_SPIHandler *handler*, uint8_t *bitsPerWord*)

```
#include <tee_spi.h>
```

Set the number of bits that will be transferred per SPI rate.

Parameters

in	<i>handler</i>	SPI handler which contains device descriptor.
in	<i>bitsPerWord</i>	bits per word.

Returns

TEE_SUCCESS - In case of success.

Example:

```
...
TEES_SPIHandler spi;
TEES_SPISetBitsPerWord(spi, 0);
...
```

4.5.4.8 TEE_Result TEES_SPISetClockSpeed (TEES_SPIHandler *handler*, uint32_t *speedHz*)

```
#include <tee_spi.h>
```

Set clock frequency rate.

Parameters

in	<i>handler</i>	SPI handler which contains device descriptor.
in	<i>speedHz</i>	Value of frequency rate.

Returns

TEE_SUCCESS - In case of success.

Example:

```
...
TEES_SPIHandler spi;
TEES_SPISetClockSpeed(spi, 0);
...
```

4.5.4.9 TEE_Result TEES_SPISetConfig (TEES_SPIHandler *handler*, const TEES_SPIConfig * *cfg*)

#include <tee_spi.h>

Initialize handler with configuration values.

Parameters

in	<i>handler</i>	SPI handler which contains device descriptor.
in	<i>cfg</i>	Pointer to SPI configuration for handler initialisation.

Returns

TEE_SUCCESS - In case of success.

Example:

```

...
TEES_SPIHandler handler;
TEES_SPIConfig cfg = {
    .speedHz = 5000000,
    .CPOL = 0,
    .CPHA = 0,
    .bitsPerWord = SPI_TYPE_WORD,
    .isDMAMode = true,
};

TEES_SPIDMAInit(pa);

TEE_Result res = TEES_SPIInit(port, NULL, &handler);
if (res != TEE_SUCCESS) {
    printf("TEST: Failed to initialize SPI, tee_err: %#x\n", res);
    return res;
}

res = TEES_SPISetConfig(handler, &cfg);
if (res != TEE_SUCCESS) {
    printf("TEST: Failed to configure SPI, tee_err: %#x\n", res);
    TEES_SPIExit(handler);
    return res;
}
...

```

4.5.4.10 TEE_Result TEES_SPISetCPHA (TEES_SPIHandler *handler*, uint8_t *cpha*)

#include <tee_spi.h>

Set SPI clock phase bit.

Parameters

in	<i>handler</i>	SPI handler which contains device descriptor.
in	<i>cpha</i>	Value of phase bit.

Returns

TEE_SUCCESS - In case of success.

Example:

```

...
TEES_SPIHandler spi;
TEES_SPISetCPHA(spi, 0);
...

```

4.5.4.11 TEE_Result TEES_SPISetCPOL (TEES_SPIHandler *handler*, uint8_t *cpol*)

```
#include <tee_spi.h>
```

Set SPI clock polarity bit.

Parameters

in	<i>handler</i>	SPI handler which contains device descriptor.
in	<i>cpol</i>	Value of polarity bit.

Returns

TEE_SUCCESS - In case of success.

Example:

```
...
TEES_SPIHandler spi;
TEES_SPISetCPOL(spi, 0);
...
```

4.5.4.12 TEE_Result TEES_SPISetDMAMode (TEES_SPIHandler *handler*, bool *isEnabled*)

```
#include <tee_spi.h>
```

Enable or disable DMA mode.

Parameters

in	<i>handler</i>	SPI handler which contains device descriptor.
in	<i>isEnabled</i>	true – enable, false - disable.

Returns

TEE_SUCCESS - In case of success.

Example:

```
...
TEES_SPIHandler spi;
TEES_SPISetDMAMode (spi, true);
...
```

4.5.4.13 TEE_Result TEES_SPIWrite (TEES_SPIHandler *handler*, TEES_SPITransfer * *tx*)

```
#include <tee_spi.h>
```

Transfer data from buffer bus to SPI.

Parameters

in	<i>handler</i>	SPI handler which contains device descriptor.
in	<i>tx</i>	Pointer to buffer to read data from.

Return values

<i>TEE_SUCCESS</i>	on success.
<i>TEE_ERROR_GENERIC</i>	on failure.

4.5.4.14 TEE_Result TEES_SPIWriteRead (TEES_SPIHandler *handler*, TEES_SPITransfer * *tx*, TEES_SPITransfer * *rx*)

```
#include <tee_spi.h>
```

Transfer data from buffer to SPI bus.

Parameters

in	<i>handler</i>	SPI handler which contains device descriptor.
in	<i>tx</i>	Pointer to buffer to read data from.
out	<i>rx</i>	Pointer to buffer to store data.

Return values

<i>TEE_SUCCESS</i>	on success.
<i>TEE_ERROR_BAD_PARAMETERS</i>	<i>rx</i> and <i>tx</i> buffers have different size.

Example:

```
TEES_SPIHandler handler;
TEES_SPIConfig cfg = {
    .speedHz = 5000000,
    .CPOL = 0,
    .CPHA = 0,
    .bitsPerWord = SPI_TYPE_WORD,
    .isDMAMode = true
};

TEES_SPIDMAInit(pa);

TEE_Result res = TEES_SPIInit(port, &cfg, &handler);
if (res != TEE_SUCCESS) {
    printf("TEST: Failed to initialize SPI, tee_err: %#x\n", res);
    return res;
}

TEES_SPITransfer tx = {
    .bufAddr = (void *) (pa + DMA_TX_OFFSET),
    .bufLen = buflen,
    .transferredLen = 0
};

TEES_SPITransfer rx = {
    .bufAddr = (void *) (pa + DMA_RX_OFFSET),
    .bufLen = buflen,
    .transferredLen = 0
};

res = TEES_SPIWriteRead(handler, &tx, &rx);
if (TEES_SPIExit(handler)) {
    TEE_Panic(0);
}

if (rx.transferredLen == 0 || tx.transferredLen == 0) {
    printf("TEST: nothing was transferred over SPI, err: %#x\n", res);
    return TEE_ERROR_GENERIC;
}
```

4.6 I2C API

Data Structures

- struct [TEES_I2CTransfer](#)
I2C data transfer buffer. [More...](#)

Typedefs

- typedef struct __TEES_I2CHandlerImpl * [TEES_I2CHandler](#)

Functions

- TEE_Result [TEES_I2CInit](#) (const char *name, uint32_t transac_len, [TEES_I2CHandler](#) *handler)
Initialize I2C device and set transfer parameters to handler.
- TEE_Result [TEES_I2CExit](#) ([TEES_I2CHandler](#) handler)
Terminate connection to I2C device.
- TEE_Result [TEES_I2CWrite](#) ([TEES_I2CHandler](#) handler, uint8_t chip, [TEES_I2CTransfer](#) *tx)
Write data buffer tx to initialized I2C device.
- TEE_Result [TEES_I2CRead](#) ([TEES_I2CHandler](#) handler, uint8_t chip, [TEES_I2CTransfer](#) *rx)
Read data buffer rx from initialized I2C device.
- TEE_Result [TEES_I2CWriteRead](#) ([TEES_I2CHandler](#) handler, uint8_t chip, [TEES_I2CTransfer](#) *tx, [TEES_I2CTransfer](#) *rx)
Write data buffer tx and read buffer rx from initialized I2C device at the same time.

4.6.1 Detailed Description

Provides set of functions to manipulate device connected with I2C bus.

4.6.2 Data Structure Documentation

4.6.2.1 struct TEES_I2CTransfer

I2C data transfer buffer.

Data Fields

void *	buf	pointer to buffer used for I2C data transfer.
size_t	len	size of buffer buf.

4.6.3 Typedef Documentation

4.6.3.1 TEES_I2CHandler

```
#include <tee_i2c.h>
```

Handler for I2C device.

4.6.4 Function Documentation

4.6.4.1 TEE_Result TEES_I2CExit (TEES_I2CHandler *handler*)

```
#include <tee_i2c.h>
```

Terminate connection to I2C device.

Parameters

in	<i>handler</i>	initialized parameters.
----	----------------	-------------------------

Return values

<i>TEE_SUCCESS</i>	in case of success.
<i>TEE_ERROR_XXX</i>	in case of errors.

4.6.4.2 TEE_Result TEES_I2CInit (const char * *name*, uint32_t *transac_len*, TEES_I2CHandler * *handler*)

```
#include <tee_i2c.h>
```

Initialize I2C device and set transfer parameters to handler.

Parameters

in	<i>name</i>	interface name.
in	<i>transac_len</i>	size of transaction.
out	<i>handler</i>	Pointer to structure which contains set up parameters.

Return values

<i>TEE_SUCCESS</i>	in case of success.
<i>TEE_ERROR_XXX</i>	in case of errors.

4.6.4.3 TEE_Result TEES_I2CRead (TEES_I2CHandler *handler*, uint8_t *chip*, TEES_I2CTransfer * *rx*)

```
#include <tee_i2c.h>
```

Read data buffer *rx* from initialized I2C device.

Parameters

in	<i>handler</i>	initialized parameters.
in	<i>chip</i>	address of target device on I2C bus.
out	<i>rx</i>	Pointer to buffer to receive.

Return values

<i>TEE_SUCCESS</i>	in case of success.
<i>TEE_ERROR_XXX</i>	in case of errors.

4.6.4.4 TEE_Result TEES_I2CWrite (TEES_I2CHandler *handler*, uint8_t *chip*, TEES_I2CTransfer * *tx*)

```
#include <tee_i2c.h>
```

Write data buffer *tx* to initialized I2C device.

Parameters

in	<i>handler</i>	initialized parameters.
in	<i>chip</i>	address of target device on I2C bus.
in	<i>tx</i>	Pointer to buffer to transfer.

Return values

<i>TEE_SUCCESS</i>	in case of success.
<i>TEE_ERROR_XXX</i>	in case of errors.

4.6.4.5 TEE_Result TEES_I2CWriteRead (TEES_I2CHandler *handler*, uint8_t *chip*, TEES_I2CTransfer * *tx*, TEES_I2CTransfer * *rx*)

```
#include <tee_i2c.h>
```

Write data buffer *tx* and read buffer *rx* from initialized I2C device at the same time.

Parameters

in	<i>handler</i>	initialized parameters.
in	<i>chip</i>	address of target device on I2C bus.
in	<i>tx</i>	Pointer to buffer to transfer.
out	<i>rx</i>	Pointer to buffer to receive.

Return values

<i>TEE_SUCCESS</i>	in case of success.
<i>TEE_ERROR_XXX</i>	in case of errors.

4.7 Trusted user interface

Data Structures

- struct [TEES_TUIScreenInfo](#)
Structure that represents information on the requested screen. [More...](#)
- struct [TEES_TUITouchInfo](#)
Structure that represents information of touch event (touch position, touch type). [More...](#)
- struct [TEES_TUIImage](#)
Structure that represents properties of image and buffers. [More...](#)
- struct [TEE_TUIImage](#)
Structure that defines a way to handle an image for label area and buttons. [More...](#)
- union [TEE_TUIImage.__unnamed__](#)
- struct [TEE_TUIImage.__unnamed__ref](#)
- struct [TEE_TUIImage.__unnamed__object](#)
- struct [TEE_TUIScreenLabel](#)
Structure that defines the contents of the TA defined label area, which is provided to support TA branding and a TA defined message. [More...](#)
- struct [TEE_TUIButton](#)
Structure that defines the content of a button. [More...](#)
- struct [TEE_TUIScreenConfiguration](#)
Structure that enables configuration of a TUI screen. [More...](#)
- struct [TEE_TUIScreenButtonInfo](#)
Structure that represents button information associated with a TUI screen for a given orientation. [More...](#)
- struct [TEE_TUIScreenInfo](#)
Structure that represents screen information for a given orientation. [More...](#)
- struct [TEE_TUIEntryField](#)
Structure that represents an entry field which acquires user inputs. [More...](#)

Macros

- #define [MIN_FONT_SIZE](#) 20
- #define [MAX_FONT_SIZE](#) 160
- #define [TEE_TUI_NUMBER_BUTTON_TYPES](#) 6

Enumerations

- enum [TEES_Result](#) {
[TEES_SUCCESS](#) = 0x00000000, [TEES_ERROR_TUI_GENERIC](#) = 0xFFFF0000, [TEES_ERROR_TUI_BAD_FORMAT](#) = 0xFFFF0005, [TEES_ERROR_TUI_INVAL_PARAM](#) = 0xFFFF0006,
[TEES_ERROR_TUI_BAD_STATE](#) = 0xFFFF0007, [TEES_ERROR_NOT_IMPLEMENTED](#) = 0xFFFF0009,
[TEES_ERROR_NOT_SUPPORTED](#) = 0xFFFF000A, [TEES_ERROR_TUI_OUT_OF_MEMORY](#) = 0xFFFF000C,
[TEES_ERROR_TUI_BUSY](#) = 0xFFFF000D }
TEES result codes.
- enum [TEES_TUITouchTypes](#) { [TEES_TOUCH_PRESSED](#), [TEES_TOUCH_RELEASED](#) }
Internal, touch event type.
- enum [TEES_bool](#) { [TEES_FALSE](#) = 0, [TEES_TRUE](#) = 1 }
Internal, bool type.
- enum [TEE_TUIEntryFieldMode](#) { [TEE_TUI_HIDDEN_MODE](#) = 0, [TEE_TUI_CLEAR_MODE](#), [TEE_TEMPRARY_CLEAR_M](#) }
}

- Entry fields mode.*

 - enum `TEE_TUIEntryFieldType` { `TEE_TUI_NUMERICAL` = 0, `TEE_TUI_ALPHANUMERICAL` }

Entry fields type.

 - enum `TEE_TUIScreenOrientation` { `TEE_TUI_PORTRAIT` = 0, `TEE_TUI_LANDSCAPE` }

Screen orientation.

 - enum `TEE_TUIButtonType` {
`TEE_TUI_CORRECTION` = 0, `TEE_TUI_OK`, `TEE_TUI_CANCEL`, `TEE_TUI_VALIDATE`,
`TEE_TUI_PREVIOUS`, `TEE_TUI_NEXT` }

Button type.

 - enum `TEE_TUIImageSource` { `TEE_TUI_NO_SOURCE` = 0, `TEE_TUI_REF_SOURCE`, `TEE_TUI_OBJECT_SOURCE` }

Image source.

Functions

- `TEES_Result TEE_TUIGetScreenInfo` (`TEES_TUIScreenInfo` *screenInfo)
Retrieves information about the screen.
- `TEES_Result TEE_TUIOpenSession` (void)
Claims an exclusive access to TUI resources.
- `TEES_Result TEE_TUICloseSession` (void)
Releases TUI resources.
- `TEES_Result TEE_TUIDrawImage` (`TEES_TUIImage` *image, uint32_t posX, uint32_t posY)
Display an image on screen.
- `TEES_Result TEE_TUIGetTouchEvent` (`TEES_TUITouchInfo` *touchInfo, uint32_t timeout)
Get a touch event.
- `TEES_Result TEE_TUICheckTextFormat` (char *inputText, uint32_t textSize, uint32_t *width, uint32_t *height, uint32_t *lastIndex)
Checks validity of the string and Retrieves its width and height.
- `TEES_Result TEE_TUIPrintString` (char *inputText, uint32_t textSize, uint32_t posX, uint32_t posY, uint8_t redColorValue, uint8_t greenColorValue, uint8_t blueColorValue, bool rotation90)
Print string on the screen.
- `TEES_Result TEE_TUIRefreshScreen` (void)
Refresh display controller.
- `TEES_Result TEE_TUIGetBuffer` (uint32_t *buffer, uint32_t posX, uint32_t posY, uint32_t W, uint32_t H)
Copy a rectangle from screen to buffer.
- `TEES_Result TEE_TUIDrawBuffer` (uint32_t *buffer, uint32_t posX, uint32_t posY, uint32_t W, uint32_t H)
Copy a rectangle from buffer to screen.
- `TEE_Result TEE_TUICheckTextFormat` (char *text, uint32_t *width, uint32_t *height, uint32_t *lastIndex)
Check whether a given text can be rendered and retrieves information.
- `TEE_Result TEE_TUIGetScreenInfo` (`TEE_TUIScreenOrientation` screenOrientation, uint32_t nbEntryFields, `TEE_TUIScreenInfo` *screenInfo)
Retrieves information about the screen.
- `TEE_Result TEE_TUIInitSession` (void)
Claims an exclusive access to TUI resources.
- `TEE_Result TEE_TUICloseSession` (void)
Releases TUI resources.
- `TEE_Result TEE_TUIDisplayScreen` (`TEE_TUIScreenConfiguration` *screenConfiguration, bool closeTUISession, `TEE_TUIEntryField` *entryFields, uint32_t entryFieldCount, `TEE_TUIButtonType` *selectedButton)
Displays a TUI screen.

4.7.1 Detailed Description

Provides a set of functions to manipulate user interface.

4.7.2 Data Structure Documentation

4.7.2.1 struct TEES_TUIScreenInfo

Structure that represents information on the requested screen.

Data Fields

uint32_t	blueBitsDepth	Available Blue bit depth.
uint32_t	grayscaleBitsDepth	Available grayscale depth.
uint32_t	greenBitsDepth	Available Green bit depth.
uint32_t	heightInch	Height in pixels per inch.
uint32_t	redBitsDepth	Available Red bit depth.
uint32_t	resolution_height	Height of screen resolution.
uint32_t	resolution_width	Width of screen resolution.
uint32_t	widthInch	Width in pixels per inch.

4.7.2.2 struct TEES_TUITouchInfo

Structure that represents information of touch event (touch position, touch type).

Data Fields

TEES_TUITouchTypes	touchType	indicates touch type.
uint32_t	xPosition	the x-coordinate of touch event.
uint32_t	yPosition	the y-coordinate of touch event.

4.7.2.3 struct TEES_TUIImage

Structure that represents properties of image and buffers.

Data Fields

uint32_t	height	the number of pixels of the height of the image.
void *	imageBuf	buffer containing the image.
size_t	imageLength	buffer size.
uint32_t	use_png_alpha	not used.
uint32_t	width	the number of pixels of the width of the image.

4.7.2.4 struct TEE_TUIImage

Structure that defines a way to handle an image for label area and buttons.

Data Fields

union TEE_TUIImage	<code>__unnamed__</code>	union of image source
<code>uint32_t</code>	<code>height</code>	the number of pixels of the height of the image.
TEE_TUIImageSource	<code>source</code>	indicates the source of the image.
<code>uint32_t</code>	<code>width</code>	the number of pixels of the width of the image.

4.7.2.5 union TEE_TUIImage.__unnamed__

Data Fields

__unnamed__	<code>object</code>	<code>TEE_TUI_OBJECT_SOURCE</code>
__unnamed__	<code>ref</code>	<code>TEE_TUI_REF_SOURCE</code>

4.7.2.6 struct TEE_TUIImage.__unnamed__.ref

Data Fields

<code>void *</code>	<code>image</code>	buffer containing the image.
<code>size_t</code>	<code>imageLength</code>	buffer size.

4.7.2.7 struct TEE_TUIImage.__unnamed__.object

Data Fields

<code>void *</code>	<code>objectID</code>	Object Identifier which refers to a Data Object
<code>size_t</code>	<code>objectIDLen</code>	Object ID length.
<code>uint32_t</code>	<code>storageID</code>	storage to use.

4.7.2.8 struct TEE_TUIScreenLabel

Structure that defines the contents of the TA defined label area, which is provided to support TA branding and a TA defined message.

Data Fields

TEE_TUIImage	image	image to be put in the label area.
uint32_t	imageXOffset	the x-coordinate for the top left corner of the image to display.
uint32_t	imageYOffset	the y-coordinate for the top left corner of the image to display.
char *	text	string to put in the label area.
uint8_t	textColor[3]	defines the color of the text in RGB form.
uint32_t	textXOffset	the x-coordinate for the top left corner of the text to render.
uint32_t	textYOffset	the y-coordinate for the top left corner of the text to render.

4.7.2.9 struct TEE_TUIButton

Structure that defines the content of a button.

Data Fields

TEE_TUIImage	image	image to associate with the button.
char *	text	string to associate with the button.

4.7.2.10 struct TEE_TUIScreenConfiguration

Structure that enables configuration of a TUI screen.

Data Fields

TEE_TUIButton *	buttons[6]	customizes the buttons compared to the default configuration.
TEE_TUIScreenLabel	label	specifies the label of the screen.
bool	requestedButtons[6]	specifies which buttons to be displayed.
TEE_TUIScreenOrientation	screenOrientation	requested screen orientation.

4.7.2.11 struct TEE_TUIScreenButtonInfo

Structure that represents button information associated with a TUI screen for a given orientation.

Data Fields

uint32_t	buttonHeight	The height in pixels of the button.
bool	buttonImageCustom	if the image of the button can be customized. false otherwise.
char *	buttonText	The value of the default label.
bool	buttonTextCustom	true if the text of the button can be customized. false otherwise.
uint32_t	buttonWidth	The width in pixels of the button.

4.7.2.12 struct TEE_TUIScreenInfo

Structure that represents screen information for a given orientation.

Data Fields

uint32_t	blueBitsDepth	Available Blue bit depth.
TEE_TUIScreenButtonInfo	buttonInfo[6]	Information defining the buttons of the screens.
uint32_t	entryFieldLabelHeight	Height in pixels of the label of an entry field.
uint32_t	entryFieldLabelWidth	Width in pixels of the label of an entry field.
uint32_t	grayscaleBitsDepth	Available grayscale depth.
uint32_t	greenBitsDepth	Available Green bit depth.
uint32_t	heightInch	Height in pixels per inch.
uint8_t	labelColor[3]	The RGB values of the default label canvas.
uint32_t	labelHeight	Height in pixels of the label canvas.
uint32_t	labelWidth	Width in pixels of the label canvas.
uint32_t	maxEntryFieldLength	The maximum number of characters that can be entered within an entry field.
uint32_t	maxEntryFields	Maximum number of entry fields that can be displayed on the screen.
uint32_t	redBitsDepth	Available Red bit depth.
uint32_t	widthInch	Width in pixels per inch.

4.7.2.13 struct TEE_TUIEntryField

Structure that represents an entry field which acquires user inputs.

Data Fields

char *	buffer	Contains the input entered by the user.
size_t	bufferLength	Contains the input entered by the user.
char *	label	The label associated with the entry field.
uint32_t	maxExpectedLength	The maximum number of characters expected for the entry field.
uint32_t	minExpectedLength	The minimum number of characters expected for the entry field.
TEE_TUIEntryFieldMode	mode	The mode to be used when displaying characters.
TEE_TUIEntryFieldType	type	The type of inputs accepted by the entry field.

4.7.3 Macro Definition Documentation

4.7.3.1 #define MAX_FONT_SIZE 160

```
#include <tees_tui.h>
```

Font size value should be between min~Max. If not error should return.

4.7.3.2 #define MIN_FONT_SIZE 20

```
#include <tees_tui.h>
```

Font size value should be between min~Max. If not error should return.

4.7.3.3 #define TEE_TUI_NUMBER_BUTTON_TYPES 6

```
#include <tui.h>
```

Number of possible buttons on TUI screen.

4.7.4 Enumeration Type Documentation

4.7.4.1 enum TEE_TUIButtonType

```
#include <tui.h>
```

Button type.

Enumerator

```
TEE_TUI_CORRECTION Correction
TEE_TUI_OK OK
TEE_TUI_CANCEL Cancel
TEE_TUI_VALIDATE Validate
TEE_TUI_PREVIOUS Previous
TEE_TUI_NEXT Next
```

4.7.4.2 enum TEE_TUIEntryFieldMode

```
#include <tui.h>
```

Entry fields mode.

Enumerator

TEE_TUI_HIDDEN_MODE hidden
TEE_TUI_CLEAR_MODE clear
TEE_TEMPRARY_CLEAR_MODE temporary clear

4.7.4.3 enum TEE_TUIEntryFieldType

```
#include <tui.h>
```

Entry fields type.

Enumerator

TEE_TUI_NUMERICAL numerical
TEE_TUI_ALPHANUMERICAL alphanumerical

4.7.4.4 enum TEE_TUIImageSource

```
#include <tui.h>
```

Image source.

Enumerator

TEE_TUI_NO_SOURCE No image is provided as input
TEE_TUI_REF_SOURCE The image source is provided as memory reference
TEE_TUI_OBJECT_SOURCE The image source is provided as a Data Object in the Trusted Storage

4.7.4.5 enum TEE_TUIScreenOrientation

```
#include <tui.h>
```

Screen orientation.

Enumerator

TEE_TUI_PORTRAIT Portrait
TEE_TUI_LANDSCAPE Landscape

4.7.4.6 enum TEES_bool

```
#include <tees_tui.h>
```

Internal, bool type.

Enumerator

```
TEES_FALSE false  
TEES_TRUE true
```

4.7.4.7 enum TEES_Result

```
#include <tees_tui.h>
```

TEES result codes.

Enumerator

```
TEES_SUCCESS Scess  
TEES_ERROR_TUI_GENERIC Generic error  
TEES_ERROR_TUI_BAD_FORMAT Bad format  
TEES_ERROR_TUI_INVALID_PARAM Invalid parameter  
TEES_ERROR_TUI_BAD_STATE Bad state  
TEES_ERROR_NOT_IMPLEMENTED Not implemented  
TEES_ERROR_NOT_SUPPORTED Not supported  
TEES_ERROR_TUI_OUT_OF_MEMORY Out of memory  
TEES_ERROR_TUI_BUSY Busy
```

4.7.4.8 enum TEES_TUITouchTypes

```
#include <tees_tui.h>
```

Internal, touch event type.

Enumerator

```
TEES_TOUCH_PRESSED Pressed  
TEES_TOUCH_RELEASED Released
```

4.7.5 Function Documentation

4.7.5.1 TEE_Result TEE_TUICheckTextFormat (char * text, uint32_t * width, uint32_t * height, uint32_t * lastIndex)

```
#include <tui.h>
```

Check whether a given text can be rendered and retrieves information.

The TEE_TUICheckTextFormat function allows a TA to check whether a given text can be rendered by the current implementation and retrieves information about the size and width that is needed to render it. TEE_TUIInitSession does not have to be called before using this function.

Parameters

in	<i>text</i>	The string to be checked.
out	<i>width</i>	Width in pixels needed to render the text.
out	<i>height</i>	Height in pixels needed to render the text.
out	<i>lastIndex</i>	Indicates the last character that has been checked. In case of success, it corresponds to the last character of the text string. In case of failure, it indicates the index of the character which causes the failure. The index starts at 0.

Return values

<i>TEE_SUCCESS</i>	In case of success
<i>TEE_ERROR_NOT_SUPPORTED</i>	If at least one of the characters present in the text string cannot be rendered.

4.7.5.2 TEE_Result TEE_TUICloseSession (void)

```
#include <tui.h>
```

Releases TUI resources.

The TEE_TUICloseSession function releases TUI resources previously acquired. This function SHOULD be called as soon as possible after the last TUI screen of the TUI session has ended in order to avoid a bad user experience.

Return values

<i>TEE_SUCCESS</i>	In case of success
<i>TEE_ERROR_BAD_STATE</i>	If the current TA is not within a TUI session initially started by a successful call to TEE_TUIInitSession. In particular, it will be returned if a TUI session has been closed automatically because the TUI session timeout has been reached or if a TUI session has been closed due to an OS specific external event such as an incoming call.
<i>TEE_ERROR_BUSY</i>	If the TUI resources are currently in use, i.e. a TUI screen is displayed. This error code can only be returned by a TEE implementation supporting multi-threading within a TA and will occur when a thread tries to close a TUI session that is displaying a TUI screen in another thread.

4.7.5.3 TEE_Result TEE_TUIDisplayScreen (TEE_TUIScreenConfiguration * *screenConfiguration*, bool *closeTUISession*, TEE_TUIEntryField * *entryFields*, uint32_t *entryFieldCount*, TEE_TUIButtonType * *selectedButton*)

```
#include <tui.h>
```

Displays a TUI screen.

The TEE_TUIDisplayScreen function displays a TUI screen. The display order of the requested entry fields is from top to bottom. This function is cancellable, i.e., if the current task's cancelled flag is set and the TA has unmasked the effects of cancellation, then this function returns earlier than the requested timeout with the error code TEE_ERROR_CANCEL. In a given session, once the first call to TEE_TUIDisplayScreen has been made and if the parameter *closeTUISession* was set to false, the screen will not be handed back to the REE until TEE_TUICloseSession is called. When not displaying a particular TEE_TUIDisplayScreen the TEE MAY continue to display the current TUI screen or MAY display a screen indicating a security TUI session is in progress. Input events that occurred before the call to TEE_TUIDisplayScreen SHALL be ignored. Note that all in and out parameters, as well as the buffers they refer to, MUST NOT reside in shared memory.

Parameters

in	<i>screenConfiguration</i>	Configures the label of the screen and optionally the buttons of the screen.
in	<i>closeTUISession</i>	If true the TUI session is automatically closed when exiting the function.
in	<i>entryFields,entryFieldCount</i>	Array of entry fields. It contains the entry fields to display on the screen and enables input from the user by filling the buffer field of the corresponding TEE_TUIEntryField structure. It is ignored if <i>entryFieldCount</i> is set to 0.
out	<i>selectedButton</i>	In case of success, it indicates which button has been selected by the user to exit the TUI screen.

Return values

<i>TEE_SUCCESS</i>	In case of success.
<i>TEE_ERROR_OUT_OF_MEMORY</i>	If the system ran out of resources.
<i>TEE_ERROR_ITEM_NOT_FOUND</i>	If at least one image provided by the TA refers to a storage denoted by a storageID which does not exist or if the corresponding Object Identifier cannot be found in the storage.
<i>TEE_ERROR_ACCESS_CONFLICT</i>	If at least one image provided by the TA refers to a Data Object in the Trusted Storage and an access right conflict was detected while opening the object.
<i>TEE_ERROR_BAD_FORMAT</i>	If at least one input image is not compliant with PNG format.
<i>TEE_ERROR_BAD_STATE</i>	If the current TA is not within a TUI session initially started by a successful call to <i>TEE_TUIInitSession</i> . In particular, it will be returned if a TUI session has been closed automatically because the TUI session timeout has been reached or if a TUI session has been closed due to an OS specific external event such as an incoming call.
<i>TEE_ERROR_BUSY</i>	If the TUI resources are currently in use, i.e. a TUI screen is displayed. This error code can only be returned by a TEE implementation supporting multi-threading within a TA and will occur when a thread tries to display a TUI screen while a TUI screen is already displayed.
<i>TEE_ERROR_CANCEL</i>	<p>If the operation has been cancelled while a TUI screen is currently displayed.</p> <ul style="list-style-type: none"> • The current UI session acquired by <i>TEE_TUIInitSession</i> is automatically closed as if <i>TEE_TUICloseSession</i> had been called. • The implementation MAY have started to fill out entry fields. In that case, entry fields will be returned with the last values entered by the user and it is up to the TA as to how it makes use of these.
<i>TEE_ERROR_EXTERNAL_CANCEL</i>	<p>If the operation has been cancelled by an external event which occurred in the REE while a TUI screen is currently displayed.</p> <ul style="list-style-type: none"> • The current UI session acquired by <i>TEE_TUIInitSession</i> is automatically closed as if <i>TEE_TUICloseSession</i> had been called. • The implementation MAY have started to fill out entry fields. In that case, entry fields MAY be returned with the last values entered by the user and it is up to the TA as to how it makes use of these.

4.7.5.4 **TEE_Result TEE_TUIGetScreenInfo (TEE_TUIScreenOrientation screenOrientation, uint32_t nbEntryFields, TEE_TUIScreenInfo * screenInfo)**

```
#include <tui.h>
```

Retrieves information about the screen.

The TEE_TUIGetScreenInfo function retrieves information about the screen depending on its orientation and the number of required entry fields. TEE_TUIInitSession does not have to be called before using this function.

Parameters

in	<i>screenOrientation</i>	Defines for which orientation screen information is requested.
in	<i>nbEntryFields</i>	Defines how many entry fields are requested.
out	<i>screenInfo</i>	Returns information on the requested screen for a given orientation.

Return values

<i>TEE_SUCCESS</i>	In case of success
<i>TEE_ERROR_NOT_SUPPORTED</i>	if the requested number of entry fields is not supported. In that case, the field <i>maxEntryFields</i> of the <i>screenInfo</i> output parameter is set to the maximum number of entry fields supported for the given orientation.

4.7.5.5 TEE_Result TEE_TUIInitSession (void)

```
#include <tui.h>
```

Claims an exclusive access to TUI resources.

The TEE_TUIInitSession function claims an exclusive access to TUI resources for the current TA. Control of screen and keyboard MAY be taken over by the TEE at this stage. This just reserves the ability to use the TUI for this particular TA and will notify other TAs that this reservation has been made and the resource is busy (i.e. those other TAs will receive TEE_ERROR_BUSY when attempting this operation). As a limited resource, the TUI session will be closed automatically whenever the TA does not display a TUI screen to interact with the user for a period of time. This period of time is equal to the value of the property `gpd.tee.tui.session.timeout`. This function MUST be called before a screen can be displayed with TEE_TUIDisplayScreen.

Return values

<i>TEE_SUCCESS</i>	In case of success.
<i>TEE_ERROR_BUSY</i>	If the TUI resources cannot be reserved.
<i>TEE_ERROR_OUT_OF_MEMORY</i>	If the system ran out of resources.

4.7.5.6 TEES_Result TEES_TUICheckTextFormat (char * *inputText*, uint32_t *textSize*, uint32_t * *width*, uint32_t * *height*, uint32_t * *lastIndex*)

```
#include <tees_tui.h>
```

Checks validity of the string and Retrieves its width and height.

The `TEES_TUICheckTextFormat` function retrieves information about the string such as width, height needed to be displayed on the screen. If the string is not valid or the string with the size as `textSize` can not be displayed on the screen, this function returns error. In the case of that the string exceeds the screen boundary, this function returns error and `lastIndex` has the index of the character causing the error in the string.

Parameters

in	<i>inputText</i>	pointer to input string to be checked in this function
in	<i>textSize</i>	size of the text. The range is 20 to 160
out	<i>width</i>	width in pixels needed to render the text
out	<i>height</i>	height in pixels needed to render the text
out	<i>lastIndex</i>	indicates the last character that has been checked

Return values

<i>TEES_SUCCESS</i>	In case of success
<i>TEES_ERROR_TUI_INVALID_PARAM</i>	if parameters are not correct

4.7.5.7 `TEES_Result TEES_TUICloseSession (void)`

```
#include <tees_tui.h>
```

Releases TUI resources.

The `TEES_TUICloseSession` function releases TUI resources previously acquired. This function SHOULD be called as soon as possible after the last TUI screen of the TUI session has ended in order to avoid a bad user experience.

Return values

<i>TEES_SUCCESS</i>	In case of success
<i>TEES_ERROR_TUI_BAD_STATE</i>	If the current TA is not within a TUI session initially started by a successful call to <code>TEES_TUIOpenSession</code> . In particular, it will be returned if a TUI session has been closed automatically because the TUI session timeout has been reached or if a TUI session has been closed due to an OS specific external event such as an incoming call.

4.7.5.8 `TEES_Result TEES_TUIDrawBuffer (uint32_t * buffer, uint32_t posX, uint32_t posY, uint32_t W, uint32_t H)`

```
#include <tees_tui.h>
```

Copy a rectangle from buffer to screen.

The TEES_TUIDrawBuffer function copies a rectangle from buffer to screen

Parameters

in	<i>buffer</i>	is a pointer to buffer
in	<i>posX,posY</i>	coordinates top left pixels of the rectangle
in	<i>W,H</i>	width and height of the rectangle

Return values

<i>TEES_SUCCESS</i>	In case of success.
<i>TEES_ERROR_TUI_BAD_STATE</i>	If TUI session is not opened.
<i>TEES_ERROR_TUI_BUSY</i>	If some TUI function is called at moment.
<i>TEES_ERROR_TUI_GENERIC</i>	If mutex unlock error has happened.
<i>TEES_ERROR_TUI_INVALID_PARAM</i>	If wrong values were passed to function.

4.7.5.9 TEES_Result TEES_TUIDrawImage (TEES_TUIImage * *image*, uint32_t *posX*, uint32_t *posY*)

```
#include <tees_tui.h>
```

Display an image on screen.

The TEES_TUIDrawImage function displays an image on screen.

Parameters

in	<i>image</i>	is a pointer to properties of image and buffers
in	<i>posX,posY</i>	coordinates top left pixels of the image

Return values

<i>TEES_SUCCESS</i>	In case of success.
<i>TEES_ERROR_TUI_BAD_FORMAT</i>	input image is not compliant with PNG format.
<i>TEES_ERROR_TUI_INVALID_PARAM</i>	input pointer type parameter is invalid or null or out of area.
<i>TEES_ERROR_TUI_OUT_OF_MEMORY</i>	: internal memory allocation fail.
<i>TEES_ERROR_TUI_BAD_STATE</i>	: unable to get a session state

4.7.5.10 **TEES_Result** TEES_TUIGetBuffer (**uint32_t** * *buffer*, **uint32_t** *posX*, **uint32_t** *posY*, **uint32_t** *W*, **uint32_t** *H*)

```
#include <tees_tui.h>
```

Copy a rectangle from screen to buffer.

The TEES_TUIGetBuffer function copies a rectangle from screen to buffer.

Parameters

in	<i>buffer</i>	is a pointer to buffer
in	<i>posX,posY</i>	coordinates top left pixels of the rectangle
in	<i>W,H</i>	width and height of the rectangle

Return values

<i>TEES_SUCCESS</i>	In case of success.
<i>TEES_ERROR_TUI_BAD_STATE</i>	If TUI session is not opened.
<i>TEES_ERROR_TUI_BUSY</i>	If some TUI function is called at moment.
<i>TEES_ERROR_TUI_GENERIC</i>	If mutex unlock error has happened.
<i>TEES_ERROR_TUI_INVALID_PARAM</i>	If wrong values were passed to function.

4.7.5.11 **TEES_Result** TEES_TUIGetScreenInfo (**TEES_TUIScreenInfo** * *screenInfo*)

```
#include <tees_tui.h>
```

Retrieves information about the screen.

The TEES_TUIGetScreenInfo function retrieves information about the screen TEES_TUIOpenSession does not have to be called before using this function.

Parameters

out	<i>screenInfo</i>	returns information on the requested screen screenInfo->width and screenInfo->height information is based on PORTRAIT MODE only
-----	-------------------	---

Return values

<i>TEES_SUCCESS</i>	In case of success
<i>TEES_ERROR_TUI_INVALID_PARAM</i>	if out parameter is null, error returned.

4.7.5.12 TEES_Result TEES_TUIGetTouchEvent (TEES_TUITouchInfo * touchInfo, uint32_t timeout)

```
#include <tees_tui.h>
```

Get a touch event.

At a given time, TA wait and get a touch event If there's no input events from device, an error is returned. A zero timeout means this function waits for a maximum time.

Parameters

out	<i>touchInfo</i>	is a pointer to information of touch event (touch position, touch type)
in	<i>timeout</i>	is a specified time to wait for an incoming event

Return values

<i>TEES_SUCCESS</i>	: In case of success.
<i>TEES_ERROR_TUI_TIMEOUT</i>	: there is no event to return within the timeout
<i>TEES_ERROR_TUI_BAD_STATE</i>	: session has expired or closed before
<i>TEES_ERROR_TUI_INVALID_PARAM</i>	input pointer type parameter is null

4.7.5.13 TEES_Result TEES_TUIOpenSession (void)

```
#include <tees_tui.h>
```

Claims an exclusive access to TUI resources.

The TEES_TUIOpenSession function claims an exclusive access to TUI resources for the current TA. This just reserves the ability to use the TUI for this particular TA and will notify other TAs that this reservation has been made and the resource is busy (i.e. those other TAs will receive TEES_ERROR_BUSY when attempting this operation). As a limited resource, the TUI session will be closed automatically whenever the TA does not display a TUI screen to interact with the user for a period of time. This period of time is equal to the value of the property `gpd.tee.tui.session.timeout`. This function MUST be called before a screen can be displayed with TEES_TUIDrawImage. This function MUST be called before TEES_TUIGetTouchEvent.

Return values

<i>TEES_SUCCESS</i>	In case of success.
<i>TEES_ERROR_TUI_BUSY</i>	If the TUI resources cannot be reserved. for example session has already opened before, this error type is returned

4.7.5.14 TEES_Result TEES_TUIPrintString (char * *inputText*, uint32_t *textSize*, uint32_t *posX*, uint32_t *posY*, uint8_t *redColorValue*, uint8_t *greenColorValue*, uint8_t *blueColorValue*, bool *rotation90*)

```
#include <tees_tui.h>
```

Print string on the screen.

The TEES_TUIPrintString function prints input string on the screen.

Parameters

in	<i>inputText</i>	pointer to the input text string to be displayed on the screen
in	<i>textSize</i>	size of the text string. The range is 20 to 160
in	<i>posX,posY</i>	coordinates top left pixels of the image
in	<i>redColorValue,greenColorValue,blueColorValue</i>	color values for the text string. The range for each parameter is 0x00 to 0xFF
in	<i>rotation90</i>	if this parameter is true, the text string is rotated to 90 degree on the screen

Return values

<i>TEES_SUCCESS</i>	In case of success.
<i>TEES_ERROR_TUI_BUSY</i>	If the TUI resources cannot be reserved. For example session has already opened before, this error type is returned
<i>TEES_ERROR_TUI_BAD_STATE</i>	If the current TA is not within a TUI session initially started by a successful call to TEES_TUIOpenSession. In particular, it will be returned if a TUI session has been closed automatically because the TUI session timeout has been reached or if a TUI session has been closed due to an OS specific external event such as an incoming call.
<i>TEES_ERROR_TUI_INVALID_PARAM</i>	if parameters are not correct
<i>TEES_ERROR_TUI_BAD_FORMAT</i>	input text string is not valid
<i>TEES_ERROR_TUI_OUT_OF_MEMORY</i>	: internal memory allocation fail.
<i>TEES_ERROR_NOT_SUPPORTED</i>	if there is a problem in font engine in TUI
<i>TEES_ERROR_TUI_GENERIC</i>	other exceptional cases that are not defined as error message

4.7.5.15 TEES_Result TEES_TUIRefreshScreen (void)

```
#include <tees_tui.h>
```

Refresh display controller.

The TEES_TUIRefreshScreen function refreshes display controller after frame buffer changing.

Return values

<i>TEES_SUCCESS</i>	In case of success
<i>TEE_ERROR_GENERIC</i>	If ioctl to display driver returned error.
<i>TEES_ERROR_TUI_BAD_STATE</i>	If TUI session is not opened.
<i>TEES_ERROR_TUI_BUSY</i>	If some TUI function is called at moment.
<i>TEES_ERROR_TUI_GENERIC</i>	If mutex unlock error has happened.

4.8 Integrity Report System API

Macros

- #define `IRS_FAIL_TZ` -1
- #define `IRS_UNKNOWN_ID_TZ` -2
- #define `IRS_UNKNOWN_INT_CMD_TZ` -3
- #define `IRS_INCORRECT_FLAG_TYPE_TZ` -4
- #define `IRS_RT_FLAGS_EMPTY_TZ` -5
- #define `IRS_RT_FLAGS_FULL_TZ` -6
- #define `IRS_INCORRECT_RT_ID_TZ` -7
- #define `IRS_DENY_READ_FROM_SMC_TZ` -8
- #define `IRS_DENY_WRITE_FROM_SMC_TZ` -9
- #define `IRS_DENY_DELETE_FROM_SMC_TZ` -10
- #define `IRS_MAX_VAL_COUNTER_TZ` -11
- #define `IRS_MAX_VAL_COUNTER_RT_TZ` -12
- #define `IRS_INCORRECT_CHECKSUM_TZ` -13
- #define `IRS_UNKNOWN_ERROR_TZ` -14
- #define `IRS_SUCCESS_TZ` 0

Enumerations

- enum `IRS_INTERNAL_CMD` {
`IRS_SET_FLAG_CMD` = 1, `IRS_SET_FLAG_VALUE_CMD`, `IRS_INC_FLAG_CMD`, `IRS_GET_FLAG_VALUE_CMD`,
`IRS_ADD_FLAG_CMD`, `IRS_DEL_FLAG_CMD` }
Internal IRS commands ids.
- enum `IRS_PARAM` {
`IRS_TYPE_BOOLEAN` = 0x00000001, `IRS_TYPE_VALUE` = 0x00000002, `IRS_TYPE_COUNTER` =
0x00000004, `IRS_NWD_RD` = 0x00000008,
`IRS_NWD_WR` = 0x00000010, `IRS_NWD_CTRL` = 0x00000020, `IRS_SWD_RD` = 0x00000040,
`IRS_SWD_WR` = 0x00000080,
`IRS_SWD_CTRL` = 0x00000100 }
Bit position of params.

Functions

- int `TEES_SetIrsFlag` (unsigned int *flag_id)
Set flag by flag_id in 1. Used only for boolean flag type.
- int `TEES_SetIrsFlagValue` (unsigned int *flag_id, unsigned int value)
Set flag by flag_id in value by value.
- int `TEES_IncIrsFlag` (unsigned int *flag_id)
Increment flag by flag_id in 1. Used only for IRS_TYPE_VALUE flag type.
- int `TEES_GetIrsFlagValue` (unsigned int *flag_id, unsigned int *value)
Get value by flag_id.
- int `TEES_AddIrsFlag` (unsigned int *flag_id, unsigned int param)
Function for control run-time flags.
- int `TEES_DelIrsFlag` (unsigned int *flag_id)
Delete run-time flag by flag_id.

4.8.1 Detailed Description

4.8.2 Macro Definition Documentation

4.8.2.1 #define IRS_DENY_DELETE_FROM_SMC_TZ -10

```
#include <irs.h>
```

Deny deleting access flag from SWd.

4.8.2.2 #define IRS_DENY_READ_FROM_SMC_TZ -8

```
#include <irs.h>
```

Deny reading access from flag from SWd.

4.8.2.3 #define IRS_DENY_WRITE_FROM_SMC_TZ -9

```
#include <irs.h>
```

Deny writing access to flag from SWd.

4.8.2.4 #define IRS_FAIL_TZ -1

```
#include <irs.h>
```

Generic error.

4.8.2.5 #define IRS_INCORRECT_CHECKSUM_TZ -13

```
#include <irs.h>
```

Flag was changed outside.

4.8.2.6 #define IRS_INCORRECT_FLAG_TYPE_TZ -4

```
#include <irs.h>
```

Incorrect flag type (can be boolean, value or counter).

4.8.2.7 #define IRS_INCORRECT_RT_ID_TZ -7

```
#include <irs.h>
```

Incorrect id of run-time flag.

4.8.2.8 #define IRS_MAX_VAL_COUNTER_RT_TZ -12

```
#include <irs.h>
```

COUNTER run-time flag got MAX value.

4.8.2.9 #define IRS_MAX_VAL_COUNTER_TZ -11

```
#include <irs.h>
```

COUNTER flag got MAX value.

4.8.2.10 #define IRS_RT_FLAGS_EMPTY_TZ -5

```
#include <irs.h>
```

Run-time flags are empty.

4.8.2.11 #define IRS_RT_FLAGS_FULL_TZ -6

```
#include <irs.h>
```

Run-time flags are full.

4.8.2.12 #define IRS_SUCCESS_TZ 0

```
#include <irs.h>
```

Success result.

4.8.2.13 #define IRS_UNKNOWN_ERROR_TZ -14

```
#include <irs.h>
```

Unknown error.

4.8.2.14 #define IRS_UNKNOWN_ID_TZ -2

```
#include <irs.h>
```

Unknown flag id.

4.8.2.15 #define IRS_UNKNOWN_INT_CMD_TZ -3

```
#include <irs.h>
```

Unknown internal command.

4.8.3 Enumeration Type Documentation

4.8.3.1 enum IRS_INTERNAL_CMD

```
#include <irs.h>
```

Internal IRS commands ids.

Enumerator

- IRS_SET_FLAG_CMD** Internal command setFlag.
- IRS_SET_FLAG_VALUE_CMD** Internal command setFlagValue.
- IRS_INC_FLAG_CMD** Internal command incFlag.
- IRS_GET_FLAG_VALUE_CMD** Internal command getFlagValue.
- IRS_ADD_FLAG_CMD** Internal command addFlag.
- IRS_DEL_FLAG_CMD** Internal command delFlag.

4.8.3.2 enum IRS_PARAM

```
#include <irs.h>
```

Bit position of params.

Enumerator

- IRS_TYPE_BOOLEAN** Boolean type.
- IRS_TYPE_VALUE** Value type.
- IRS_TYPE_COUNTER** Counter type.
- IRS_NWD_RD** Read from NWd.
- IRS_NWD_WR** Write to NWd.
- IRS_NWD_CTRL** Control NWd.
- IRS_SWD_RD** Read from SWd.
- IRS_SWD_WR** Write to SWd.
- IRS_SWD_CTRL** Control SWd.

4.8.4 Function Documentation

4.8.4.1 int TEES_AddIrsFlag (unsigned int * *flag_id*, unsigned int *param*)

```
#include <irs.h>
```

Function for control run-time flags.

Add new run-time flag to `rt_irs_flags` and fill params by incoming param. Increment of `rt_flags_num`(current numbers of run-time flags).

Parameters

out	<i>flag_id</i>	return a pointer to new run-time flag identifier.
in	<i>param</i>	32 bits with data (using enum IRS_PARAM).

Return values

0	no error.
<i>IRS_error</i>	code on failure.

4.8.4.2 int TEES_DellrsFlag (unsigned int * *flag_id*)

```
#include <irs.h>
```

Delete run-time flag by `flag_id`.

Parameters

in	<i>flag_id</i>	Pointer to flag identifier.
----	----------------	-----------------------------

Return values

0	no error.
<i>IRS_error</i>	code on failure.

4.8.4.3 int TEES_GetIrsFlagValue (unsigned int * *flag_id*, unsigned int * *value*)

```
#include <irs.h>
```

Get value by *flag_id*.

Get value of flag by *flag_id*. Used only for [IRS_TYPE_COUNTER](#) flag type.

Parameters

in	<i>flag_id</i>	Pointer to flag identifier.
out	<i>value</i>	Pointer to output value.

Return values

0	no error.
<i>IRS_error</i>	code on failure.

4.8.4.4 int TEES_IncIrsFlag (unsigned int * *flag_id*)

```
#include <irs.h>
```

Increment flag by *flag_id* in 1. Used only for [IRS_TYPE_VALUE](#) flag type.

Parameters

in	<i>flag_id</i>	Pointer to flag identifier.
----	----------------	-----------------------------

Return values

0	no error.
<i>IRS_error</i>	code on failure.

4.8.4.5 int TEES_SetIrsFlag (unsigned int * *flag_id*)

```
#include <irs.h>
```

Set flag by *flag_id* in 1. Used only for boolean flag type.

Parameters

in	<i>flag_id</i>	Pointer to flag identifier.
----	----------------	-----------------------------

Return values

0	no error.
<i>IRS_error</i>	code on failure.

4.8.4.6 int TEES_SetIrsFlagValue (unsigned int * *flag_id*, unsigned int *value*)

```
#include <irs.h>
```

Set flag by *flag_id* in value by value.

Parameters

in	<i>flag_id</i>	Pointer to flag identifier. Used only for IRS_TYPE_BOOLEAN flag type.
in	<i>value</i>	inputed value.

Return values

0	no error.
<i>IRS_error</i>	code on failure.

4.9 Miscellaneous extensions

Functions

- int [TEES_GetTaskDataSize](#) (size_t *data_size)
Get used size of data memory of the current Trusted Application instance.
- void * [TEES_Duplwhsmem](#) (void *address, uint32_t size)
Make long-life duplicate of an Interworld Shared memory buffer.
- TEE_Result [TEES_IsREESharedMemory](#) (uint32_t accessFlags, const void *buffer, size_t size)
Check if buffer is shared with REE.
- TEE_Result [TEES_IsPhysMemoryProtected](#) (uint32_t flags, uint64_t base, size_t size)
Check if physical memory is protected.

4.9.1 Detailed Description

Provides set of miscellaneous Samsung's extension of TEE Internal API.

4.9.2 Function Documentation

4.9.2.1 void* [TEES_Duplwhsmem](#) (void * *address*, uint32_t *size*)

```
#include <tees_extension.h>
```

Make long-life duplicate of an Interworld Shared memory buffer.

The [TEES_Duplwhsmem](#) will make a copy of Interworld Shared memory buffer, that was passed in `TEE_Param[]` on TA entry. This copy became accessible by current Trusted Application instance during current and all the following TA entries, even if the buffer is not passed in `TEE_Param[]` anymore or TA entry have no `TEE_Param` argument at all. If buffer became not needed, TA may release it by calling [munmap\(\)](#).

Parameters

in	<i>address</i>	Interworld Shared memory buffer.
in	<i>size</i>	Buffer size.

Returns

Pointer to a copied buffer or NULL on error.

4.9.2.2 int TEES_GetTaskDataSize (size_t * data_size)

```
#include <tees_extension.h>
```

Get used size of data memory of the current Trusted Application instance.

Parameters

out	<i>data_size</i>	Output data memory size.
-----	------------------	--------------------------

Return values

0	On success.
-1	On error. In this case errno is set with appropriate error code.

4.9.2.3 TEE_Result TEES_IsPhysMemoryProtected (uint32_t flags, uint64_t base, size_t size)

```
#include <tees_extension.h>
```

Check if physical memory is protected.

The TEES_IsMemoryProtected examines memory buffer passed in parameters buffer and size to determine whether the physical memory region is protected secure memory. Note that this function operates on physical addresses, not virtual.

Parameters

in	<i>flags</i>	Special flags which may alter the behavior; reserved.
in	<i>base</i>	A physical address pointing to the buffer to be examined.
in	<i>size</i>	Size of the buffer to be examined.

Return values

<i>TEE_SUCCESS</i>	Memory is present and protected by the corresponding driver.
<i>TEE_ERROR_ACCESS_DENIED</i>	Memory is present, but is either not protected or disabled.
<i>TEE_ERROR_BAD_PARAMETERS</i>	Parameters don't pass sanity check or memory is not present.
<i>TEE_ERROR_NOT_IMPLEMENTED</i>	This function is not supported on this target.

4.9.2.4 TEE_Result TEES_IsREESharedMemory (uint32_t accessFlags, const void * buffer, size_t size)

```
#include <tees_extension.h>
```

Check if buffer is shared with REE.

The TEES_IsREESharedMemory examines memory buffer passed in parameters buffer and size to determine whether Trusted Application instance can access buffer accordingly to requested access-Flags and buffer can be accessed by REE. If the characteristics of the buffer are compatible with accessFlags and REE can access this buffer, then the function returns TEE_SUCCESS. Otherwise, it returns TEE_ERROR_ACCESS_DENIED. Function should be used only to determine whether buffer shared. To determine whether buffer accessible exclusively by Trusted Application (not shared) use the TEE_CheckMemoryAccessRights.

The parameter accessFlags can contain one or more of the following flags:

- TEE_MEMORY_ACCESS_READ: Check that the buffer is entirely readable by the current Trusted Application instance.
- TEE_MEMORY_ACCESS_WRITE: Check that the buffer is entirely writable by the current Trusted Application instance. All other flags are reserved for future use and MUST be set to 0.

This function MUST NOT panic for any reason.

Parameters

in	<i>accessFlags</i>	The access flags to check.
in	<i>buffer</i>	Pointer of the buffer to check.
in	<i>size</i>	Size of the buffer to check.

Return values

<i>TEE_SUCCESS</i>	If the entire buffer allows the requested accesses.
<i>TEE_ERROR_ACCESS_DENIED</i>	If at least one byte in the buffer is not accessible with the requested accesses or REE may not access this buffer.
<i>TEE_ERROR_BAD_PARAMETERS</i>	If passed zero or unknown accessFlags.

4.10 RPMB API

Functions

- TEE_Result [TEES_RPMBRead](#) (uint32_t partition, uint32_t offset, uint8_t *data, uint32_t data_size, uint8_t type_flag)
Read data from RPMB Storage.
- TEE_Result [TEES_RPMBWrite](#) (uint32_t partition, uint32_t offset, uint8_t *data, uint32_t data_size, uint8_t type_flag)
Write data to RPMB Storage.
- TEE_Result [TEES_RPMBCheckEnable](#) (void)
Check RPMB availability.

4.10.1 Detailed Description

Provides set of functions to use(read/write) RPMB storage.

4.10.2 Function Documentation

4.10.2.1 TEE_Result TEES_RPMBCheckEnable (void)

```
#include <rpmb.h>
```

Check RPMB availability.

Return values

<i>TEE_SUCCESS</i>	in case of success
<i>error</i>	code in case of errors <ul style="list-style-type: none"> • TEE_ERROR_NOT_IMPLEMENTED - not implemented on current device • TEE_ERROR_NOT_SUPPORTED - RPMB Key is not programmed (RPMB is disabled) • other implementation-defined error codes are possible

4.10.2.2 TEE_Result TEES_RPMBRead (uint32_t *partition*, uint32_t *offset*, uint8_t * *data*, uint32_t *data_size*, uint8_t *type_flag*)

```
#include <rpmb.h>
```

Read data from RPMB Storage.

Parameters

in	<i>partition</i>	Partition number of RPMB storage
in	<i>offset</i>	Offset value of data. The start address is 0 in each partition
out	<i>data</i>	Buffer for data which read from RPMB
in	<i>data_size</i>	Size of data to be read from RPMB
in	<i>type_flag</i>	rpmb data size type (block or byte)

Return values

<i>TEE_SUCCESS</i>	in case of success
<i>error</i>	code in case of errors

4.10.2.3 TEE_Result TEES_RPMBWrite (uint32_t *partition*, uint32_t *offset*, uint8_t * *data*, uint32_t *data_size*, uint8_t *type_flag*)

```
#include <rpmb.h>
```

Write data to RPMB Storage.

Parameters

in	<i>partition</i>	Partition number of RPMB storage.
in	<i>offset</i>	Offset value of data. The start address is 0 in each partition
in	<i>data</i>	Buffer for data which write to RPMB
in	<i>data_size</i>	Size of data to be write to RPMB
in	<i>type_flag</i>	rpmb data size type (block or byte)

Return values

<i>TEE_SUCCESS</i>	in case of success
<i>error</i>	code in case of errors

4.11 Thread support library API

Data Structures

- struct `__pthread_once_t`
- struct `__pthread_attr_t`
- struct `__pthread_mutex_t`
- struct `__pthread_cond_t`
- struct `__pthread_condattr_t`

Macros

- #define `PTHREAD_STACK_MIN` (PAGE_SIZE)
- #define `MUTEX_STATE_UNLOCKED` 0
- #define `MUTEX_STATE_LOCKED` 1
- #define `PTHREAD_MUTEX_INITIALIZER` {{ `MUTEX_STATE_UNLOCKED` }, `PTHREAD_MUTEX_DEFAULT`, 0, 0 }
- #define `PTHREAD_ERRORCHECK_MUTEX_INITIALIZER_NP` { { `MUTEX_STATE_UNLOCKED` }, `PTHREAD_MUTEX_ERRORCHECK`, 0, 0 }
- #define `PTHREAD_RECURSIVE_MUTEX_INITIALIZER_NP` { { `MUTEX_STATE_UNLOCKED` }, `PTHREAD_MUTEX_RECURSIVE`, 0, 0 }
- #define `PTHREAD_ONCE_INIT` { `ATOMIC_INITIALIZER` }
- #define `PTHREAD_COND_INITIALIZER` { `ATOMIC_INITIALIZER` }
- *Object for static initialization of `pthread_cond_t` variables.*
- #define `pthread_sigmask`(how, set, oldset) sigprocmask(how, set, oldset)
Set signal mask for specified thread.

Typedefs

- typedef struct pthread_impl `pthread_impl_t`
- typedef uintptr_t `pthread_t`
- typedef uint32_t `pthread_mutexattr_t`
- typedef struct `__pthread_mutex_t` `pthread_mutex_t`
- typedef unsigned `pthread_key_t`
- typedef struct `__pthread_attr_t` `pthread_attr_t`
- typedef struct `__pthread_once_t` `pthread_once_t`
- typedef struct `__pthread_cond_t` `pthread_cond_t`
- typedef struct `__pthread_cond_t` `pthread_condattr_t`

Enumerations

- enum {
`PTHREAD_MUTEX_NORMAL` = 0, `PTHREAD_MUTEX_RECURSIVE`, `PTHREAD_MUTEX_ERRORCHECK`,
`PTHREAD_MUTEX_DEFAULT` = `PTHREAD_MUTEX_NORMAL`,
`PTHREAD_MUTEX_DESTROYED` = -1, `PTHREAD_MUTEX_ERRORCHECK_NP` = `PTHREAD_MUTEX_ERRORCHECK`,
`PTHREAD_MUTEX_RECURSIVE_NP` = `PTHREAD_MUTEX_RECURSIVE` }
types and states for mutex

Functions

- int [pthread_attr_init](#) ([pthread_attr_t](#) *attr)
The [pthread_attr_init\(\)](#) function initialize attribute struct.
- int [pthread_attr_destroy](#) ([pthread_attr_t](#) *attr)
The [pthread_attr_destroy\(\)](#) function destroy attribute struct.
- int [pthread_attr_setstacksize](#) ([pthread_attr_t](#) *attr, [size_t](#) size)
The [pthread_attr_setstacksize\(\)](#) function sets size of stack.
- int [pthread_create](#) ([pthread_t](#) *thread, const [pthread_attr_t](#) *attr, void *(*start_routine)(void *), void *arg)
The [pthread_create\(\)](#) function starts a new thread in the calling process. The new thread starts execution by invoking [start_routine\(\)](#) arg is passed as the sole argument of [start_routine\(\)](#).
- int [pthread_join](#) ([pthread_t](#) thread, void **retval)
Wait for the thread specified by *thread* to terminate. If that thread has already terminated, then returns immediately. The thread specified by *thread* must be joinable.
- int [pthread_once](#) ([pthread_once_t](#) *once_control, void(*init_routine)(void))
If any thread in a process with a *once_control* parameter makes a call to [pthread_once\(\)](#), the first call will summon the [init_routine\(\)](#), but subsequent calls will not. The *once_control* parameter determines whether the associated initialization routine has been called. The [init_routine\(\)](#) is complete upon return of [pthread_once\(\)](#).
- void * [pthread_getspecific](#) ([pthread_key_t](#) key)
Return the value currently bound to the specified *key* on behalf of the calling thread.
- int [pthread_setspecific](#) ([pthread_key_t](#) key, const void *value)
Associate a thread-specific *value* with a *key* obtained via a previous call to [pthread_key_create\(\)](#).
- int [pthread_key_create](#) ([pthread_key_t](#) *key, void(*destructor)(void *))
Create data key for data manipulation functions ([pthread_getspecific\(\)](#), [pthread_setspecific\(\)](#)). Multiple threads can call data manipulation functions with the same key. In this case all threads will have separate data.
- int [pthread_key_delete](#) ([pthread_key_t](#) key)
Delete data key and destructor associated with *key*. After key deletion there is no destructor will be called on thread exits.
- int [pthread_mutexattr_init](#) ([pthread_mutexattr_t](#) *attr)
Initialize mutex attributes object and initialize attributes with default values.
- int [pthread_mutexattr_destroy](#) ([pthread_mutexattr_t](#) *attr)
Destroy attributes object and make all attribute values are uninitialized.
- int [pthread_mutexattr_gettype](#) (const [pthread_mutexattr_t](#) *attr, int *type)
Get mutex type attribute associated with *attr* parameter.
- int [pthread_mutexattr_settype](#) ([pthread_mutexattr_t](#) *attr, int type)
Set mutex type attribute associated with *attr* parameter.
- int [pthread_mutex_lock](#) ([pthread_mutex_t](#) *mutex)
Lock mutex or wait while another thread is unlock currently locked mutex.
- int [pthread_mutex_trylock](#) ([pthread_mutex_t](#) *mutex)
Lock mutex or fail if mutex is already locked.
- int [pthread_mutex_unlock](#) ([pthread_mutex_t](#) *mutex)
Release lock on currently locked mutex.
- int [pthread_mutex_destroy](#) ([pthread_mutex_t](#) *mutex)
Destroy mutex object and make all associated data are uninitialized.
- int [pthread_mutex_init](#) ([pthread_mutex_t](#) *mutex, const [pthread_mutexattr_t](#) *attr)
Initialize mutex object *mutex* with attributes given by *attr* parameter. If *attr* parameter is NULL then default attributes will be used.
- int [pthread_cond_destroy](#) ([pthread_cond_t](#) *cond)
Destroy conditional variable object and make it uninitialized.
- int [pthread_cond_init](#) ([pthread_cond_t](#) *cond, const [pthread_condattr_t](#) *attr)

- Initialize conditional variable object `cond` with attributes given by `attr` parameter.*
- int `pthread_cond_signal` (`pthread_cond_t *cond`)
Unblock at least one of the threads that are blocked on the specified condition variable `cond` (if any threads are blocked on `cond`).
 - int `pthread_cond_broadcast` (`pthread_cond_t *cond`)
Wake up all threads locked by conditional variable `cond`.
 - int `pthread_cond_wait` (`pthread_cond_t *cond`, `pthread_mutex_t *mutex`)
Block on condition variable `cond`, while another thread will unblock this variable by `pthread_cond_broadcast()` / `pthread_cond_signal()` call.
 - `pthread_t pthread_self` (void)
Obtain ID of the calling thread.
 - int `pthread_kill` (`pthread_t thread`, int sig)
Send signal to specified thread.

4.11.1 Detailed Description

4.11.2 Data Structure Documentation

4.11.2.1 struct `__pthread_once_t`

Struct to avoid direct assignment of its single field, which is meant to be threaded atomically.

Data Fields

<code>atomic_t</code>	<code>already_executed</code>	
-----------------------	-------------------------------	--

4.11.2.2 struct `__pthread_attr_t`

Pthread attribute object type (not implemented yet).

Data Fields

<code>size_t</code>	<code>stacksize</code>	
---------------------	------------------------	--

4.11.2.3 struct `__pthread_mutex_t`

Pthread mutex object type.

Data Fields

unsigned	<code>attr</code>	
<code>pthread_t</code>	<code>caller_thread</code>	
unsigned	<code>count</code>	
<code>atomic_t</code>	<code>state</code>	

4.11.2.4 struct `__pthread_cond_t`

Struct to avoid direct assignment of its single field, which is meant to be threaded atomically.

Data Fields

<code>atomic_t</code>	<code>status</code>	
-----------------------	---------------------	--

4.11.2.5 struct `__pthread_condattr_t`

Condition variable attributes type. Not implemented

4.11.3 Macro Definition Documentation

4.11.3.1 `#define MUTEX_STATE_LOCKED 1`

```
#include <pthread.h>
```

Lock state for mutex.

4.11.3.2 `#define MUTEX_STATE_UNLOCKED 0`

```
#include <pthread.h>
```

Unlock state for mutex.

4.11.3.3 `#define PTHREAD_ERRORCHECK_MUTEX_INITIALIZER_NP { { MUTEX_STATE_UNLOCKED }, PTHREAD_MUTEX_ERRORCHECK, 0, 0 }`

```
#include <pthread.h>
```

`pthread_mutex_t` object for static mutex initialization with errorcheck.

4.11.3.4 `#define PTHREAD_MUTEX_INITIALIZER { { MUTEX_STATE_UNLOCKED }, PTHREAD_MUTEX_DEFAULT, 0, 0 }`

```
#include <pthread.h>
```

`pthread_mutex_t` object for static mutex initialization.

4.11.3.5 `#define PTHREAD_ONCE_INIT { ATOMIC_INITIALIZER }`

```
#include <pthread.h>
```

Object for static initialization of `pthread_once_t` variables.

4.11.3.6 #define PTHREAD_RECURSIVE_MUTEX_INITIALIZER_NP { { MUTEX_STATE_UNLOCKED }, PTHREAD_MUTEX_RECURSIVE, 0, 0 }

#include <pthread.h>

pthread_mutex_t object for static recursive mutex initialization.

4.11.3.7 #define pthread_sigmask(how, set, oldset) sigprocmask(how, set, oldset)

#include <pthread.h>

Set signal mask for specified thread.

Parameters

in	<i>how</i>	defines how to set mask, must be one of: SIG_BLOCK, SIG_UNBLOCK, SIG_SETMASK
in	<i>set</i>	if not NULL, signals mask to act on, otherwise ignored
in	<i>oldset</i>	if not NULL then here stored previous value of the signal mask

Return values

0	On success.
<i>EINVAL</i>	The value specified by <i>how</i> is invalid
<i>EFAULT</i>	The <i>set</i> or <i>oldset</i> argument points outside the process's allocated address space

Example:

```
sigset_t set, oldset;
memset(&set, 0xff, sizeof(set));
int err = pthread_sigmask(SIG_BLOCK, &set, &oldset);
if (!err) {
    do_some_work(...);
}
```

4.11.3.8 #define PTHREAD_STACK_MIN (PAGE_SIZE)

#include <pthread.h>

Minimum stack size.

4.11.4 Typedef Documentation

4.11.4.1 pthread_attr_t

#include <pthread.h>

Type for pthread attributes.

4.11.4.2 pthread_cond_t

```
#include <pthread.h>
```

Type for pthread conditional variable.

4.11.4.3 pthread_condattr_t

```
#include <pthread.h>
```

Type for attributes of pthread conditional variable.

4.11.4.4 pthread_impl_t

```
#include <pthread.h>
```

Internal pthread info representation.

4.11.4.5 pthread_key_t

```
#include <pthread.h>
```

Type for unique per-thread keys.

4.11.4.6 pthread_mutex_t

```
#include <pthread.h>
```

Pthread mutex type.

4.11.4.7 pthread_mutexattr_t

```
#include <pthread.h>
```

Mutex attribute variable.

4.11.4.8 pthread_once_t

```
#include <pthread.h>
```

Type for pthreads, created once.

4.11.4.9 pthread_t

```
#include <pthread.h>
```

Unique thread identification variable.

4.11.5 Enumeration Type Documentation

4.11.5.1 anonymous enum

```
#include <pthread.h>
```

types and states for mutex

Enumerator

- PTHREAD_MUTEX_NORMAL** Regular mutex.
- PTHREAD_MUTEX_RECURSIVE** Mutex with the concept of a lock count.
- PTHREAD_MUTEX_ERRORCHECK** Mutex with error checking.
- PTHREAD_MUTEX_DEFAULT** Regular mutex.
- PTHREAD_MUTEX_DESTROYED** Mutex state after [pthread_mutex_destroy\(\)](#).
- PTHREAD_MUTEX_ERRORCHECK_NP** Same as [PTHREAD_MUTEX_ERRORCHECK](#).
- PTHREAD_MUTEX_RECURSIVE_NP** Same as [PTHREAD_MUTEX_RECURSIVE](#).

4.11.6 Function Documentation

4.11.6.1 int pthread_attr_destroy (pthread_attr_t * attr)

```
#include <pthread.h>
```

The [pthread_attr_destroy\(\)](#) function destroy attribute struct.

Parameters

in	<i>attr</i>	is attribute struct.
----	-------------	----------------------

Return values

0	On success.
<i>EINVAL</i>	Invalid attr address

4.11.6.2 int pthread_attr_init (pthread_attr_t * attr)

```
#include <pthread.h>
```

The `pthread_attr_init()` function initialize attribute struct.

Parameters

in	<i>attr</i>	is attribute struct.
----	-------------	----------------------

Return values

0	On success.
<i>EINVAL</i>	Invalid attr address

4.11.6.3 int pthread_attr_setstacksize (pthread_attr_t * attr, size_t size)

```
#include <pthread.h>
```

The `pthread_attr_setstacksize()` function sets size of stack.

Parameters

in	<i>attr</i>	is attribute struct.
in	<i>size</i>	is wished size of stack.

Return values

0	On success.
<i>EINVAL</i>	size is less than PTHREAD_STACK_MIN.

4.11.6.4 int pthread_cond_broadcast (pthread_cond_t * cond)

```
#include <pthread.h>
```

Wake up all threads locked by conditional variable `cond`.

Parameters

out	<i>cond</i>	A pointer to the <code>pthread_cond_t</code> object for which user wants to unblock the threads.
-----	-------------	--

Return values

0	If successful.
<i>EINVAL</i>	The value <code>cond</code> does not refer to an initialized condition variable.

Example:

```
pthread_mutex_t rsrc_lock;
pthread_cond_t rsrc_add;
unsigned int resources;
void get_resources(int amount)
{
    pthread_mutex_lock(&rsrc_lock);
    while (resources < amount)
        pthread_cond_wait(&rsrc_add, &rsrc_lock);
    resources -= amount;
    pthread_mutex_unlock(&rsrc_lock);
}
void add_resources(int amount)
{
    pthread_mutex_lock(&rsrc_lock);
    resources += amount;
    pthread_cond_broadcast(&rsrc_add);
    pthread_mutex_unlock(&rsrc_lock);
}
```

4.11.6.5 int pthread_cond_destroy (pthread_cond_t * cond)

```
#include <pthread.h>
```

Destroy conditional variable object and make it uninitialized.

Parameters

out	<i>cond</i>	A pointer to the <code>pthread_cond_t</code> object that to destroy.
-----	-------------	--

Return values

0	On success.
<i>EBUSY</i>	The implementation has detected an attempt to destroy the object referenced by <code>cond</code> while it is referenced (for example, while being used in a <code>pthread_cond_wait()</code>) by another thread.
<i>EINVAL</i>	The value specified by <code>cond</code> is invalid.

Example:

```

struct list {
    pthread_mutex_t lm;
    ...
}
struct elt {
    key k;
    int busy;
    pthread_cond_t notbusy;
    ...
}
delete_elt(struct list *lp, struct elt *ep)
{
    pthread_mutex_lock(&lp->lm);
    assert(ep->busy);
    ... remove ep from list ...
    ep->busy = 0;
    pthread_cond_broadcast(&ep->notbusy);
    pthread_mutex_unlock(&lp->lm);
    pthread_cond_destroy(&ep->notbusy);
    free(ep);
}

```

4.11.6.6 int pthread_cond_init (pthread_cond_t * cond, const pthread_condattr_t * attr)

```
#include <pthread.h>
```

Initialize conditional variable object `cond` with attributes given by `attr` parameter.

Parameters

out	<code>cond</code>	A pointer to the <code>pthread_cond_t</code> object to initialize.
in	<code>attr</code>	NULL, or a pointer to a <code>pthread_condattr_t</code> object that specifies the attributes that you want to use for the <code>cond</code> .

Return values

<code>0</code>	On success it returns 0.
<code>EAGAIN</code>	The system lacked the necessary resources (other than memory) to initialise another condition variable.
<code>ENOMEM</code>	Insufficient memory exists to initialise the condition variable.
<code>EBUSY</code>	The implementation has detected an attempt to re-initialise the object referenced by <code>cond</code> , a previously initialised, but not yet destroyed, condition variable.
<code>EINVAL</code>	The value specified by <code>attr</code> is invalid.

Example:

```

// initialize a condition variable to its default value
ret = pthread_cond_init(&cv, NULL);

```

4.11.6.7 int pthread_cond_signal (pthread_cond_t * cond)

```
#include <pthread.h>
```

Unblock at least one of the threads that are blocked on the specified condition variable `cond` (if any threads are blocked on `cond`).

Parameters

out	<code>cond</code>	A pointer to the pthread_cond_t object for which user wants to unblock the threads.
-----	-------------------	---

Return values

0	If successful.
EINVAL	The value <code>cond</code> does not refer to an initialized condition variable.

Example:

```
pthread_mutex_t count_lock;
pthread_cond_t count_nonzero;
unsigned count;
void decrement_count()
{
    pthread_mutex_lock(&count_lock);
    while (count == 0)
        pthread_cond_wait(&count_nonzero, &count_lock);
    count = count - 1;
    pthread_mutex_unlock(&count_lock);
}
void increment_count()
{
    pthread_mutex_lock(&count_lock);
    if (count == 0)
        pthread_cond_signal(&count_nonzero);
    count = count + 1;
    pthread_mutex_unlock(&count_lock);
}
```

4.11.6.8 int pthread_cond_wait (pthread_cond_t * cond, pthread_mutex_t * mutex)

```
#include <pthread.h>
```

Block on condition variable `cond`, while another thread will unblock this variable by [pthread_cond_broadcast\(\)](#) / [pthread_cond_signal\(\)](#) call.

Parameters

out	<code>cond</code>	A pointer to the pthread_cond_t object that user wants the threads to block on.
out	<code>mutex</code>	The mutex that to unlock.

Return values

0	On success.
<i>EINVAL</i>	The value specified by <code>cond</code> or <code>mutex</code> is invalid.
<i>EINVAL</i>	Different mutexes were supplied for concurrent <code>pthread_cond_wait()</code> operations on the same condition variable.
<i>EPERM</i>	The mutex was not owned by the current thread at the time of the call.

Example:

```
pthread_mutex_t myMutex;
pthread_cond_t cond;
pthread_attr_t attr;
int cont;
void *anotherFunc(void*)
{
    printf("anotherFunc\n");
    pthread_mutex_lock(&myMutex);
    printf("waiting...\n");
    pthread_cond_wait(&cond, &myMutex);
    cont += 10;
    printf("slot\n");
    pthread_mutex_unlock(&myMutex);
    printf("mutex unlocked anotherFunc\n");
    printf("Done anotherFunc\n");
    pthread_exit(NULL);
}
```

4.11.6.9 int pthread_create (pthread_t * thread, const pthread_attr_t * attr, void (*)(void *) start_routine, void * arg)

```
#include <pthread.h>
```

The `pthread_create()` function starts a new thread in the calling process. The new thread starts execution by invoking `start_routine()` `arg` is passed as the sole argument of `start_routine()`.

Parameters

out	<i>thread</i>	NULL, or a pointer to a <code>pthread_t</code> object where the function can store the thread ID of the new thread.
in	<i>attr</i>	NULL, a pointer to a <code>pthread_attr_t</code> structure that specifies the attributes of the new thread.
in	<i>start_routine</i>	The routine where the thread begins, with <code>arg</code> as its only argument.
in	<i>arg</i>	single parameter of <code>start_routine()</code> function.

Return values

0	On success.
<i>EAGAIN</i>	The system lacked the necessary resources to create another thread, or the system-imposed limit on the total number of threads in a process would be exceeded.
<i>EPERM</i>	The caller does not have appropriate permission to set the required scheduling parameters or scheduling policy.
<i>EINVAL</i>	The attributes specified by <code>attr</code> are invalid.

Example:

```

void *foo(void *i) {
    int a = *((int *) i);
    free(i);
}

int main()
{
    pthread_t thread;
    int *arg = malloc(sizeof(*arg));
    *arg = 10;
    pthread_create(&thread, 0, foo, arg);
    return 0;
}

```

4.11.6.10 void* pthread_getspecific (pthread_key_t key)

```
#include <pthread.h>
```

Return the value currently bound to the specified key on behalf of the calling thread.

Parameters

in	key	The key associated with the data that user wants to get.
----	-----	--

Returns

Pointer to the data value, or NULL.

Example:

```

pthread_key_t buffer_key;

void buffer_key_destruct( void *value )
{
    free( value );
    pthread_setspecific( buffer_key, NULL );
}

char *lookup( void )
{
    char *string;
    string = (char *)pthread_getspecific( buffer_key );
    if( string == NULL ) {
        string = (char *) malloc( 32 );
        sprintf( string, "This is thread %d\n", pthread_self() );
        pthread_setspecific( buffer_key, (void *)string );
    }
    return( string );
}

void *function( void *arg )
{
    while( 1 ) {
        puts( lookup() );
    }
    return( 0 );
}

int main( void )
{
    pthread_key_create( &buffer_key,
                      &buffer_key_destruct );
    pthread_create( NULL, NULL, &function, NULL );
    // Let the threads run for 60 seconds
    sleep( 60 );

    return EXIT_SUCCESS;
}

```

4.11.6.11 `int pthread_join (pthread_t thread, void ** retval)`

```
#include <pthread.h>
```

Wait for the thread specified by `thread` to terminate. If that thread has already terminated, then returns immediately. The thread specified by `thread` must be joinable.

Parameters

in	<code>thread</code>	Target thread which termination is waited.
out	<code>retval</code>	NULL, or a double pointer to a location where the function can store the value passed to <code>pthread_exit()</code> by the target thread.

Return values

<code>0</code>	If successful.
<code>EINVAL</code>	The implementation has detected that the value specified by <code>thread</code> does not refer to a joinable thread.
<code>ESRCH</code>	No thread could be found corresponding to that specified by the given thread ID.
<code>EDEADLK</code>	A deadlock was detected or the value of <code>thread</code> specifies the calling thread.

Example:

```
void* thread_function(void)
{
    char *a = malloc(10);
    strcpy(a,"hello world");
    pthread_exit((void*)a);
}
int main()
{
    pthread_t thread_id;
    char *b;

    pthread_create (&thread_id, NULL,&thread_function, NULL);
    pthread_join(thread_id,(void**)&b);
    printf("b is %s",b);
}
```

4.11.6.12 `int pthread_key_create (pthread_key_t * key, void(*)(void *) destructor)`

```
#include <pthread.h>
```

Create data key for data manipulation functions (`pthread_getspecific()`, `pthread_setspecific()`). Multiple threads can call data manipulation functions with the same key. In this case all threads will have separate data.

Parameters

out	<code>key</code>	A pointer to a <code>pthread_key_t</code> object where the function can store the new key.
in	<code>destructor</code>	(optional) pointer to function to be called when you destroy the key.

Return values

0	On success.
<i>EINVAL</i>	The key value is invalid.

Example:

```
pthread_key_t buffer_key;

void buffer_key_destruct( void *value )
{
    free( value );
    pthread_setspecific( buffer_key, NULL );
}

char *lookup( void )
{
    char *string;
    string = (char *)pthread_getspecific( buffer_key );
    if( string == NULL ) {
        string = (char *) malloc( 32 );
        sprintf( string, "This is thread %d\n", pthread_self() );
        pthread_setspecific( buffer_key, (void *)string );
    }
    return( string );
}

void *function( void *arg )
{
    while( 1 ) {
        puts( lookup() );
    }
    return( 0 );
}

int main( void )
{
    pthread_key_create( &buffer_key,
                      &buffer_key_destruct );
    pthread_create( NULL, NULL, &function, NULL );
    // Let the threads run for 60 seconds
    sleep( 60 );

    return EXIT_SUCCESS;
}
```

4.11.6.13 int pthread_key_delete (pthread_key_t key)

```
#include <pthread.h>
```

Delete data key and destructor associated with *key*. After key deletion there is no destructor will be called on thread exits.

Parameters

in	<i>key</i>	The key, created by calling <code>pthread_key_create()</code> to delete.
----	------------	--

Return values

0	On success.
<i>EAGAIN</i>	The system lacked the necessary resources to create another thread-specific data key, or the system-imposed limit on the total number of keys per process { <code>PTHREAD_KEYS_MAX</code> } has been exceeded.
<i>ENOMEM</i>	Insufficient memory exists to create the key.

Example:

```
pthread_key_t buffer_key;
void buffer_key_destruct( void *value )
{
    free( value );
    pthread_setspecific( buffer_key, NULL );
}

char *lookup( void )
{
    char *string;

    string = (char *)pthread_getspecific( buffer_key );
    if( string == NULL ) {
        string = (char *) malloc( 32 );
        sprintf( string, "This is thread %d\n", pthread_self() );
        pthread_setspecific( buffer_key, (void *)string );
    }

    return( string );
}

void *function( void *arg )
{
    while( 1 ) {
        puts( lookup() );
    }
    return( 0 );
}

int main( void )
{
    pthread_key_create( &buffer_key,
                      &buffer_key_destruct );
    pthread_key_delete( buffer_key );
    return EXIT_SUCCESS;
}
```

4.11.6.14 int pthread_kill (pthread_t thread, int sig)

```
#include <pthread.h>
```

Send signal to specified thread.

Parameters

in	<i>thread</i>	thread to send signal
in	<i>sig</i>	signal id to send

Return values

0	On success.
<i>EINVAL</i>	The value specified by <i>sig</i> not a valid signal identifier
<i>EPERM</i>	The signal id specified by <i>sig</i> is not allowed to send from unprivileged task
<i>ESRCH</i>	Specified thread not found

Example:

```
int err = pthread_kill(thread, sig);
if (!err) {
    do_some_work(...);
}
```

4.11.6.15 int pthread_mutex_destroy (pthread_mutex_t * mutex)

```
#include <pthread.h>
```

Destroy mutex object and make all associated data are uninitialized.

Parameters

out	<i>mutex</i>	A pointer to the pthread_mutex_t object to destroy.
-----	--------------	---

Return values

0	On success.
<i>EBUSY</i>	The implementation has detected an attempt to destroy the object referenced by <i>mutex</i> while it is locked or referenced (for example, while being used in a pthread_cond_wait()) by another thread.
<i>EINVAL</i>	The value specified by <i>mutex</i> is invalid.

Example:

```
pthread_mutex_t demoMutex;
pthread_mutex_trylock(&demoMutex);
pthread_t writeToFile = pthread_self ();
unsigned short iterate;
for (iterate = 0; iterate < 10000; iterate++) {
    fprintf (fp, " %d ", iterate, 4);
    fprintf (fp, " %lu ", writeToFile, sizeof (pthread_t));
    fprintf (fp, "\n", writeToFile, 1);
}
pthread_mutex_unlock (&demoMutex);
pthread_mutex_destroy (&demoMutex);
pthread_exit (NULL);
```

4.11.6.16 int pthread_mutex_init (pthread_mutex_t * mutex, const pthread_mutexattr_t * attr)

```
#include <pthread.h>
```

Initialize mutex object *mutex* with attributes given by *attr* parameter. If *attr* parameter is NULL then default attributes will be used.

Parameters

out	<i>mutex</i>	A pointer to the pthread_mutex_t object to initialize.
in	<i>attr</i>	NULL, or a pointer to a pthread_mutexattr_t object that specifies the attributes that you want to use for the mutex.

Return values

0	On success.
<i>EAGAIN</i>	The system lacked the necessary resources (other than memory) to initialise another mutex.
<i>ENOMEM</i>	Insufficient memory exists to initialise the mutex.
<i>EPERM</i>	The caller does not have the privilege to perform the operation
<i>EBUSY</i>	The implementation has detected an attempt to re-initialise the object referenced by <code>mutex</code> , a previously initialised, but not yet destroyed, mutex.
<i>EINVAL</i>	The value specified by <code>attr</code> is invalid.

Example:

```
pthread_mutex_t demoMutex;
pthread_mutex_init(&Mutex, &Attr);
pthread_mutex_trylock(&demoMutex);
pthread_t    writeToFile = pthread_self ();
unsigned short iterate;
for (iterate = 0; iterate < 10000; iterate++) {
    fprintf (fp, "%d ", iterate, 4);
    fprintf (fp, "%lu ", writeToFile, sizeof (pthread_t));
    fprintf (fp, "\n", writeToFile, 1);
}
pthread_mutex_unlock (&demoMutex);
pthread_exit (NULL);
```

4.11.6.17 int pthread_mutex_lock (pthread_mutex_t * mutex)

```
#include <pthread.h>
```

Lock mutex or wait while another thread is unlock currently locked mutex.

Parameters

out	<i>mutex</i>	A pointer to the <code>pthread_mutex_t</code> object to lock.
-----	--------------	---

Return values

0	On success.
<i>EINVAL</i>	The value specified by <code>mutex</code> does not refer to an initialized mutex object.
<i>EAGAIN</i>	The mutex could not be acquired because the maximum number of recursive locks for mutex has been exceeded.
<i>EDEADLK</i>	The current thread already owns the mutex.

Example:

```
pthread_mutex_t count_mutex;
long long count;

void increment_count()
{
    pthread_mutex_lock(&count_mutex);
    count = count + 1;
    pthread_mutex_unlock(&count_mutex);
}
```

4.11.6.18 int pthread_mutex_trylock (pthread_mutex_t * mutex)

```
#include <pthread.h>
```

Lock mutex or fail if mutex is already locked.

Parameters

out	<i>mutex</i>	A pointer to the pthread_mutex_t object to lock.
-----	--------------	--

Return values

0	On success.
<i>EBUSY</i>	The mutex could not be acquired because it was already locked.
<i>EINVAL</i>	The value specified by <i>mutex</i> does not refer to an initialized mutex object.
<i>EAGAIN</i>	The mutex could not be acquired because the maximum number of recursive locks for <i>mutex</i> has been exceeded.

Example:

```
pthread_mutex_t demoMutex;
pthread_mutex_trylock(&demoMutex);
pthread_t      writeToFile = pthread_self ();
unsigned short iterate;
for (iterate = 0; iterate < 10000; iterate++) {
    fprintf (fp, " %d ", iterate, 4);
    fprintf (fp, " %lu ", writeToFile, sizeof (pthread_t));
    fprintf (fp, "\n", writeToFile, 1);
}
pthread_mutex_unlock (&demoMutex);
pthread_exit (NULL);
```

4.11.6.19 int pthread_mutex_unlock (pthread_mutex_t * mutex)

```
#include <pthread.h>
```

Release lock on currently locked mutex.

Parameters

out	<i>mutex</i>	A pointer to the pthread_mutex_t object to unlock.
-----	--------------	--

Return values

0	On success.
<i>EPERM</i>	The current thread does not own the mutex.

Example:

```
pthread_mutex_t demoMutex;
pthread_mutex_trylock(&demoMutex);
pthread_t      writeToFile = pthread_self ();
unsigned short iterate;
for (iterate = 0; iterate < 10000; iterate++) {
    fprintf (fp, " %d ", iterate, 4);
    fprintf (fp, " %lu ", writeToFile, sizeof (pthread_t));
    fprintf (fp, "\n", writeToFile, 1);
}
pthread_mutex_unlock (&demoMutex);
pthread_exit (NULL);
```

4.11.6.20 int pthread_mutexattr_destroy (pthread_mutexattr_t * attr)

```
#include <pthread.h>
```

Destroy attributes object and make all attribute values are uninitialized.

Parameters

in	<i>attr</i>	A pointer to the pthread_mutexattr_t object that user wants to get the attribute from.
----	-------------	--

Return values

0	On success.
<i>EINVAL</i>	The value specified by <i>attr</i> is invalid.

Example:

```
pthread_mutex_t mutex;

int main(int argc, char **argv)
{
    int rc = 0;
    pthread_mutexattr_t mta;

    printf("Create a default mutex attribute\n");
    rc = pthread_mutexattr_init(&mta);
    printf("Create the mutex using a mutex attributes object\n");
    rc = pthread_mutex_init(&mutex, &mta);
    printf("Destroy mutex attribute\n");
    rc = pthread_mutexattr_destroy(&mta);
    printf("Destroy mutex\n");
    rc = pthread_mutex_destroy(&mutex);
    return 0;
}
```

4.11.6.21 int pthread_mutexattr_gettype (const pthread_mutexattr_t * attr, int * type)

```
#include <pthread.h>
```

Get mutex type attribute associated with *attr* parameter.

Parameters

in	<i>attr</i>	A pointer to the pthread_mutexattr_t object to get the attribute from.
out	<i>type</i>	A pointer to location where the function can store the type.

Return values

0	On success.
EINVAL	The value specified by <i>attr</i> is invalid.

Example:

```
pthread_mutex_t Mutex;
int type;
pthread_mutexattr_t Attr;
pthread_mutexattr_init(&Attr);
pthread_mutexattr_settype(&Attr,
    PTHREAD_MUTEX_RECURSIVE);
pthread_mutexattr_gettype(&Attr, &type);
pthread_mutex_init(&Mutex, &Attr);
```

4.11.6.22 int pthread_mutexattr_init (pthread_mutexattr_t * attr)

```
#include <pthread.h>
```

Initialize mutex attributes object and initialize attributes with default values.

Parameters

in	<i>attr</i>	A pointer to the pthread_mutexattr_t object to initialize.
----	-------------	--

Return values

0	On success.
ENOMEM	Insufficient memory exists to initialize the mutex attributes object.

Example:

```
pthread_mutex_t mutex;

int main(int argc, char **argv)
{
    int rc = 0;
    pthread_mutexattr_t mta;

    printf("Create a default mutex attribute\n");
    rc = pthread_mutexattr_init(&mta);
    printf("Create the mutex using a mutex attributes object\n");
    rc = pthread_mutex_init(&mutex, &mta);
    printf("Destroy mutex attribute\n");
    rc = pthread_mutexattr_destroy(&mta);
    printf("Destroy mutex\n");
    rc = pthread_mutex_destroy(&mutex);
    return 0;
}
```

4.11.6.23 int pthread_mutexattr_settype (pthread_mutexattr_t * attr, int type)

```
#include <pthread.h>
```

Set mutex type attribute associated with `attr` parameter.

Parameters

out	<code>attr</code>	A pointer to the pthread_mutexattr_t object to set the attribute in.
in	<code>type</code>	Parameter type can be one of the following values: <ul style="list-style-type: none"> • PTHREAD_MUTEX_NORMAL - does not detect deadlocks. • PTHREAD_MUTEX_ERRORCHECK - lock function tries to lock already locked function will fail with error. • PTHREAD_MUTEX_RECURSIVE - allow lock already locked mutex by the same thread. • PTHREAD_MUTEX_DEFAULT - equal to PTHREAD_MUTEX_NORMAL.

Return values

0	On success.
EINVAL	The value <code>type</code> is invalid.
EINVAL	The value specified by <code>attr</code> is invalid.

Example:

```
pthread_mutex_t      mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutexattr_t  mta;

pthread_mutexattr_init(&mta);
pthread_mutexattr_settype(&mta, PTHREAD_MUTEX_RECURSIVE);
pthread_mutex_init(&mutex, &mta);
```

4.11.6.24 int pthread_once (pthread_once_t * once_control, void(*)(void) init_routine)

```
#include <pthread.h>
```

If any thread in a process with a `once_control` parameter makes a call to [pthread_once\(\)](#), the first call will summon the `init_routine()`, but subsequent calls will not. The `once_control` parameter determines whether the associated initialization routine has been called. The `init_routine()` is complete upon return of [pthread_once\(\)](#).

Parameters

out	<code>once_control</code>	A pointer to a pthread_once_t object that the function uses to determine whether or not to run the initialization code.
in	<code>init_routine</code>	The function that new thread will run.

Return values

<code>0</code>	Upon successful completion.
<code>EINVAL</code>	The implementation has detected that the value specified by <code>thread</code> does not refer to a joinable thread.
<code>ESRCH</code>	No thread could be found corresponding to that specified by the given thread ID.
<code>EDEADLK</code>	A deadlock was detected or the value of <code>thread</code> specifies the calling thread.

Example:

```
pthread_once_t once_control = PTHREAD_ONCE_INIT;
void library_init( void )
{
    // initialize the library
}

void library_entry_point1( void )
{
    pthread_once( &once_control, library_init );
    // do stuff for library_entry_point1...
}

void library_entry_point2( void )
{
    pthread_once( &once_control, library_init );
    // do stuff for library_entry_point1...
}
```

4.11.6.25 pthread_t pthread_self (void)

```
#include <pthread.h>
```

Obtain ID of the calling thread.

Returns

the ID of the calling thread.

Example:

```
pthread_t id = pthread_self();
if (id == wanted_id) {
    ...
}
```

4.11.6.26 int pthread_setspecific (pthread_key_t key, const void * value)

```
#include <pthread.h>
```

Associate a thread-specific value with a key obtained via a previous call to [pthread_key_create\(\)](#).

Parameters

out	<i>key</i>	The key associated with the data to set.
in	<i>value</i>	Pointer to buffer where to store value.

Return values

0	If successful.
ENOMEM	Insufficient memory to store threads specific data value.
EINVAL	Invalid thread specific data key.

Example:

```
pthread_key_t buffer_key;

void buffer_key_destruct( void *value )
{
    free( value );
    pthread_setspecific( buffer_key, NULL );
}

char *lookup( void )
{
    char *string;
    string = (char *)pthread_getspecific( buffer_key );
    if( string == NULL ) {
        string = (char *) malloc( 32 );
        sprintf( string, "This is thread %d\n", pthread_self() );
        pthread_setspecific( buffer_key, (void *)string );
    }
    return( string );
}

void *function( void *arg )
{
    while( 1 ) {
        puts( lookup() );
    }
    return( 0 );
}

int main( void )
{
    pthread_key_create( &buffer_key,
                      &buffer_key_destruct );
    pthread_create( NULL, NULL, &function, NULL );
    // Let the threads run for 60 seconds
    sleep( 60 );

    return EXIT_SUCCESS;
}
```

4.12 Math library API

Macros

- #define `isnan(x)` `_isnan(x)`
- #define `M_E` 2.7182818284590452354
- #define `M_PI` 3.14159265358979323846
- #define `M_PI_2` 1.57079632679489661923
- #define `M_PI_4` 0.78539816339744830962

Functions

- `_const_ double atan` (double x)
Arc tangent function.
- `_const_ float atanf` (float x)
Arc tangent function.
- `_const_ double atan2` (double y, double x)
Arc tangent function of two variables.
- `_const_ float atan2f` (float y, float x)
Arc tangent function of two variables.
- `_const_ double ceil` (double x)
ceiling function: smallest integral value not less than argument
- `_const_ float ceilf` (float x)
ceiling function: smallest integral value not less than argument
- `_const_ double pow` (double base, double exp)
Calculate the exponentiation.
- `_const_ float powf` (float base, float exp)
Calculate the exponentiation.
- `_const_ double sqrt` (double x)
Calculate the square root of x.
- `_const_ float sqrtf` (float x)
Calculate the square root of x.
- `_const_ double fabs` (double x)
Calculate the absolute value of x.
- `_const_ float fabsf` (float x)
Calculate the absolute value of x.
- `_const_ double round` (double x)
Calculate the integral value that is the nearest to x.
- `_const_ float roundf` (float x)
Calculate the integral value that is the nearest to x.
- `_const_ double sin` (double x)
Calculate the sine of an angle of x radians.
- `_const_ float sinf` (float x)
Calculate the sine of an angle of x radians.
- `_const_ double cos` (double x)
Calculate the cosine of an angle of x radians.
- `_const_ float cosf` (float x)
Calculate the cosine of an angle of x radians.
- `_const_ double exp` (double x)
base-e exponential function

- `_const_ float expf` (float x)
base-e exponential function
- `_const_ double log` (double x)
Calculate the natural logarithm.
- `_const_ float logf` (float x)
Calculate the natural logarithm.
- `_const_ double logb` (double x)
Calculate exponent of a floating-point value.
- `_const_ float logbf` (float x)
Calculate exponent of a floating-point value.
- `_const_ double fmax` (double x, double y)
Determine maximum of two floating-point numbers.
- `_const_ float fmaxf` (float x, float y)
Determine maximum of two floating-point numbers.
- `_const_ double fmin` (double x, double y)
Determine minimum of two floating-point numbers.
- `_const_ float fminf` (float x, float y)
Determine minimum of two floating-point numbers.
- `_const_ double scalbn` (double x, int exp)
Multiply floating-point number by integral power of radix.
- `_const_ float scalbnf` (float x, int exp)
Multiply floating-point number by integral power of radix.
- `_const_ double copysign` (double x, double y)
Copy sign of a number.
- `_const_ float copysignf` (float x, float y)
Copy sign of a number.
- `_const_ double floor` (double x)
Get largest integral value not greater than argument.
- `_const_ float floorf` (float x)
Get largest integral value not greater than argument.
- `_const_ double hypot` (double x, double y)
Euclidean distance function.
- `_const_ float hypotf` (float x, float y)
Euclidean distance function.
- `double modf` (double x, double *iptr)
extract signed integral and fractional values from floating-point number
- `float modff` (float x, float *iptr)
extract signed integral and fractional values from floating-point number

4.12.1 Detailed Description

4.12.2 Macro Definition Documentation

4.12.2.1 #define isnan(x) _isnan(x)

```
#include <math.h>
```

Parameters

in	x	- value to check for NaN
----	---	--------------------------

Returns

0 - is number, !0 - NaN

4.12.2.2 #define M_E 2.7182818284590452354

```
#include <math.h>
```

e - base of the natural logarithm

4.12.2.3 #define M_PI 3.14159265358979323846

```
#include <math.h>
```

pi - the ratio of a circle's circumference to its diameter

4.12.2.4 #define M_PI_2 1.57079632679489661923

```
#include <math.h>
```

pi/2

4.12.2.5 #define M_PI_4 0.78539816339744830962

```
#include <math.h>
```

pi/4

4.12.3 Function Documentation

4.12.3.1 _const_double atan (double x)

```
#include <math.h>
```

Arc tangent function.

Parameters

in	x	function argument
----	---	-------------------

Returns

arc tangent of function argument

4.12.3.2 `_const_double atan2 (double y, double x)`

```
#include <math.h>
```

Arc tangent function of two variables.

Parameters

in	y	function argument
in	x	function argument

Returns

principal value of the arc tangent of y/x in radians

4.12.3.3 `_const_float atan2f (float y, float x)`

```
#include <math.h>
```

Arc tangent function of two variables.

Parameters

in	y	function argument
in	x	function argument

Returns

principal value of the arc tangent of y/x in radians

4.12.3.4 `_const_float atanf (float x)`

```
#include <math.h>
```

Arc tangent function.

Parameters

in	x	function argument
----	---	-------------------

Returns

arc tangent of function argument

4.12.3.5 `_const_double ceil (double x)`

```
#include <math.h>
```

ceiling function: smallest integral value not less than argument

Parameters

in	x	function argument
----	---	-------------------

Returns

ceiling of x

4.12.3.6 `_const_float ceilf (float x)`

```
#include <math.h>
```

ceiling function: smallest integral value not less than argument

Parameters

in	x	function argument
----	---	-------------------

Returns

ceiling of x

4.12.3.7 `_const_double copysign (double x, double y)`

```
#include <math.h>
```

Copy sign of a number.

Parameters

in	x	function argument
in	y	sign source

Returns

value whose absolute value matches that of x, but whose sign bit matches that of y

4.12.3.8 `_const_float copysignf (float x, float y)`

```
#include <math.h>
```

Copy sign of a number.

Parameters

in	x	function argument
in	y	sign source

Returns

value whose absolute value matches that of x, but whose sign bit matches that of y

4.12.3.9 `_const_double cos (double x)`

```
#include <math.h>
```

Calculate the cosine of an angle of x radians.

Parameters

in	x	function argument in radians
----	---	------------------------------

Returns

the result of calculation

4.12.3.10 `_const_float cosf (float x)`

```
#include <math.h>
```

Calculate the cosine of an angle of x radians.

Parameters

in	x	function argument in radians
----	---	------------------------------

Returns

the result of calculation

4.12.3.11 `_const_double exp (double x)`

```
#include <math.h>
```

base-e exponential function

Parameters

in	x	function argument in radians
----	---	------------------------------

Returns

value of e raised to the power of x

4.12.3.12 `_const_float expf (float x)`

```
#include <math.h>
```

base-e exponential function

Parameters

in	x	function argument in radians
----	---	------------------------------

Returns

value of e raised to the power of x

4.12.3.13 `_const_double fabs (double x)`

```
#include <math.h>
```

Calculate the absolute value of x.

Parameters

in	x	function argument
----	---	-------------------

Returns

the result of calculation

4.12.3.14 `_const_float fabsf (float x)`

```
#include <math.h>
```

Calculate the absolute value of x.

Parameters

in	x	function argument
----	---	-------------------

Returns

the result of calculation

4.12.3.15 `_const_double floor (double x)`

```
#include <math.h>
```

Get largest integral value not greater than argument.

Parameters

in	x	function argument
----	---	-------------------

Returns

the floor of x

4.12.3.16 `_const_float floorf (float x)`

```
#include <math.h>
```

Get largest integral value not greater than argument.

Parameters

in	x	function argument
----	---	-------------------

Returns

the floor of x

4.12.3.17 `_const_double fmax (double x, double y)`

```
#include <math.h>
```

Determine maximum of two floating-point numbers.

Parameters

in	<i>x</i>	function argument
in	<i>y</i>	function argument

Returns

maximal value of *x* or *y*

4.12.3.18 `_const_float fmaxf (float x, float y)`

```
#include <math.h>
```

Determine maximum of two floating-point numbers.

Parameters

in	<i>x</i>	function argument
in	<i>y</i>	function argument

Returns

maximal value of *x* or *y*

4.12.3.19 `_const_double fmin (double x, double y)`

```
#include <math.h>
```

Determine minimum of two floating-point numbers.

Parameters

in	<i>x</i>	function argument
in	<i>y</i>	function argument

Returns

minimal value of *x* or *y*

4.12.3.20 `_const_float fminf (float x, float y)`

```
#include <math.h>
```

Determine minimum of two floating-point numbers.

Parameters

in	<i>x</i>	function argument
in	<i>y</i>	function argument

Returns

minimal value of *x* or *y*

4.12.3.21 `_const_double hypot (double x, double y)`

```
#include <math.h>
```

Euclidean distance function.

Parameters

in	<i>x</i>	function argument
in	<i>y</i>	function argument

Returns

length of a right-angled triangle with sides of length *x* and *y*

4.12.3.22 `_const_float hypotf (float x, float y)`

```
#include <math.h>
```

Euclidean distance function.

Parameters

in	<i>x</i>	function argument
in	<i>y</i>	function argument

Returns

length of a right-angled triangle with sides of length *x* and *y*

4.12.3.23 `_const_double log (double x)`

```
#include <math.h>
```

Calculate the natural logarithm.

Parameters

in	x	function argument
----	---	-------------------

Returns

the result of calculation

4.12.3.24 `_const_double logb (double x)`

```
#include <math.h>
```

Calculate exponent of a floating-point value.

Parameters

in	x	function argument
----	---	-------------------

Returns

the result of calculation

4.12.3.25 `_const_float logbf (float x)`

```
#include <math.h>
```

Calculate exponent of a floating-point value.

Parameters

in	x	function argument
----	---	-------------------

Returns

the result of calculation

4.12.3.26 `_const_float logf (float x)`

```
#include <math.h>
```

Calculate the natural logarithm.

Parameters

in	<i>x</i>	function argument
----	----------	-------------------

Returns

the result of calculation

4.12.3.27 `double modf (double x, double * iptr)`

```
#include <math.h>
```

extract signed integral and fractional values from floating-point number

Parameters

in	<i>x</i>	function argument
in,out	<i>iptr</i>	integral part is stored in the location pointed to by iptr

Returns

return the fractional part of *x*

4.12.3.28 `float modff (float x, float * iptr)`

```
#include <math.h>
```

extract signed integral and fractional values from floating-point number

Parameters

in	<i>x</i>	function argument
in,out	<i>iptr</i>	integral part is stored in the location pointed to by iptr

Returns

return the fractional part of *x*

4.12.3.29 `_const_double pow (double base, double exp)`

```
#include <math.h>
```

Calculate the exponentiation.

Parameters

in	<i>base</i>	base value
in	<i>exp</i>	exponent value

Returns

base raised to the power exponent

4.12.3.30 `_const_float powf (float base, float exp)`

```
#include <math.h>
```

Calculate the exponentiation.

Parameters

in	<i>base</i>	base value
in	<i>exp</i>	exponent value

Returns

base raised to the power exponent

4.12.3.31 `_const_double round (double x)`

```
#include <math.h>
```

Calculate the integral value that is the nearest to x.

Parameters

in	<i>x</i>	function argument
----	----------	-------------------

Returns

the result of calculation

4.12.3.32 `_const_float roundf (float x)`

```
#include <math.h>
```

Calculate the integral value that is the nearest to x.

Parameters

in	x	function argument
----	---	-------------------

Returns

the result of calculation

4.12.3.33 `_const_double scalbn (double x, int exp)`

```
#include <math.h>
```

Multiply floating-point number by integral power of radix.

Parameters

in	x	function argument
in	exp	exponenta

Returns

result of calculation

4.12.3.34 `_const_float scalbnf (float x, int exp)`

```
#include <math.h>
```

Multiply floating-point number by integral power of radix.

Parameters

in	x	function argument
in	exp	exponenta

Returns

result of calculation

4.12.3.35 `_const_double sin (double x)`

```
#include <math.h>
```

Calculate the sine of an angle of x radians.

Parameters

in	x	function argument in radians
----	---	------------------------------

Returns

the result of calculation

4.12.3.36 `_const_float sinf (float x)`

```
#include <math.h>
```

Calculate the sine of an angle of x radians.

Parameters

in	x	function argument in radians
----	---	------------------------------

Returns

the result of calculation

4.12.3.37 `_const_double sqrt (double x)`

```
#include <math.h>
```

Calculate the square root of x.

Parameters

in	x	function argument
----	---	-------------------

Returns

the result of calculation

4.12.3.38 `_const_float sqrtf (float x)`

```
#include <math.h>
```

Calculate the square root of x.

Parameters

in	x	function argument
----	---	-------------------

Returns

the result of calculation

4.13 Message queue library API

Macros

- #define `MQ_MAX_NAME` 1024

Typedefs

- typedef int `mqd_t`

Functions

- `mqd_t mq_open` (const char *pathname, int flags,...)
Create new message queue or open an existing queue.
- int `mq_unlink` (const char *pathname)
Remove specified message queue name.
- int `mq_close` (`mqd_t` fd)
Close a message queue descriptor.
- int `mq_send` (`mqd_t` fd, const char *msg_ptr, size_t msg_len, unsigned msg_prio)
Send a message to a message queue.
- `ssize_t mq_receive` (`mqd_t` fd, char *msg_ptr, size_t msg_len, unsigned *msg_prio)
Receive a message from a message queue.

4.13.1 Detailed Description

4.13.2 Macro Definition Documentation

4.13.2.1 #define MQ_MAX_NAME 1024

```
#include <mqqueue.h>
```

max length of the mq name

4.13.3 Typedef Documentation

4.13.3.1 mqd_t

```
#include <mqqueue.h>
```

type for message queue functions

4.13.4 Function Documentation

4.13.4.1 `int mq_close (mqd_t fd)`

```
#include <mqqueue.h>
```

Close a message queue descriptor.

Parameters

in	<i>fd</i>	message queue descriptor
----	-----------	--------------------------

Returns

0 on success
-1 with `errno` on error

4.13.4.2 `mqd_t mq_open (const char * pathname, int flags, ...)`

```
#include <mqqueue.h>
```

Create new message queue or open an existing queue.

Parameters

in	<i>pathname</i>	mqqueue identifier
in	<i>flags</i>	control flags

Returns

message queue descriptor on success
-1 with `errno` on error

4.13.4.3 `ssize_t mq_receive (mqd_t fd, char * msg_ptr, size_t msg_len, unsigned * msg_prio)`

```
#include <mqqueue.h>
```

Receive a message from a message queue.

Parameters

in	<i>fd</i>	message queue descriptor
in, out	<i>msg_ptr</i>	message pointer
in	<i>msg_len</i>	length of the message
in	<i>msg_prio</i>	priority of the message

Returns

number of bytes in the received message on success
-1 with `errno` on error

4.13.4.4 `int mq_send (mqd_t fd, const char * msg_ptr, size_t msg_len, unsigned msg_prio)`

```
#include <mqeue.h>
```

Send a message to a message queue.

Parameters

in	<i>fd</i>	message queue descriptor
in,out	<i>msg_ptr</i>	message pointer
in	<i>msg_len</i>	length of the message
in	<i>msg_prio</i>	priority of the message

Returns

0 on success
-1 with `errno` on error

4.13.4.5 `int mq_unlink (const char * pathname)`

```
#include <mqeue.h>
```

Remove specified message queue name.

Parameters

in	<i>pathname</i>	name of the path
----	-----------------	------------------

Returns

0 on success
-1 with `errno` on error

4.14 Socket library API

Functions

- int [accept](#) (int sockfd, struct sockaddr *addr, int *addrlen)
Accept a connection on a socket.
- int [bind](#) (int sockfd, const struct sockaddr *addr, int addrlen)
Bind a name to a socket.
- int [connect](#) (int sockfd, const struct sockaddr *addr, int addrlen)
Initiate a connection on a socket.
- int [getsockopt](#) (int sockfd, int level, int optname, void *optval, int *optlen)
Get options on socket.
- int [setsockopt](#) (int sockfd, int level, int optname, void *optval, int optlen)
Get options on socket.
- int [listen](#) (int sockfd, int backlog)
Listen for connections on a socket.
- int [socket](#) (int socket_family, int socket_type, int protocol)
Create endpoint for communication.
- [ssize_t recv](#) (int sockfd, void *buf, size_t len, int flags)
Receive a message from a socket.
- [ssize_t send](#) (int sockfd, const void *buf, size_t len, int flags)
Send a message to a socket.
- int [socketpair](#) (int socket_family, int socket_type, int protocol, int sv[2])
Create a pair of connected sockets.
- [ssize_t recvmsg](#) (int sockfd, struct msghdr *msg, int flags)
Receive multiple message on a socket.
- [ssize_t sendmsg](#) (int sockfd, struct msghdr *msg, int flags)
Send multiple message on a socket.

4.14.1 Detailed Description

4.14.2 Function Documentation

4.14.2.1 int accept (int sockfd, struct sockaddr * addr, int * addrlen)

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

Accept a connection on a socket.

Parameters

in	<i>sockfd</i>	listening socket descriptor
in	<i>addr</i>	pointer to a sockaddr structure
in,out	<i>addrlen</i>	size of structure pointed to by addr

Returns

nonnegative file descriptor for the accepted socket on success
-1 with errno on error

4.14.2.2 `int bind (int sockfd, const struct sockaddr * addr, int addrlen)`

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

Bind a name to a socket.

Parameters

in	<i>sockfd</i>	socket descriptor
in	<i>addr</i>	address to the socket
in	<i>addrlen</i>	size of the address structure

Returns

0 on success
-1 with `errno` on error

4.14.2.3 `int connect (int sockfd, const struct sockaddr * addr, int addrlen)`

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

Initiate a connection on a socket.

Parameters

in	<i>sockfd</i>	socket descriptor
in	<i>addr</i>	address to the socket
in	<i>addrlen</i>	size of the address structure

Returns

0 on success
-1 with `errno` on error

4.14.2.4 `int getsockopt (int sockfd, int level, int optname, void * optval, int * optlen)`

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

Get options on socket.

Parameters

in	<i>sockfd</i>	socket descriptor
in	<i>level</i>	socket API level
in	<i>optname</i>	specified options
in, out	<i>optval</i>	buffer in which the value of the requested options are to be returned
in, out	<i>optlen</i>	the size of the buffer pointed to by <code>optval</code>

Returns

0 on success
-1 with `errno` on error

4.14.2.5 int listen (int sockfd, int backlog)

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

Listen for connections on a socket.

Parameters

in	<i>sockfd</i>	socket descriptor
in	<i>backlog</i>	maximum length to which the queue of pending connections for sockfd may grow

Returns

0 on success
-1 with errno on error

4.14.2.6 ssize_t recv (int sockfd, void * buf, size_t len, int flags)

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

Receive a message from a socket.

Parameters

in	<i>sockfd</i>	socket descriptor
in	<i>buf</i>	pointer to a message buffer
in	<i>len</i>	message length
in	<i>flags</i>	type of receiving

Returns

number of bytes received on success
-1 with errno on error

4.14.2.7 ssize_t recvmsg (int sockfd, struct msghdr * msg, int flags)

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

Receive multiple message on a socket.

Parameters

in	<i>sockfd</i>	socket descriptor
in	<i>msg</i>	pointer to an array of msghdr structures
in	<i>flags</i>	type of receiving

Returns

number of messages received in msg on success
-1 with errno on error

4.14.2.8 `ssize_t send (int sockfd, const void * buf, size_t len, int flags)`

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

Send a message to a socket.

Parameters

in	<i>sockfd</i>	socket descriptor
in,out	<i>buf</i>	pointer to a message buffer
in	<i>len</i>	message length
in	<i>flags</i>	type of sending

Returns

number of bytes sent on success
-1 with `errno` on error

4.14.2.9 `ssize_t sendmsg (int sockfd, struct msghdr * msg, int flags)`

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

Send multiple message on a socket.

Parameters

in	<i>sockfd</i>	socket descriptor
out	<i>msg</i>	pointer to an array of <code>msghdr</code> structures
in	<i>flags</i>	type of sending

Returns

number of messages sent from `msg` on success
-1 with `errno` on error

4.14.2.10 `int setsockopt (int sockfd, int level, int optname, void * optval, int optlen)`

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

Get options on socket.

Parameters

in	<i>sockfd</i>	socket descriptor
in	<i>level</i>	socket API level
in	<i>optname</i>	specified options
in,out	<i>optval</i>	buffer in which the value of the requested options are to be returned
in	<i>optlen</i>	the size of the buffer pointed to by <code>optval</code>

Returns

0 on success
-1 with `errno` on error

4.14.2.11 int socket (int *socket_family*, int *socket_type*, int *protocol*)

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

Create endpoint for communication.

Parameters

in	<i>socket_family</i>	communication domain
in	<i>socket_type</i>	communication semantics specifier
in	<i>protocol</i>	particular protocol to be used with the socket

Returns

nonnegative file descriptor for the new socket on success
-1 with `errno` on error

4.14.2.12 int socketpair (int *socket_family*, int *socket_type*, int *protocol*, int *sv*[2])

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

Create a pair of connected sockets.

Parameters

in	<i>socket_family</i>	communication domain
in	<i>socket_type</i>	communication semantics specifier
in	<i>protocol</i>	particular protocol to be used with the socket
in	<i>sv</i>	file descriptors

Returns

0 on success
-1 with `errno` on error

4.15 Auxiliary API

Files

- file [error.h](#)
- file [mman.h](#)
- file [phys.h](#)

Data Structures

- struct [cpu_set_t](#)
- struct [tm](#)
- struct [__uuid_t](#)
wrapper for `uuid` type. [More...](#)

Macros

- #define [assert](#)(expr) `__assert__(expr, __FILE__, __LINE__)`
Abort the program if assertion is false.
- #define [EPERM](#) 1 /* Operation not permitted */
- #define [ENOENT](#) 2 /* No such file or directory */
- #define [ESRCH](#) 3 /* No such process */
- #define [EINTR](#) 4 /* Interrupted system call */
- #define [EIO](#) 5 /* I/O error */
- #define [ENXIO](#) 6 /* No such device or address */
- #define [E2BIG](#) 7 /* Argument list too long */
- #define [ENOEXEC](#) 8 /* Exec format error */
- #define [EBADF](#) 9 /* Bad file number */
- #define [ECHILD](#) 10 /* No child processes */
- #define [EAGAIN](#) 11 /* Try again */
- #define [ENOMEM](#) 12 /* Out of memory */
- #define [EACCES](#) 13 /* Permission denied */
- #define [EFAULT](#) 14 /* Bad address */
- #define [ENOTBLK](#) 15 /* Block device required */
- #define [EBUSY](#) 16 /* Device or resource busy */
- #define [EEXIST](#) 17 /* File exists */
- #define [EXDEV](#) 18 /* Cross-device link */
- #define [ENODEV](#) 19 /* No such device */
- #define [ENOTDIR](#) 20 /* Not a directory */
- #define [EISDIR](#) 21 /* Is a directory */
- #define [EINVAL](#) 22 /* Invalid argument */
- #define [ENFILE](#) 23 /* File table overflow */
- #define [EMFILE](#) 24 /* Too many [open](#) files */
- #define [ENOTTY](#) 25 /* Not a typewriter */
- #define [ETXTBSY](#) 26 /* Text file busy */
- #define [EFBIG](#) 27 /* File too large */
- #define [ENOSPC](#) 28 /* No space left on device */
- #define [ESPIPE](#) 29 /* Illegal seek */
- #define [EROFS](#) 30 /* Read-only file system */
- #define [EMLINK](#) 31 /* Too many links */
- #define [EPIPE](#) 32 /* Broken pipe */

- #define **EDOM** 33 /* Math argument out of domain of func */
- #define **ERANGE** 34 /* Math result not representable */
- #define **EDEADLK** 35 /* Resource deadlock would occur */
- #define **ENAMETOOLONG** 36 /* File name too long */
- #define **ENOLCK** 37 /* No record locks available */
- #define **ENOSYS** 38 /* Function not implemented */
- #define **ENOTEMPTY** 39 /* Directory not empty */
- #define **ELOOP** 40 /* Too many symbolic links encountered */
- #define **EWOULDBLOCK EAGAIN** /* Operation would block */
- #define **ENOMSG** 42 /* No message of desired type */
- #define **EIDRM** 43 /* Identifier removed */
- #define **ECHRNG** 44 /* Channel number out of range */
- #define **EL2NSYNC** 45 /* Level 2 not synchronized */
- #define **EL3HLT** 46 /* Level 3 halted */
- #define **EL3RST** 47 /* Level 3 reset */
- #define **ELNRNG** 48 /* Link number out of range */
- #define **EUNATCH** 49 /* Protocol driver not attached */
- #define **ENOCSI** 50 /* No CSI structure available */
- #define **EL2HLT** 51 /* Level 2 halted */
- #define **EBADE** 52 /* Invalid exchange */
- #define **EBADR** 53 /* Invalid request descriptor */
- #define **EXFULL** 54 /* Exchange full */
- #define **ENOANO** 55 /* No anode */
- #define **EBADRQC** 56 /* Invalid request code */
- #define **EBADSLT** 57 /* Invalid slot */
- #define **EDEADLOCK EDEADLK**
- #define **EBFONT** 59 /* Bad font file format */
- #define **ENOSTR** 60 /* Device not a stream */
- #define **ENODATA** 61 /* No data available */
- #define **ETIME** 62 /* Timer expired */
- #define **ENOSR** 63 /* Out of streams resources */
- #define **ENONET** 64 /* Machine is not on the network */
- #define **ENOPKG** 65 /* Package not installed */
- #define **EREMOTE** 66 /* Object is remote */
- #define **ENOLINK** 67 /* Link has been severed */
- #define **EADV** 68 /* Advertise error */
- #define **ESRMNT** 69 /* Srmount error */
- #define **ECOMM** 70 /* Communication error on **send** */
- #define **EPROTO** 71 /* Protocol error */
- #define **EMULTIHOP** 72 /* Multihop attempted */
- #define **EDOTDOT** 73 /* RFS specific error */
- #define **EBADMSG** 74 /* Not a data message */
- #define **EOVERFLOW** 75 /* Value too large for defined data type */
- #define **ENOTUNIQ** 76 /* Name not unique on network */
- #define **EBADFD** 77 /* File descriptor in bad state */
- #define **EREMCHG** 78 /* Remote address changed */
- #define **ELIBACC** 79 /* Can not access a needed shared library */
- #define **ELIBBAD** 80 /* Accessing a corrupted shared library */
- #define **ELIBSCN** 81 /* .lib section in a.out corrupted */
- #define **ELIBMAX** 82 /* Attempting to link in too many shared libraries */
- #define **ELIBEXEC** 83 /* Cannot exec a shared library directly */
- #define **EILSEQ** 84 /* Illegal byte sequence */
- #define **ERESTART** 85 /* Interrupted system call should be restarted */
- #define **ESTRPIPE** 86 /* Streams pipe error */
- #define **EUSERS** 87 /* Too many users */

- #define ENOTSOCK 88 /* Socket operation on non-socket */
- #define EDESTADDRREQ 89 /* Destination address required */
- #define EMSGSIZE 90 /* Message too long */
- #define EPROTOTYPE 91 /* Protocol wrong type for socket */
- #define ENOPROTOOPT 92 /* Protocol not available */
- #define EPROTONOSUPPORT 93 /* Protocol not supported */
- #define ESOCKTNOSUPPORT 94 /* Socket type not supported */
- #define EOPNOTSUPP 95 /* Operation not supported on transport endpoint */
- #define EPFNOSUPPORT 96 /* Protocol family not supported */
- #define EAFNOSUPPORT 97 /* Address family not supported by protocol */
- #define EADDRINUSE 98 /* Address already in use */
- #define EADDRNOTAVAIL 99 /* Cannot assign requested address */
- #define ENETDOWN 100 /* Network is down */
- #define ENETUNREACH 101 /* Network is unreachable */
- #define ENETRESET 102 /* Network dropped connection because of reset */
- #define ECONNABORTED 103 /* Software caused connection abort */
- #define ECONNRESET 104 /* Connection reset by peer */
- #define ENOBUFS 105 /* No buffer space available */
- #define EISCONN 106 /* Transport endpoint is already connected */
- #define ENOTCONN 107 /* Transport endpoint is not connected */
- #define ESHUTDOWN 108 /* Cannot send after transport endpoint shutdown */
- #define ETOOMANYREFS 109 /* Too many references: cannot splice */
- #define ETIMEDOUT 110 /* Connection timed out */
- #define ECONNREFUSED 111 /* Connection refused */
- #define EHOSTDOWN 112 /* Host is down */
- #define EHOSTUNREACH 113 /* No route to host */
- #define EALREADY 114 /* Operation already in progress */
- #define EINPROGRESS 115 /* Operation now in progress */
- #define ESTALE 116 /* Stale file handle */
- #define EUCLEAN 117 /* Structure needs cleaning */
- #define ENOTNAM 118 /* Not a XENIX named type file */
- #define ENAVAIL 119 /* No XENIX semaphores available */
- #define EISNAM 120 /* Is a named type file */
- #define EREMOTEIO 121 /* Remote I/O error */
- #define EDQUOT 122 /* Quota exceeded */
- #define ECANCELED 125 /* Operation canceled */
- #define ENOKEY 126 /* Required key not available */
- #define errno (*get_errno_addr())
- #define PRId16 __16_PREFIX "d"
- #define PRId32 "d"
- #define PRId64 __64_PREFIX "d"
- #define PRIu16 __16_PREFIX "u"
- #define PRIu32 "u"
- #define PRIu64 __64_PREFIX "u"
- #define PRIx16 __16_PREFIX "x"
- #define PRIx32 "x"
- #define PRIx64 __64_PREFIX "x"
- #define SCNd16 __16_PREFIX "d"
- #define SCNd32 "d"
- #define SCNd64 __64_PREFIX "d"
- #define SCNu16 __16_PREFIX "u"
- #define SCNu32 "u"
- #define SCNu64 __64_PREFIX "u"
- #define SCNx16 __16_PREFIX "x"
- #define SCNx32 "x"

- #define `SCNx64 __64_PREFIX "x"`
- #define `CHAR_BIT 8`
- #define `SCHAR_MAX (127)`
- #define `SCHAR_MIN (-128)`
- #define `UCHAR_MAX (255)`
- #define `USHRT_MAX (0xFFFFU)`
- #define `SHRT_MAX (32767)`
- #define `SHRT_MIN (-32768)`
- #define `INT_MAX ((int)(~0U>>1))`
- #define `INT_MIN (-INT_MAX - 1)`
- #define `LONG_MAX ((long)(~0UL>>1))`
- #define `LONG_MIN (-LONG_MAX - 1)`
- #define `UINT_MAX (~0U)`
- #define `ULONG_MAX (~0UL)`
- #define `ULLONG_MAX (~0ULL)`
- #define `LLONG_MAX ((long long)(~0ULL>>1))`
- #define `LLONG_MIN ((long long)(-LLONG_MAX - 1))`
- #define `_POSIX_THREAD_KEYS_MAX 12`
- #define `PTHREAD_KEYS_MAX _POSIX_THREAD_KEYS_MAX`
- #define `M_CACHE_PAGES 1`
- #define `AUTO_BUFFER_SIZE 1024`
- #define `BIT_PER_CPU (1)`
- #define `MAX_CPUS (32)`
- #define `BITS_TO_CPU_MASK(bits) (((bits) + BITS_PER_LONG - 1) / BITS_PER_LONG)`
- #define `BITMAP_ELT(cpu) ((cpu) / BITS_PER_LONG)`
- #define `__CPUMASK(cpu) (1L << ((cpu) % BITS_PER_LONG))`
- #define `DECLARE_BITMAP(name, bits) unsigned long name[BITS_TO_CPU_MASK(bits)]`
- #define `CPU_ZERO(cpusetp)`
- #define `CPU_SET(cpu, cpusetp)`
- #define `CPU_CLR(cpu, cpusetp)`
- #define `CPU_ISSET(cpu, cpusetp)`
- #define `EOF (-1)`
- #define `MAP_ANONYMOUS (1 << 0)`
- #define `MAP_POPULATE (1 << 1)`
- #define `MAP_FIXED (1 << 2)`
- #define `MAP_PRIVATE (1 << 3)`
- #define `MAP_SHARED (1 << 4)`
- #define `PROT_NONE 0`
- #define `PROT_READ 1`
- #define `PROT_WRITE 2`
- #define `PROT_EXEC 4`
- #define `PGOFF_SHIFT 12`
- #define `PHYS_DEV_NAME "phys://"`
- #define `PHYS_DEV_NAME_LEN (sizeof(PHYS_DEV_NAME) - 1)`
- #define `MAP_PHYS_NON_SECURE (1 << 29)`
- #define `MAP_PHYS_NON_CACHED (1 << 28)`
- #define `MAP_FAILED ((void *)-1)`
- #define `NUM_SECONDS_IN_MIN (60)`
- #define `NUM_MILLIS_IN_SEC (1000)`
- #define `NUM_NANOS_IN_USEC (1000)`
- #define `NUM_NANOS_IN_MILLI (1000000L)`
- #define `NUM_NANOS_IN_SEC (1000000000ULL)`
- #define `TEMP_FAILURE_RETRY(expression)`
Recall function if it was interrupted by signal.
- #define `UUID_STRING_LEN 37`
- #define `uuid_unparse_lower(uu, out) uuid_unparse((uu), (out))`
- #define `uuid_generate_time(x) uuid_generate(x)`

Typedefs

- typedef int [errno_t](#)
- typedef int [ssize_t](#)
- typedef int [pid_t](#)
- typedef int [uid_t](#)
- typedef int [gid_t](#)
- typedef long long [time_t](#)
- typedef int64_t [off_t](#)
- typedef unsigned int [mode_t](#)
- typedef struct [__uuid_t](#) [__uuid_t](#)
- typedef [__uuid_t](#) [uuid_t](#)
- typedef void(* [constraint_handler_t](#)) (const char *restrict msg, void *restrict ptr, [errno_t](#) error)

Functions

- static int [isspace](#) (int c)
Check for white-space characters. These are: space, form-feed ('\f'), newline ('\n'), carriage return ('\r'), horizontal tab ('\t'), and vertical tab ('\v').
- static int [isascii](#) (int c)
Check whether c is a 7-bit unsigned char value that fits into the ASCII character set.
- static int [isupper](#) (int c)
Check for an uppercase letter.
- static int [islower](#) (int c)
Check for a lowercase letter.
- static int [isalpha](#) (int c)
Check for an alphabetic character; it is equivalent to (isupper(c) || islower(c))
- static int [isdigit](#) (int c)
Check for a digit (0 through 9)
- static int [isalnum](#) (int c)
Check for an alphanumeric character; it is equivalent to (isalpha(c) || isdigit(c)).
- static int [isblank](#) (int c)
Check for a blank character; that is, a space or a tab.
- static int [isxdigit](#) (int c)
Check for hexadecimal digits, that is, one of 0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F.
- static int [isprint](#) (int c)
Check for any printable character including space.
- static int [isgraph](#) (int c)
Check for any printable character except space.
- static int [ispunct](#) (int c)
Check for any printable character which is not a space or an alphanumeric character.
- static int [iscntrl](#) (int c)
Check for a control character.
- static int [toupper](#) (int c)
Convert lowercase letter to uppercase.
- static int [tolower](#) (int c)
Convert uppercase letter to lowercase.
- int * [get_errno_addr](#) (void)
This function should NOT be used directly, use 'errno' instead.
- int [open](#) (const char *pathname, int flags,...)

- Open a device by specifying its namespace path. Device driver operations should be previously registered with the namespace framework.
- int `printf_no_alloc` (const char *fmt,...)
Prints to ringbuffer. If resulting string exceeds AUTO_BUFFER_SIZE, cuts it off to AUTO_BUFFER_SIZE.
 - int `sched_yield` (void)
Causes the calling thread to relinquish the CPU. The thread is moved to the end of the queue for its static priority and a new thread gets to run.
 - int `sched_setaffinity` (pid_t pid, size_t cpusetsize, cpu_set_t *mask)
Sets the CPU affinity mask of the process whose ID is pid to the value specified by mask. If pid is zero, then the calling process is used.
 - int `sched_getaffinity` (pid_t pid, size_t cpusetsize, cpu_set_t *mask)
Writes the affinity mask of the process whose ID is pid into the cpu_set_t structure pointed to by mask.
 - int `printf` (const char *fmt,...)
Format and print string.
 - int `fprintf` (FILE *_restrict_ stream, const char *_restrict_ fmt,...)
The fprintf() is equivalent to the printf(), but uses a custom file handle.
 - int `vfprintf` (FILE *_restrict_ stream, const char *_restrict_ fmt, va_list ap)
The vfprintf() is equivalent to the fprintf(), but uses an argument list.
 - int `fflush` (FILE *stream)
Flush a stream.
 - int `sprintf` (char *s, const char *fmt,...)
format and string and save it to 's'.
 - int `printf_s` (const char *restrict fmt,...)
format and print string.
 - int `vsprintf_s` (char *restrict s, rsize_t n, const char *restrict format, va_list arg)
Write formatted data from variable length argument list to string.
 - int `vsprintf_s` (char *restrict s, rsize_t n, const char *restrict format, va_list arg)
Write formatted data from variable argument list to sized buffer.
 - int `sprintf_s` (char *restrict s, rsize_t n, const char *restrict format,...)
format string and save it to 's'.
 - int `snprintf_s` (char *restrict s, rsize_t n, const char *restrict format,...)
Format string and save it to 's'.
 - int `sscanf_s` (const char *restrict s, const char *restrict format,...)
Reads data safely from buf and stores them according to parameter fmt into the locations given by the additional arguments.
 - int `vsscanf_s` (const char *restrict s, const char *restrict format, va_list arg)
vsscanf unformat a buffer into a list of arguments.
 - int `snprintf` (char *s, size_t count, const char *fmt,...)
format and string and save it to 's'.
 - int `vsprintf` (char *buffer, const char *format, va_list args)
Write formatted data from variable argument list to string.
 - int `vsprintf` (char *buffer, size_t size, const char *format, va_list args)
Write formatted data from variable argument list to sized buffer.
 - int `sscanf` (const char *buf, const char *fmt,...)
Reads data from buf and stores them according to parameter fmt into the locations given by the additional arguments.
 - int `asprintf` (char **strp, const char *fmt,...)
print to allocated string.
 - int `vasprintf` (char **strp, const char *fmt, va_list args)
print to allocated string.
 - int `vasprintf_s` (char **strp, const char *fmt, va_list args)
Print to allocated string in secure mode.

- int [putchar](#) (int ch)
writes a character to log.
- int [puts](#) (const char *s)
writes the string s and a trailing newline to log(dmesg).
- int [vsscanf](#) (const char *buffer, const char *fmt, va_list args)
vsscanf unformat a buffer into a list of arguments.
- [errno_t memcpy_s](#) (void *restrict dest, rsize_t dest_max, const void *restrict src, rsize_t n)
Check arguments and copy src sized memory area to dest. Memory must not be overlapped. To copy overlapped area use [memmove_s\(\)](#) function.
- [errno_t memmove_s](#) (void *dest, rsize_t dest_max, const void *src, rsize_t n)
Check arguments and copy src sized memory area to dest. Memory may be overlapped.
- [errno_t memset_s](#) (void *block, rsize_t block_max, int c, rsize_t n)
Check arguments and fill n bytes of block sized memory with c constant value.
- size_t [strnlen_s](#) (const char *s, size_t s_max)
Check arguments and calculate the length of the s fixed-size string, excluding the terminating null byte ('\0').
- [errno_t strcpy_s](#) (char *restrict dest, rsize_t dest_max, const char *restrict src)
Check arguments and copy the src string, including the terminating null byte ('\0'), to the dest sized buffer. If n is bigger than size of src then the remaining characters (after '\0') are unspecified.
- [errno_t strncpy_s](#) (char *restrict dest, rsize_t dest_max, const char *restrict src, rsize_t n)
Check arguments and copy at most n bytes of the src string, including the terminating null byte ('\0') to the dest sized buffer.
- [errno_t strcat_s](#) (char *restrict dest, rsize_t dest_max, const char *restrict src)
Check arguments and append the src string to the dest string, overwriting the terminating null byte ('\0') at the end of dest, and add a terminating null byte.
- [errno_t strncat_s](#) (char *restrict dest, rsize_t dest_max, const char *restrict src, rsize_t n)
Check arguments and append at most n bytes of src string to the dest string, overwriting the terminating null byte ('\0') at the end of dest and then adds a terminating null byte.
- [errno_t strerror_s](#) (char *buf, rsize_t bufmax, [errno_t](#) errnum)
Check arguments and return a pointer to a buf string that describes the errnum error code.
- void * [memcpy](#) (void *dest, const void *src, size_t n)
Copy memory area from src to dst. The memory must not overlap. To copy overlapped area use [memmove\(\)](#) function.
- void * [memmove](#) (void *dest, const void *src, size_t n)
Copy memory area from src to dest. The memory may overlap.
- int [memcmp](#) (const void *s1, const void *s2, size_t n)
Compare fist n bytes of memory pointed by s1 and s2.
- void * [memset](#) (void *block, int c, size_t size)
Fill size bytes with a constant value.
- size_t [strlen](#) (const char *s)
Calculate the length of the string s, excluding the terminating null byte ('\0').
- size_t [strnlen](#) (const char *s, size_t n)
Calculate the length of the fixed-size string s, excluding the terminating null byte ('\0').
- char * [strcpy](#) (char *dest, const char *src)
Copy the string pointed to by src, including the terminating null byte ('\0'), to the buffer pointed to by dest.
- char * [strncpy](#) (char *dest, const char *src, size_t n)
Copy at most n bytes of the string pointed to by src, including the terminating null byte ('\0'), to the buffer pointed to by dest.
- char * [strdup](#) (const char *s)
Return a pointer to a new string which is a duplicate of the string s. Memory for the new string is obtained with [malloc\(\)](#).
- char * [strstr](#) (const char *text, const char *pattern)
Find the first occurrence of the substring pattern in the string text. The terminating null bytes ('\0') are not compared.

- char * **strncat** (char *dest, const char *src, size_t n)
Append the src string to the dest string, overwriting the terminating null byte ('\0') at the end of dest, and then adds a terminating null byte.
- size_t **strlen** (char *dest, const char *src, size_t n)
Append the NUL-terminated string src to the end of dest string, overwriting the terminating null byte ('\0') at the end of dest, and guarantee to NUL-terminate the result.
- char * **strcat** (char *dest, const char *src)
Append the src string to the dest string, overwriting the terminating null byte ('\0') at the end of dest, and then adds a terminating null byte.
- int **strcmp** (const char *s1, const char *s2)
Compare the two strings s1 and s2.
- int **strncmp** (const char *s1, const char *s2, size_t n)
Compare at most n bytes of two strings s1 and s2.
- char * **strrchr** (const char *s, int c)
Find last occurrence of character c in string s.
- const char * **strerror** (int errnum)
Return a pointer to a string that describes the error code passed in the argument errnum.
- void * **memchr** (const void *s, int c, size_t n)
Find a character in an area of memory.
- char * **strchr** (const char *s, int c)
Find first occurrence of character c in string s.
- char * **strchrnul** (const char *s, int c)
Find first occurrence of character c in string s.
- size_t **strspn** (const char *s, const char *accept)
Calculate the length (in bytes) of the initial segment of s which consists entirely of bytes in accept.
- size_t **strcspn** (const char *s, const char *reject)
Calculate the length of the initial segment of s which consists entirely of bytes not in reject.
- char * **strtok** (char *str, const char *delim)
Function breaks a string into a sequence of zero or more nonempty tokens.
- char * **strtok_r** (char *_restrict_ s, const char *_restrict_ sep, char **_restrict_ p)
Function breaks a string into a sequence of zero or more nonempty tokens.
- int **strerror_r** (int errnum, char *buf, size_t buflen)
Returns the error string in the user-supplied buf of length buflen. XSI-compliant version.
- int **ioctl** (int fd, int request, unsigned long data)
Manipulates the underlying device parameters of special files.
- void * **mmap** (void *addr, size_t len, int prot, int flags, int fd, off_t offset)
- int **munmap** (void *addr, size_t length)
- int **nanosleep** (struct timespec *req, struct timespec *rem)
*nanosleep() suspends the execution of the calling thread until either at least the time specified in *req has elapsed, or the delivery of a signal that triggers the invocation of a handler in the calling thread or that terminates the process. If the call is interrupted by a signal handler, nanosleep() returns -1, sets errno to EINTR, and writes the remaining time into the structure pointed to by rem unless rem is NULL. The value of *req can then be used to call nanosleep() again and complete the specified pause.*
- int **clock_gettime** (clockid_t clk_id, struct timespec *ts)
The function retrieves the time of the specified clock clk_id. This function is non-blocking, except CLOCK_REALTIME case. In this case function can sleep and can be interrupted with -1 result and EINTR errno.
- time_t **time** (time_t *time)
Get the current calendar time as a value of type time_t.
- int **alarm** (unsigned int seconds)
Set the real-time timer to expire in seconds seconds.
- int **setitimer** (int which, const struct itimerval *new_value, struct itimerval *old_value)
Set value of an interval timer.

- int [getitimer](#) (int which, struct itimerval *curr_value)
Get value of an interval timer.
- unsigned int [arm_timer_get_frequency](#) (void)
The function retrieves the frequency of ARM timer.
- void [timeadd](#) (const struct timespec *a, const struct timespec *b, struct timespec *res)
The function adds two dates.
- void [timesub](#) (const struct timespec *a, const struct timespec *b, struct timespec *res)
The function subtracts two dates.
- int64_t [timespec_to_ms](#) (const struct timespec *a)
The function converts time from timespec format to milliseconds.
- uint64_t [timespec_to_nsec](#) (const struct timespec *a)
The function converts time from timespec format to nanoseconds.
- void [ms_to_timespec](#) (int64_t t, struct timespec *a)
The function converts time in milliseconds to to timespec format.
- void [_exit](#) (int status)
Terminate the calling process "immediately". Any open file descriptors belonging to the process are closed; process's parent is sent a SIGCHLD signal.
- void [_exit_thread](#) (unsigned long status)
Terminate the calling thread "immediately".
- int [profil](#) (unsigned short *buf, size_t bufsiz, size_t offset, unsigned int scale)
Provide a means to find out in what areas your program spends most of its time. The argument `buf` points to `bufsiz` bytes of core. Every virtual 10 milliseconds, the user's program counter (PC) is examined: `offset` is subtracted and the result is multiplied by `scale` and divided by 65536. If the resulting value is less than `bufsiz`, then the corresponding entry in `buf` is incremented. If `buf` is NULL, profiling is disabled.
- [pid_t getpid](#) (void)
Return the process ID of the calling process.
- [pid_t gettid](#) (void)
Return the thread ID of the calling thread.
- int [getcpu](#) (void)
Return the number of CPU on which current thread is performed.
- int [getcluster](#) (void)
Return the cluster id on which current thread is performed.
- int [unlink](#) (const char *pathname)
Remove a link to a file.
- int [ftruncate](#) (int fd, int size)
Cause file referenced by `fd` to be truncated to a `size` of precisely length bytes.
- int [close](#) (int fd)
Deallocate the file descriptor indicated by `fd`. To deallocate means to make the file descriptor available for return by subsequent calls to `open()` or other functions that allocate file descriptors. All outstanding record locks owned by the process on the file associated with the file descriptor shall be removed (that is, unlocked).
- [ssize_t read](#) (int fd, void *buf, size_t count)
Attempt to read `count` bytes from the file associated with the open file descriptor, `fd`, into the buffer pointed to by `buf`.
- [ssize_t write](#) (int fd, const void *buf, size_t count)
Attempt to write `count` bytes from buffer pointed to by `buf` to the file associated with the open file descriptor, `fd`.
- int [fstat](#) (int fd, struct stat *buf)
Get status of a file with a descriptor `fd`.
- int [stat](#) (const char *pathname, struct stat *buf)
Get status of a file `pathname`.
- int [lseek](#) (int fd, int offset, int whence)
Reposition read/write file offset.

- void **uuid_unparse** (const **uuid_t** *uu, char *out)
Convert binary representation of UUID to string.
- void **uuid_unparse_upper** (const **uuid_t** *uu, char *out)
Convert binary representation of UUID to string.
- int **uuid_parse** (const char *in, **uuid_t** *uu)
Convert an input UUID string into binary representation.
- void **uuid_generate** (**uuid_t** *out)
The *uuid_generate* function creates a new universally unique identifier (UUID).
- int **uuid_is_null** (const **uuid_t** *uu)
Check if UUID is null.
- void **uuid_clear** (**uuid_t** *uu)
set value to zero UUID.
- int **uuid_compare** (const **uuid_t** *uu1, const **uuid_t** *uu2)
Compare the two supplied uuid variables *uu1* and *uu2* to each other.
- **constraint_handler_t** **set_constraint_handler_s** (**constraint_handler_t** handler)
Set the *handler* to be handler.
- void **invoke_constraint_handler_s** (const char *msg, const char *file, const char *function, uint32_t line, **errno_t** error)
Print *msg* if constraint was caused.
- void **abort_handler_s** (const char *restrict msg, void *restrict ptr, **errno_t** error)
Abort system if constraint was caused.
- void **ignore_handler_s** (const char *restrict msg, void *restrict ptr, **errno_t** error)
Returns to the caller without performing any actions.
- void **exit** (int status)
Cause normal process termination and return the value of *status* & 0377 to the parent.
- static __inline__ int **abs** (int j)
Compute the absolute value of the integer argument *__n*.
- void **abort** (void)
Cause abnormal process termination.
- long **strtol** (const char *nptr, char **endptr, int base)
Convert the initial part of the string in *nptr* to a long integer value according to the given *base*, which must be between 2 and 36 inclusive, or be the special value 0.
- unsigned long **strtoul** (const char *cp, char **endp, int base)
Convert the initial part of the string in *nptr* to a unsigned long integer value according to the given *base*, which must be between 2 and 36 inclusive, or be the special value 0.
- double **strtod** (const char *nptr, char **endptr)
the initial portion of the string pointed to by *nptr* to double.
- long long **strtoll** (const char *nptr, char **endptr, int base)
Convert the initial part of the string in *nptr* to a long long integer value according to the given *base*, which must be between 2 and 36 inclusive, or be the special value 0.
- unsigned long long **strtoull** (const char *cp, char **endp, int base)
Convert the initial part of the string in *nptr* to a unsigned long long integer value according to the given *base*, which must be between 2 and 36 inclusive, or be the special value 0.
- float **strtof** (const char *nptr, char **endptr)
the initial portion of the string pointed to by *nptr* to float.
- long double **strtold** (const char *nptr, char **endptr)
the initial portion of the string pointed to by *nptr* to long double.
- int **atexit** (void(*func)(void))
Register the given function to be called at normal process termination.
- void * **malloc** (size_t size)
Allocate *size* bytes and return a pointer to the allocated memory.
- void **free** (void *ptr)

- Free the memory space pointer to by ptr.*

 - void * **calloc** (size_t nmemb, size_t size)
Allocate memory for an array of nmemb elements of size bytes each and return a pointer to the allocated memory.
 - void * **realloc** (void *ptr, size_t size)
Change the size of the memory block pointed to by ptr to size bytes.
 - void **qsort** (void *base, size_t nmemb, size_t size, int(*compar)(const void *, const void *))
Sort an array.
 - void **qsort_r** (void *base, size_t nmemb, size_t size, int(*compar)(const void *, const void *, void *), void *arg)
Sort an array.

Variables

- FILE * **stdin**
Standard input stream (stub).
- FILE * **stdout**
Standard output stream.
- FILE * **stderr**
Standard error stream.

4.15.1 Detailed Description

4.15.2 Data Structure Documentation

4.15.2.1 struct cpu_set_t

cpu set structure

Data Fields

unsigned long	bits[(((1)*(32))+BITS_PER_LONG-1)/BITS_PER_LONG]	
---------------	--	--

4.15.2.2 struct tm

time structure

Data Fields

int	tm_hour	Hours. [0-23]
int	tm_isdst	DST. [-1/0/1]
int	tm_mday	Day. [1-31]
int	tm_min	Minutes. [0-59]
int	tm_mon	Month. [0-11]
int	tm_sec	Seconds. [0-60] (1 leap second)
int	tm_wday	Day of week. [0-6]
int	tm_yday	Days in year.[0-365]
int	tm_year	Year - 1900.

4.15.2.3 struct __uuid_t

wrapper for uuid type.

Data Fields

uint8_t	clockSeqAndNode[8]	unique identifier for specific system.
uint16_t	timeHiAndVersion	time specific part of UUID and version number.
uint32_t	timeLow	time specific part of UUID.
uint16_t	timeMid	time specific part of UUID.

4.15.3 Macro Definition Documentation

4.15.3.1 #define __CPUMASK(*cpu*) (1L << ((*cpu*) % BITS_PER_LONG))

```
#include <sched.h>
```

The value of cpu mask

4.15.3.2 #define _POSIX_THREAD_KEYS_MAX 12

```
#include <limits.h>
```

The number of data keys per process.

POSIX requires to support at least 128 keys (Linux supports up to 1024), but we can't afford such luxury

4.15.3.3 #define assert(*expr*) __assert__(*expr*, __FILE__, __LINE__)

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

Abort the program if assertion is false.

Parameters

in	<i>expr</i>	expression to check
----	-------------	---------------------

Returns

- if *expr* == true - do nothing
- if *expr* != true - in debug mode terminates application

4.15.3.4 #define AUTO_BUFFER_SIZE 1024

```
#include <print_no_alloc.h>
```

Buffer size.

4.15.3.5 #define BIT_PER_CPU (1)

```
#include <sched.h>
```

The number cpu bits

4.15.3.6 #define BITMAP_ELT(*cpu*) ((*cpu*) / BITS_PER_LONG)

```
#include <sched.h>
```

The bit map element

4.15.3.7 #define BITS_TO_CPU_MASK(*bits*) (((*bits*) + BITS_PER_LONG - 1) / BITS_PER_LONG)

```
#include <sched.h>
```

The number bits of cpu mask

4.15.3.8 #define CHAR_BIT 8

```
#include <limits.h>
```

Number of bits in a type char

4.15.3.9 #define CPU_CLR(*cpu*, *cpusetp*)

```
#include <sched.h>
```

Value:

```
{
    size_t __cpu = (cpu);
    __cpu < CHAR_BIT * (sizeof(cpu_set_t))
    ? ((cpusetp)->bits)[BITMAP_ELT(__cpu)] &= ~__CPUMASK (__cpu)
    : 0;
}
```

The macros for clear cpu

4.15.3.10 #define CPU_ISSET(*cpu*, *cpusetp*)

```
#include <sched.h>
```

Value:

```
{
    size_t __cpu = (cpu);
    __cpu < CHAR_BIT * (sizeof(cpu_set_t))
    ? (((cpusetp)->bits)[BITMAP_ELT(__cpu)] & __CPUMASK (__cpu)) != 0
    : 0;
}
```

The macros for check is spu is set

4.15.3.11 #define CPU_SET(*cpu*, *cpusetp*)

```
#include <sched.h>
```

Value:

```
{
    size_t __cpu = (cpu);
    __cpu < CHAR_BIT * (sizeof(cpu_set_t))
    ? (((cpusetp)->bits)[BITMAP_ELT(__cpu)] |= __CPUMASK (__cpu))
    : 0;
}
```

The macros for set cpu to cpusetp

4.15.3.12 #define CPU_ZERO(*cpusetp*)

```
#include <sched.h>
```

Value:

```
{
    size_t i;
    size_t imax = (sizeof(cpu_set_t)) / sizeof(unsigned long);
    unsigned long *bits = (cpusetp)->bits;
    for (i = 0; i < imax; ++i)
        bits[i] = 0;
}
```

The macros for set cpu to zero

4.15.3.13 #define DECLARE_BITMAP(*name*, *bits*) unsigned long name[BITS_TO_CPU_MASK(bits)]

```
#include <sched.h>
```

The short type of bitmap

4.15.3.14 #define E2BIG 7 /* Argument list too long */

```
#include <core/error.h>
```

Argument list too long.

4.15.3.15 #define EACCES 13 /* Permission denied */

```
#include <core/error.h>
```

Permission denied.

4.15.3.16 #define EADDRINUSE 98 /* Address already in use */

```
#include <core/error.h>
```

Address already in use.

4.15.3.17 #define EADDRNOTAVAIL 99 /* Cannot assign requested address */

```
#include <core/error.h>
```

Cannot assign requested address.

4.15.3.18 #define EADV 68 /* Advertise error */

```
#include <core/error.h>
```

Advertise error.

4.15.3.19 #define EAFNOSUPPORT 97 /* Address family not supported by protocol */

```
#include <core/error.h>
```

Address family not supported by protocol.

4.15.3.20 #define EAGAIN 11 /* Try again */

```
#include <core/error.h>
```

Try again.

4.15.3.21 #define EALREADY 114 /* Operation already in progress */

```
#include <core/error.h>
```

Operation already in progress.

4.15.3.22 #define EBADE 52 /* Invalid exchange */

```
#include <core/error.h>
```

Invalid exchange.

4.15.3.23 #define EBADF 9 /* Bad file number */

```
#include <core/error.h>
```

Bad file number.

4.15.3.24 #define EBADFD 77 /* File descriptor in bad state */

```
#include <core/error.h>
```

File descriptor in bad state.

4.15.3.25 #define EBADMSG 74 /* Not a data message */

```
#include <core/error.h>
```

Not a data message.

4.15.3.26 #define EBADR 53 /* Invalid request descriptor */

```
#include <core/error.h>
```

Invalid request descriptor.

4.15.3.27 #define EBADRQC 56 /* Invalid request code */

```
#include <core/error.h>
```

Invalid request code.

4.15.3.28 #define EBADSLT 57 /* Invalid slot */

```
#include <core/error.h>
```

Invalid slot.

4.15.3.29 #define EBFONT 59 /* Bad font file format */

```
#include <core/error.h>
```

Bad font file format.

4.15.3.30 #define EBUSY 16 /* Device or resource busy */

```
#include <core/error.h>
```

Device or resource busy.

4.15.3.31 #define ECANCELED 125 /* Operation canceled */

```
#include <core/error.h>
```

Operation canceled.

4.15.3.32 #define ECHILD 10 /* No child processes */

```
#include <core/error.h>
```

No child processes.

4.15.3.33 #define ECHRNG 44 /* Channel number out of range */

```
#include <core/error.h>
```

Channel number out of range.

4.15.3.34 #define ECOMM 70 /* Communication error on send */

```
#include <core/error.h>
```

Communication error on send.

4.15.3.35 #define ECONNABORTED 103 /* Software caused connection abort */

```
#include <core/error.h>
```

Software caused connection abort.

4.15.3.36 #define ECONNREFUSED 111 /* Connection refused */

```
#include <core/error.h>
```

Connection refused.

4.15.3.37 #define ECONNRESET 104 /* Connection reset by peer */

```
#include <core/error.h>
```

Connection reset by peer.

4.15.3.38 #define EDEADLK 35 /* Resource deadlock would occur */

```
#include <core/error.h>
```

Resource deadlock would occur.

4.15.3.39 #define EDEADLOCK EDEADLK

```
#include <core/error.h>
```

Resource deadlock would occur.

4.15.3.40 #define EDESTADDRREQ 89 /* Destination address required */

```
#include <core/error.h>
```

Destination address required.

4.15.3.41 #define EDOM 33 /* Math argument out of domain of func */

```
#include <core/error.h>
```

Math argument out of domain of func.

4.15.3.42 #define EDOTDOT 73 /* RFS specific error */

```
#include <core/error.h>
```

RFS specific error.

4.15.3.43 #define EDQUOT 122 /* Quota exceeded */

```
#include <core/error.h>
```

Quota exceeded.

4.15.3.44 #define EEXIST 17 /* File exists */

```
#include <core/error.h>
```

File exists.

4.15.3.45 #define EFAULT 14 /* Bad address */

```
#include <core/error.h>
```

Bad address.

4.15.3.46 #define EFBIG 27 /* File too large */

```
#include <core/error.h>
```

File too large.

4.15.3.47 #define EHOSTDOWN 112 /* Host is down */

```
#include <core/error.h>
```

Host is down.

4.15.3.48 #define EHOSTUNREACH 113 /* No route to host */

```
#include <core/error.h>
```

No route to host.

4.15.3.49 #define EIDRM 43 /* Identifier removed */

```
#include <core/error.h>
```

Identifier removed.

4.15.3.50 #define EILSEQ 84 /* Illegal byte sequence */

```
#include <core/error.h>
```

Illegal byte sequence.

4.15.3.51 #define EINPROGRESS 115 /* Operation now in progress */

```
#include <core/error.h>
```

Operation now in progress.

4.15.3.52 #define EINTR 4 /* Interrupted system call */

```
#include <core/error.h>
```

Interrupted system call.

4.15.3.53 #define EINVAL 22 /* Invalid argument */

```
#include <core/error.h>
```

Invalid argument.

4.15.3.54 #define EIO 5 /* I/O error */

```
#include <core/error.h>
```

I/O error.

4.15.3.55 #define EISCONN 106 /* Transport endpoint is already connected */

```
#include <core/error.h>
```

Transport endpoint is already connected.

4.15.3.56 #define EISDIR 21 /* Is a directory */

```
#include <core/error.h>
```

Is a directory.

4.15.3.57 #define EISNAM 120 /* Is a named type file */

```
#include <core/error.h>
```

Is a named type file.

4.15.3.58 #define EL2HLT 51 /* Level 2 halted */

```
#include <core/error.h>
```

Level 2 halted.

4.15.3.59 #define EL2NSYNC 45 /* Level 2 not synchronized */

```
#include <core/error.h>
```

Level 2 not synchronized.

4.15.3.60 #define EL3HLT 46 /* Level 3 halted */

```
#include <core/error.h>
```

Level 3 halted.

4.15.3.61 #define EL3RST 47 /* Level 3 reset */

```
#include <core/error.h>
```

Level 3 reset.

4.15.3.62 #define ELIBACC 79 /* Can not access a needed shared library */

```
#include <core/error.h>
```

Can not access a needed shared library.

4.15.3.63 #define ELIBBAD 80 /* Accessing a corrupted shared library */

```
#include <core/error.h>
```

Accessing a corrupted shared library.

4.15.3.64 #define ELIBEXEC 83 /* Cannot exec a shared library directly */

```
#include <core/error.h>
```

Cannot exec a shared library directly.

4.15.3.65 #define ELIBMAX 82 /* Attempting to link in too many shared libraries */

```
#include <core/error.h>
```

Attempting to link in too many shared libraries.

4.15.3.66 #define ELIBSCN 81 /* .lib section in a.out corrupted */

```
#include <core/error.h>
```

.lib section in a.out corrupted.

4.15.3.67 #define ELNRNG 48 /* Link number out of range */

```
#include <core/error.h>
```

Link number out of range.

4.15.3.68 #define ELOOP 40 /* Too many symbolic links encountered */

```
#include <core/error.h>
```

Too many symbolic links encountered.

4.15.3.69 #define EMFILE 24 /* Too many open files */

```
#include <core/error.h>
```

Too many open files.

4.15.3.70 #define EMLINK 31 /* Too many links */

```
#include <core/error.h>
```

Too many links.

4.15.3.71 #define EMSGSIZE 90 /* Message too long */

```
#include <core/error.h>
```

Message too long.

4.15.3.72 #define EMULTIHOP 72 /* Multihop attempted */

```
#include <core/error.h>
```

Multihop attempted.

4.15.3.73 #define ENAMETOOLONG 36 /* File name too long */

```
#include <core/error.h>
```

File name too long.

4.15.3.74 #define ENAVAIL 119 /* No XENIX semaphores available */

```
#include <core/error.h>
```

No XENIX semaphores available.

4.15.3.75 #define ENETDOWN 100 /* Network is down */

```
#include <core/error.h>
```

Network is down.

4.15.3.76 #define ENETRESET 102 /* Network dropped connection because of reset */

```
#include <core/error.h>
```

Network dropped connection because of reset.

4.15.3.77 #define ENETUNREACH 101 /* Network is unreachable */

```
#include <core/error.h>
```

Network is unreachable.

4.15.3.78 #define ENFILE 23 /* File table overflow */

```
#include <core/error.h>
```

File table overflow.

4.15.3.79 #define ENOANO 55 /* No anode */

```
#include <core/error.h>
```

No anode.

4.15.3.80 #define ENOBUFS 105 /* No buffer space available */

```
#include <core/error.h>
```

No buffer space available.

4.15.3.81 #define ENOCSI 50 /* No CSI structure available */

```
#include <core/error.h>
```

No CSI structure available.

4.15.3.82 #define ENODATA 61 /* No data available */

```
#include <core/error.h>
```

No data available.

4.15.3.83 #define ENODEV 19 /* No such device */

```
#include <core/error.h>
```

No such device.

4.15.3.84 #define ENOENT 2 /* No such file or directory */

```
#include <core/error.h>
```

No such file or directory.

4.15.3.85 #define ENOEXEC 8 /* Exec format error */

```
#include <core/error.h>
```

Exec format error.

4.15.3.86 #define ENOKEY 126 /* Required key not available */

```
#include <core/error.h>
```

Required key not available.

4.15.3.87 #define ENOLCK 37 /* No record locks available */

```
#include <core/error.h>
```

No record locks available.

4.15.3.88 #define ENOLINK 67 /* Link has been severed */

```
#include <core/error.h>
```

Link has been severed.

4.15.3.89 #define ENOMEM 12 /* Out of memory */

```
#include <core/error.h>
```

Out of memory.

4.15.3.90 #define ENOMSG 42 /* No message of desired type */

```
#include <core/error.h>
```

No message of desired type.

4.15.3.91 #define ENONET 64 /* Machine is not on the network */

```
#include <core/error.h>
```

Machine is not on the network.

4.15.3.92 #define ENOPKG 65 /* Package not installed */

```
#include <core/error.h>
```

Package not installed.

4.15.3.93 #define ENOPROTOOPT 92 /* Protocol not available */

```
#include <core/error.h>
```

Protocol not available.

4.15.3.94 #define ENOSPC 28 /* No space left on device */

```
#include <core/error.h>
```

No space left on device.

4.15.3.95 #define ENOSR 63 /* Out of streams resources */

```
#include <core/error.h>
```

Out of streams resources.

4.15.3.96 #define ENOSTR 60 /* Device not a stream */

```
#include <core/error.h>
```

Device not a stream.

4.15.3.97 #define ENOSYS 38 /* Function not implemented */

```
#include <core/error.h>
```

Function not implemented.

4.15.3.98 #define ENOTBLK 15 /* Block device required */

```
#include <core/error.h>
```

Block device required.

4.15.3.99 #define ENOTCONN 107 /* Transport endpoint is not connected */

```
#include <core/error.h>
```

Transport endpoint is not connected.

4.15.3.100 #define ENOTDIR 20 /* Not a directory */

```
#include <core/error.h>
```

Not a directory.

4.15.3.101 #define ENOTEMPTY 39 /* Directory not empty */

```
#include <core/error.h>
```

Directory not empty.

4.15.3.102 #define ENOTNAM 118 /* Not a XENIX named type file */

```
#include <core/error.h>
```

Not a XENIX named type file.

4.15.3.103 #define ENOTSOCK 88 /* Socket operation on non-socket */

```
#include <core/error.h>
```

Socket operation on non-socket.

4.15.3.104 #define ENOTTY 25 /* Not a typewriter */

```
#include <core/error.h>
```

Not a typewriter.

4.15.3.105 #define ENOTUNIQ 76 /* Name not unique on network */

```
#include <core/error.h>
```

Name not unique on network.

4.15.3.106 #define ENXIO 6 /* No such device or address */

```
#include <core/error.h>
```

No such device or address.

4.15.3.107 #define EOF (-1)

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

EOF - symbol signifying End Of File.

4.15.3.108 #define EOPNOTSUPP 95 /* Operation not supported on transport endpoint */

```
#include <core/error.h>
```

Operation not supported on transport endpoint.

4.15.3.109 #define EOVERFLOW 75 /* Value too large for defined data type */

```
#include <core/error.h>
```

Value too large for defined data type.

4.15.3.110 #define EPERM 1 /* Operation not permitted */

```
#include <core/error.h>
```

Operation not permitted.

4.15.3.111 #define EPNOSUPPORT 96 /* Protocol family not supported */

```
#include <core/error.h>
```

Protocol family not supported.

4.15.3.112 #define EPIPE 32 /* Broken pipe */

```
#include <core/error.h>
```

Broken pipe.

4.15.3.113 #define EPROTO 71 /* Protocol error */

```
#include <core/error.h>
```

Protocol error.

4.15.3.114 #define EPROTONOSUPPORT 93 /* Protocol not supported */

```
#include <core/error.h>
```

Protocol not supported.

4.15.3.115 #define EPROTOTYPE 91 /* Protocol wrong type for socket */

```
#include <core/error.h>
```

Protocol wrong type for socket.

4.15.3.116 #define ERANGE 34 /* Math result not representable */

```
#include <core/error.h>
```

Math result not representable.

4.15.3.117 #define EREMCHG 78 /* Remote address changed */

```
#include <core/error.h>
```

Remote address changed.

4.15.3.118 #define EREMOTE 66 /* Object is remote */

```
#include <core/error.h>
```

Object is remote.

4.15.3.119 #define EREMOTEIO 121 /* Remote I/O error */

```
#include <core/error.h>
```

Remote I/O error.

4.15.3.120 #define ERESTART 85 /* Interrupted system call should be restarted */

```
#include <core/error.h>
```

Interrupted system call should be restarted.

4.15.3.121 #define EROFS 30 /* Read-only file system */

```
#include <core/error.h>
```

Read-only file system.

4.15.3.122 #define errno (*get_errno_addr())

```
#include <errno.h>
```

number of last error

Contains numeric code of last error, list of possible error codes is specified in POSIX.1-2001 or in [core/error.h](#) file.

4.15.3.123 #define ESHUTDOWN 108 /* Cannot send after transport endpoint shutdown */

```
#include <core/error.h>
```

Cannot send after transport endpoint shutdown.

4.15.3.124 #define ESOCKTNOSUPPORT 94 /* Socket type not supported */

```
#include <core/error.h>
```

Socket type not supported.

4.15.3.125 #define ESPIPE 29 /* Illegal seek */

```
#include <core/error.h>
```

Illegal seek.

4.15.3.126 #define ESRCH 3 /* No such process */

```
#include <core/error.h>
```

No such process.

4.15.3.127 #define ESRMNT 69 /* Srmount error */

```
#include <core/error.h>
```

Srmount error.

4.15.3.128 #define ESTALE 116 /* Stale file handle */

```
#include <core/error.h>
```

Stale file handle.

4.15.3.129 #define ESTRPIPE 86 /* Streams pipe error */

```
#include <core/error.h>
```

Streams pipe error.

4.15.3.130 #define ETIME 62 /* Timer expired */

```
#include <core/error.h>
```

Timer expired.

4.15.3.131 #define ETIMEDOUT 110 /* Connection timed out */

```
#include <core/error.h>
```

Connection timed out.

4.15.3.132 #define ETOOMANYREFS 109 /* Too many references: cannot splice */

```
#include <core/error.h>
```

Too many references: cannot splice.

4.15.3.133 #define ETXTBSY 26 /* Text file busy */

```
#include <core/error.h>
```

Text file busy.

4.15.3.134 #define EUCLEAN 117 /* Structure needs cleaning */

```
#include <core/error.h>
```

Structure needs cleaning.

4.15.3.135 #define EUNATCH 49 /* Protocol driver not attached */

```
#include <core/error.h>
```

Protocol driver not attached.

4.15.3.136 #define EUSERS 87 /* Too many users */

```
#include <core/error.h>
```

Too many users.

4.15.3.137 #define EWOULDBLOCK EAGAIN /* Operation would block */

```
#include <core/error.h>
```

Operation would block.

4.15.3.138 #define EXDEV 18 /* Cross-device link */

```
#include <core/error.h>
```

Cross-device link.

4.15.3.139 #define EXFULL 54 /* Exchange full */

```
#include <core/error.h>
```

Exchange full.

4.15.3.140 #define INT_MAX ((int)(~0U>>1))

```
#include <limits.h>
```

Maximum value for an object of type `int`. Minimum Acceptable Value: 2 147 483 647

4.15.3.141 #define INT_MIN (-INT_MAX - 1)

```
#include <limits.h>
```

Minimum value for an object of type `int`. Maximum Acceptable Value: -2 147 483 647

4.15.3.142 #define LLONG_MAX ((long long)(~0ULL>>1))

```
#include <limits.h>
```

Maximum value for an object of type `long long`. Minimum Acceptable Value: +9223372036854775807

4.15.3.143 #define LLONG_MIN ((long long)(-LLONG_MAX - 1))

```
#include <limits.h>
```

Minimum value for an object of type `long long`. Maximum Acceptable Value: -9223372036854775807

4.15.3.144 #define LONG_MAX ((long)(~0UL>>1))

```
#include <limits.h>
```

Maximum value for an object of type `long`. Minimum Acceptable Value: +2 147 483 647

4.15.3.145 #define LONG_MIN (-LONG_MAX - 1)

```
#include <limits.h>
```

Minimum value for an object of type `long`. Maximum Acceptable Value: -2 147 483 647

4.15.3.146 #define M_CACHE_PAGES 1

```
#include <malloc.h>
```

For setting number of pages that not to be munmapped to reuse them quickly again.

4.15.3.147 #define MAP_ANONYMOUS (1 << 0)

```
#include <core/mman.h>
```

The mapping is not backed by any file; its contents are initialized to zero.

4.15.3.148 #define MAP_FAILED ((void *)-1)

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

Error return value from `mmap()`.

4.15.3.149 #define MAP_FIXED (1 << 2)

```
#include <core/mman.h>
```

Don't interpret `addr` as a hint: place the mapping at exactly that address. `addr` must be a multiple of the page size.

4.15.3.150 #define MAP_PHYS_NON_CACHED (1 << 28)

```
#include <driver/mem/phys.h>
```

Memory will be mapped as non-cached.

4.15.3.151 #define MAP_PHYS_NON_SECURE (1 << 29)

```
#include <driver/mem/phys.h>
```

Non-secure memory region will be mapped.

4.15.3.152 #define MAP_POPULATE (1 << 1)

```
#include <core/mman.h>
```

Populate (prefault) page tables for a mapping.

4.15.3.153 #define MAP_PRIVATE (1 << 3)

```
#include <core/mman.h>
```

Create a private copy-on-write mapping. Updates to the mapping are not visible to other processes mapping the same file, and are not carried through to the underlying file.

4.15.3.154 #define MAP_SHARED (1 << 4)

```
#include <core/mman.h>
```

Share this mapping. Updates to the mapping are visible to other processes that map this file, and are carried through to the underlying file.

4.15.3.155 #define MAX_CPUS (32)

```
#include <sched.h>
```

The maximum cpu number

4.15.3.156 #define NUM_MILLIS_IN_SEC (1000)

```
#include <time.h>
```

The number millisecond in second

4.15.3.157 #define NUM_NANOS_IN_MILLI (1000000L)

```
#include <time.h>
```

The number nanosecond in millisecond

4.15.3.158 #define NUM_NANOS_IN_SEC (1000000000ULL)

```
#include <time.h>
```

The number nanosecond in second

4.15.3.159 #define NUM_NANOS_IN_USEC (1000)

```
#include <time.h>
```

The number nanosecond in microsecond

4.15.3.160 #define NUM_SECONDS_IN_MIN (60)

```
#include <time.h>
```

The number second in minute

4.15.3.161 #define PGOFF_SHIFT 12

```
#include <core/mman.h>
```

Offset shift.

4.15.3.162 #define PHYS_DEV_NAME "phys://"

```
#include <driver/mem/phys.h>
```

Name of device to physical memory access.

4.15.3.163 #define PHYS_DEV_NAME_LEN (sizeof(PHYS_DEV_NAME) - 1)

```
#include <driver/mem/phys.h>
```

Length of the name of device to physical memory access.

4.15.3.164 #define PRId16 __16_PREFIX "d"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the value of type int16_t

4.15.3.165 #define PRId32 "d"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the value of type int32_t

4.15.3.166 #define PRId64 __64_PREFIX "d"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the value of type int64_t

4.15.3.167 #define PRId16 __16_PREFIX "u"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the value of type uint16_t

4.15.3.168 #define PRlu32 "u"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the value of type uint32_t

4.15.3.169 #define PRlu64 __64_PREFIX "u"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the value of type uint64_t

4.15.3.170 #define PRlx16 __16_PREFIX "x"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the 16-bit value in hex

4.15.3.171 #define PRlx32 "x"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the 32-bit value in hex

4.15.3.172 #define PRlx64 __64_PREFIX "x"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the 64-bit value in hex

4.15.3.173 #define PROT_EXEC 4

```
#include <core/mman.h>
```

Pages may be executed.

4.15.3.174 #define PROT_NONE 0

```
#include <core/mman.h>
```

Pages may not be accessed.

4.15.3.175 #define PROT_READ 1

```
#include <core/mman.h>
```

Pages may be read.

4.15.3.176 #define PROT_WRITE 2

```
#include <core/mman.h>
```

Pages may be written.

4.15.3.177 #define PTHREAD_KEYS_MAX _POSIX_THREAD_KEYS_MAX

```
#include <limits.h>
```

Maximum number of data keys that can be created by a process. Minimum Acceptable Value: [_POSIX_THREAD_KEYS_MAX](#)

4.15.3.178 #define SCHAR_MAX (127)

```
#include <limits.h>
```

Maximum value for an object of type `signed char`. Minimum Acceptable Value: 127

4.15.3.179 #define SCHAR_MIN (-128)

```
#include <limits.h>
```

Minimum value for an object of type `signed char`. Maximum Acceptable Value: -128

4.15.3.180 #define SCNd16 __16_PREFIX "d"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the value of type `int16_t`

4.15.3.181 #define SCNd32 "d"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the value of type `int32_t`

4.15.3.182 #define SCNd64 __64_PREFIX "d"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the value of type int64_t

4.15.3.183 #define SCNu16 __16_PREFIX "u"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the value of type uint16_t

4.15.3.184 #define SCNu32 "u"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the value of type uint32_t

4.15.3.185 #define SCNu64 __64_PREFIX "u"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the value of type uint64_t

4.15.3.186 #define SCNx16 __16_PREFIX "x"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the 16-bit value in hex

4.15.3.187 #define SCNx32 "x"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the 32-bit value in hex

4.15.3.188 #define SCNx64 __64_PREFIX "x"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the 64-bit value in hex

4.15.3.189 #define SHRT_MAX (32767)

```
#include <limits.h>
```

Maximum value for an object of type `signed short int`. Minimum Acceptable Value: 32767

4.15.3.190 #define SHRT_MIN (-32768)

```
#include <limits.h>
```

Minimum value for an object of type `signed short int`. Maximum Acceptable Value: -32768

4.15.3.191 #define TEMP_FAILURE_RETRY(*expression*)

```
#include <unistd.h>
```

Value:

```
{
  typeof (expression) _result; \
  do { \
    _result = (expression); \
  } while (_result == ((typeof (expression)) -1) && errno == EINTR); \
  _result; }
```

Recall function if it was interrupted by signal.

Parameters

in	<i>expression</i>	macro argument.
----	-------------------	-----------------

Returns

result of last function call.

4.15.3.192 #define UCHAR_MAX (255)

```
#include <limits.h>
```

Maximum value for an object of type `unsigned char`. Minimum Acceptable Value: 255

4.15.3.193 #define UINT_MAX (~0U)

```
#include <limits.h>
```

Maximum value for an object of type `unsigned`. Minimum Acceptable Value: 4 294 967 295

4.15.3.194 #define ULLONG_MAX (~0ULL)

```
#include <limits.h>
```

Maximum value for an object of type `unsigned long long`. Minimum Acceptable Value: 18446744073709551615

4.15.3.195 #define ULONG_MAX (~0UL)

```
#include <limits.h>
```

Maximum value for an object of type `unsigned long`. Minimum Acceptable Value: 4 294 967 295

4.15.3.196 #define USHRT_MAX (0xFFFFU)

```
#include <limits.h>
```

Maximum value for an object of type `unsigned short`. Minimum Acceptable Value: 65 535

4.15.3.197 #define uuid_generate_time(x) uuid_generate(x)

```
#include <uuid/uuid.h>
```

The `uuid_generate` function creates a new universally unique identifier (UUID), based on secure timer value.

4.15.3.198 #define UUID_STRING_LEN 37

```
#include <uuid/uuid.h>
```

Length of string representation of UUID in format (with terminating '\0'): xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

4.15.3.199 #define uuid_unparse_lower(uu, out) uuid_unparse((uu), (out))

```
#include <uuid/uuid.h>
```

Convert binary representation of UUID to string in lowercase format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

4.15.4 Typedef Documentation

4.15.4.1 __uuid_t

```
#include <uuid/uuid.h>
```

Provide UUID type as the structure.

4.15.4.2 constraint_handler_t

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

Used for runtime-constraint handler description.

4.15.4.3 errno_t

```
#include <errno.h>
```

Used for error description in safe C functions (*_s). See Annex K of C11 standard.

4.15.4.4 gid_t

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

Used for groups IDs.

4.15.4.5 mode_t

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

Used for some file attributes.

4.15.4.6 off_t

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

Used for file sizes.

4.15.4.7 pid_t

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

Used for process IDs.

4.15.4.8 ssize_t

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

Used for a count of bytes or an error indication.

4.15.4.9 time_t

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

Used for time in seconds.

4.15.4.10 uid_t

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

Used for user IDs.

4.15.4.11 uuid_t

```
#include <uuid/uuid.h>
```

Provide UUID type for users.

4.15.5 Function Documentation

4.15.5.1 void _exit (int status)

```
#include <unistd.h>
```

Terminate the calling process "immediately". Any open file descriptors belonging to the process are closed; process's parent is sent a SIGCHLD signal.

Cause process termination without calling of functions registered with [atexit\(\)](#) and return the value of `status & 0377` to the parent.

Parameters

in	<i>status</i>	value to return to parent process.
in	<i>status</i>	function argument.

4.15.5.2 void _exit_thread (unsigned long status)

```
#include <unistd.h>
```

Terminate the calling thread "immediately".

Parameters

in	<i>status</i>	value to return to parent process.
----	---------------	------------------------------------

4.15.5.3 void abort_handler_s (const char *restrict msg, void *restrict ptr, errno_t error)

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

Abort system if constraint was caused.

Parameters

in	<i>msg</i>	pointer to the error message.
in	<i>ptr</i>	pointer to an implementation-defined object or a NULL.
in	<i>error</i>	positive value of type <code>errno_t</code> .

4.15.5.4 static __inline__ int abs (int j) [static]

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

Compute the absolute value of the integer argument `__n`.

Parameters

in	<i>j</i>	function argument.
----	----------	--------------------

Returns

the result of computation.

4.15.5.5 int alarm (unsigned int seconds)

```
#include <time.h>
```

Set the real-time timer to expire in `seconds` seconds.

Parameters

in	<i>seconds</i>	Time in second.
----	----------------	-----------------

Returns

The return value indicates how many seconds remain before the previous alarm would have been sent. If there is no previous alarm, alarm returns zero.

4.15.5.6 unsigned int arm_timer_get_frequency (void)

```
#include <time.h>
```

The function retrieves the frequency of ARM timer.

Return values

<i>frequency</i>	of ARM timer in HZ.
------------------	---------------------

4.15.5.7 int asprintf (char ** strp, const char * fmt, ...)

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

print to allocated string.

Parameters

out	<i>strp</i>	Pointer to newly allocated string or NULL(if the function failed).
in	<i>fmt</i>	Format string.
in,out	...	Variable arguments for printing.

The functions [asprintf\(\)](#) and [vasprintf\(\)](#) are analogs of [sprintf\(\)](#) and [vsprintf\(\)](#), except that they allocate a string large enough to hold the output including the terminating null byte, and return a pointer to it via the first argument. This pointer should be passed to [free\(\)](#) to release the allocated storage when it is no longer needed.

These functions are Blowfish extensions, not in C or POSIX. They are also available under *BSD and GNU/Linux. The GNU/Linux implementation leaves *strp* in undefined state in case of error.

Return values

≥ 0	number of bytes printed if succeed
-1	if failed, and set <i>strp</i> to NULL

4.15.5.8 int atexit (void(*)(void) func)

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

Register the given function to be called at normal process termination.

Parameters

in	<i>func</i>	pointer to the function.
----	-------------	--------------------------

Returns

- 0 if successful
- nonzero otherwise

4.15.5.9 void* calloc (size_t nmemb, size_t size)

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

Allocate memory for an array of `nmemb` elements of `size` bytes each and return a pointer to the allocated memory.

Note

The memory is set to zero.

Parameters

in	<i>nmemb</i>	function argument.
in	<i>size</i>	function argument.

Return values

<i>address</i>	pointer to allocated memory on success.
<i>NULL</i>	if <code>size</code> is equal to zero.
<i>NULL</i>	and sets <code>errno</code> to <code>ENOMEM</code> if there is no enough memory to be allocated.

4.15.5.10 int clock_gettime (clockid_t clk_id, struct timespec * ts)

```
#include <time.h>
```

The function retrieves the time of the specified clock `clk_id`. This function is non-blocking, except `CLOCK_REALTIME` case. In this case function can sleep and can be interrupted with `-1` result and `EINTR` `errno`.

Parameters

in	<i>clk_id</i>	Specified clock.
out	<i>ts</i>	timespec structs argument.

Return values

<i>0</i>	on success.
<i>-1</i>	on failure. <code>errno</code> variable is set appropriately.

4.15.5.11 int close (int *fd*)

```
#include <unistd.h>
```

Deallocate the file descriptor indicated by *fd*. To deallocate means to make the file descriptor available for return by subsequent calls to [open\(\)](#) or other functions that allocate file descriptors. All outstanding record locks owned by the process on the file associated with the file descriptor shall be removed (that is, unlocked).

Parameters

in	<i>fd</i>	file descriptor.
----	-----------	------------------

Return values

0	on success.
-1	on error. errno is set appropriately.

4.15.5.12 void exit (int *status*)

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

Cause normal process termination and return the value of *status* & 0377 to the parent.

Parameters

in	<i>status</i>	function argument.
----	---------------	--------------------

4.15.5.13 int fflush (FILE * *stream*)

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

Flush a stream.

Parameters

in	<i>stream</i>	A writeable file handle.
----	---------------	--------------------------

Return values

=0	if succeed.
EOF	if error occurred.

4.15.5.14 int fprintf (FILE *_restrict_stream, const char *_restrict_fmt, ...)

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

The `fprintf()` is equivalent to the `printf()`, but uses a custom file handle.

Parameters

in	<i>stream</i>	A writeable file handle.
in	<i>fmt</i>	Format specification of output string.
in,out	...	Arguments used for printing.

Return values

≥ 0	number of characters printed if succeed.
-1	if error occurred.

Example:

```
fprintf(stdout, "Hello, %s!\n", "world");
```

4.15.5.15 void free (void * ptr)

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

Free the memory space pointer to by `ptr`.

Note

if `free(ptr)` has already been called before, undefined behaviour occurs.

Parameters

in	<i>ptr</i>	- pointer returned by a previous call to <code>malloc()</code> , <code>calloc()</code> , or <code>realloc()</code> .
----	------------	--

4.15.5.16 `int fstat (int fd, struct stat * buf)`

```
#include <unistd.h>
```

Get status of a file with a descriptor `fd`.

Parameters

in	<i>fd</i>	File descriptor.
out	<i>buf</i>	Pointer to structure to store status information.

Return values

0	on success.
-1	on error. <code>errno</code> is set appropriately.

4.15.5.17 `int ftruncate (int fd, int size)`

```
#include <unistd.h>
```

Cause file referenced by `fd` to be truncated to a `size` of precisely length bytes.

Parameters

in	<i>fd</i>	file descriptor.
in	<i>size</i>	function argument.

Return values

0	on success.
-1	on error. <code>errno</code> is set appropriately.

4.15.5.18 `int* get_errno_addr (void)`

```
#include <errno.h>
```

This function should NOT be used directly, use '`errno`' instead.

Returns

thread local pointer to `errno`

4.15.5.19 int getcluster (void)

```
#include <unistd.h>
```

Return the cluster id on which current thread is performed.

Note

Arch-specific function. Assumes the presence of two clusters.

Returns

function result.

4.15.5.20 int getcpu (void)

```
#include <unistd.h>
```

Return the number of CPU on which current thread is performed.

Returns

function result.

4.15.5.21 int getitimer (int which, struct itimerval * curr_value)

```
#include <time.h>
```

Get value of an interval timer.

Parameters

in	<i>which</i>	setting for the timer (one of ITIMER_REAL , ITIMER_VIRTUAL , or ITIMER_PROF).
out	<i>curr_value</i>	The function getitimer() fills the structure pointed to by <i>curr_value</i> .

Return values

0	on success.
-1	on failure. errno variable is set appropriately.

4.15.5.22 pid_t getpid (void)

```
#include <unistd.h>
```

Return the process ID of the calling process.

Returns

function result.

4.15.5.23 pid_t gettid (void)

```
#include <unistd.h>
```

Return the thread ID of the calling thread.

Returns

function result.

4.15.5.24 void ignore_handler_s (const char *restrict msg, void *restrict ptr, errno_t error)

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

Returns to the caller without performing any actions.

Parameters

in	<i>msg</i>	pointer to the error message.
in	<i>ptr</i>	pointer to an implementation-defined object or a NULL.
in	<i>error</i>	positive value of type <code>errno_t</code> .

4.15.5.25 void invoke_constraint_handler_s (const char * msg, const char * file, const char * function, uint32_t line, errno_t error)

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

Print `msg` if constraint was caused.

Parameters

in	<i>msg</i>	pointer to the error message.
in	<i>file</i>	name of file where constraint was caused.
in	<i>function</i>	name of function where constraint was caused.
in	<i>line</i>	number of line where constraint was caused.
in	<i>error</i>	positive value of type <code>errno_t</code> .

4.15.5.26 `int ioctl (int fd, int request, unsigned long data)`

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

Manipulates the underlying device parameters of special files.

Parameters

in	<i>fd</i>	Open file descriptor.
in	<i>request</i>	Device-dependent request code.
in,out	<i>data</i>	Optional data associated with request.

Returns

usually, on success zero is returned, on error -1 is returned and `errno` is set appropriately. Strictly speaking, return status is device-dependent.

Performs a variety of control functions on a device opened by `open()` system call. The request argument and an optional third argument (of varying type) are device driver implementation-defined. Example:

```
ret = ioctl (fd, STORAGE_GET_IMAGE_SNAPSHOT, 0);
if (ret == -1)
    return errno;
```

4.15.5.27 `static int isalnum (int c)` [`static`]

```
#include <ctype.h>
```

Check for an alphanumeric character; it is equivalent to `(isalpha(c) || isdigit(c))`.

Parameters

in	<i>c</i>	symbol for checking
----	----------	---------------------

Returns

- non-zero if *c* is an alphanumeric character;
- 0 otherwise.

4.15.5.28 static int isalpha (int c) [static]

```
#include <ctype.h>
```

Check for an alphabetic character; it is equivalent to (isupper(c) || islower(c))

Parameters

in	c	symbol for checking
----	---	---------------------

Returns

- non-zero if *c* is an alphabetic character;
- 0 otherwise.

4.15.5.29 static int isascii (int c) [static]

```
#include <ctype.h>
```

Check whether *c* is a 7-bit unsigned char value that fits into the ASCII character set.

Parameters

in	c	symbol for checking
----	---	---------------------

Returns

- non-zero if *c* is a 7-bit US-ASCII character code between 0 and octal 0177 inclusive;
- 0 otherwise.

4.15.5.30 static int isblank (int c) [static]

```
#include <ctype.h>
```

Check for a blank character; that is, a space or a tab.

Parameters

in	c	symbol for checking
----	---	---------------------

Returns

- non-zero if *c* is a blank character;
- 0 otherwise.

4.15.5.31 static int iscntrl (int c) [static]

```
#include <ctype.h>
```

Check for a control character.

Parameters

in	c	symbol for checking
----	---	---------------------

Returns

- non-zero if *c* is a control character;
- 0 otherwise.

4.15.5.32 static int isdigit (int c) [static]

```
#include <ctype.h>
```

Check for a digit (0 through 9)

Parameters

in	c	symbol for checking
----	---	---------------------

Returns

- non-zero if *c* is a decimal digit character;
- 0 otherwise.

4.15.5.33 static int isgraph (int c) [static]

```
#include <ctype.h>
```

Check for any printable character except space.

Parameters

in	c	symbol for checking
----	---	---------------------

Returns

- non-zero if *c* is a character with a visible representation;
- 0 otherwise.

4.15.5.34 static int islower (int c) [static]

```
#include <ctype.h>
```

Check for an lowercase letter.

Parameters

in	c	symbol for checking
----	---	---------------------

Returns

- non-zero if c is a lowercase letter;
- 0 otherwise.

4.15.5.35 static int isprint (int c) [static]

```
#include <ctype.h>
```

Check for any printable character including space.

Parameters

in	c	symbol for checking
----	---	---------------------

Returns

- non-zero if c is a printable character;
- 0 otherwise.

4.15.5.36 static int ispunct (int c) [static]

```
#include <ctype.h>
```

Check for any printable character which is not a space or an alphanumeric character.

Parameters

in	c	symbol for checking
----	---	---------------------

Returns

- non-zero if c is a punctuation character;
- 0 otherwise.

4.15.5.37 static int isspace (int c) [static]

```
#include <ctype.h>
```

Check for white-space characters. These are: space, form-feed ('\f'), newline ('\n'), carriage return ('\r'), horizontal tab ('\t'), and vertical tab ('\v').

Parameters

in	c	symbol for checking
----	---	---------------------

Returns

- non-zero if *c* is a white-space character;
- 0 otherwise.

4.15.5.38 static int isupper (int c) [static]

```
#include <ctype.h>
```

Check for an uppercase letter.

Parameters

in	c	symbol for checking
----	---	---------------------

Returns

- non-zero if *c* is an uppercase letter;
- 0 otherwise.

4.15.5.39 static int isxdigit (int c) [static]

```
#include <ctype.h>
```

Check for hexadecimal digits, that is, one of 0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F.

Parameters

in	c	symbol for checking
----	---	---------------------

Returns

- non-zero if *c* is a hexadecimal digit character;
- 0 otherwise.

4.15.5.40 `int lseek (int fd, int offset, int whence)`

```
#include <unistd.h>
```

Reposition read/write file offset.

Available modes:

- `SEEK_SET` - offset is set to `offset` bytes.
- `SEEK_CUR` - offset is set to current location plus `offset` bytes.
- `SEEK_END` - offset is set to the size of the file plus `offset` bytes.

Parameters

in	<i>fd</i>	File descriptor.
in	<i>offset</i>	New offset value.
in	<i>whence</i>	Mode of operation.

Returns

- resulting offset location from the beginning of the file - on success.
- -1 and set `errno` to indicate the error - otherwise.

4.15.5.41 `void* malloc (size_t size)`

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

Allocate `size` bytes and return a pointer to the allocated memory.

Parameters

in	<i>size</i>	- size of memory to be allocated.
----	-------------	-----------------------------------

Return values

<i>address</i>	pointer to allocated memory on success.
<code>NULL</code>	if <code>size</code> is equal to zero.
<code>NULL</code>	and sets <code>errno</code> to <code>ENOMEM</code> if there is no enough memory to be allocated.

4.15.5.42 void* memchr (const void * s, int c, size_t n)

```
#include <string.h>
```

Find a character in an area of memory.

Parameters

in	<i>s</i>	pointer to the memory area.
in	<i>c</i>	the byte to search for.
in	<i>n</i>	the size of the area.

Returns

address of the first occurrence of *c*, or NULL.

4.15.5.43 int memcmp (const void * s1, const void * s2, size_t n)

```
#include <string.h>
```

Compare first *n* bytes of memory pointed by *s1* and *s2*.

Parameters

in	<i>s1</i>	first memory region to compare.
in	<i>s2</i>	second memory region to compare.
in	<i>n</i>	number of bytes to compare.

Return values

0	memory regions have the same values.
<0	region <i>s1</i> has lower value than <i>s2</i> .
>0	region <i>s1</i> has greater value than <i>s2</i> .

4.15.5.44 void* memcpy (void * dest, const void * src, size_t n)

```
#include <string.h>
```

Copy memory area from *src* to *dst*. The memory must not overlap. To copy overlapped area use [memmove\(\)](#) function.

Parameters

out	<i>dest</i>	destination area.
in	<i>src</i>	source area.
in	<i>n</i>	number of bytes to copy.

Returns

pointer to `dest` area.

4.15.5.45 `errno_t memcpy_s (void *restrict dest, rsize_t dest_max, const void *restrict src, rsize_t n)`

```
#include <string.h>
```

Check arguments and copy `src` sized memory area to `dest`. Memory must not be overlapped. To copy overlapped area use `memmove_s()` function.

Parameters

out	<code>dest</code>	destination area.
in	<code>dest_max</code>	max number of bytes to modify in the dest.
in	<code>src</code>	source area.
in	<code>n</code>	number of bytes to copy.

Return values

<code>0</code>	function completed work with success.
<code>!=0</code>	function completed work with error.

4.15.5.46 `void* memmove (void * dest, const void * src, size_t n)`

```
#include <string.h>
```

Copy memory area from `src` to `dest`. The memory may overlap.

Parameters

out	<code>dest</code>	destination area.
in	<code>src</code>	source area.
in	<code>n</code>	number of bytes to copy.

Returns

pointer to `dest` area.

4.15.5.47 `errno_t memmove_s (void * dest, rsize_t dest_max, const void * src, rsize_t n)`

```
#include <string.h>
```

Check arguments and copy `src` sized memory area to `dest`. Memory may be overlapped.

Parameters

out	<i>dest</i>	destination area.
in	<i>dest_max</i>	max number of bytes to modify in the dest.
in	<i>src</i>	source area.
in	<i>n</i>	number of bytes to copy.

Return values

0	function completed work with success.
!=0	function completed work with error.

4.15.5.48 `void* memset (void * block, int c, size_t size)`

```
#include <string.h>
```

Fill `size` bytes with a constant value.

Parameters

out	<i>block</i>	address of memory region to be filled.
in	<i>c</i>	value to fill.
in	<i>size</i>	number of bytes to fill.

Returns

pointer to `block` area.

4.15.5.49 `errno_t memset_s (void * block, rsize_t block_max, int c, rsize_t n)`

```
#include <string.h>
```

Check arguments and fill `n` bytes of `block` sized memory with `c` constant value.

Parameters

out	<i>block</i>	address of memory region to be filled.
in	<i>block_max</i>	max number of bytes for copy to block.
in	<i>c</i>	value to fill.
in	<i>n</i>	number of bytes to fill.

Return values

<i>0</i>	function completed work with success.
<i>!=0</i>	function completed work with error.

4.15.5.50 void* mmap (void * *addr*, size_t *len*, int *prot*, int *flags*, int *fd*, off_t *offset*)

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

The `mmap()` function establishes a mapping between a process' address space and a file or shared memory object.

Parameters

in	<i>addr</i>	starting address.
in	<i>len</i>	specify the length of the mapping.
in	<i>prot</i>	Describes the desired memory protection of the mapping Protection flags are defined in core/mman.h header. Possible values: <ul style="list-style-type: none"> • PROT_READ - Data can be read. • PROT_WRITE - Data can be written. • PROT_EXEC - Data can be executed. • PROT_NONE - Data cannot be accessed.
in	<i>flags</i>	Provides other information about the handling of the mapped data. All flags except MAP_PHYS_NON_SECURE and MAP_PHYS_NON_CACHED are defined in core/mman.h header. Possible values: <ul style="list-style-type: none"> • MAP_ANONYMOUS - The mapping is not backed by any file; its contents are initialized to zero. • MAP_POPULATE - Populate (prefault) page tables for a mapping. • MAP_FIXED - Don't interpret <i>addr</i> as a hint: place the mapping at exactly that address. • MAP_PRIVATE - Create a private copy-on-write mapping. Updates to the mapping are not visible to other processes mapping the same file, and are not carried through to the underlying file. • MAP_SHARED - Share this mapping. Updates to the mapping are visible to other processes that map this file, and are carried through to the underlying file. • MAP_PHYS_NON_SECURE - Use non-secure mapping (phys_mmap driver specific, i.e. for mappings created in phys:// namespace). Defined in driver/mem/phys.h header. • MAP_PHYS_NON_CACHED - Use non-cached mapping (phys_mmap driver specific, i.e. for mappings created in phys:// namespace). Defined in driver/mem/phys.h header.
in	<i>fd</i>	file descriptor.
in	<i>offset</i>	offset.

Returns

Upon successful completion, the [mmap\(\)](#) function returns the address at which the mapping was placed (*pa*); otherwise, it returns a value of [MAP_FAILED](#).

Example:

```
address = mmap(0, len, PROT_READ, MAP_PRIVATE, fildes, some_offset);
```

4.15.5.51 void ms_to_timespec (int64_t t, struct timespec * a)

```
#include <time.h>
```

The function converts time in milliseconds to to timespec format.

Parameters

in	t	time in milliseconds.
out	a	Pointer to timespec structure to save result.

4.15.5.52 int munmap (void * addr, size_t length)

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

Tries to unmap memory area at addr of length bytes previously allocated and mmaped by `mmap()` system call.

Parameters

in	addr	address where memory was previously mapped.
in	length	size of the memory area.

Return values

0	no error.
-1	on failure. <code>errno</code> variable is set appropriately. <ul style="list-style-type: none"> • <code>EINVAL</code> - invalid length specified. • <code>EPERM</code> - memory area other than heap is specified.

Example:

```
ret = munmap(buf, 0x8000);
if (ret) {
    printf("munmap error\n");
    panic(1);
}
```

4.15.5.53 int nanosleep (struct timespec * req, struct timespec * rem)

```
#include <time.h>
```

`nanosleep()` suspends the execution of the calling thread until either at least the time specified in `*req` has elapsed, or the delivery of a signal that triggers the invocation of a handler in the calling thread or that terminates the process. If the call is interrupted by a signal handler, `nanosleep()` returns -1, sets `errno` to `EINTR`, and writes the remaining time into the structure pointed to by `rem` unless `rem` is NULL. The value of `*req` can then be used to call `nanosleep()` again and complete the specified pause.

Parameters

in	<code>req</code>	Pointer to requested time.
in	<code>rem</code>	Pointer to remained time.

Return values

0	on success.
-1	on failure. <code>errno</code> variable is set appropriately.

Example:

```
struct timespec req;
struct timespec rem;
req.tv_sec = TIME_TO_WAIT;
req.tv_nsec = 0;

if(nanosleep(&req, &rem) != 0) {
    printf("TEST_NAME: nanosleep() failed, errno: %d\n", errno);
}
```

4.15.5.54 int open (const char * pathname, int flags, ...)

```
#include <fcntl.h>
```

Open a device by specifying its namespace path. Device driver operations should be previously registered with the namespace framework.

Parameters

in	<code>pathname</code>	namespace path for specific device
in	<code>flags</code>	file access mode
in	...	(optional) file mode

Return values

>0	a descriptor associated with the device.
-1	on error and sets <code>errno</code> to appropriated value: <ul style="list-style-type: none"> • <code>EINVAL</code> – pathname is invalid or device not registered; • <code>ENOMEM</code> – there is not enough memory for internal operations; • <code>EMFILE</code> – to many device descriptors opened by the current thread; • Other values are device driver implementation defined

Example:

```
int func(void) {
    int fd = open("dev://crypto", O_RDWR);
    if (fd < 0)
        return errno;
}
```

4.15.555 int printf (const char * *fmt*, ...)

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

Format and print string.

Parameters

in	<i>fmt</i>	Format specification of output string.
in,out	...	Arguments used for printing.

Return values

>=0	number of characters printed if succeed.
-1	if error occurred.

Example:

```
int a,b;
float c,d;

a = 15;
b = a / 2;
printf("%d\n",b);
printf("%3d\n",b);
printf("%03d\n",b);

c = 15.3;
d = c / 3;
printf("%3.2f\n",d);
```

4.15.5.56 int printf_no_alloc (const char * *fmt*, ...)

```
#include <print_no_alloc.h>
```

Prints to ringbuffer. If resulting string exceeds AUTO_BUFFER_SIZE, cuts it off to AUTO_BUFFER_SIZE.

Parameters

in	<i>fmt</i>	Format specification of output string.
in,out	...	Arguments used for printing.

Return values

≥ 0	number of characters printed if succeed.
-1	if error occurred.

Example:

```
printf_no_alloc("Receive_signal_test: never reach here.\n");
```

4.15.5.57 int printf_s (const char *restrict *fmt*, ...)

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

format and print string.

Parameters

in	<i>fmt</i>	Format specification of output string.
in	...	Arguments used for printing.

Return values

≥ 0	number of characters printed if succeed.
-1	if error occurred.

4.15.5.58 int profil (unsigned short * *buf*, size_t *bufsiz*, size_t *offset*, unsigned int *scale*)

```
#include <unistd.h>
```

Provide a means to find out in what areas your program spends most of its time. The argument *buf* points to *bufsiz* bytes of core. Every virtual 10 milliseconds, the user's program counter (PC) is examined: *offset* is subtracted and the result is multiplied by *scale* and divided by 65536. If the resulting value is less than *bufsiz*, then the corresponding entry in *buf* is incremented. If *buf* is NULL, profiling is disabled.

Parameters

in	<i>buf</i>	function argument.
in	<i>bufsiz</i>	function argument.
in	<i>offset</i>	function argument.
in	<i>scale</i>	function argument.

Returns

zero is always returned.

4.15.5.59 int putchar (int *ch*)

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

writes a character to log.

writes the character *ch*, cast to an unsigned char, to the log.

Parameters

in	<i>ch</i>	Character to print.
----	-----------	---------------------

Return values

<i>character</i>	written as an unsigned char cast to an int.
<i>EOF</i>	on error.

4.15.5.60 int puts (const char * s)

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

writes the string *s* and a trailing newline to log(dmesg).

Parameters

in	<i>s</i>	String to print.
----	----------	------------------

Return values

≥ 0	on success.
<i>EOF</i>	on error.

4.15.5.61 void qsort (void * base, size_t nmemb, size_t size, int(*)(const void *, const void *) compar)

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

Sort an array.

Parameters

in,out	<i>base</i>	pointer to start of the array.
in	<i>nmemb</i>	number or elements in array.
in	<i>size</i>	size of each element in array.
in	<i>compar</i>	comparison function.

4.15.5.62 void qsort_r (void * base, size_t nmemb, size_t size, int(*)(const void *, const void *, void *) compar, void * arg)

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

Sort an array.

Parameters

in,out	<i>base</i>	pointer to start of the array.
in	<i>nmemb</i>	number or elements in array.
in	<i>size</i>	size of each element in array.
in	<i>compar</i>	comparison function.
in	<i>arg</i>	argument, which passed directly as 3-rd parameter to <i>compar</i> .

4.15.5.63 `ssize_t read (int fd, void * buf, size_t count)`

```
#include <unistd.h>
```

Attempt to read `count` bytes from the file associated with the open file descriptor, `fd`, into the buffer pointed to by `buf`.

Parameters

in	<i>fd</i>	File descriptor.
out	<i>buf</i>	Pointer to buffer to write.
in	<i>count</i>	Number of bytes to read.

Returns

- non-negative integer indicating the number of bytes actually read - on success.
- -1 and set `errno` to indicate the error - otherwise.

4.15.5.64 `void* realloc (void * ptr, size_t size)`

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

Change the size of the memory block pointed to by `ptr` to `size` bytes.

Note

The contents will be unchanged in the range from the start of the region up to the minimum of the old and new sizes. If the new size is larger than the old size, the added memory will not be initialized. If `ptr` is NULL, then the call is equivalent to `malloc(size)`, for all values of `size`; if `size` is equal to zero, and `ptr` is not NULL, then the call is equivalent to `free(ptr)`. Unless `ptr` is NULL, it must have been returned by an earlier call to `malloc()`, `calloc()` or `realloc()`. If the area pointed to was moved, a `free(ptr)` is done. If `realloc()` fails, the original block is left untouched; it is not freed or moved.

Parameters

in	<i>ptr</i>	function argument.
in	<i>size</i>	function argument.

Return values

<i>address</i>	pointer to the newly allocated memory, which is properly aligned for any built-in type and may be different from <code>ptr</code> .
<i>NULL</i>	if the request fails. <ul style="list-style-type: none"> • either NULL or a pointer suitable to be passed to <code>free()</code> if <code>size</code> was equal to 0.

4.15.5.65 int sched_getaffinity (pid_t pid, size_t cpusetsize, cpu_set_t * mask)

```
#include <sched.h>
```

Writes the affinity mask of the process whose ID is pid into the `cpu_set_t` structure pointed to by mask.

Parameters

in	<i>pid</i>	process ID.
in	<i>cpusetsize</i>	size (in bytes) of mask.
in,out	<i>mask</i>	pointer to CPU affinity mask.

Return values

0	function completed work with success.
-1	function completed work with error and errno is set appropriately.

4.15.5.66 int sched_setaffinity (pid_t pid, size_t cpusetsize, cpu_set_t * mask)

```
#include <sched.h>
```

Sets the CPU affinity mask of the process whose ID is pid to the value specified by mask. If pid is zero, then the calling process is used.

Parameters

in	<i>pid</i>	process ID.
in	<i>cpusetsize</i>	the length (in bytes) of the data pointed to by mask.
in,out	<i>mask</i>	pointer to CPU affinity mask.

Return values

0	function completed work with success.
-1	function completed work with error and errno is set appropriately.

4.15.5.67 int sched_yield (void)

```
#include <sched.h>
```

Causes the calling thread to relinquish the CPU. The thread is moved to the end of the queue for its static priority and a new thread gets to run.

Return values

0	function completed work with success.
-1	function completed work with error and errno is set appropriately.

4.15.5.68 `constraint_handler_t set_constraint_handler_s (constraint_handler_t handler)`

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

Set the handler to be handler.

Parameters

in	<i>handler</i>	runtime-constraint handler.
----	----------------	-----------------------------

Returns

pointer to the previously registered handler.

4.15.5.69 `int setitimer (int which, const struct itimerval * new_value, struct itimerval * old_value)`

```
#include <time.h>
```

Set value of an interval timer.

Parameters

in	<i>which</i>	Setting for the timer (one of <code>ITIMER_REAL</code> , <code>ITIMER_VIRTUAL</code> , or <code>ITIMER_PROF</code>).
in	<i>new_value</i>	The function <code>setitimer()</code> sets the specified timer to the value in <i>new_value</i> .
out	<i>old_value</i>	If <i>old_value</i> is non-NULL, the old value of the timer is stored there.

Return values

0	on success.
-1	on failure. <code>errno</code> variable is set appropriately.

4.15.5.70 `int snprintf (char * s, size_t count, const char * fmt, ...)`

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

format and string and save it to 's'.

Parameters

out	<i>s</i>	Pointer to an array of char elements where the resulting string will be stored.
in	<i>count</i>	size of <i>s</i> in bytes.
in	<i>fmt</i>	Format specification of output string.
in,out	...	Arguments used for printing.

Return values

<i>number</i>	of characters printed.
<i>number</i>	of characters printed if succeed.
-1	if error occurred.

Example:

```

char buffer[200];
int i, j;
double fp;
char *s = "baltimore";
char c;

int main(void)
{
    c = 'l';
    i = 35;
    fp = 1.7320508;

    j = snprintf(buffer, 6, "%s\n", s);
    j += snprintf(buffer+j, 6, "%c\n", c);
    j += snprintf(buffer+j, 6, "%d\n", i);
    j += snprintf(buffer+j, 6, "%f\n", fp);
    printf("string:\n%s\ncharacter count = %d\n", buffer, j);
}

```

4.15.5.71 int snprintf_s (char *restrict s, rsize_t n, const char *restrict format, ...)

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

Format string and save it to 's'.

Parameters

out	<i>s</i>	Pointer to an array of char elements where the resulting string will be stored.
in	<i>n</i>	Size of <i>s</i> in bytes.
in	<i>format</i>	Format specification of output string.
in	...	Arguments used for printing.

Return values

<i>number</i>	Number of characters printed if succeed.
-1	if error occurred.

4.15.5.72 int sprintf (char * s, const char * fmt, ...)

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

format and string and save it to 's'.

Parameters

out	s	Pointer to an array of char elements where the resulting string will be stored.
in	fmt	Format specification of output string.
in,out	...	Arguments used for printing.

Return values

number	of characters printed if succeed.
-1	if error occurred.

Example:

```
char str[80];
sprintf(str, "Value of Pi = %f", M_PI);
```

4.15.5.73 int sprintf_s (char *restrict s, rsize_t n, const char *restrict format, ...)

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

format string and save it to 's'.

Parameters

out	s	Pointer to an array of char elements where the resulting string will be stored.
in	n	Size of s in bytes.
in	format	Format specification of output string.
in	...	Arguments used for printing.

Return values

number	of characters printed if succeed.
-1	if encoding error occurred.
0	if constrain violation occurred.

4.15.5.74 int sscanf (const char * buf, const char * fmt, ...)

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

Reads data from buf and stores them according to parameter fmt into the locations given by the additional arguments.

Parameters

in	<i>buf</i>	Pointer to C string that the function processes as its source to retrieve the data.
in	<i>fmt</i>	Pointer to C string that contains a format string that follows the same specifications as format in scanf.

Return values

<i>number</i>	of items read, This count can match the expected number of items or be less (even zero) in the case of a matching failure.
<i>EOF</i>	in case of an input failure before any data could be successfully interpreted.

Example:

```
char sentence []="Rudolph is 12 years old";
char str [20];
int i;

sscanf (sentence,"%s %*s %d",str,&i);
printf ("%s -> %d\n", str, i);
```

4.15.5.75 int sscanf_s (const char *restrict s, const char *restrict format, ...)

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

Reads data safely from buf and stores them according to parameter fmt into the locations given by the additional arguments.

Parameters

in	<i>s</i>	Pointer to C string that the function processes as its source to retrieve the data.
in	<i>format</i>	Pointer to C string that contains a format string that follows the same specifications as format in scanf.

Return values

<i>number</i>	of items read, this count can match the expected number of items or be less (even zero) in the case of a matching failure.
<i>EOF</i>	in case of an input failure before any data could be successfully interpreted.

4.15.5.76 int stat (const char * *pathname*, struct stat * *buf*)

```
#include <unistd.h>
```

Get status of a file *pathname*.

Parameters

in	<i>pathname</i>	Path to file.
out	<i>buf</i>	Pointer to structure to store status information.

Return values

0	on success.
-1	on error. <code>errno</code> is set appropriately.

4.15.5.77 char* strcat (char * *dest*, const char * *src*)

```
#include <string.h>
```

Append the *src* string to the *dest* string, overwriting the terminating null byte ('\0') at the end of *dest*, and then adds a terminating null byte.

Parameters

in, out	<i>dest</i>	Pointer to string to append.
in	<i>src</i>	Pointer to source string.

Returns

pointer to the resulting string *dest*.

Example:

```
int main ()
{
    char src[50], dest[50];

    strcpy(src, "This is source");
    strcpy(dest, "This is destination");

    strcat(dest, src);

    printf("Final destination string : |%s|", dest);

    return 0;
}
```

4.15.5.78 `errno_t strcat_s (char *restrict dest, rsize_t dest_max, const char *restrict src)`

```
#include <string.h>
```

Check arguments and append the `src` string to the `dest` string, overwriting the terminating null byte ('\0') at the end of `dest`, and add a terminating null byte.

Parameters

in,out	<code>dest</code>	Pointer to string to append.
in	<code>dest_max</code>	max number of bytes to modify in the <code>dest</code> .
in	<code>src</code>	Pointer to source string.

Return values

<code>0</code>	function completed work with success.
<code>!=0</code>	function completed work with error.

4.15.5.79 `char* strchr (const char * s, int c)`

```
#include <string.h>
```

Find first occurrence of character `c` in string `s`.

Parameters

in	<code>s</code>	Pointer to string to search in.
in	<code>c</code>	character to search.

Returns

pointer to matched character, or NULL if not found.

Example:

```
char str[] = "This is a sample string";
char * pch;
printf ("Looking for the 's' character in \"%s\"...\n",str);
pch=strchr(str,'s');
while (pch != NULL)
{
    printf ("found at %d\n",pch-str+1);
    pch = strchr(pch+1,'s');
}
```

4.15.5.80 char* strchrnul (const char * s, int c)

```
#include <string.h>
```

Find first occurrence of character *c* in string *s*.

Parameters

in	<i>s</i>	Pointer to string to search in.
in	<i>c</i>	character to search.

Returns

pointer to matched character, or a pointer to the null byte at the end of *s* (i.e., *s*+strlen(*s*)) if the character is not found.

4.15.5.81 int strcmp (const char * s1, const char * s2)

```
#include <string.h>
```

Compare the two strings *s1* and *s2*.

Parameters

in	<i>s1</i>	Pointer to first string to compare.
in	<i>s2</i>	Pointer to second string to compare.

Return values

0	strings are equal.
<0	string <i>s1</i> has lower value than <i>s2</i> .
>0	string <i>s1</i> has greater value than <i>s2</i> .

Example:

```
int main ()
{
    char str1[15];
    char str2[15];
    int ret;

    strcpy(str1, "abcdef");
    strcpy(str2, "ABCDEF");

    ret = strcmp(str1, str2);

    if(ret > 0)
    {
        printf("str1 is less than str2");
    }
    else if(ret < 0)
    {
        printf("str2 is less than str1");
    }
    else
    {
        printf("str1 is equal to str2");
    }

    return 0;
}
```

4.15.5.82 char* strcpy (char * dest, const char * src)

```
#include <string.h>
```

Copy the string pointed to by `src`, including the terminating null byte ('\0'), to the buffer pointed to by `dest`.

Parameters

out	<code>dest</code>	pointer to destination string.
in	<code>src</code>	pointer to source string.

Returns

pointer to destination string `dest`.

Example:

```
int main()
{
    char src[40];
    char dest[100];

    memset(dest, '\0', sizeof(dest));
    strcpy(src, "This is example");
    strcpy(dest, src);

    printf("Final copied string : %s\n", dest);

    return 0;
}
```

4.15.5.83 errno_t strcpy_s (char *restrict dest, rsize_t dest_max, const char *restrict src)

```
#include <string.h>
```

Check arguments and copy the `src` string, including the terminating null byte ('\0'), to the `dest` sized buffer. If `n` is bigger than size of `src` then the remaining characters (after '\0') are unspecified.

Parameters

out	<code>dest</code>	pointer to destination string.
in	<code>dest_max</code>	max number of bytes to modify in the dest.
in	<code>src</code>	pointer to source string.

Return values

0	function completed work with success.
!=0	function completed work with error.

4.15.5.84 `size_t strcspn (const char * s, const char * reject)`

```
#include <string.h>
```

Calculate the length of the initial segment of `s` which consists entirely of bytes not in `reject`.

Parameters

in	<code>s</code>	Pointer to string to search in.
in	<code>reject</code>	String containing the characters not to match.

Returns

number of bytes in the initial segment of `s` which are not in the string `reject`.

4.15.5.85 `char* strdup (const char * s)`

```
#include <string.h>
```

Return a pointer to a new string which is a duplicate of the string `s`. Memory for the new string is obtained with `malloc()`.

Parameters

in	<code>s</code>	pointer to source string.
----	----------------	---------------------------

Returns

pointer to duplicated string.

4.15.5.86 `const char* strerror (int errnum)`

```
#include <string.h>
```

Return a pointer to a string that describes the error code passed in the argument `errnum`.

Parameters

in	<code>errnum</code>	error code.
----	---------------------	-------------

Returns

pointer to description.

4.15.5.87 `int strerror_r (int errnum, char * buf, size_t buflen)`

`#include <string.h>`

Returns the error string in the user-supplied `buf` of length `buflen`. XSI-compliant version.

Parameters

in	<i>errnum</i>	Error code.
in	<i>buf</i>	Pointer to the buffer.
in	<i>buflen</i>	Buffer length.

Return values

0	on success.
-1	on error and <code>errno</code> is set.

4.15.5.88 `errno_t strerror_s (char * buf, rsize_t bufmax, errno_t errnum)`

`#include <string.h>`

Check arguments and return a pointer to a `buf` string that describes the `errnum` error code.

Parameters

in	<i>buf</i>	pointer to a user-provided buffer.
in	<i>bufmax</i>	max size of a user-provided buffer.
in	<i>errnum</i>	error code.

Return values

0	function completed work with success.
<i>!=0</i>	function completed work with error.

4.15.5.89 `size_t strlcat (char * dest, const char * src, size_t n)`

`#include <string.h>`

Append the NUL-terminated string `src` to the end of `dest` string, overwriting the terminating null byte ('`\0`') at the end of `dest`, and guarantee to NUL-terminate the result.

Parameters

in,out	<i>dest</i>	Pointer to string to append.
in	<i>src</i>	Pointer to source string.
in	<i>n</i>	Maximal number of bytes to copy if string does not have <code>\0</code> terminator.

Returns

The total length of the string they tried to create. Count does not include NUL.

Example:

```
int main ()
{
    char src[50], dst[50];

    strcpy(src, "It is source.");
    strcpy(dst, "It is dst.");

    strlcat(dst, src, sizeof(dst));

    return 0;
}
```

4.15.5.90 size_t strlen (const char * s)

```
#include <string.h>
```

Calculate the length of the string *s*, excluding the terminating null byte ('\0').

Parameters

in	<i>s</i>	Pointer to string.
----	----------	--------------------

Returns

number of bytes in the string.

Example:

```
int main ()
{
    char string[32] = "hello, world";
    printf("String length is %zu\n", strlen(string));
    return 0;
}
```

4.15.5.91 char* strncat (char * dest, const char * src, size_t n)

```
#include <string.h>
```

Append the *src* string to the *dest* string, overwriting the terminating null byte ('\0') at the end of *dest*, and then adds a terminating null byte.

Parameters

in,out	<i>dest</i>	Pointer to string to append.
in	<i>src</i>	Pointer to source string.
in	<i>n</i>	maximal number of bytes to copy if string does not have \0 terminator.

Returns

pointer to the resulting string `dest`.

Example:

```
int main ()
{
    char src[50], dest[50];

    strcpy(src, "This is source");
    strcpy(dest, "This is destination");

    strncat(dest, src, 10);

    printf("Final destination string : |%s|", dest);

    return 0;
}
```

4.15.5.92 `errno_t strncat_s (char *restrict dest, rsize_t dest_max, const char *restrict src, rsize_t n)`

```
#include <string.h>
```

Check arguments and append at most `n` bytes of `src` string to the `dest` string, overwriting the terminating null byte (`'\0'`) at the end of `dest` and then adds a terminating null byte.

Parameters

in,out	<code>dest</code>	Pointer to string to append.
in	<code>dest_max</code>	max number of bytes to modify in the <code>dest</code> .
in	<code>src</code>	Pointer to source string.
in	<code>n</code>	maximal number of bytes to copy if string does not have <code>\0</code> terminator.

Return values

<code>0</code>	function completed work with success.
<code>!=0</code>	function completed work with error.

4.15.5.93 `int strncmp (const char * s1, const char * s2, size_t n)`

```
#include <string.h>
```

Compare at most `n` bytes of two strings `s1` and `s2`.

Parameters

in	<i>s1</i>	Pointer to first string to compare.
in	<i>s2</i>	Pointer to second string to compare.
in	<i>n</i>	maximal number of bytes to compare if string does not have \0 terminator.

Return values

0	strings are equal.
<0	string <i>s1</i> has lower value than <i>s2</i> .
>0	string <i>s1</i> has greater value than <i>s2</i> .

Example:

```
int main ()
{
    char str1[15];
    char str2[15];
    int ret;

    strcpy(str1, "abcdef");
    strcpy(str2, "ABCDEF");

    ret = strncmp(str1, str2, 4);

    if(ret > 0)
    {
        printf("str1 is less than str2");
    }
    else if(ret < 0)
    {
        printf("str2 is less than str1");
    }
    else
    {
        printf("str1 is equal to str2");
    }

    return 0;
}
```

4.15.5.94 char* strncpy (char * *dest*, const char * *src*, size_t *n*)

```
#include <string.h>
```

Copy at most *n* bytes of the string pointed to by *src*, including the terminating null byte ('\0'), to the buffer pointed to by *dest*.

Parameters

out	<i>dest</i>	pointer to destination string.
in	<i>src</i>	pointer to source string.
in	<i>n</i>	maximal number of bytes to copy if string does not have \0 terminator.

Returns

pointer to destination string *dest*.

Example:

```
int main()
{
    char src[40];
    char dest[100];

    memset(dest, '\0', sizeof(dest));
    strncpy(src, "This is example", 10);
    strncpy(dest, src, 10);

    printf("Final copied string : %s\n", dest);

    return 0;
}
```

4.15.5.95 `errno_t strncpy_s (char *restrict dest, rsize_t dest_max, const char *restrict src, rsize_t n)`

```
#include <string.h>
```

Check arguments and copy at most *n* bytes of the *src* string, including the terminating null byte ('\0') to the *dest* sized buffer.

Parameters

out	<i>dest</i>	pointer to destination string.
in	<i>dest_max</i>	max number of bytes to modify in the dest.
in	<i>src</i>	pointer to source string.
in	<i>n</i>	maximal number of bytes to copy if string does not have \0 terminator.

Return values

0	function completed work with success.
!=0	function completed work with error.

4.15.5.96 `size_t strlen (const char * s, size_t n)`

```
#include <string.h>
```

Calculate the length of the fixed-size string *s*, excluding the terminating null byte ('\0').

Parameters

in	<i>s</i>	Pointer to string.
in	<i>n</i>	maximal number of bytes to scan.

Returns

number of bytes in the string, or `n` if string is longer than `n`.

Example:

```
int main ()
{
    char string[32] = "hello, world";
    printf("String length is %zu\n", strlen(string, 10));
    return 0;
}
```

4.15.5.97 `size_t strlen_s (const char * s, size_t s_max)`

```
#include <string.h>
```

Check arguments and calculate the length of the `s` fixed-size string, excluding the terminating null byte (`'\0'`).

Parameters

in	<code>s</code>	pointer to string.
in	<code>s_max</code>	maximal number of bytes to scan.

Return values

<code>0</code>	<code>s</code> is a null pointer.
<code>!=0</code>	number of symbols that precede the terminating null character.

4.15.5.98 `char* strchr (const char * s, int c)`

```
#include <string.h>
```

Find last occurrence of character `c` in string `s`.

Parameters

in	<code>s</code>	Pointer to string to search in.
in	<code>c</code>	character to search.

Returns

pointer to matched character, or NULL if not found.

Example:

```
char str[] = "This is a sample string";
char * pch;
printf ("Looking for the 's' character in \"%s\"...\n",str);
pch=strrchr(str,'s');
while (pch != NULL)
{
    printf ("found at %d\n",pch-str+1);
    pch = strrchr(pch+1,'s');
}
```

4.15.5.99 size_t strspn (const char * s, const char * accept)

```
#include <string.h>
```

Calculate the length (in bytes) of the initial segment of s which consists entirely of bytes in accept.

Parameters

in	s	Pointer to string to search in.
in	accept	String containing the characters to match.

Returns

number of bytes in the initial segment of s which consist only of bytes from accept.

4.15.5.100 char* strstr (const char * text, const char * pattern)

```
#include <string.h>
```

Find the first occurrence of the substring pattern in the string text. The terminating null bytes ('\0') are not compared.

Parameters

in	text	Pointer to buffer with text for scan.
in	pattern	Pointer to pattern to find in text.

Returns

pointer to substring, or NULL if substring is not found.

Example:

```
int main()
{
    const char text [20] = "ExamplePoint";
    const char pattern [10] = "Point";
    char *ret;

    ret = strstr(text, pattern);

    printf("The substring is: %s\n", ret);

    return 0;
}
```

4.15.5.101 double strtod (const char * *nptr*, char ** *endptr*)

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

the initial portion of the string pointed to by *nptr* to double.

Note

The expected form of the (initial portion of the) string is optional leading white space, an optional plus ('+') or minus sign ('-') and then either

- (i) a decimal number.
- (ii) a hexadecimal number.
- (iii) an infinity.
- (iv) a NAN (not-a-number).

A decimal number consists of a nonempty sequence of decimal digits possibly containing a radix character (decimal point, locale-dependent, usually '.'), optionally followed by a decimal exponent. A decimal exponent consists of an 'E' or 'e', followed by an optional plus or minus sign, followed by a nonempty sequence of decimal digits, and indicates multiplication by a power of 10. A hexadecimal number consists of a "0x" or "0X" followed by a nonempty sequence of hexadecimal digits possibly containing a radix character, optionally followed by a binary exponent. A binary exponent consists of a 'P' or 'p', followed by an optional plus or minus sign, followed by a nonempty sequence of decimal digits, and indicates multiplication by a power of 2. At least one of radix character and binary exponent must be present. An infinity is either "INF" or "INFINITY", disregarding case.

Parameters

in	<i>nptr</i>	The start of the string.
in	<i>endptr</i>	A pointer to the end of the parsed string will be placed here.

Return values

<i>result</i>	conversion is success.
<i>HUGE_VAL</i>	if an overflow occurs. errno is set to ERANGE .
<i>-HUGE_VAL</i>	if an underflow occurs. errno is set to ERANGE .

4.15.5.102 float strtof (const char * *nptr*, char ** *endptr*)

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

the initial portion of the string pointed to by *nptr* to float.

Note

The expected form of the (initial portion of the) string is optional leading white space as recognized by `isspace(3)`, an optional plus ('+') or minus sign ('-') and then either

- (i) a decimal number.
- (ii) a hexadecimal number.
- (iii) an infinity.
- (iv) a NAN (not-a-number).

A decimal number consists of a nonempty sequence of decimal digits possibly containing a radix character (decimal point, locale-dependent, usually '.'), optionally followed by a decimal exponent. A decimal exponent consists of an 'E' or 'e', followed by an optional plus or minus sign, followed by a nonempty sequence of decimal digits, and indicates multiplication by a power of 10. A hexadecimal number consists of a "0x" or "0X" followed by a nonempty sequence of hexadecimal digits possibly containing a radix character, optionally followed by a binary exponent. A binary exponent consists of a 'P' or 'p', followed by an optional plus or minus sign, followed by a nonempty sequence of decimal digits, and indicates multiplication by a power of 2. At least one of radix character and binary exponent must be present. An infinity is either "INF" or "INFINITY", disregarding case.

Parameters

in	<i>nptr</i>	The start of the string.
in	<i>endptr</i>	A pointer to the end of the parsed string will be placed here.

Return values

<i>result</i>	conversion is success.
<i>HUGE_VAL</i>	if an overflow occurs. errno is set to ERANGE .
<i>-HUGE_VAL</i>	if an underflow occurs. errno is set to ERANGE .

4.15.5.103 char* strtok (char * *str*, const char * *delim*)

```
#include <string.h>
```

Function breaks a string into a sequence of zero or more nonempty tokens.

This function is MT-unsafe, i.e. it's not safe to call in a multithreaded programs

Parameters

in	<i>str</i>	The string to be parsed.
in	<i>delim</i>	The argument specifies a set of bytes that delimit the tokens in the parsed string.

Returns

a pointer to a null-terminated string containing the next token.

```
char str[] = "- This, a sample string.";
char *pch;
printf("Splitting string \"%s\" into tokens:\n",str);
pch = strtok(str, " ,.-");
while (pch != NULL)
{
    printf("%s\n", pch);
    pch = strtok(NULL, " ,.-");
}
return 0;
```

4.15.5.104 char* strtok_r (char *_restrict_s, const char *_restrict_sep, char **_restrict_p)

```
#include <string.h>
```

Function breaks a string into a sequence of zero or more nonempty tokens.

Parameters

in	s	The string to be parsed.
in	sep	The argument specifies a set of bytes that delimit the tokens in the parsed string.
out	p	used internally by <code>strtok_r()</code> in order to maintain context between successive calls that parse the same string.

Returns

a pointer to a null-terminated string containing the next token.

4.15.5.105 long strtol (const char * nptr, char ** endptr, int base)

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

Convert the initial part of the string in `nptr` to a long integer value according to the given `base`, which must be between 2 and 36 inclusive, or be the special value 0.

Note

The string may begin with an arbitrary amount of white spaces followed by a single optional '+' or '-' sign. If `base` is zero or 16, the string may then include a "0x" prefix, and the number will be read in base 16; otherwise, a zero base is taken as 10 (decimal) unless the next character is '0', in which case it is taken as 8 (octal). The remainder of the string is converted to a long int value in the obvious manner, stopping at the first character which is not a valid digit in the given `base`. (In bases above 10, the letter 'A' in either uppercase or lowercase represents 10, 'B' represents 11, and so forth, with 'Z' representing 35.)

Parameters

in	nptr	The start of the string
in	endptr	A pointer to the end of the parsed string will be placed here
in	base	The number base to use

Returns

- result of conversion on success
- LONG_MIN if an underflow occurs. `errno` is set to `ERANGE`.
- LONG_MAX if an overflow occurs. `errno` is set to `ERANGE`.

4.15.5.106 long double strtold (const char * *nptr*, char ** *endptr*)

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

the initial portion of the string pointed to by `nptr` to long double.

Note

The expected form of the (initial portion of the) string is optional leading white space as recognized by `isspace(3)`, an optional plus (+) or minus sign (-) and then either

- (i) a decimal number.
- (ii) a hexadecimal number.
- (iii) an infinity.
- (iv) a NAN (not-a-number).

A decimal number consists of a nonempty sequence of decimal digits possibly containing a radix character (decimal point, locale-dependent, usually '.'), optionally followed by a decimal exponent. A decimal exponent consists of an 'E' or 'e', followed by an optional plus or minus sign, followed by a nonempty sequence of decimal digits, and indicates multiplication by a power of 10. A hexadecimal number consists of a "0x" or "0X" followed by a nonempty sequence of hexadecimal digits possibly containing a radix character, optionally followed by a binary exponent. A binary exponent consists of a 'P' or 'p', followed by an optional plus or minus sign, followed by a nonempty sequence of decimal digits, and indicates multiplication by a power of 2. At least one of radix character and binary exponent must be present. An infinity is either "INF" or "INFINITY", disregarding case.

Parameters

in	<i>nptr</i>	The start of the string.
in	<i>endptr</i>	A pointer to the end of the parsed string will be placed here.

Return values

<i>result</i>	conversion is success.
<i>HUGE_VAL</i>	if an overflow occurs. <code>errno</code> is set to <code>ERANGE</code> .
<i>-HUGE_VAL</i>	if an underflow occurs. <code>errno</code> is set to <code>ERANGE</code> .

4.15.5.107 long long strtoll (const char * *nptr*, char ** *endptr*, int *base*)

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

Convert the initial part of the string in `nptr` to a long long integer value according to the given `base`, which must be between 2 and 36 inclusive, or be the special value 0.

Note

The string may begin with an arbitrary amount of white spaces (as determined by `isspace(3)`) followed by a single optional '+' or '-' sign. If `base` is zero or 16, the string may then include a "0x" prefix, and the number will be read in `base` 16; otherwise, a zero `base` is taken as 10 (decimal) unless the next character is '0', in which case it is taken as 8 (octal). The remainder of the string is converted to a long int value in the obvious manner, stopping at the first character which is not a valid digit in the given `base`. (In bases above 10, the letter 'A' in either uppercase or lowercase represents 10, 'B' represents 11, and so forth, with 'Z' representing 35.)

Parameters

in	<i>nptr</i>	The start of the string
in	<i>endptr</i>	A pointer to the end of the parsed string will be placed here
in	<i>base</i>	The number base to use

Returns

- result of conversion on success
- `LLONG_MIN` if an underflow occurs. `errno` is set to `ERANGE`.
- `LLONG_MAX` if an overflow occurs. `errno` is set to `ERANGE`.

4.15.5.108 unsigned long strtoul (const char * *cp*, char ** *endp*, int *base*)

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

Convert the initial part of the string in `nptr` to a unsigned long integer value according to the given `base`, which must be between 2 and 36 inclusive, or be the special value 0.

Note

The string may begin with an arbitrary amount of white spaces followed by a single optional '+' or '-' sign. If `base` is zero or 16, the string may then include a "0x" prefix, and the number will be read in `base` 16; otherwise, a zero `base` is taken as 10 (decimal) unless the next character is '0', in which case it is taken as 8 (octal). The remainder of the string is converted to a long int value in the obvious manner, stopping at the first character which is not a valid digit in the given `base`. (In bases above 10, the letter 'A' in either uppercase or lowercase represents 10, 'B' represents 11, and so forth, with 'Z' representing 35.)

Parameters

in	<i>cp</i>	The start of the string.
in	<i>endp</i>	A pointer to the end of the parsed string will be placed here.
in	<i>base</i>	The number base to use.

Return values

<i>result</i>	conversion is success.
<code>ULONG_MAX</code>	if an overflow occurs. <code>errno</code> is set to <code>ERANGE</code> .

4.15.5.109 unsigned long long strtoull (const char * *cp*, char ** *endp*, int *base*)

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

Convert the initial part of the string in `nptr` to a unsigned long long integer value according to the given `base`, which must be between 2 and 36 inclusive, or be the special value 0.

Note

The string may begin with an arbitrary amount of white spaces (as determined by `isspace(3)`) followed by a single optional '+' or '-' sign. If `base` is zero or 16, the string may then include a "0x" prefix, and the number will be read in `base` 16; otherwise, a zero `base` is taken as 10 (decimal) unless the next character is '0', in which case it is taken as 8 (octal). The remainder of the string is converted to a long int value in the obvious manner, stopping at the first character which is not a valid digit in the given `base`. (In bases above 10, the letter 'A' in either uppercase or lowercase represents 10, 'B' represents 11, and so forth, with 'Z' representing 35.)

Parameters

in	<i>cp</i>	The start of the string.
in	<i>endp</i>	A pointer to the end of the parsed string will be placed here.
in	<i>base</i>	The number base to use.

Return values

<i>result</i>	conversion is success.
<code>ULONG_MAX</code>	if an overflow occurs. <code>errno</code> is set to <code>ERANGE</code> .

4.15.5.110 time_t time (time_t * *time*)

```
#include <time.h>
```

Get the current calendar time as a value of type `time_t`.

Parameters

in	<i>time</i>	Pointer to an object of type <code>time_t</code> .
----	-------------	--

Returns

The current calendar time as a `time_t` object.

4.15.5.111 void timeadd (const struct timespec * a, const struct timespec * b, struct timespec * res)

```
#include <time.h>
```

The function adds two dates.

Parameters

in	<i>a</i>	Pointer to parameter to add.
in	<i>b</i>	Pointer to parameter to add.
out	<i>res</i>	Pointer to parameter to save result.

4.15.5.112 int64_t timespec_to_ms (const struct timespec * a)

```
#include <time.h>
```

The function converts time from timespec format to milliseconds.

Parameters

in	<i>a</i>	Pointer to timespec structure.
----	----------	--------------------------------

Return values

<i>time</i>	in milliseconds.
-------------	------------------

4.15.5.113 uint64_t timespec_to_nsec (const struct timespec * a)

```
#include <time.h>
```

The function converts time from timespec format to nanoseconds.

Parameters

in	<i>a</i>	Pointer to timespec structure.
----	----------	--------------------------------

Return values

<i>time</i>	in nanoseconds.
-------------	-----------------

4.15.5.114 void timesub (const struct timespec * a, const struct timespec * b, struct timespec * res)

```
#include <time.h>
```

The function subtracts two dates.

Parameters

in	<i>a</i>	Pointer to parameter to subtract.
in	<i>b</i>	Pointer to parameter to subtract.
out	<i>res</i>	Pointer to parameter to save result.

4.15.5.115 static int tolower (int c) [static]

```
#include <ctype.h>
```

Convert uppercase letter to lowercase.

Parameters

in	<i>c</i>	symbol for converting
----	----------	-----------------------

Returns

- the value of the converted letter,
- *c* if the conversion was not possible.

4.15.5.116 static int toupper (int c) [static]

```
#include <ctype.h>
```

Convert lowercase letter to uppercase.

Parameters

in	<i>c</i>	symbol for converting
----	----------	-----------------------

Returns

- the value of the converted letter,
- *c* if the conversion was not possible.

4.15.5.117 int unlink (const char * *pathname*)

```
#include <unistd.h>
```

Remove a link to a file.

Parameters

in	<i>pathname</i>	name of the file to remove link on it.
----	-----------------	--

Return values

0	on success.
-1	on error. <code>errno</code> is set appropriately.

4.15.5.118 void uuid_clear (uuid_t * *uu*)

```
#include <uuid/uuid.h>
```

set value to zero UUID.

Parameters

in	<i>uu</i>	value to cleanup.
----	-----------	-------------------

4.15.5.119 int uuid_compare (const uuid_t * *uu1*, const uuid_t * *uu2*)

```
#include <uuid/uuid.h>
```

Compare the two supplied uuid variables `uu1` and `uu2` to each other.

Parameters

in	<i>uu1</i>	first value to compare.
in	<i>uu2</i>	second value to compare.

Return values

0	- values are the same.
<0	<code>uu2</code> value is lexicographically greater than <code>uu1</code> .
>0	<code>uu2</code> value is lexicographically less than <code>uu1</code> .

4.15.5.120 void uuid_generate (uuid_t * out)

```
#include <uuid/uuid.h>
```

The `uuid_generate` function creates a new universally unique identifier (UUID).

Parameters

out	<i>out</i>	generated value.
-----	------------	------------------

4.15.5.121 int uuid_is_null (const uuid_t * uu)

```
#include <uuid/uuid.h>
```

Check if UUID is null.

Parameters

in	<i>uu</i>	value to check.
----	-----------	-----------------

Return values

0	UUID is not NULL.
1	otherwise.

4.15.5.122 int uuid_parse (const char * in, uuid_t * uu)

```
#include <uuid/uuid.h>
```

Convert an input UUID string into binary representation.

Function converts the UUID string given by `in` into the binary representation. The input UUID is a string of the form 1b4e28ba-2fa1-11d2-883f-b9a761bde3fb (in `printf(3)` format "%08x-%04x-%04x-%04x-%012x", 36 bytes plus the trailing '\0').

Parameters

in	<i>in</i>	string value to convert.
out	<i>uu</i>	binary UUID representation.

Return values

0	upon successfully parsing the input string.
-1	on failure.

4.15.5.123 void uuid_unparse (const uuid_t * uu, char * out)

```
#include <uuid/uuid.h>
```

Convert binary representation of UUID to string.

Parameters

in	<i>uu</i>	value to convert.
out	<i>out</i>	textual representation in lowercase format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

4.15.5.124 void uuid_unparse_upper (const uuid_t * uu, char * out)

```
#include <uuid/uuid.h>
```

Convert binary representation of UUID to string.

Parameters

in	<i>uu</i>	value to convert.
out	<i>out</i>	textual representation in uppercase format XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX.

4.15.5.125 int vasprintf (char ** strp, const char * fmt, va_list args)

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

print to allocated string.

Parameters

in,out	<i>args</i>	arguments for printing.
out	<i>strp</i>	Pointer to newly allocated string or NULL(if the function failed).
in	<i>fmt</i>	Format string.
in,out	...	Variable arguments for printing.

The functions [asprintf\(\)](#) and [vasprintf\(\)](#) are analogs of [sprintf\(\)](#) and [vsprintf\(\)](#), except that they allocate a string large enough to hold the output including the terminating null byte, and return a pointer to it via the first argument. This pointer should be passed to [free\(\)](#) to release the allocated storage when it is no longer needed.

These functions are Blowfish extensions, not in C or POSIX. They are also available under *BSD and GNU/Linux. The GNU/Linux implementation leaves *strp* in undefined state in case of error.

Return values

≥ 0	number of bytes printed if succeed
-1	if failed, and set <i>strp</i> to NULL

4.15.5.126 int vasprintf_s (char ** *strp*, const char * *fmt*, va_list *args*)

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

Print to allocated string in secure mode.

Parameters

out	<i>strp</i>	Pointer to newly allocated string or NULL(if the function failed).
in	<i>fmt</i>	Format string.
in,out	<i>args</i>	arguments for printing.

Return values

≥ 0	number of bytes printed if succeed
-1	if failed, and set <i>strp</i> to NULL

4.15.5.127 int vfprintf (FILE *_restrict_ *stream*, const char *_restrict_ *fmt*, va_list *ap*)

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

The `vfprintf()` is equivalent to the `fprintf()`, but uses an argument list.

Parameters

in	<i>stream</i>	A writeable file handle.
in	<i>fmt</i>	Format specification of output string.
in	<i>ap</i>	Value identifying a variable arguments list initialized with <code>va_start</code> . <code>va_list</code> is a special type.

Return values

≥ 0	number of characters printed if succeed.
-1	if error occurred.

4.15.5.128 int vsnprintf (char * buffer, size_t size, const char * format, va_list args)

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

Write formatted data from variable argument list to sized buffer.

Parameters

out	<i>buffer</i>	Pointer to a buffer where the resulting C-string is stored.
in	<i>size</i>	size of the buffer.
in	<i>format</i>	C string that contains a format string.
in,out	<i>args</i>	A value identifying a variable arguments list initialized with va_start. va_list is a special type.

Returns

The number of characters that would have been written if size had been sufficiently large, not counting the terminating null character. If an encoding error occurs, a negative number is returned. Notice that only when this returned value is non-negative and less than size, the string has been completely written.

Example:

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

void PrintFError ( const char * format, ... )
{
    char buffer[256];
    va_list args;
    va_start (args, format);
    vsnprintf (buffer,256,format, args);
    perror (buffer);
    va_end (args);
}

int main ()
{
    FILE * pFile;
    char szFileName []="myfile.txt";

    pFile = fopen (szFileName,"r");
    if (pFile == NULL)
        PrintFError ("Error opening '%s'",szFileName);
    else
    {
        fclose (pFile);
    }
    return 0;
}
```

4.15.5.129 int vsnprintf_s (char *restrict s, rsize_t n, const char *restrict format, va_list arg)

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

Write formatted data from variable argument list to sized buffer.

Parameters

out	<i>s</i>	Pointer to a buffer where the resulting C-string is stored.
in	<i>n</i>	size of the buffer.
in	<i>format</i>	C string that contains a format string.
in	<i>arg</i>	A value identifying a variable length arguments list initialized with <code>va_start</code> . <code>va_list</code> is a special type.

Returns

The number of characters that would have been written if size had been sufficiently large, not counting the terminating null character. If an encoding error or constrain violation occur, a negative number is returned. Notice that only when this returned value is non-negative and less than size, the string has been completely written.

4.15.5.130 int vsprintf (char * *buffer*, const char * *format*, va_list *args*)

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

Write formatted data from variable argument list to string.

Parameters

out	<i>buffer</i>	Pointer to a buffer where the resulting C-string is stored.
in	<i>format</i>	C string that contains a format string.
in,out	<i>args</i>	A value identifying a variable arguments list initialized with <code>va_start</code> . <code>va_list</code> is a special type.

Return values

<i>number</i>	of characters printed if succeed.
-1	if error occurred.

Example:

```

#include <stdio.h>
#include <stdarg.h>

void PrintFError ( const char * format, ... )
{
    char buffer[256];
    va_list args;
    va_start (args, format);
    vsprintf (buffer,format, args);
    perror (buffer);
    va_end (args);
}

int main ()
{
    FILE * pFile;
    char szFileName[]="myfile.txt";

    pFile = fopen (szFileName,"r");
    if (pFile == NULL)
        PrintFError ("Error opening '%s'",szFileName);
    else
    {
        fclose (pFile);
    }
    return 0;
}

```

4.15.5.131 int vsprintf_s (char *restrict s, rsize_t n, const char *restrict format, va_list arg)

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

Write formatted data from variable length argument list to string.

Parameters

out	<i>s</i>	Pointer to a buffer where the resulting C-string is stored.
in	<i>n</i>	buffer size in characters.
in	<i>format</i>	C string that contains a format string.
in	<i>arg</i>	A value identifying a variable arguments list initialized. with va_start. va_list is a special type.

Return values

<i>number</i>	of characters printed if succeed.
0	if constrain violation occurred.
-1	if encoding error occurred.

4.15.5.132 int vsscanf (const char * buffer, const char * fmt, va_list args)

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

vsscanf unformat a buffer into a list of arguments.

Parameters

in	<i>buffer</i>	input buffer.
in	<i>fmt</i>	format of buffer.
in	<i>args</i>	value identifying a variable arguments list initialized with <i>va_start</i> . <i>va_list</i> is a special type.

Return values

≥ 0	On success, the function returns the number of items in the argument list successfully filled. This count can match the expected number of items or be less -even zero- in the case of a matching failure.
<i>EOF</i>	In the case of an input failure before any data could be successfully interpreted.

4.15.5.133 int vsscanf_s (const char *restrict s, const char *restrict format, va_list arg)

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

vsscanf unformat a buffer into a list of arguments.

Parameters

in	<i>s</i>	Input buffer.
in	<i>format</i>	Format of buffer.
in	<i>arg</i>	Value identifying a variable arguments list initialized with <i>va_start</i> . <i>va_list</i> is a special type.

Return values

≥ 0	On success, the function returns the number of items in the argument list successfully filled. This count can match the expected number of items or be less -even zero- in the case of a matching failure.
<i>EOF</i>	In the case of an input failure before any data could be successfully interpreted.

4.15.5.134 ssize_t write (int fd, const void * buf, size_t count)

```
#include <unistd.h>
```

Attempt to write *count* bytes from buffer pointed to by *buf* to the file associated with the open file descriptor, *fd*.

Parameters

in	<i>fd</i>	File descriptor.
in	<i>buf</i>	Pointer to buffer from to read.
in	<i>count</i>	Number of bytes to write.

Returns

- non-negative integer indicating the number of bytes actually written - on success.
- -1 and set [errno](#) to indicate the error - otherwise.

4.15.6 Variable Documentation

4.15.6.1 FILE* stdin

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

Standard input stream (stub).

PRIVATE

4.16 Teesl_smc

Macros

- #define `U(n)` ((unsigned int)(n))
Defined for compatibility now.

Typedefs

- typedef int `TEES_SMCHandle`
SMC interface handle.
- typedef struct `smc_data` `TEES_SMCData`
SMC command data.

Functions

- TEE_Result `TEES_SMCHandleInit` (`TEES_SMCHandle` *handle)
Initialize handler fields.
- TEE_Result `TEES_SMCHandleFini` (`TEES_SMCHandle` handle)
release SMC interface handle.
- TEE_Result `TEES_SMCHandleCommand` (`TEES_SMCHandle` handle, `TEES_SMCData` *data)
Send SMC command to EL3.

4.16.1 Detailed Description

4.16.2 Function Documentation

4.16.2.1 TEE_Result TEES_SMCHandleCommand (TEES_SMCHandle handle, TEES_SMCData * data)

```
#include <tee_smc.h>
```

Send SMC command to EL3.

Parameters

in	<i>handle</i>	SMC handle which contains device descriptor.
in,out	<i>data</i>	SMC command data.

Return values

<code>TEE_SUCCESS</code>	on success, error code otherwise.
--------------------------	-----------------------------------

Example:

```

...
TEES_SMCHandle handle;
TEES_SMCHandle data = {
    .arg_types = SMC_ARG_TYPES(SMC_ARG_TYPE_VALUE,
                               SMC_ARG_TYPE_VALUE,
                               0, 0, 0, 0, 0);
    .args.arg[0] = CREATE_SMC_CMD(SMC_TYPE_FAST,           // SMC command code
                                 SMC_AARCH_32,
                                 SMC_SIP_SERVICE_MASK,
                                 0x42),
    .args.arg[1] = 5000000,                               // some SMC argument
};

TEE_Result res = TEES_SMCInit(&handle);
if (res != TEE_SUCCESS) {
    printf("TEST: Failed to initialize SMC, tee_err: %#x\n", res);
    return res;
}

res = TEES_SMCCommand(handle, &data);
if (res != TEE_SUCCESS) {
    printf("TEST: Failed to call SMC, tee_err: %#x\n", res);
    TEES_SMCFinis(handle);
    return res;
}

TEES_SMCFinis(handle);
...

```

4.16.2.2 TEE_Result TEES_SMCFinis (TEES_SMCHandle handle)

```
#include <tee_smc.h>
```

release SMC interface handle.

Parameters

in	<i>handle</i>	SMC handle which contains device descriptor.
----	---------------	--

Return values

<i>TEE_SUCCESS</i>	on success, error code otherwise.
--------------------	-----------------------------------

4.16.2.3 TEE_Result TEES_SMCInit (TEES_SMCHandle * handle)

```
#include <tee_smc.h>
```

Initialize handler fields.

Parameters

out	<i>handle</i>	Pointer to SMC handle which contains device descriptor.
-----	---------------	---

Return values

<i>TEE_SUCCESS</i>	on success, error code otherwise.
--------------------	-----------------------------------

4.17 Tee_sockets

Data Structures

- struct [TEE_iSocket_s](#)
iSocket instance Please refer to GPD_SPE_100 specification for detailed description. Basic rules are following: [More...](#)
- struct [TEE_tcpSocket_Setup_s](#)
TCP Setup structure. [More...](#)
- struct [TEE_udpSocket_Setup_s](#)
UDP Setup structure. [More...](#)
- struct [TEE_udpSocket_Change_s](#)
UDP change addr and port IOCTL structure. TEE_UDP_CHANGE functions are implementation as synonyms. Both server_addr and server_port must be provided for either call. In case of error returned Client should try to open new socket as usual. [More...](#)*
- struct [TEE_tlsSocket_PSK_Info_s](#)
Pre-Shared Key (PSK). When PSK is used, the TA needs to provide the key and a key identity to the TLS implementation. This structure holds that information Not supported. [More...](#)
- struct [TEE_tlsSocket_SRP_Info_s](#)
Secure Remote Password (SRP). When SRP is used, the TA needs to provide the password and the user identity to the TLS implementation. This structure holds that information. Not supported. [More...](#)
- struct [TEE_tlsSocket_ClientPDC_s](#)
This structure holds the opaque client certificate for the TA as well as the corresponding private key. This is used to provide pre-installed certificates for the TA authentication on Server. [More...](#)
- struct [TEE_tlsSocket_ServerPDC_s](#)
If the server Root public key has been pre-distributed to the TA, this structure holds the TEE_ObjectHandle to that key. If desirable, Server Root credentials could be provided as bulkCertChain - this is GP specs extension. publicKey is used by default. [More...](#)
- struct [TEE_tlsSocket_CertStorageCred_s](#)
Void type for future usage. Applications SHALL pass a NULL pointer. The intention is to have this structure hold handles or references to either trusted root certificates or a proper client certificate inside a future certificate storage of the TEE. [More...](#)
- struct [TEE_tlsSocket_Credentials_s](#)
Structure holding server and client credentials. [More...](#)
- union [TEE_tlsSocket_Credentials_s.__unnamed__](#)
- struct [TEE_tlsSocket_CallbackInfo_s](#)
Callback description structure. [More...](#)
- struct [TEE_tlsSocket_Setup_s](#)
TLS Setup structure. [More...](#)
- union [TEE_tlsSocket_Setup_s.__unnamed__](#)
- struct [TEE_tlsSocket_CB_Data_s](#)
IOCTL definitions. [More...](#)

Typedefs

- typedef struct __TEE_iSocketHandle * [TEE_iSocketHandle](#)
iSocket context handle
- typedef const struct [TEE_iSocket_s](#) [TEE_iSocket](#)
iSocket instance Please refer to GPD_SPE_100 specification for detailed description. Basic rules are following:
- typedef enum [TEE_ipSocket_ipVersion_e](#) [TEE_ipSocket_ipVersion](#)
IP version.

- typedef struct [TEE_tcpSocket_Setup_s](#) [TEE_tcpSocket_Setup](#)
TCP Setup structure.
- typedef struct [TEE_udpSocket_Setup_s](#) [TEE_udpSocket_Setup](#)
UDP Setup structure.
- typedef struct [TEE_udpSocket_Change_s](#) [TEE_udpSocket_Change](#)
UDP change addr and port IOCTL structure. TEE_UDP_CHANGE functions are implementation as synonyms. Both server_addr and server_port must be provided for either call. In case of error returned Client should try to open new socket as usual.*
- typedef struct [__TEES_lwtHandle](#) * [TEES_lwtHandle](#)
Handle representing active IWT connection. Must be treated as opaque structure.
- typedef enum [TEE_tlsSocket_tlsVersion_e](#) [TEE_tlsSocket_tlsVersion](#)
TLS protocol version to use.
- typedef enum [TEE_tlsSocket_CipherSuites_e](#) [TEE_tlsSocket_CipherSuites](#)
Cryptosuite ID definitions.
- typedef struct [TEE_tlsSocket_PSK_Info_s](#) [TEE_tlsSocket_PSK_Info](#)
Pre-Shared Key (PSK). When PSK is used, the TA needs to provide the key and a key identity to the TLS implementation. This structure holds that information Not supported.
- typedef struct [TEE_tlsSocket_SRP_Info_s](#) [TEE_tlsSocket_SRP_Info](#)
Secure Remote Password (SRP). When SRP is used, the TA needs to provide the password and the user identity to the TLS implementation. This structure holds that information. Not supported.
- typedef struct [TEE_tlsSocket_ClientPDC_s](#) [TEE_tlsSocket_ClientPDC](#)
This structure holds the opaque client certificate for the TA as well as the corresponding private key. This is used to provide pre-installed certificates for the TA authentication on Server.
- typedef struct [TEE_tlsSocket_ServerPDC_s](#) [TEE_tlsSocket_ServerPDC](#)
If the server Root public key has been pre-distributed to the TA, this structure holds the TEE_ObjectHandle to that key. If desirable, Server Root credentials could be provided as bulkCertChain - this is GP specs extension. publicKey is used by default.
- typedef struct [TEE_tlsSocket_CertStorageCred_s](#) [TEE_tlsSocket_CertStorageCred](#)
Void type for future usage. Applications SHALL pass a NULL pointer. The intention is to have this structure hold handles or references to either trusted root certificates or a proper client certificate inside a future certificate storage of the TEE.
- typedef enum [TEE_tlsSocket_ClientCredentialType_e](#) [TEE_tlsSocket_ClientCredentialType](#)
This specifies what kind of client credentials the TA has.
- typedef enum [TEE_tlsSocket_ServerCredentialType_e](#) [TEE_tlsSocket_ServerCredentialType](#)
This specifies what kind of server credentials a remote node has.
- typedef struct [TEE_tlsSocket_Credentials_s](#) [TEE_tlsSocket_Credentials](#)
Structure holding server and client credentials.
- typedef enum [TEE_tlsSocket_CallbackReasonType_e](#) [TEE_tlsSocket_CallbackReasonType](#)
Callback types.
- typedef struct [TEE_tlsSocket_CallbackInfo_s](#) [TEE_tlsSocket_CallbackInfo](#)
Callback description structure.
- typedef TEE_Result(* [TEE_tlsCallback](#)) ([TEE_iSocketHandle](#) ctx, [TEE_tlsSocket_CallbackInfo](#) *cbInfo, void *cbData, uint32_t *cbDataLength)
Callback function. This is specification extension. Used to allow client perform custom checks of certificate chain, OCSP response. etc. cbData buffer is valid only in the callback context.
- typedef enum [TEE_tlsSocket_StatusRequestType_e](#) [TEE_tlsSocket_StatusRequestType](#)
OCSP stapling certificate status request type.
- typedef enum [TEE_tlsSocket_ExtensionFlags_e](#) [TEE_tlsSocket_ExtensionFlags](#)
Certificate/OCSP validation mode and callback control flags.
- typedef struct [TEE_tlsSocket_Setup_s](#) [TEE_tlsSocket_Setup](#)
TLS Setup structure.
- typedef struct [TEE_tlsSocket_CB_Data_s](#) [TEE_tlsSocket_CB_Data](#)
IOCTL definitions.

Enumerations

- enum {
TEE_ISOCKET_ERROR_PROTOCOL = 0xF1007001, **TEE_ISOCKET_ERROR_REMOTE_CLOSED** = 0xF1007002, **TEE_ISOCKET_ERROR_TIMEOUT** = 0xF1007003, **TEE_ISOCKET_ERROR_OUT_OF_RESOURCES** = 0xF1007004,
TEE_ISOCKET_ERROR_LARGE_BUFFER = 0xF1007005, **TEE_ISOCKET_WARNING_PROTOCOL** = 0xF1007006, **TEE_ISOCKET_ERROR_HOSTNAME** = 0xF1007007 }
iSocket common errors
- enum { **TEE_ISOCKET_SERVER_NAME_MAX_LENGTH** = 255 }
Maximum server IPv4 address string length.
- enum **TEE_ipSocket_ipVersion_e** { **TEE_IP_VERSION_DC** = 0, **TEE_IP_VERSION_4** = 1, **TEE_IP_VERSION_6** = 2 }
IP version.
- enum { **TEE_ISOCKET_TCP_API_VERSION** = 0x01010000 }
TCP iSocket API version. Used to ensure API structures matching.
- enum { **TEE_ISOCKET_PROTOCOLID_TCP** = 0x65 }
TCP Protocol identifier.
- enum { **TEE_ISOCKET_TCP_WARNING_UNKNOWN_OUT_OF_BAND** = 0xF1010002 }
TCP Instance specific errors.
- enum { **TEE_ISOCKET_UDP_API_VERSION** = 0x01000000 }
UDP iSocket API version. Used to ensure API structures matching.
- enum { **TEE_ISOCKET_PROTOCOLID_UDP** = 0x66 }
UDP Protocol identifier.
- enum { **TEE_ISOCKET_UDP_WARNING_UNKNOWN_OUT_OF_BAND** = 0xF1020002 }
UDP Instance specific errors.
- enum { **TEE_UDP_CHANGEADDR** = 0x66000001, **TEE_UDP_CHANGEPORT** = 0x66000002 }
UDP IOCTL codes.
- enum { **TEES_IWT_LISTENER_NAME_MAX_LENGTH** = 91 }
TEES_IWT_LISTENER_NAME_MAX_LENGTH.
- enum **protocol_error_code** {
NO_ERROR = 0, **TEE_ISOCKET_IWC_ERROR_CHANNEL** = 0x81000000, **TEE_ISOCKET_IWC_ERROR_TIMEOUT** = 0x81000001, **TEE_ISOCKET_IWC_ERROR_NOT_IMPLEMENTED** = 0x81000002,
TEE_ISOCKET_IWC_ERROR_INVALID_VERSION = 0x81000003, **TEE_ISOCKET_IWC_ERROR_SWD_CLIENT_AUTH_FAIL** = 0x81000004, **TEE_ISOCKET_NET_ERROR_GENERIC** = 0x81010000, **TEE_ISOCKET_NET_ERROR_BAD_PARAMETER** = 0x81010001,
TEE_ISOCKET_NET_ERROR_BUFFER_TOO_SMALL = 0x81010002, **TEE_ISOCKET_NET_ERROR_LARGE_BUFFER** = 0x81010003, **TEE_ISOCKET_NET_ERROR_OUT_OF_RESOURCES** = 0x81010004, **TEE_ISOCKET_NET_ERROR_OUT_OF_ORDER** = 0x81010005,
TEE_ISOCKET_NET_ERROR_HOSTNAME_UNKNOWN = 0x81010006, **TEE_ISOCKET_NET_ERROR_HOSTNAME_NOT_FOUND** = 0x81010007, **TEE_ISOCKET_NET_ERROR_HOSTNAME_TRYAGAIN** = 0x81010008, **TEE_ISOCKET_NET_ERROR_CONNECTION_REFUSED** = 0x81010009,
TEE_ISOCKET_NET_ERROR_CONNECTION_REFUSED = 0x8101000A, **TEE_ISOCKET_NET_ERROR_NET_UNREACHABLE** = 0x8101000B, **TEE_ISOCKET_NET_ERROR_REMOTE_CLOSED** = 0x8101000C, **TEE_ISOCKET_NET_ERROR_TIMEOUT** = 0x8101000D,
TEE_ISOCKET_NET_ERROR_DATA_REMAIN = 0x8101000E, **TEE_ISOCKET_TLS_ERROR_CERT_PARSING** = 0x80000000, **TEE_ISOCKET_TLS_ERROR_CRL_PARSING** = 0x80000001, **TEE_ISOCKET_TLS_ERROR_CERT_EXPIRED** = 0x80000002,
TEE_ISOCKET_TLS_ERROR_CERT_SIGN_VERIFICATION = 0x80000003, **TEE_ISOCKET_TLS_ERROR_ECDHE_GEN_KEY_FAILED** = 0x81030010, **TEE_ISOCKET_TLS_ERROR_ECDHE_SHARED_SECRET** = 0x81030011, **TEE_ISOCKET_TLS_ERROR_ECDSA_SIGNATURE_VERIFY_FAILED** = 0x81030012,
TEE_ISOCKET_TLS_ERROR_ECDHE_SERIALIZING = 0x81030013, **TEE_ISOCKET_TLS_ERROR_CERT_COMMON_NAME_MISMATCH** = 0x81030014, **TEE_ISOCKET_TLS_ERROR_UNEXPECTED_MESSAGE** = 0x81030015, **TEE_ISOCKET_TLS_ERROR_HANDSHAKE_FAILED** = 0x81030016,
TEE_ISOCKET_TLS_ERROR_CERT_IS_TOO_LONG = 0x81030017, **TEE_ISOCKET_TLS_ERROR_NO_ALERT_PRESENT** = 0x81030018 }

```

= 0x81030018, TEE_ISOCKET_TLS_ERROR_ALERT_PENDING = 0x81030019, TEE_ISOCKET_TLS_ERROR_USER_CAN
= 0x8103001A,
TEE_ISOCKET_TLS_ERROR_CERT_UNKNOWN_CA = 0x8103001B, TEE_ISOCKET_TLS_ERROR_CERT_UNSUPPORTE
= 0x8103001C, TEE_ISOCKET_TLS_ERROR_CERT_REVOKED = 0x8103001D, TEE_ISOCKET_TLS_ERROR_CERT_STAT
= 0x8103001E,
TEE_ISOCKET_TLS_ALERT_CLOSE_NOTIFY = 0x81031000, TEE_ISOCKET_TLS_ALERT_UNEXPECTED_MSG
= 0x81031010, TEE_ISOCKET_TLS_ALERT_BAD_RECORD_MAC = 0x81031020, TEE_ISOCKET_TLS_ALERT_DECRYPT
= 0x81031021,
TEE_ISOCKET_TLS_ALERT_RECORD_OVERFLOW = 0x81031022, TEE_ISOCKET_TLS_ALERT_DECOMP_FAILED
= 0x81031030, TEE_ISOCKET_TLS_ALERT_HANDSHAKE_FAILED = 0x81031040, TEE_ISOCKET_TLS_ALERT_NO_CE
= 0x81031041,
TEE_ISOCKET_TLS_ALERT_BAD_CERTIFICATE = 0x81031042, TEE_ISOCKET_TLS_ALERT_UNSUPPORTED_CERT
= 0x81031043, TEE_ISOCKET_TLS_ALERT_CERT_REVOKED = 0x81031044, TEE_ISOCKET_TLS_ALERT_CERT_EXPIR
= 0x81031045,
TEE_ISOCKET_TLS_ALERT_CERT_UNKNOWN = 0x81031046, TEE_ISOCKET_TLS_ALERT_ILLEGAL_PARAMETER
= 0x81031047, TEE_ISOCKET_TLS_ALERT_UNKNOWN_CA = 0x81031048, TEE_ISOCKET_TLS_ALERT_ACCESS_DEN
= 0x81031049,
TEE_ISOCKET_TLS_ALERT_DECODE_ERROR = 0x81031050, TEE_ISOCKET_TLS_ALERT_DECRYPT_ERROR
= 0x81031051, TEE_ISOCKET_TLS_ALERT_EXPORT_RESTRICTED = 0x81031060, TEE_ISOCKET_TLS_ALERT_PROTC
= 0x81031070,
TEE_ISOCKET_TLS_ALERT_INSUFFICIENT_SECURITY = 0x81031071, TEE_ISOCKET_TLS_ALERT_INTERNAL_ERROR
= 0x81031080, TEE_ISOCKET_TLS_ALERT_INAPPROPRIATE_FALLBACK = 0x81031086, TEE_ISOCKET_TLS_ALERT_
= 0x81031090,
TEE_ISOCKET_TLS_ALERT_NO_RENEGOTIATION = 0x81031100, TEE_ISOCKET_TLS_ALERT_MISSING_EXTENSION
= 0x81031109, TEE_ISOCKET_TLS_ALERT_UNSUPPORTED_EXTENSION = 0x81031110, TEE_ISOCKET_TLS_ALERT_C
= 0x81031111,
TEE_ISOCKET_TLS_ALERT_UNRECOGNIZED_NAME = 0x81031112, TEE_ISOCKET_TLS_ALERT_BAD_CERT_STATUS
= 0x81031113, TEE_ISOCKET_TLS_ALERT_BAD_CERT_HASH_VALUE = 0x81031114, TEE_ISOCKET_TLS_ALERT_UNK
= 0x81031115,
TEE_ISOCKET_TLS_ALERT_CERT_REQUIRED = 0x81031116 }

```

Propriate protocol specific error codes. According to GPD_SPE_010 specification, TEE error code range 0x80000000..0x8FFFFFFF is reserved for implementation specific error. In addition, TEE Socket Subsystem considers protocol errors as specification extension and includes specification ID into code: 0x8 | 3 digit BCD spec ID | error code.

- enum { **TEE_ISOCKET_TLS_API_VERSION** = 0x01030000 }
TEE iSocket API version. Used to enshure API structures matching.
- enum { **TEE_ISOCKET_PROTOCOLID_TLS** = 0x67 }
TLS Protocol identifier.
- enum {
TEE_ISOCKET_TLS_ERROR_REJECTED_SUITE = 0xF1030001, **TEE_ISOCKET_TLS_ERROR_VERSION**
= 0xF1030002, **TEE_ISOCKET_TLS_ERROR_UNSUPPORTED_SUITE** = 0xF1030003, **TEE_ISOCKET_TLS_ERROR_HAN**
= 0xF1030004,
TEE_ISOCKET_TLS_ERROR_AUTHENTICATION = 0xF1030005, **TEE_ISOCKET_TLS_ERROR_DATA**
= 0xF1030006 }
TLS Instance specific errors.
- enum **TEE_tlsSocket_tlsVersion_e** { **TEE_TLS_VERSION_ALL** = 0, **TEE_TLS_VERSION_1v2** = 1 }
TLS protocol version to use.
- enum **TEE_tlsSocket_CipherSuites_e** {
TLS_NULL_WITH_NULL_NULL = 0x0000, **TLS_RSA_WITH_3DES_EDE_CBC_SHA** = 0x000A,
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA = 0x0013, **TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA**
= 0x0016,
TLS_RSA_WITH_AES_128_CBC_SHA = 0x002F, **TLS_DHE_DSS_WITH_AES_128_CBC_SHA** =
0x0032, **TLS_DHE_RSA_WITH_AES_128_CBC_SHA** = 0x0033, **TLS_RSA_WITH_AES_256_CBC_SHA**
= 0x0035,
TLS_DHE_DSS_WITH_AES_256_CBC_SHA = 0x0038, **TLS_DHE_RSA_WITH_AES_256_CBC_SHA** =
0x0039, **TLS_RSA_WITH_AES_128_CBC_SHA256** = 0x003C, **TLS_RSA_WITH_AES_256_CBC_SHA256**

```

= 0x003D,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256 = 0x0040, TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
= 0x0067, TLS_DHE_DSS_WITH_AES_256_CBC_SHA256 = 0x006A, TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
= 0x006B,
TLS_PSK_WITH_3DES_EDE_CBC_SHA = 0x008B, TLS_PSK_WITH_AES_128_CBC_SHA = 0x008C,
TLS_PSK_WITH_AES_256_CBC_SHA = 0x008D, TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA =
0x008F,
TLS_DHE_PSK_WITH_AES_128_CBC_SHA = 0x0090, TLS_DHE_PSK_WITH_AES_256_CBC_SHA =
0x0091, TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA = 0x0093, TLS_RSA_PSK_WITH_AES_128_CBC_SHA
= 0x0094,
TLS_RSA_PSK_WITH_AES_256_CBC_SHA = 0x0095, TLS_RSA_WITH_AES_128_GCM_SHA256 =
0x009C, TLS_RSA_WITH_AES_256_GCM_SHA384 = 0x009D, TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
= 0x009E,
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 = 0x009F, TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
= 0x00A2, TLS_DHE_DSS_WITH_AES_256_GCM_SHA384 = 0x00A3, TLS_PSK_WITH_AES_128_GCM_SHA256
= 0x00A8,
TLS_PSK_WITH_AES_256_GCM_SHA384 = 0x00A9, TLS_DHE_PSK_WITH_AES_128_GCM_SHA256
= 0x00AA, TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 = 0x00AB, TLS_RSA_PSK_WITH_AES_128_GCM_SHA256
= 0x00AC,
TLS_RSA_PSK_WITH_AES_256_GCM_SHA384 = 0x00AD, TLS_PSK_WITH_AES_128_CBC_SHA256
= 0x00AE, TLS_PSK_WITH_AES_256_CBC_SHA384 = 0x00AF, TLS_DHE_PSK_WITH_AES_128_CBC_SHA256
= 0x00B2,
TLS_DHE_PSK_WITH_AES_256_CBC_SHA384 = 0x00B3, TLS_RSA_PSK_WITH_AES_128_CBC_SHA256
= 0x00B6, TLS_RSA_PSK_WITH_AES_256_CBC_SHA384 = 0x00B7, TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
= 0xC008,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA = 0xC009, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
= 0xC00A, TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA = 0xC012, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
= 0xC013,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA = 0xC014, TLS_SRP_SHA_WITH_3DES_EDE_CBC_SHA
= 0xC01A, TLS_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA = 0xC01B, TLS_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA
= 0xC01C,
TLS_SRP_SHA_WITH_AES_128_CBC_SHA = 0xC01D, TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA
= 0xC01E, TLS_SRP_SHA_DSS_WITH_AES_128_CBC_SHA = 0xC01F, TLS_SRP_SHA_WITH_AES_256_CBC_SHA
= 0xC020,
TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA = 0xC021, TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA
= 0xC022, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 = 0xC023, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
= 0xC024,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 = 0xC027, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
= 0xC028, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 = 0xC02B, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
= 0xC02C,
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 = 0xC02F, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
= 0xC030, TLS_ECDHE_PSK_WITH_3DES_EDE_CBC_SHA = 0xC034, TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA
= 0xC035,
TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA = 0xC036, TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256
= 0xC037, TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384 = 0xC038, TLS_RSA_WITH_AES_128_CCM
= 0xC09C,
TLS_RSA_WITH_AES_256_CCM = 0xC09D, TLS_DHE_RSA_WITH_AES_128_CCM = 0xC09E,
TLS_DHE_RSA_WITH_AES_256_CCM = 0xC09F, TLS_PSK_WITH_AES_128_CCM = 0xC0A4,
TLS_PSK_WITH_AES_256_CCM = 0xC0A5, TLS_DHE_PSK_WITH_AES_128_CCM = 0xC0A6,
TLS_DHE_PSK_WITH_AES_256_CCM = 0xC0A7 }

```

Cryptosuite ID definitions.

- enum `TEE_tlsSocket_ClientCredentialType_e` { `TEE_TLS_CLIENT_CRED_NONE` = 0, `TEE_TLS_CLIENT_CRED_PDC` = 1, `TEE_TLS_CLIENT_CRED_CSC` = 2 }

This specifies what kind of client credentials the TA has.

- enum `TEE_tlsSocket_ServerCredentialType_e` { `TEE_TLS_SERVER_CRED_PDC` = 0, `TEE_TLS_SERVER_CRED_CSC` = 1 }

This specifies what kind of server credentials a remote node has.

- enum `TEE_tlsSocket_CallbackReasonType_e` {
`TEE_ISOCKET_TLS_CB_CHECK_CERT_CHAIN = 1`, `TEE_ISOCKET_TLS_CB_BAD_CERT_CHAIN = 2`,
`TEE_ISOCKET_TLS_CB_CHECK_OCSP_STATUS = 11`, `TEE_ISOCKET_TLS_CB_UNKNOWN_OCSP_STATUS = 12`,
`TEE_ISOCKET_TLS_CB_REVOKED_OCSP_STATUS = 13` }
Callback types.
- enum `TEE_tlsSocket_StatusRequestType_e` { `TEE_ISOCKET_TLS_OCSP_STATUS_REQUEST_NO = 0`, `TEE_ISOCKET_TLS_OCSP_STATUS_REQUEST = 1` }
OCSP stapling certificate status request type.
- enum `TEE_tlsSocket_ExtensionFlags_e` {
`TEE_ISOCKET_TLS_CERT_NAME_CHECK_CLIENT = 0x00000001`, `TEE_ISOCKET_TLS_CERT_KEYUSAGE_CHECK_CLIENT = 0x00000002`,
`TEE_ISOCKET_TLS_CERT_NOTIFY_CLIENT = 0x00000004`, `TEE_ISOCKET_TLS_OCSP_CHECK_CLIENT = 0x00010000`,
`TEE_ISOCKET_TLS_OCSP_CHECK_ADVISORY = 0x00020000`, `TEE_ISOCKET_TLS_OCSP_CHECK_MANDATORY = 0x00040000` }
Certificate/OCSP validation mode and callback control flags.
- enum { `TEE_ISOCKET_TLS_MAX_ALPN_LIST_LENGTH = 16` }
- enum { `TEE_TLS_BINDING_INFO = 0x67000001` }
IOCTL codes.

Functions

- `TEE_Result TEES_IwtOpenChannel` (`const char *listenerName`, `TEES_IwtHandle *iwtCtx`)
Open interworld transport (IWT) connection (channel) to the NWd Listener "listenerName".
- `TEE_Result TEES_IwtWrite` (`TEES_IwtHandle iwtCtx`, `const void *buf`, `uint32_t *length`)
Write length bytes from buf to the active IWT connection iwtCtx.
- `TEE_Result TEES_IwtRead` (`TEES_IwtHandle iwtCtx`, `void *buf`, `uint32_t *length`)
Read length bytes to buf from the active IWT connection iwtCtx.
- `TEE_Result TEES_IwtCloseChannel` (`TEES_IwtHandle iwtCtx`)
Close active IWT connection iwtCtx.

Variables

- `const TEE_iSocket *const TEE_tcpSocket`
Public TCP instance pointer.
- `const TEE_iSocket *const TEE_udpSocket`
Public UDP instance pointer.
- `const TEE_iSocket *const TEE_tlsSocket`
Public TLS instance pointer.

4.17.1 Detailed Description

4.17.2 Data Structure Documentation

4.17.2.1 struct TEE_iSocket_s

iSocket instance Please refer to GPD_SPE_100 specification for detailed description. Basic rules are following:

- Specific Interface instance structure is exported from its shared library by dedicated pointer (TEE_tcpSocket for TCP and TEE_tlsSocket for TLS);
- Each protocol connection starts with `open()` and ends with `close()` call and is represented by `TEE_iSocketHandle` context (ctx);
- ctx is [out] parameter for `open()` and [in] for other functions;
- setup [in] is pointer to protocol specific structure (`TEE_tcpSocket_Setup` or `TEE_tlsSocket_Setup`);
- buf is [in] for `send()`, [out] for `recv()` and [in,out] for `ioctl()`, length is [in,out] and other parameters are [in];
- NULL parameters are not valide except for `close()`;
- *length = 0 is not valid for `recv()/send()`.

Data Fields

- uint32_t `TEE_iSocketVersion`
- uint8_t `protocolID`
- TEE_Result(* `open`)(`TEE_iSocketHandle` *ctx, void *setup, uint32_t *protocolError)
- TEE_Result(* `close`)(`TEE_iSocketHandle` ctx)
- TEE_Result(* `send`)(`TEE_iSocketHandle` ctx, const void *buf, uint32_t *length, uint32_t timeout)
- TEE_Result(* `recv`)(`TEE_iSocketHandle` ctx, void *buf, uint32_t *length, uint32_t timeout)
- uint32_t(* `error`)(`TEE_iSocketHandle` ctx)
- TEE_Result(* `ioctl`)(`TEE_iSocketHandle` ctx, uint32_t commandCode, void *buf, uint32_t *length)

Field Documentation

TEE_Result(* TEE_iSocket_s::close) (TEE_iSocketHandle ctx)

used to close connection

uint32_t(* TEE_iSocket_s::error) (TEE_iSocketHandle ctx)

used to get specific protocol error in case of TEE_ISOCKET_ERROR_PROTOCOL

TEE_Result(* TEE_iSocket_s::ioctl) (TEE_iSocketHandle ctx, uint32_t commandCode, void *buf, uint32_t *length)

used to send ioctl to opened connection

TEE_Result(* TEE_iSocket_s::open) (TEE_iSocketHandle *ctx, void *setup, uint32_t *protocolError)

used to open connection

uint8_t TEE_iSocket_s::protocolID

Protocol identifier

TEE_Result(* TEE_iSocket_s::recv) (TEE_iSocketHandle ctx, void *buf, uint32_t *length, uint32_t timeout)

used to receive data over opened connection

TEE_Result(* TEE_iSocket_s::send) (TEE_iSocketHandle ctx, const void *buf, uint32_t *length, uint32_t timeout)

used to send data over opened connection

uint32_t TEE_iSocket_s::TEE_iSocketVersion

The specification version number

4.17.2.2 struct TEE_tcpSocket_Setup_s

TCP Setup structure.

Data Fields

uint32_t	apiVersion	Must be TEE_ISOCKET_TCP_API_VERSION
TEE_ipSocket_ipVersion	ipVersion	Must be TEE_IP_VERSION_4
uint32_t	openTimeout	Connection open timeout
char *	server_addr	Pointer to IPv4 address or DNS name string e.g., "10.0.0.5", "www.samsung.com"
int	server_port	Server port

4.17.2.3 struct TEE_udpSocket_Setup_s

UDP Setup structure.

Data Fields

uint32_t	apiVersion	Must be TEE_ISOCKET_UDP_API_VERSION
TEE_ipSocket_ipVersion	ipVersion	Must be TEE_IP_VERSION_4
uint32_t	openTimeout	Connection open timeout
char *	server_addr	Pointer to IPv4 address or DNS name string e.g., "10.0.0.5", "www.samsung.com"
int	server_port	Server port

4.17.2.4 struct TEE_udpSocket_Change_s

UDP change addr and port IOCTL structure. TEE_UDP_CHANGE* functions are implementation as synonyms. Both server_addr and server_port must be provided for either call. In case of error returned Client should try to open new socket as usual.

Data Fields

char	server_addr[TEE_ISOCKET_SERVER_NAME_MAX_LENGTH]	IPv4 address or DNS name string
int	server_port	Server port

4.17.2.5 struct TEE_tlsSocket_PSK_Info_s

Pre-Shared Key (PSK). When PSK is used, the TA needs to provide the key and a key identity to the TLS implementation. This structure holds that information Not supported.

Data Fields

char *	pskIdentity	
TEE_ObjectHandle	pskKey	

4.17.2.6 struct TEE_tlsSocket_SRP_Info_s

Secure Remote Password (SRP). When SRP is used, the TA needs to provide the password and the user identity to the TLS implementation. This structure holds that information. Not supported.

Data Fields

char *	srpIdentity	
char *	srpPassword	

4.17.2.7 struct TEE_tlsSocket_ClientPDC_s

This structure holds the opaque client certificate for the TA as well as the corresponding private key. This is used to provide pre-installed certificates for the TA authentication on Server.

Data Fields

char *	bulkCertChain	
uint32_t	bulkSize	
TEE_ObjectHandle	privateKey	

4.17.2.8 struct TEE_tlsSocket_ServerPDC_s

If the server Root public key has been pre-distributed to the TA, this structure holds the TEE_ObjectHandle to that key. If desirable, Server Root credentials could be provided as bulkCertChain - this is GP specs extension. publicKey is used by default.

Data Fields

char *	bulkCertChain	
uint32_t	bulkSize	
TEE_ObjectHandle	publicKey	

4.17.2.9 struct TEE_tlsSocket_CertStorageCred_s

Void type for future usage. Applications SHALL pass a NULL pointer. The intention is to have this structure hold handles or references to either trusted root certificates or a proper client certificate inside a future certificate storage of the TEE.

4.17.2.10 struct TEE_tlsSocket_Credentials_s

Structure holding server and client credentials.

Data Fields

union TEE_tlsSocket_Credentials_s	__unnamed__	
union TEE_tlsSocket_Credentials_s	__unnamed__	
TEE_tlsSocket_ClientCredentialType	clientCredType	Client credentials provisioning type
TEE_tlsSocket_ServerCredentialType	serverCredType	Server credentials provisioning type

4.17.2.11 union TEE_tlsSocket_Credentials_s.__unnamed__

Data Fields

TEE_tlsSocket_CertStorageCred *	rootCertStore	Certificate storage - for future extension
TEE_tlsSocket_ServerPDC *	serverCred	Predistributed (explicitly provided)

4.17.2.12 struct TEE_tlsSocket_CallbackInfo_s

Callback description structure.

Data Fields

uint32_t	protocolError	
TEE_tlsSocket_CallbackReasonType	reason	
TEE_Result	result	

4.17.2.13 struct TEE_tlsSocket_Setup_s

TLS Setup structure.

Data Fields

union TEE_tlsSocket_Setup_s	__unnamed__	PSK or SRP - not supported
TEE_tlsSocket_tlsVersion	acceptServerVersion	TLS version, MUST be TEE_TLS_VERSION_1v2
TEE_tlsSocket_CipherSuites *	allowedCipherSuites	Pointer to an array of allowed cipher suites terminated by the constant 0 (TLS_NULL_WITH_NULL_NULL). If suite is not supported it will be ignored. If noone suite in the list is supported the error will be returned.
char **	alpnList	ALPN TLS extension (RFC 7301). OPTIONAL. List of maximum TEE_ISOCKET_TLS_MAX_ALPN_LIST_LENGTH null terminated strings terminated with extra NULL pointer
uint32_t	apiVersion	Must be set to TEE_ISOCKET_TLS_API_VERSION
TEE_iSocketHandle *	baseContext	Lower level connection handle. Should be pointer according to GP Spec
TEE_iSocket *	baseSocket	Pointer to the lower level protocol (TCP) instance
TEE_tlsSocket_Credentials *	credentials	Server certificate and Client certificate
TEE_tlsSocket_ExtensionFlags	extFlags	TEE TLS extension options control flags. Callback must be set as well
TEE_tlsSocket_StatusRequestType	ocspStatusType	OCSP stapling certificate status request type. The only supported type is TEE_ISOCKET_TLS_OCSP_STATUS_REQUEST (RFC 6066). The Server must support this extension else handshake will fail. To be correctly verified OCSP response should be signed with Responder certificate provided with response. Responder certificate must be marked as id-pkix-ocsp-nocheck or must be signed with the second CA from Server certificate chain or with Root certificate. Response valid age is one week. Client is responsible to verify all other response types/cases itself.
char *	serverName	Pointer to Server fully qualified DNS hostname string. OPTIONAL. Is used: 1) in Server Name Indication TLS Extension (RFC 6066); 2) to check matching with Subject IDs in Server certificate if TEE_ISOCKET_TLS_CERT_NAME_CHECK flag is set in extFlags: <ul style="list-style-type: none"> • check order is: DNS ID, CN ID; • leftmost '*' label in CERTIFICATE name is supported, e.g. "*.testserver.com". If Client requires other check rules than implemented, it could set this

4.17.2.14 union TEE_tlsSocket_Setup_s.__unnamed__

Data Fields

TEE_tlsSocket_PSK_Info *	PSKInfo	Pre-Shared secret key - not supported
TEE_tlsSocket_SRP_Info *	SRPInfo	Secure Remote Password - not supported

4.17.2.15 struct TEE_tlsSocket_CB_Data_s

IOCTL definitions.

This structure is returned in the output buffer by the ioctl function TEE_TLS_BINDING_INFO. It provides “TLS-Unique” channel binding information according to [RFC 5929].

Data Fields

uint8_t	cb_data[]	
uint32_t	cb_data_size	

4.17.3 Typedef Documentation

4.17.3.1 typedef const struct TEE_iSocket_s TEE_iSocket

```
#include <tee_isocket.h>
```

iSocket instance Please refer to GPD_SPE_100 specification for detailed description. Basic rules are following:

- Specific Interface instance structure is exported from its shared library by dedicated pointer (TEE_tcpSocket for TCP and TEE_tlsSocket for TLS);
- Each protocol connection starts with `open()` and ends with `close()` call and is represented by `TEE_iSocketHandle` context (ctx);
- ctx is [out] parameter for `open()` and [in] for other functions;
- setup [in] is pointer to protocol specific structure (`TEE_tcpSocket_Setup` or `TEE_tlsSocket_Setup`);
- buf is [in] for `send()`, [out] for `recv()` and [in,out] for `ioctl()`, length is [in,out] and other parameters are [in];
- NULL parameters are not valide except for `close()`;
- *length = 0 is not valid for `recv()/send()`.

4.17.3.2 typedef struct TEE_tlsSocket_CB_Data_s TEE_tlsSocket_CB_Data

```
#include <tee_tlsocket.h>
```

IOCTL definitions.

This structure is returned in the output buffer by the ioctl function TEE_TLS_BINDING_INFO. It provides “TLS-Unique” channel binding information according to [RFC 5929].

4.17.4 Enumeration Type Documentation

4.17.4.1 anonymous enum

```
#include <tee_isocket.h>
```

iSocket common errors

Enumerator

TEE_ISOCKET_ERROR_PROTOCOL Protocol specific error. Use error() function to get detailed code

TEE_ISOCKET_ERROR_REMOTE_CLOSED The remote host has closed the connection

TEE_ISOCKET_ERROR_TIMEOUT Timeout occurs. Not fatal error

TEE_ISOCKET_ERROR_OUT_OF_RESOURCES Failed to allocate resources for the socket

TEE_ISOCKET_ERROR_LARGE_BUFFER Buffer is too large to be sent in one datagram

TEE_ISOCKET_WARNING_PROTOCOL Protocol specific warning. Not fatal error. Use error() function

TEE_ISOCKET_ERROR_HOSTNAME The provided hostname cannot be resolved

4.17.4.2 anonymous enum

```
#include <tee_isocket.h>
```

Maximum server IPv4 address string length.

Enumerator

TEE_ISOCKET_SERVER_NAME_MAX_LENGTH Maximum server IPv4 address string length

4.17.4.3 anonymous enum

```
#include <tee_tcpsocket.h>
```

TCP iSocket API version. Used to ensure API structures matching.

Enumerator

TEE_ISOCKET_TCP_API_VERSION Currently supported version

4.17.4.4 anonymous enum

```
#include <tee_tcpsocket.h>
```

TCP Protocol identifier.

Enumerator

TEE_ISOCKET_PROTOCOLID_TCP GP TCP protocol ID

4.17.4.5 anonymous enum

```
#include <tee_tcpsocket.h>
```

TCP Instance specific errors.

Enumerator

TEE_ISOCKET_TCP_WARNING_UNKNOWN_OUT_OF_BAND A protocol message was received that is not supported

4.17.4.6 anonymous enum

```
#include <tee_udpsocket.h>
```

UDP iSocket API version. Used to ensure API structures matching.

Enumerator

TEE_ISOCKET_UDP_API_VERSION Currently supported version

4.17.4.7 anonymous enum

```
#include <tee_udpsocket.h>
```

UDP Protocol identifier.

Enumerator

TEE_ISOCKET_PROTOCOLID_UDP GP UDP protocol ID

4.17.4.8 anonymous enum

```
#include <tee_udpsocket.h>
```

UDP Instance specific errors.

Enumerator

TEE_ISOCKET_UDP_WARNING_UNKNOWN_OUT_OF_BAND A protocol message was received that is not supported

4.17.4.9 anonymous enum

```
#include <tees_iwt.h>
```

TEES_IWT_LISTENER_NAME_MAX_LENGTH.

Enumerator

TEES_IWT_LISTENER_NAME_MAX_LENGTH The maximum allowed listener name length

4.17.4.10 anonymous enum

```
#include <tee_tlsocket.h>
```

TLS iSocket API version. Used to ensure API structures matching.

Enumerator

TEE_ISOCKET_TLS_API_VERSION Currently supported version

4.17.4.11 anonymous enum

```
#include <tee_tlsocket.h>
```

TLS Protocol identifier.

Enumerator

TEE_ISOCKET_PROTOCOLID_TLS GP TLS protocol ID

4.17.4.12 anonymous enum

```
#include <tee_tlsocket.h>
```

TLS Instance specific errors.

Enumerator

TEE_ISOCKET_TLS_ERROR_REJECTED_SUITE The server rejected all the offered cipher suites

TEE_ISOCKET_TLS_ERROR_VERSION The server only supports lower version of TLS than allowed

TEE_ISOCKET_TLS_ERROR_UNSUPPORTED_SUITE Cryptosuite is not implemented or supported

TEE_ISOCKET_TLS_ERROR_HANDSHAKE An error occurred during the TLS handshake

TEE_ISOCKET_TLS_ERROR_AUTHENTICATION The server could not be authenticated

TEE_ISOCKET_TLS_ERROR_DATA Wrong formatted or not anticipated data

4.17.4.13 enum protocol_error_code

```
#include <protocol_errors.h>
```

Propriate protocol specific error codes. According to GPD_SPE_010 specification, TEE error code range 0x80000000..0x8FFFFFFF is reserved for implementation specific error. In addition, TEE Socket Subsystem considers protocol errors as specification extension and includes specification ID into code: 0x8 | 3 digit BCD spec ID | error code.

Enumerator

TEE_ISOCKET_IWC_ERROR_CHANNEL IWC Fatal error. Connection could not be used anymore and must be closed

TEE_ISOCKET_IWC_ERROR_TIMEOUT IWC watchdog timeout. Fatal error. Connection must be closed

TEE_ISOCKET_IWC_ERROR_NOT_IMPLEMENTED Proxy function is not implemented

TEE_ISOCKET_IWC_ERROR_INVALID_VERSION Communication channel structure version mismatch

TEE_ISOCKET_IWC_ERROR_SWD_CLIENT_AUTH_FAILED Client TA authentication failed

TEE_ISOCKET_NET_ERROR_GENERIC Connection generic error

TEE_ISOCKET_NET_ERROR_BAD_PARAMETERS Bad parameters

TEE_ISOCKET_NET_ERROR_BUFFER_TOO_SMALL Buffer too small

TEE_ISOCKET_NET_ERROR_LARGE_BUFFER Buffer too large

TEE_ISOCKET_NET_ERROR_OUT_OF_RESOURCES Could not allocate socket resources

TEE_ISOCKET_NET_ERROR_OUT_OF_MEMORY Not enough memory

TEE_ISOCKET_NET_ERROR_HOSTNAME_UNKNOWN Unknown host

TEE_ISOCKET_NET_ERROR_HOSTNAME_NOTRESOLVED Unknown host name

TEE_ISOCKET_NET_ERROR_HOSTNAME_TRYAGAIN Host currently unavailable, try again later

TEE_ISOCKET_NET_ERROR_COMMUNICATION Connection error EIO

TEE_ISOCKET_NET_ERROR_CONNECTION_REFUSED Connection refused

TEE_ISOCKET_NET_ERROR_NET_UNREACHABLE Unknown or unreachable network

TEE_ISOCKET_NET_ERROR_REMOTE_CLOSED Remote host closed connection

TEE_ISOCKET_NET_ERROR_TIMEOUT Connection timeout

TEE_ISOCKET_NET_ERROR_DATA_REMAIN Data remain

TEE_ISOCKET_TLS_ERROR_CERT_PARSING Certificate parsing error. The same as TEE_ERROR_CERT_PARSING

TEE_ISOCKET_TLS_ERROR_CRL_PARSING Certificate parsing error. The same as TEE_ERROR_CRL_PARSING

TEE_ISOCKET_TLS_ERROR_CERT_EXPIRED Certificate expired. The same as TEE_ERROR_CERT_EXPIRED

TEE_ISOCKET_TLS_ERROR_CERT_SIGN_VERIFICATION Certificate sign verification error. The same as TEE_ERROR_CERT_VERIFICATION

TEE_ISOCKET_TLS_ERROR_ECDHE_GEN_KEY ECDHE key generation error

TEE_ISOCKET_TLS_ERROR_ECDHE_SHARED_SECRET Shared secrete calculation error

TEE_ISOCKET_TLS_ERROR_ECDHE_UNSUPPORTED_CURVE Unsupported EC curve

TEE_ISOCKET_TLS_ERROR_ECDHE_SERIALIZING ECDHE parameters serialization error

TEE_ISOCKET_TLS_ERROR_CERT_COMMON_NAME_VERIFICATION Certificate Common Name verification failed

TEE_ISOCKET_TLS_ERROR_UNEXPECTED_MESSAGE Unkown, bad formatted or unexpected TLS handshake message

TEE_ISOCKET_TLS_ERROR_HANDSHAKE_UNEXPECTED_PARAMETER Handshake unexpected parameter

TEE_ISOCKET_TLS_ERROR_CERT_IS_TOO_LONG Certificate is too long

TEE_ISOCKET_TLS_ERROR_NO_ALERT_PRESENT No alert recieved from peer

TEE_ISOCKET_TLS_ERROR_ALERT_PENDING Alert pending

TEE_ISOCKET_TLS_ERROR_USER_CANCELED Certificate chain is rejected by Client TA

TEE_ISOCKET_TLS_ERROR_CERT_UNKNOWN_CA Wrong CA in certificate chain

TEE_ISOCKET_TLS_ERROR_CERT_UNSUPPORTED Certificate has not supported attributes

TEE_ISOCKET_TLS_ERROR_CERT_REVOKED Certificate revoked

TEE_ISOCKET_TLS_ERROR_CERT_STATUS_UNKNOWN Unknown certificate revocation status

TEE_ISOCKET_TLS_ALERT_CLOSE_NOTIFY Connection will be closed

TEE_ISOCKET_TLS_ALERT_UNEXPECTED_MSG Unexpected TLS handshake message

TEE_ISOCKET_TLS_ALERT_BAD_RECORD_MAC Bad message MAC

TEE_ISOCKET_TLS_ALERT_DECRYPT_FAILED Message decryption failed

TEE_ISOCKET_TLS_ALERT_RECORD_OVERFLOW Record overflow

TEE_ISOCKET_TLS_ALERT_DECOMP_FAILED Message decompression failed

TEE_ISOCKET_TLS_ALERT_HANDSHAKE_FAILED Handshake failed

TEE_ISOCKET_TLS_ALERT_NO_CERTIFICATE No certificate recieved

TEE_ISOCKET_TLS_ALERT_BAD_CERTIFICATE Bad certificate recieved

TEE_ISOCKET_TLS_ALERT_UNSUPPORTED_CERT Unsupported certificate format/parameters

TEE_ISOCKET_TLS_ALERT_CERT_REVOKED Certificate revoked

TEE_ISOCKET_TLS_ALERT_CERT_EXPIRED Unsupported certificate

TEE_ISOCKET_TLS_ALERT_CERT_UNKNOWN Unknown certificate

TEE_ISOCKET_TLS_ALERT_ILLEGAL_PARAMETER Illegal parameter

TEE_ISOCKET_TLS_ALERT_UNKNOWN_CA Unknown certificate CA

TEE_ISOCKET_TLS_ALERT_ACCESS_DENIED Access denied

TEE_ISOCKET_TLS_ALERT_DECODE_ERROR Message decode error

TEE_ISOCKET_TLS_ALERT_DECRYPT_ERROR Message decryption error

TEE_ISOCKET_TLS_ALERT_EXPORT_RESTRICTED Export restricted

TEE_ISOCKET_TLS_ALERT_PROTOCOL_VERSION Unsupported TLS version

TEE_ISOCKET_TLS_ALERT_INSUFFICIENT_SECURITY Too vulnarable TLS version

TEE_ISOCKET_TLS_ALERT_INTERNAL_ERROR Internal error

TEE_ISOCKET_TLS_ALERT_INAPPROPRIATE_FALLBACK Invalid connection retry attempt from a client

TEE_ISOCKET_TLS_ALERT_USER_CANCELED Canceled by user

TEE_ISOCKET_TLS_ALERT_NO_RENEGOTIATION Renegotiation error

TEE_ISOCKET_TLS_ALERT_MISSING_EXTENSION Missing extension

TEE_ISOCKET_TLS_ALERT_UNSUPPORTED_EXTENSION Unsupported extension

TEE_ISOCKET_TLS_ALERT_CERT_UNOBTAINABLE Client certificate unobtainable

TEE_ISOCKET_TLS_ALERT_UNRECOGNIZED_NAME Wrong server name specified

TEE_ISOCKET_TLS_ALERT_BAD_CERT_STATUS_RESPONSE Bad OCSP response

TEE_ISOCKET_TLS_ALERT_BAD_CERT_HASH_VALUE Bar client certificate hash

TEE_ISOCKET_TLS_ALERT_UNKNOWN_PSK_IDENTITY Unknown PSK identity

TEE_ISOCKET_TLS_ALERT_CERT_REQUIRED Client certificate not provided

4.17.4.14 enum TEE_ipSocket_ipVersion_e

```
#include <tee_isocket.h>
```

IP version.

Enumerator

TEE_IP_VERSION_DC Either IP version - not supported
TEE_IP_VERSION_4 IPv4 - the only supported IP protocol version
TEE_IP_VERSION_6 IPv6 - not supported

4.17.4.15 enum TEE_tlsSocket_CallbackReasonType_e

```
#include <tee_tlssocket.h>
```

Callback types.

Enumerator

TEE_ISOCKET_TLS_CB_CHECK_CERT_CHAIN Check server certificate chain
TEE_ISOCKET_TLS_CB_BAD_CERT_CHAIN Bad server certificate chain - informative only
TEE_ISOCKET_TLS_CB_CHECK_OCSP_STATUS Check OCSP response
TEE_ISOCKET_TLS_CB_UNKNOWN_OCSP_STATUS Unknown OCSP response status
TEE_ISOCKET_TLS_CB_REVOKED_OCSP_STATUS Revoked OCSP response status obtained

4.17.4.16 enum TEE_tlsSocket_ClientCredentialType_e

```
#include <tee_tlssocket.h>
```

This specifies what kind of client credentials the TA has.

Enumerator

TEE_TLS_CLIENT_CRED_NONE No client credentials
TEE_TLS_CLIENT_CRED_PDC Predistributed (explicitly provided)
TEE_TLS_CLIENT_CRED_CSC Certificate storage - for future extension

4.17.4.17 enum TEE_tlsSocket_ExtensionFlags_e

```
#include <tee_tlssocket.h>
```

Certificate/OCSP validation mode and callback control flags.

Enumerator

TEE_ISOCKET_TLS_CERT_NAME_CHECK_CLIENT Client will perform Server name check
TEE_ISOCKET_TLS_CERT_KEYUSAGE_CHECK_CLIENT Client will perform key usage check
TEE_ISOCKET_TLS_CERT_NOTIFY_CLIENT Call Client callback at any case
TEE_ISOCKET_TLS_OCSP_CHECK_CLIENT Client will perform OCSP stapling response check
TEE_ISOCKET_TLS_OCSP_CHECK_ADVISORY Check OCSP stapling response but let Client decide
TEE_ISOCKET_TLS_OCSP_CHECK_MANDATORY Fail if OCSP status unknown or revoked but notify Client

4.17.4.18 enum TEE_tlsSocket_ServerCredentialType_e

```
#include <tee_tlsocket.h>
```

This specifies what kind of server credentials a remote node has.

Enumerator

TEE_TLS_SERVER_CRED_PDC Predistributed (explicitly provided)
TEE_TLS_SERVER_CRED_CSC Certificate storage - for future extension

4.17.4.19 enum TEE_tlsSocket_StatusRequestType_e

```
#include <tee_tlsocket.h>
```

OCSP stapling certificate status request type.

Enumerator

TEE_ISOCKET_TLS_OCSP_STATUS_REQUEST_NO No certificate status request
TEE_ISOCKET_TLS_OCSP_STATUS_REQUEST OCSP stapling certificate status request - RFC 6066

4.17.4.20 enum TEE_tlsSocket_tlsVersion_e

```
#include <tee_tlsocket.h>
```

TLS protocol version to use.

Enumerator

TEE_TLS_VERSION_ALL Any supported TLS protocol version. Only TLS 1.2 is supported
TEE_TLS_VERSION_1v2 TLS 1.2 protocol version - the only supported version

4.17.5 Function Documentation

4.17.5.1 TEE_Result TEES_IwtCloseChannel (TEES_IwtHandle iwtCtx)

```
#include <tees_iwt.h>
```

Close active IWT connection *iwtCtx*.

Parameters

in	<i>iwtCtx</i>	Handle representing active IWT connection
----	---------------	---

Return values

TEE_SUCCESS	no error
TEE_ERROR_*	on failure

4.17.5.2 TEE_Result TEES_IwtOpenChannel (const char * *listenerName*, TEES_IwtHandle * *iwtCtx*)

```
#include <tees_iwt.h>
```

Open interworld transport (IWT) connection (channel) to the NWd Listener "listenerName".

Parameters

in	<i>listenerName</i>	Pointer to an array storing Listener name. Listener name must correspond to Listener's UNIX domain socket, which must have following name: /dev/socket/iwt/"listenerName". Listener name length must not exceed TEES_IWT_LISTENER_NAME_MAX_LENGTH
out	<i>iwtCtx</i>	Pointer to the handle representing active IWT connection

Return values

<i>TEE_SUCCESS</i>	no error
<i>TEE_ERROR_*</i>	on failure

4.17.5.3 TEE_Result TEES_IwtRead (TEES_IwtHandle *iwtCtx*, void * *buf*, uint32_t * *length*)

```
#include <tees_iwt.h>
```

Read *length* bytes to *buf* from the active IWT connection *iwtCtx*.

Parameters

in	<i>iwtCtx</i>	Handle representing active IWT connection
in	<i>buf</i>	Pointer to buffer containing data to be read
in,out	<i>length</i>	pointer to data length to be read on input and actually read on output

Return values

<i>TEE_SUCCESS</i>	no error
<i>TEE_ERROR_*</i>	on failure

4.17.5.4 TEE_Result TEES_IwtWrite (TEES_IwtHandle *iwtCtx*, const void * *buf*, uint32_t * *length*)

```
#include <tees_iwt.h>
```

Write *length* bytes from *buf* to the active IWT connection *iwtCtx*.

Parameters

in	<i>iwtCtx</i>	Handle representing active IWT connection
in	<i>buf</i>	Pointer to buffer containing data to be written
in,out	<i>length</i>	pointer to data length to be written on input and actually written on output

Return values

<i>TEE_SUCCESS</i>	no error
<i>TEE_ERROR_*</i>	on failure

5 Data Structure Documentation

5.1 __TEE_SC_CardKeyRef

Data Fields

- uint8_t [scKeyID](#)
- uint8_t [scKeyVersion](#)

5.1.1 Detailed Description

This type defines the reference to the card keys which shall be used for the secure channel.

5.1.2 Field Documentation

5.1.2.1 uint8_t __TEE_SC_CardKeyRef::scKeyID

key identifier of the SC card key

5.1.2.2 uint8_t __TEE_SC_CardKeyRef::scKeyVersion

key version of the SC card key

5.2 __TEE_SC_DeviceKeyRef

Data Fields

- TEE_SC_KeyType [scKeyType](#)
 - union {
 - TEE_ObjectHandle [scBaseKeyHandle](#)
 - [TEE_SC_KeySetRef](#) [scKeySetRef](#)
- ```
};
```

### 5.2.1 Detailed Description

This type defines the reference to the device keys which shall be used for the secure channel.

## 5.2.2 Field Documentation

### 5.2.2.1 union { ... }

union of keys

### 5.2.2.2 TEE\_SC\_KeyType \_\_TEE\_SC\_DeviceKeyRef::scKeyType

type of SC keys

## 5.3 \_\_TEE\_SC\_DeviceKeyRef.\_\_unnamed\_\_

### Data Fields

- TEE\_ObjectHandle [scBaseKeyHandle](#)
- TEE\_SC\_KeySetRef [scKeySetRef](#)

### 5.3.1 Field Documentation

#### 5.3.1.1

SC base key (acc. to SCP02)

#### 5.3.1.2

Key-ENC, Key-MAC (acc. to SCP02, SCP03)

## 5.4 \_\_TEE\_SC\_KeySetRef

### Data Fields

- TEE\_ObjectHandle [scKeyEncHandle](#)
- TEE\_ObjectHandle [scKeyMacHandle](#)

### 5.4.1 Detailed Description

This type can be used to define a key set with Key-MAC and Key-ENC according to SCP '02' and SCP '03'.

## 5.4.2 Field Documentation

### 5.4.2.1 TEE\_ObjectHandle \_\_TEE\_SC\_KeySetRef::scKeyEncHandle

the Key-ENC (static encryption key)

### 5.4.2.2 TEE\_ObjectHandle \_\_TEE\_SC\_KeySetRef::scKeyMacHandle

the Key-MAC (static MAC key)

## 5.5 \_\_TEE\_SC\_OID

### Data Fields

- uint8\_t \* [buffer](#)
- uint32\_t [bufferLen](#)

### 5.5.1 Detailed Description

This type defines the type of protocol which shall be used for the secure channel.

### 5.5.2 Field Documentation

#### 5.5.2.1 uint8\_t\* \_\_TEE\_SC\_OID::buffer

the value of the OID

#### 5.5.2.2 uint32\_t \_\_TEE\_SC\_OID::bufferLen

length of the SC OID

## 5.6 \_\_TEE\_SC\_Params

### Data Fields

- uint8\_t [scType](#)
- [TEE\\_SC\\_OID](#) [scOID](#)
- TEE\_SC\_SecurityLevel [scSecurityLevel](#)
- [TEE\\_SC\\_CardKeyRef](#) [scCardKeyRef](#)
- [TEE\\_SC\\_DeviceKeyRef](#) [scDeviceKeyRef](#)

## 5.6.1 Detailed Description

This type defines the parameters which are needed to set up a secure channel.

## 5.6.2 Field Documentation

### 5.6.2.1 TEE\_SC\_CardKeyRef \_\_TEE\_SC\_Params::scCardKeyRef

reference to SC card keys

### 5.6.2.2 TEE\_SC\_DeviceKeyRef \_\_TEE\_SC\_Params::scDeviceKeyRef

reference to SC device keys

### 5.6.2.3 TEE\_SC\_OID \_\_TEE\_SC\_Params::scOID

the SC type defined by OID

### 5.6.2.4 TEE\_SC\_SecurityLevel \_\_TEE\_SC\_Params::scSecurityLevel

the SC security level

### 5.6.2.5 uint8\_t \_\_TEE\_SC\_Params::scType

the SC type

## 5.7 \_\_TEE\_SEAID

### Data Fields

- uint8\_t \* [buffer](#)
- uint32\_t [bufferLen](#)

### 5.7.1 Detailed Description

This type is used to pass the AID of the Applet that a TA wants to communicate with.

## 5.7.2 Field Documentation

### 5.7.2.1 uint8\_t\* \_\_TEE\_SEAID::buffer

the value of the applet's AID

### 5.7.2.2 uint32\_t \_\_TEE\_SEAID::bufferLen

length of the applet's AID

## 5.8 \_\_TEE\_SEReaderProperties

### Data Fields

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

### 5.8.1 Detailed Description

This type is used to return information about a Secure Element reader.

### 5.8.2 Field Documentation

#### 5.8.2.1 bool \_\_TEE\_SEReaderProperties::selectResponseEnable

true if the response to a SELECT is available in the TEE

#### 5.8.2.2 bool \_\_TEE\_SEReaderProperties::sePresent

true if SE is present in the reader

#### 5.8.2.3 bool \_\_TEE\_SEReaderProperties::teeOnly

true if only accessible via the TEE

## 5.9 smc\_data

SMC command description.

## Data Fields

- uint64\_t [arg\\_types](#)
- struct smc\_args [args](#)

### 5.9.1 Detailed Description

SMC command description.

### 5.9.2 Field Documentation

#### 5.9.2.1 uint64\_t smc\_data::arg\_types

Argument types description

#### 5.9.2.2 struct smc\_args smc\_data::args

Arguments array

## 5.10 stat

### Data Fields

- uint32\_t [st\\_mode](#)
- uint32\_t [st\\_uid](#)
- uint32\_t [st\\_gid](#)
- uint32\_t [st\\_size](#)

### 5.10.1 Detailed Description

Stat struct

### 5.10.2 Field Documentation

#### 5.10.2.1 uint32\_t stat::st\_gid

Group ID of file

#### 5.10.2.2 uint32\_t stat::st\_mode

Mode of file

#### 5.10.2.3 uint32\_t stat::st\_size

Size of file

#### 5.10.2.4 uint32\_t stat::st\_uid

User ID of file

## 6 File Documentation

---

### 6.1 assert.h File Reference

Assertion support.

#### Macros

- #define [assert](#)(expr) `__assert__(expr, __FILE__, __LINE__)`  
*Abort the program if assertion is false.*

#### 6.1.1 Detailed Description

Assertion support.

Copyright

(C) 2013-2018, Samsung Electronics Co., Ltd.

### 6.2 cache.h File Reference

cache maintenance

#### Functions

- int [TEES\\_DcacheFlush](#) (const void \*addr, size\_t nbytes)  
*Perform data cache flush.*
- int [TEES\\_DcacheClean](#) (const void \*addr, size\_t nbytes)  
*Perform data cache clean.*

#### 6.2.1 Detailed Description

cache maintenance

Copyright

(C) 2013-2016, Samsung Electronics Co., Ltd.

## 6.3 core/error.h File Reference

### Macros

- #define [EPERM](#) 1 /\* Operation not permitted \*/
- #define [ENOENT](#) 2 /\* No such file or directory \*/
- #define [ESRCH](#) 3 /\* No such process \*/
- #define [EINTR](#) 4 /\* Interrupted system call \*/
- #define [EIO](#) 5 /\* I/O error \*/
- #define [ENXIO](#) 6 /\* No such device or address \*/
- #define [E2BIG](#) 7 /\* Argument list too long \*/
- #define [ENOEXEC](#) 8 /\* Exec format error \*/
- #define [EBADF](#) 9 /\* Bad file number \*/
- #define [ECHILD](#) 10 /\* No child processes \*/
- #define [EAGAIN](#) 11 /\* Try again \*/
- #define [ENOMEM](#) 12 /\* Out of memory \*/
- #define [EACCES](#) 13 /\* Permission denied \*/
- #define [EFAULT](#) 14 /\* Bad address \*/
- #define [ENOTBLK](#) 15 /\* Block device required \*/
- #define [EBUSY](#) 16 /\* Device or resource busy \*/
- #define [EEXIST](#) 17 /\* File exists \*/
- #define [EXDEV](#) 18 /\* Cross-device link \*/
- #define [ENODEV](#) 19 /\* No such device \*/
- #define [ENOTDIR](#) 20 /\* Not a directory \*/
- #define [EISDIR](#) 21 /\* Is a directory \*/
- #define [EINVAL](#) 22 /\* Invalid argument \*/
- #define [ENFILE](#) 23 /\* File table overflow \*/
- #define [EMFILE](#) 24 /\* Too many open files \*/
- #define [ENOTTY](#) 25 /\* Not a typewriter \*/
- #define [ETXTBSY](#) 26 /\* Text file busy \*/
- #define [EFBIG](#) 27 /\* File too large \*/
- #define [ENOSPC](#) 28 /\* No space left on device \*/
- #define [ESPIPE](#) 29 /\* Illegal seek \*/
- #define [EROFS](#) 30 /\* Read-only file system \*/
- #define [EMLINK](#) 31 /\* Too many links \*/
- #define [EPIPE](#) 32 /\* Broken pipe \*/
- #define [EDOM](#) 33 /\* Math argument out of domain of func \*/
- #define [ERANGE](#) 34 /\* Math result not representable \*/
- #define [EDEADLK](#) 35 /\* Resource deadlock would occur \*/
- #define [ENAMETOOLONG](#) 36 /\* File name too long \*/
- #define [ENOLCK](#) 37 /\* No record locks available \*/
- #define [ENOSYS](#) 38 /\* Function not implemented \*/
- #define [ENOTEMPTY](#) 39 /\* Directory not empty \*/
- #define [ELOOP](#) 40 /\* Too many symbolic links encountered \*/
- #define [EWOULDBLOCK EAGAIN](#) /\* Operation would block \*/
- #define [ENOMSG](#) 42 /\* No message of desired type \*/
- #define [EIDRM](#) 43 /\* Identifier removed \*/
- #define [ECHRNG](#) 44 /\* Channel number out of range \*/
- #define [EL2NSYNC](#) 45 /\* Level 2 not synchronized \*/
- #define [EL3HLT](#) 46 /\* Level 3 halted \*/
- #define [EL3RST](#) 47 /\* Level 3 reset \*/
- #define [ELNRNG](#) 48 /\* Link number out of range \*/
- #define [EUNATCH](#) 49 /\* Protocol driver not attached \*/
- #define [ENOCSI](#) 50 /\* No CSI structure available \*/

- #define [EL2HLT](#) 51 /\* Level 2 halted \*/
- #define [EBADE](#) 52 /\* Invalid exchange \*/
- #define [EBADR](#) 53 /\* Invalid request descriptor \*/
- #define [EXFULL](#) 54 /\* Exchange full \*/
- #define [ENOANO](#) 55 /\* No anode \*/
- #define [EBADRQC](#) 56 /\* Invalid request code \*/
- #define [EBADSLT](#) 57 /\* Invalid slot \*/
- #define [EDEADLOCK](#) [EDEADLK](#)
- #define [EBFONT](#) 59 /\* Bad font file format \*/
- #define [ENOSTR](#) 60 /\* Device not a stream \*/
- #define [ENODATA](#) 61 /\* No data available \*/
- #define [ETIME](#) 62 /\* Timer expired \*/
- #define [ENOSR](#) 63 /\* Out of streams resources \*/
- #define [ENONET](#) 64 /\* Machine is not on the network \*/
- #define [ENOPKG](#) 65 /\* Package not installed \*/
- #define [EREMOTE](#) 66 /\* Object is remote \*/
- #define [ENOLINK](#) 67 /\* Link has been severed \*/
- #define [EADV](#) 68 /\* Advertise error \*/
- #define [ESRMNT](#) 69 /\* Srmount error \*/
- #define [ECOMM](#) 70 /\* Communication error on [send](#) \*/
- #define [EPROTO](#) 71 /\* Protocol error \*/
- #define [EMULTIHOP](#) 72 /\* Multihop attempted \*/
- #define [EDOTDOT](#) 73 /\* RFS specific error \*/
- #define [EBADMSG](#) 74 /\* Not a data message \*/
- #define [EOVERFLOW](#) 75 /\* Value too large for defined data type \*/
- #define [ENOTUNIQ](#) 76 /\* Name not unique on network \*/
- #define [EBADFD](#) 77 /\* File descriptor in bad state \*/
- #define [EREMCHG](#) 78 /\* Remote address changed \*/
- #define [ELIBACC](#) 79 /\* Can not access a needed shared library \*/
- #define [ELIBBAD](#) 80 /\* Accessing a corrupted shared library \*/
- #define [ELIBSCN](#) 81 /\* .lib section in a.out corrupted \*/
- #define [ELIBMAX](#) 82 /\* Attempting to link in too many shared libraries \*/
- #define [ELIBEXEC](#) 83 /\* Cannot exec a shared library directly \*/
- #define [EILSEQ](#) 84 /\* Illegal byte sequence \*/
- #define [ERESTART](#) 85 /\* Interrupted system call should be restarted \*/
- #define [ESTRPIPE](#) 86 /\* Streams pipe error \*/
- #define [EUSERS](#) 87 /\* Too many users \*/
- #define [ENOTSOCK](#) 88 /\* Socket operation on non-socket \*/
- #define [EDESTADDRREQ](#) 89 /\* Destination address required \*/
- #define [EMSGSIZE](#) 90 /\* Message too long \*/
- #define [EPROTOTYPE](#) 91 /\* Protocol wrong type for [socket](#) \*/
- #define [ENOPROTOOPT](#) 92 /\* Protocol not available \*/
- #define [EPROTONOSUPPORT](#) 93 /\* Protocol not supported \*/
- #define [ESOCKTNOSUPPORT](#) 94 /\* Socket type not supported \*/
- #define [EOPNOTSUPP](#) 95 /\* Operation not supported on transport endpoint \*/
- #define [EPFNOSUPPORT](#) 96 /\* Protocol family not supported \*/
- #define [EAFNOSUPPORT](#) 97 /\* Address family not supported by protocol \*/
- #define [EADDRINUSE](#) 98 /\* Address already in use \*/
- #define [EADDRNOTAVAIL](#) 99 /\* Cannot assign requested address \*/
- #define [ENETDOWN](#) 100 /\* Network is down \*/
- #define [ENETUNREACH](#) 101 /\* Network is unreachable \*/
- #define [ENETRESET](#) 102 /\* Network dropped connection because of reset \*/
- #define [ECONNABORTED](#) 103 /\* Software caused connection [abort](#) \*/
- #define [ECONNRESET](#) 104 /\* Connection reset by peer \*/
- #define [ENOBUFS](#) 105 /\* No buffer space available \*/

- #define **EISCONN** 106 /\* Transport endpoint is already connected \*/
- #define **ENOTCONN** 107 /\* Transport endpoint is not connected \*/
- #define **ESHUTDOWN** 108 /\* Cannot **send** after transport endpoint shutdown \*/
- #define **ETOOMANYREFS** 109 /\* Too many references: cannot splice \*/
- #define **ETIMEDOUT** 110 /\* Connection timed out \*/
- #define **ECONNREFUSED** 111 /\* Connection refused \*/
- #define **EHOSTDOWN** 112 /\* Host is down \*/
- #define **EHOSTUNREACH** 113 /\* No route to host \*/
- #define **EALREADY** 114 /\* Operation already in progress \*/
- #define **EINPROGRESS** 115 /\* Operation now in progress \*/
- #define **ESTALE** 116 /\* Stale file handle \*/
- #define **EUCLEAN** 117 /\* Structure needs cleaning \*/
- #define **ENOTNAM** 118 /\* Not a XENIX named type file \*/
- #define **ENAVAIL** 119 /\* No XENIX semaphores available \*/
- #define **EISNAM** 120 /\* Is a named type file \*/
- #define **EREMOTEIO** 121 /\* Remote I/O error \*/
- #define **EDQUOT** 122 /\* Quota exceeded \*/
- #define **ECANCELED** 125 /\* Operation canceled \*/
- #define **ENOKEY** 126 /\* Required key not available \*/

## 6.4 core/mman.h File Reference

### Macros

- #define **MAP\_ANONYMOUS** (1 << 0)
- #define **MAP\_POPULATE** (1 << 1)
- #define **MAP\_FIXED** (1 << 2)
- #define **MAP\_PRIVATE** (1 << 3)
- #define **MAP\_SHARED** (1 << 4)
- #define **PROT\_NONE** 0
- #define **PROT\_READ** 1
- #define **PROT\_WRITE** 2
- #define **PROT\_EXEC** 4
- #define **PGOFF\_SHIFT** 12

## 6.5 sys/mman.h File Reference

Memory mapping declarations.

### Macros

- #define **MAP\_FAILED** ((void \*)-1)

### Functions

- void \* **mmap** (void \*addr, size\_t len, int prot, int flags, int fd, **off\_t** offset)
- int **munmap** (void \*addr, size\_t length)

## 6.5.1 Detailed Description

Memory mapping declarations.

Copyright

(C) 2013, Samsung Electronics Co., Ltd.

## 6.6 ctype.h File Reference

Critical sections support.

### Functions

- static int `isspace` (int c)  
*Check for white-space characters. These are: space, form-feed ('\f'), newline ('\n'), carriage return ('\r'), horizontal tab ('\t'), and vertical tab ('\v').*
- static int `isascii` (int c)  
*Check whether c is a 7-bit unsigned char value that fits into the ASCII character set.*
- static int `isupper` (int c)  
*Check for an uppercase letter.*
- static int `islower` (int c)  
*Check for a lowercase letter.*
- static int `isalpha` (int c)  
*Check for an alphabetic character; it is equivalent to (isupper(c) || islower(c))*
- static int `isdigit` (int c)  
*Check for a digit (0 through 9)*
- static int `isalnum` (int c)  
*Check for an alphanumeric character; it is equivalent to (isalpha(c) || isdigit(c)).*
- static int `isblank` (int c)  
*Check for a blank character; that is, a space or a tab.*
- static int `isxdigit` (int c)  
*Check for hexadecimal digits, that is, one of 0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F.*
- static int `isprint` (int c)  
*Check for any printable character including space.*
- static int `isgraph` (int c)  
*Check for any printable character except space.*
- static int `ispunct` (int c)  
*Check for any printable character which is not a space or an alphanumeric character.*
- static int `isctrl` (int c)  
*Check for a control character.*
- static int `toupper` (int c)  
*Convert lowercase letter to uppercase.*
- static int `tolower` (int c)  
*Convert uppercase letter to lowercase.*

## 6.6.1 Detailed Description

Critical sections support.

Copyright

Copyright (c) 1982, 1988, 1991, 1993 The Regents of the University of California. All rights reserved.  
(c) UNIX System Laboratories, Inc. Portions copyright (c) 2009-2014, ARM Limited and Contributors.

## 6.7 driver.h File Reference

User-space driver API declarations.

### Data Structures

- struct [fops](#)  
*Structure that contains file operations callbacks supported by the driver. If user wants the driver to carry out some additional action, then handlers should be assigned to the appropriate callbacks. [More...](#)*
- struct [usr\\_drv\\_info](#)  
*Structure that contains information describing the driver. [More...](#)*

### Functions

- int [TEES\\_InitDriver](#) (char \*name, struct [fops](#) \*fops, unsigned int drvid, struct [usr\\_drv\\_info](#) \*\*info)  
*Register user driver within namespace system under name, using mask of file operations fops and drvid of served asset.*
- int [TEES\\_FiniDriver](#) (struct [usr\\_drv\\_info](#) \*info)  
*Allow to release driver that was registered by using struct [usr\\_drv\\_info](#).*
- int [TEES\\_RegisterDriver](#) (char \*name, struct [fops](#) \*fops, unsigned int drvid, struct [usr\\_drv\\_info](#) \*\*info) \_deprecated\_  
*Register user driver within namespace system under name, using mask of file operations fops and drvid of served asset.*
- int [TEES\\_ReleaseDriver](#) (struct [usr\\_drv\\_info](#) \*\*info) \_deprecated\_  
*Allow to release driver that was registered by using struct [usr\\_drv\\_info](#).*
- int [TEES\\_CompleteRequest](#) (struct [drv\\_info](#) \*filp, long ret)  
*Complete deferred request(read, write, etc.) to driver.*
- void \* [TEES\\_AcquireUserBuffer](#) (struct [drv\\_info](#) \*filp, uint64\_t addr, const size\_t size, int prot)  
*Get shared mapped area to use as buffer.*
- int [TEES\\_ReleaseUserBuffer](#) (const void \*addr, const size\_t size)  
*Deletes the shared mapped area for the specified address range.*

### 6.7.1 Detailed Description

User-space driver API declarations.

Copyright

(C) 2014, Samsung Electronics Co., Ltd.

## 6.8 driver/mem/phys.h File Reference

### Macros

- #define `PHYS_DEV_NAME` "phys://"
- #define `PHYS_DEV_NAME_LEN` (sizeof(`PHYS_DEV_NAME`) - 1)
- #define `MAP_PHYS_NON_SECURE` (1 << 29)
- #define `MAP_PHYS_NON_CACHED` (1 << 28)

## 6.9 errno.h File Reference

system error numbers

### Macros

- #define `errno` (`*get_errno_addr()`)

### Typedefs

- typedef int `errno_t`

### Functions

- int \* `get_errno_addr` (void)  
*This function should NOT be used directly, use 'errno' instead.*

### 6.9.1 Detailed Description

system error numbers

Copyright

(C) 2013, Samsung Electronics Co., Ltd.

## 6.10 fcntl.h File Reference

Fcntl declarations.

### Functions

- int `open` (const char \*pathname, int flags,...)  
*Open a device by specifying its namespace path. Device driver operations should be previously registered with the namespace framework.*

## 6.10.1 Detailed Description

Fcntl declarations.

Copyright

(C) 2013, Samsung Electronics Co., Ltd.

## 6.11 inttypes.h File Reference

integer type declarations.

### Macros

- #define [PRIId16](#) \_\_16\_PREFIX "d"
- #define [PRIId32](#) "d"
- #define [PRIId64](#) \_\_64\_PREFIX "d"
- #define [PRIu16](#) \_\_16\_PREFIX "u"
- #define [PRIu32](#) "u"
- #define [PRIu64](#) \_\_64\_PREFIX "u"
- #define [PRIx16](#) \_\_16\_PREFIX "x"
- #define [PRIx32](#) "x"
- #define [PRIx64](#) \_\_64\_PREFIX "x"
- #define [SCNd16](#) \_\_16\_PREFIX "d"
- #define [SCNd32](#) "d"
- #define [SCNd64](#) \_\_64\_PREFIX "d"
- #define [SCNu16](#) \_\_16\_PREFIX "u"
- #define [SCNu32](#) "u"
- #define [SCNu64](#) \_\_64\_PREFIX "u"
- #define [SCNx16](#) \_\_16\_PREFIX "x"
- #define [SCNx32](#) "x"
- #define [SCNx64](#) \_\_64\_PREFIX "x"

### 6.11.1 Detailed Description

integer type declarations.

Copyright

(C) 2013-2016, Samsung Electronics Co., Ltd.

The following macros shall be defined. Each expands to a character string literal containing a conversion specifier, possibly modified by a length modifier, suitable for use within the format argument of a formatted output function when converting the corresponding integer type. These macros have the general form of PRI followed by the conversion specifier (for ex.: d, u, x), followed by a name corresponding to a similar type name in [<stdint.h>](#).

Example:

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

int main(void)
{
 uint32_t var = 0xDEADCODE;

 printf("print in hex: 0x%" PRIx32 "\n", var);
 return 0;
}
```

## 6.12 irs.h File Reference

Integrity Report System.

### Macros

- #define [IRS\\_FAIL\\_TZ](#) -1
- #define [IRS\\_UNKNOWN\\_ID\\_TZ](#) -2
- #define [IRS\\_UNKNOWN\\_INT\\_CMD\\_TZ](#) -3
- #define [IRS\\_INCORRECT\\_FLAG\\_TYPE\\_TZ](#) -4
- #define [IRS\\_RT\\_FLAGS\\_EMPTY\\_TZ](#) -5
- #define [IRS\\_RT\\_FLAGS\\_FULL\\_TZ](#) -6
- #define [IRS\\_INCORRECT\\_RT\\_ID\\_TZ](#) -7
- #define [IRS\\_DENY\\_READ\\_FROM\\_SMC\\_TZ](#) -8
- #define [IRS\\_DENY\\_WRITE\\_FROM\\_SMC\\_TZ](#) -9
- #define [IRS\\_DENY\\_DELETE\\_FROM\\_SMC\\_TZ](#) -10
- #define [IRS\\_MAX\\_VAL\\_COUNTER\\_TZ](#) -11
- #define [IRS\\_MAX\\_VAL\\_COUNTER\\_RT\\_TZ](#) -12
- #define [IRS\\_INCORRECT\\_CHECKSUM\\_TZ](#) -13
- #define [IRS\\_UNKNOWN\\_ERROR\\_TZ](#) -14
- #define [IRS\\_SUCCESS\\_TZ](#) 0

### Enumerations

- enum [IRS\\_INTERNAL\\_CMD](#) {  
[IRS\\_SET\\_FLAG\\_CMD](#) = 1, [IRS\\_SET\\_FLAG\\_VALUE\\_CMD](#), [IRS\\_INC\\_FLAG\\_CMD](#), [IRS\\_GET\\_FLAG\\_VALUE\\_CMD](#),  
[IRS\\_ADD\\_FLAG\\_CMD](#), [IRS\\_DEL\\_FLAG\\_CMD](#) }  
*Internal IRS commands ids.*
- enum [IRS\\_PARAM](#) {  
[IRS\\_TYPE\\_BOOLEAN](#) = 0x00000001, [IRS\\_TYPE\\_VALUE](#) = 0x00000002, [IRS\\_TYPE\\_COUNTER](#) =  
0x00000004, [IRS\\_NWD\\_RD](#) = 0x00000008,  
[IRS\\_NWD\\_WR](#) = 0x00000010, [IRS\\_NWD\\_CTRL](#) = 0x00000020, [IRS\\_SWD\\_RD](#) = 0x00000040,  
[IRS\\_SWD\\_WR](#) = 0x00000080,  
[IRS\\_SWD\\_CTRL](#) = 0x00000100 }  
*Bit position of params.*

### Functions

- int [TEES\\_SetIrsFlag](#) (unsigned int \*flag\_id)  
*Set flag by flag\_id in 1. Used only for boolean flag type.*
- int [TEES\\_SetIrsFlagValue](#) (unsigned int \*flag\_id, unsigned int value)  
*Set flag by flag\_id in value by value.*
- int [TEES\\_IncIrsFlag](#) (unsigned int \*flag\_id)  
*Increment flag by flag\_id in 1. Used only for IRS\_TYPE\_VALUE flag type.*
- int [TEES\\_GetIrsFlagValue](#) (unsigned int \*flag\_id, unsigned int \*value)  
*Get value by flag\_id.*
- int [TEES\\_AddIrsFlag](#) (unsigned int \*flag\_id, unsigned int param)  
*Function for control run-time flags.*
- int [TEES\\_DelIrsFlag](#) (unsigned int \*flag\_id)  
*Delete run-time flag by flag\_id.*

### 6.12.1 Detailed Description

Integrity Report System.

Copyright

(C) 2016, Samsung Electronics Co., Ltd.

## 6.13 limits.h File Reference

Implementation-defined constants.

### Macros

- #define CHAR\_BIT 8
- #define SCHAR\_MAX (127)
- #define SCHAR\_MIN (-128)
- #define UCHAR\_MAX (255)
- #define USHRT\_MAX (0xFFFFU)
- #define SHRT\_MAX (32767)
- #define SHRT\_MIN (-32768)
- #define INT\_MAX ((int)(~0U>>1))
- #define INT\_MIN (-INT\_MAX - 1)
- #define LONG\_MAX ((long)(~0UL>>1))
- #define LONG\_MIN (-LONG\_MAX - 1)
- #define UINT\_MAX (~0U)
- #define ULONG\_MAX (~0UL)
- #define ULLONG\_MAX (~0ULL)
- #define LLONG\_MAX ((long long)(~0ULL>>1))
- #define LLONG\_MIN ((long long)(-LLONG\_MAX - 1))
- #define \_POSIX\_THREAD\_KEYS\_MAX 12
- #define PTHREAD\_KEYS\_MAX \_POSIX\_THREAD\_KEYS\_MAX

### 6.13.1 Detailed Description

Implementation-defined constants.

Copyright

(C) 2013-2016, Samsung Electronics Co., Ltd.

Header shall define macros and symbolic constants for various limits. Different categories of limits are described below, representing various limits on resources that the implementation imposes on applications. All macros and symbolic constants defined in this header shall be suitable for use in #if preprocessing directives.

## 6.14 malloc.h File Reference

Memory allocation definitions.

### Macros

- #define [M\\_CACHE\\_PAGES](#) 1

### 6.14.1 Detailed Description

Memory allocation definitions.

Copyright

(C) 2013-2016, Samsung Electronics Co., Ltd.

## 6.15 math.h File Reference

Math library.

### Macros

- #define [isnan\(x\)](#) [\\_isnan\(x\)](#)
- #define [M\\_E](#) 2.7182818284590452354
- #define [M\\_PI](#) 3.14159265358979323846
- #define [M\\_PI\\_2](#) 1.57079632679489661923
- #define [M\\_PI\\_4](#) 0.78539816339744830962

### Functions

- `_const_ double` [atan](#) (double x)  
*Arc tangent function.*
- `_const_ float` [atanf](#) (float x)  
*Arc tangent function.*
- `_const_ double` [atan2](#) (double y, double x)  
*Arc tangent function of two variables.*
- `_const_ float` [atan2f](#) (float y, float x)  
*Arc tangent function of two variables.*
- `_const_ double` [ceil](#) (double x)  
*ceiling function: smallest integral value not less than argument*
- `_const_ float` [ceilf](#) (float x)  
*ceiling function: smallest integral value not less than argument*
- `_const_ double` [pow](#) (double base, double `exp`)  
*Calculate the exponentiation.*
- `_const_ float` [powf](#) (float base, float `exp`)

- Calculate the exponentiation.*

  - `_const_ double sqrt` (double x)

*Calculate the square root of x.*
- `_const_ float sqrtf` (float x)

*Calculate the square root of x.*
- `_const_ double fabs` (double x)

*Calculate the absolute value of x.*
- `_const_ float fabsf` (float x)

*Calculate the absolute value of x.*
- `_const_ double round` (double x)

*Calculate the integral value that is the nearest to x.*
- `_const_ float roundf` (float x)

*Calculate the integral value that is the nearest to x.*
- `_const_ double sin` (double x)

*Calculate the sine of an angle of x radians.*
- `_const_ float sinf` (float x)

*Calculate the sine of an angle of x radians.*
- `_const_ double cos` (double x)

*Calculate the cosine of an angle of x radians.*
- `_const_ float cosf` (float x)

*Calculate the cosine of an angle of x radians.*
- `_const_ double exp` (double x)

*base-e exponential function*
- `_const_ float expf` (float x)

*base-e exponential function*
- `_const_ double log` (double x)

*Calculate the natural logarithm.*
- `_const_ float logf` (float x)

*Calculate the natural logarithm.*
- `_const_ double logb` (double x)

*Calculate exponent of a floating-point value.*
- `_const_ float logbf` (float x)

*Calculate exponent of a floating-point value.*
- `_const_ double fmax` (double x, double y)

*Determine maximum of two floating-point numbers.*
- `_const_ float fmaxf` (float x, float y)

*Determine maximum of two floating-point numbers.*
- `_const_ double fmin` (double x, double y)

*Determine minimum of two floating-point numbers.*
- `_const_ float fminf` (float x, float y)

*Determine minimum of two floating-point numbers.*
- `_const_ double scalbn` (double x, int exp)

*Multiply floating-point number by integral power of radix.*
- `_const_ float scalbnf` (float x, int exp)

*Multiply floating-point number by integral power of radix.*
- `_const_ double copysign` (double x, double y)

*Copy sign of a number.*
- `_const_ float copysignf` (float x, float y)

*Copy sign of a number.*
- `_const_ double floor` (double x)

*Get largest integral value not greater than argument.*

- `_const_ float floorf` (float x)  
*Get largest integral value not greater than argument.*
- `_const_ double hypot` (double x, double y)  
*Euclidean distance function.*
- `_const_ float hypotf` (float x, float y)  
*Euclidean distance function.*
- `double modf` (double x, double \*iptr)  
*extract signed integral and fractional values from floating-point number*
- `float modff` (float x, float \*iptr)  
*extract signed integral and fractional values from floating-point number*

### 6.15.1 Detailed Description

Math library.

Copyright

(C) 2015 Samsung Electronics Co., Ltd.

## 6.16 mqueue.h File Reference

POSIX message queue declarations.

### Macros

- `#define MQ_MAX_NAME` 1024

### Typedefs

- `typedef int mqd_t`

### Functions

- `mqd_t mq_open` (const char \*pathname, int flags,...)  
*Create new message queue or open an existing queue.*
- `int mq_unlink` (const char \*pathname)  
*Remove specified message queue name.*
- `int mq_close` (mqd\_t fd)  
*Close a message queue descriptor.*
- `int mq_send` (mqd\_t fd, const char \*msg\_ptr, size\_t msg\_len, unsigned msg\_prio)  
*Send a message to a message queue.*
- `ssize_t mq_receive` (mqd\_t fd, char \*msg\_ptr, size\_t msg\_len, unsigned \*msg\_prio)  
*Receive a message from a message queue.*

### 6.16.1 Detailed Description

POSIX message queue declarations.

Copyright

(C) 2013 - 2018, Samsung Electronics Co., Ltd.

## 6.17 print\_no\_alloc.h File Reference

### Macros

- #define [AUTO\\_BUFFER\\_SIZE](#) 1024

### Functions

- int [printf\\_no\\_alloc](#) (const char \*fmt,...)  
*Prints to ringbuffer. If resulting string exceeds AUTO\_BUFFER\_SIZE, cuts it off to AUTO\_BUFFER\_SIZE.*

### 6.17.1 Detailed Description

Copyright

(C) 2013, Samsung Electronics Co., Ltd.

## 6.18 protocol\_errors.h File Reference

iSocket Protocol error extended codes definitions

### Enumerations

- enum [protocol\\_error\\_code](#) {  
**NO\_ERROR** = 0, **TEE\_ISOCKET\_IWC\_ERROR\_CHANNEL** = 0x81000000, **TEE\_ISOCKET\_IWC\_ERROR\_TIMEOUT** = 0x81000001, **TEE\_ISOCKET\_IWC\_ERROR\_NOT\_IMPLEMENTED** = 0x81000002, **TEE\_ISOCKET\_IWC\_ERROR\_INVALID\_VERSION** = 0x81000003, **TEE\_ISOCKET\_IWC\_ERROR\_SWD\_CLIENT\_AUTH\_FAILURE** = 0x81000004, **TEE\_ISOCKET\_NET\_ERROR\_GENERIC** = 0x81010000, **TEE\_ISOCKET\_NET\_ERROR\_BAD\_PARAMETER** = 0x81010001, **TEE\_ISOCKET\_NET\_ERROR\_BUFFER\_TOO\_SMALL** = 0x81010002, **TEE\_ISOCKET\_NET\_ERROR\_LARGE\_BUFFER** = 0x81010003, **TEE\_ISOCKET\_NET\_ERROR\_OUT\_OF\_RESOURCES** = 0x81010004, **TEE\_ISOCKET\_NET\_ERROR\_OUT\_OF\_MEMORY** = 0x81010005, **TEE\_ISOCKET\_NET\_ERROR\_HOSTNAME\_UNKNOWN** = 0x81010006, **TEE\_ISOCKET\_NET\_ERROR\_HOSTNAME\_NOT\_FOUND** = 0x81010007, **TEE\_ISOCKET\_NET\_ERROR\_HOSTNAME\_TRYAGAIN** = 0x81010008, **TEE\_ISOCKET\_NET\_ERROR\_CONNECTION\_REFUSED** = 0x8101000A, **TEE\_ISOCKET\_NET\_ERROR\_NET\_UNREACHABLE** = 0x8101000B, **TEE\_ISOCKET\_NET\_ERROR\_REMOTE\_CLOSED** = 0x8101000C, **TEE\_ISOCKET\_NET\_ERROR\_TIMEOUT** = 0x8101000D, ...  
}

```

= 0x8101000D,
TEE_ISOCKET_NET_ERROR_DATA_REMAIN = 0x8101000E, TEE_ISOCKET_TLS_ERROR_CERT_PARSING
= 0x80000000, TEE_ISOCKET_TLS_ERROR_CRL_PARSING = 0x80000001, TEE_ISOCKET_TLS_ERROR_CERT_EXPIR
= 0x80000002,
TEE_ISOCKET_TLS_ERROR_CERT_SIGN_VERIFICATION = 0x80000003, TEE_ISOCKET_TLS_ERROR_ECDHE_GEN_K
= 0x81030010, TEE_ISOCKET_TLS_ERROR_ECDHE_SHARED_SECRET = 0x81030011, TEE_ISOCKET_TLS_ERROR_EC
= 0x81030012,
TEE_ISOCKET_TLS_ERROR_ECDHE_SERIALIZING = 0x81030013, TEE_ISOCKET_TLS_ERROR_CERT_COMMON_NAM
= 0x81030014, TEE_ISOCKET_TLS_ERROR_UNEXPECTED_MESSAGE = 0x81030015, TEE_ISOCKET_TLS_ERROR_HA
= 0x81030016,
TEE_ISOCKET_TLS_ERROR_CERT_IS_TOO_LONG = 0x81030017, TEE_ISOCKET_TLS_ERROR_NO_ALERT_PRESENT
= 0x81030018, TEE_ISOCKET_TLS_ERROR_ALERT_PENDING = 0x81030019, TEE_ISOCKET_TLS_ERROR_USER_CAN
= 0x8103001A,
TEE_ISOCKET_TLS_ERROR_CERT_UNKNOWN_CA = 0x8103001B, TEE_ISOCKET_TLS_ERROR_CERT_UNSUPPORTED
= 0x8103001C, TEE_ISOCKET_TLS_ERROR_CERT_REVOKED = 0x8103001D, TEE_ISOCKET_TLS_ERROR_CERT_STAT
= 0x8103001E,
TEE_ISOCKET_TLS_ALERT_CLOSE_NOTIFY = 0x81031000, TEE_ISOCKET_TLS_ALERT_UNEXPECTED_MSG
= 0x81031010, TEE_ISOCKET_TLS_ALERT_BAD_RECORD_MAC = 0x81031020, TEE_ISOCKET_TLS_ALERT_DECRYPT
= 0x81031021,
TEE_ISOCKET_TLS_ALERT_RECORD_OVERFLOW = 0x81031022, TEE_ISOCKET_TLS_ALERT_DECOMP_FAILED
= 0x81031030, TEE_ISOCKET_TLS_ALERT_HANDSHAKE_FAILED = 0x81031040, TEE_ISOCKET_TLS_ALERT_NO_CE
= 0x81031041,
TEE_ISOCKET_TLS_ALERT_BAD_CERTIFICATE = 0x81031042, TEE_ISOCKET_TLS_ALERT_UNSUPPORTED_CERT
= 0x81031043, TEE_ISOCKET_TLS_ALERT_CERT_REVOKED = 0x81031044, TEE_ISOCKET_TLS_ALERT_CERT_EXPIR
= 0x81031045,
TEE_ISOCKET_TLS_ALERT_CERT_UNKNOWN = 0x81031046, TEE_ISOCKET_TLS_ALERT_ILLEGAL_PARAMETER
= 0x81031047, TEE_ISOCKET_TLS_ALERT_UNKNOWN_CA = 0x81031048, TEE_ISOCKET_TLS_ALERT_ACCESS_DEN
= 0x81031049,
TEE_ISOCKET_TLS_ALERT_DECODE_ERROR = 0x81031050, TEE_ISOCKET_TLS_ALERT_DECRYPT_ERROR
= 0x81031051, TEE_ISOCKET_TLS_ALERT_EXPORT_RESTRICTED = 0x81031060, TEE_ISOCKET_TLS_ALERT_PROTC
= 0x81031070,
TEE_ISOCKET_TLS_ALERT_INSUFFICIENT_SECURITY = 0x81031071, TEE_ISOCKET_TLS_ALERT_INTERNAL_ERROR
= 0x81031080, TEE_ISOCKET_TLS_ALERT_INAPPROPRIATE_FALLBACK = 0x81031086, TEE_ISOCKET_TLS_ALERT_
= 0x81031090,
TEE_ISOCKET_TLS_ALERT_NO_RENEGOTIATION = 0x81031100, TEE_ISOCKET_TLS_ALERT_MISSING_EXTENSION
= 0x81031109, TEE_ISOCKET_TLS_ALERT_UNSUPPORTED_EXTENSION = 0x81031110, TEE_ISOCKET_TLS_ALERT_C
= 0x81031111,
TEE_ISOCKET_TLS_ALERT_UNRECOGNIZED_NAME = 0x81031112, TEE_ISOCKET_TLS_ALERT_BAD_CERT_STATUS
= 0x81031113, TEE_ISOCKET_TLS_ALERT_BAD_CERT_HASH_VALUE = 0x81031114, TEE_ISOCKET_TLS_ALERT_UNK
= 0x81031115,
TEE_ISOCKET_TLS_ALERT_CERT_REQUIRED = 0x81031116 }

```

*Propriate protocol specific error codes. According to GPD\_SPE\_010 specification, TEE error code range 0x80000000..0x8FFFFFFF is reserved for implementation specific error. In addition, TEE Socket Subsystem considers protocol errors as specification extension and includes specification ID into code: 0x8 | 3 digit BCD spec ID | error code.*

## 6.18.1 Detailed Description

iSocket Protocol error extended codes definitions

Copyright

(C) 2016-2017, Samsung Electronics Co., Ltd.

## 6.19 pthread.h File Reference

POSIX compatible thread library.

### Data Structures

- struct [\\_\\_pthread\\_once\\_t](#)
- struct [\\_\\_pthread\\_attr\\_t](#)
- struct [\\_\\_pthread\\_mutex\\_t](#)
- struct [\\_\\_pthread\\_cond\\_t](#)
- struct [\\_\\_pthread\\_condattr\\_t](#)

### Macros

- #define [PTHREAD\\_STACK\\_MIN](#) (PAGE\_SIZE)
- #define [MUTEX\\_STATE\\_UNLOCKED](#) 0
- #define [MUTEX\\_STATE\\_LOCKED](#) 1
- #define [PTHREAD\\_MUTEX\\_INITIALIZER](#) {{ [MUTEX\\_STATE\\_UNLOCKED](#) }, [PTHREAD\\_MUTEX\\_DEFAULT](#), 0, 0 }
- #define [PTHREAD\\_ERRORCHECK\\_MUTEX\\_INITIALIZER\\_NP](#) { { [MUTEX\\_STATE\\_UNLOCKED](#) }, [PTHREAD\\_MUTEX\\_ERRORCHECK](#), 0, 0 }
- #define [PTHREAD\\_RECURSIVE\\_MUTEX\\_INITIALIZER\\_NP](#) { { [MUTEX\\_STATE\\_UNLOCKED](#) }, [PTHREAD\\_MUTEX\\_RECURSIVE](#), 0, 0 }
- #define [PTHREAD\\_ONCE\\_INIT](#) { [ATOMIC\\_INITIALIZER](#) }
- #define [PTHREAD\\_COND\\_INITIALIZER](#) { [ATOMIC\\_INITIALIZER](#) }
- *Object for static initialization of [pthread\\_cond\\_t](#) variables.*
- #define [pthread\\_sigmask](#)(how, set, oldset) sigprocmask(how, set, oldset)  
*Set signal mask for specified thread.*

### Typedefs

- typedef struct pthread\_impl [pthread\\_impl\\_t](#)
- typedef uintptr\_t [pthread\\_t](#)
- typedef uint32\_t [pthread\\_mutexattr\\_t](#)
- typedef struct [\\_\\_pthread\\_mutex\\_t](#) [pthread\\_mutex\\_t](#)
- typedef unsigned [pthread\\_key\\_t](#)
- typedef struct [\\_\\_pthread\\_attr\\_t](#) [pthread\\_attr\\_t](#)
- typedef struct [\\_\\_pthread\\_once\\_t](#) [pthread\\_once\\_t](#)
- typedef struct [\\_\\_pthread\\_cond\\_t](#) [pthread\\_cond\\_t](#)
- typedef struct [\\_\\_pthread\\_cond\\_t](#) [pthread\\_condattr\\_t](#)

### Enumerations

- enum {  
[PTHREAD\\_MUTEX\\_NORMAL](#) = 0, [PTHREAD\\_MUTEX\\_RECURSIVE](#), [PTHREAD\\_MUTEX\\_ERRORCHECK](#),  
[PTHREAD\\_MUTEX\\_DEFAULT](#) = [PTHREAD\\_MUTEX\\_NORMAL](#),  
[PTHREAD\\_MUTEX\\_DESTROYED](#) = -1, [PTHREAD\\_MUTEX\\_ERRORCHECK\\_NP](#) = [PTHREAD\\_MUTEX\\_ERRORCHECK](#),  
[PTHREAD\\_MUTEX\\_RECURSIVE\\_NP](#) = [PTHREAD\\_MUTEX\\_RECURSIVE](#) }  
*types and states for mutex*

## Functions

- int [pthread\\_attr\\_init](#) ([pthread\\_attr\\_t](#) \*attr)  
The [pthread\\_attr\\_init\(\)](#) function initialize attribute struct.
- int [pthread\\_attr\\_destroy](#) ([pthread\\_attr\\_t](#) \*attr)  
The [pthread\\_attr\\_destroy\(\)](#) function destroy attribute struct.
- int [pthread\\_attr\\_setstacksize](#) ([pthread\\_attr\\_t](#) \*attr, [size\\_t](#) size)  
The [pthread\\_attr\\_setstacksize\(\)](#) function sets size of stack.
- int [pthread\\_create](#) ([pthread\\_t](#) \*thread, const [pthread\\_attr\\_t](#) \*attr, void \*(\*start\_routine)(void \*), void \*arg)  
The [pthread\\_create\(\)](#) function starts a new thread in the calling process. The new thread starts execution by invoking [start\\_routine\(\)](#) arg is passed as the sole argument of [start\\_routine\(\)](#).
- int [pthread\\_join](#) ([pthread\\_t](#) thread, void \*\*retval)  
Wait for the thread specified by *thread* to terminate. If that thread has already terminated, then returns immediately. The thread specified by *thread* must be joinable.
- int [pthread\\_once](#) ([pthread\\_once\\_t](#) \*once\_control, void(\*init\_routine)(void))  
If any thread in a process with a *once\_control* parameter makes a call to [pthread\\_once\(\)](#), the first call will summon the [init\\_routine\(\)](#), but subsequent calls will not. The *once\_control* parameter determines whether the associated initialization routine has been called. The [init\\_routine\(\)](#) is complete upon return of [pthread\\_once\(\)](#).
- void \* [pthread\\_getspecific](#) ([pthread\\_key\\_t](#) key)  
Return the value currently bound to the specified *key* on behalf of the calling thread.
- int [pthread\\_setspecific](#) ([pthread\\_key\\_t](#) key, const void \*value)  
Associate a thread-specific *value* with a *key* obtained via a previous call to [pthread\\_key\\_create\(\)](#).
- int [pthread\\_key\\_create](#) ([pthread\\_key\\_t](#) \*key, void(\*destructor)(void \*))  
Create data key for data manipulation functions ([pthread\\_getspecific\(\)](#), [pthread\\_setspecific\(\)](#)). Multiple threads can call data manipulation functions with the same key. In this case all threads will have separate data.
- int [pthread\\_key\\_delete](#) ([pthread\\_key\\_t](#) key)  
Delete data key and destructor associated with *key*. After key deletion there is no destructor will be called on thread exits.
- int [pthread\\_mutexattr\\_init](#) ([pthread\\_mutexattr\\_t](#) \*attr)  
Initialize mutex attributes object and initialize attributes with default values.
- int [pthread\\_mutexattr\\_destroy](#) ([pthread\\_mutexattr\\_t](#) \*attr)  
Destroy attributes object and make all attribute values are uninitialized.
- int [pthread\\_mutexattr\\_gettype](#) (const [pthread\\_mutexattr\\_t](#) \*attr, int \*type)  
Get mutex type attribute associated with *attr* parameter.
- int [pthread\\_mutexattr\\_settype](#) ([pthread\\_mutexattr\\_t](#) \*attr, int type)  
Set mutex type attribute associated with *attr* parameter.
- int [pthread\\_mutex\\_lock](#) ([pthread\\_mutex\\_t](#) \*mutex)  
Lock mutex or wait while another thread is unlock currently locked mutex.
- int [pthread\\_mutex\\_trylock](#) ([pthread\\_mutex\\_t](#) \*mutex)  
Lock mutex or fail if mutex is already locked.
- int [pthread\\_mutex\\_unlock](#) ([pthread\\_mutex\\_t](#) \*mutex)  
Release lock on currently locked mutex.
- int [pthread\\_mutex\\_destroy](#) ([pthread\\_mutex\\_t](#) \*mutex)  
Destroy mutex object and make all associated data are uninitialized.
- int [pthread\\_mutex\\_init](#) ([pthread\\_mutex\\_t](#) \*mutex, const [pthread\\_mutexattr\\_t](#) \*attr)  
Initialize mutex object *mutex* with attributes given by *attr* parameter. If *attr* parameter is NULL then default attributes will be used.
- int [pthread\\_cond\\_destroy](#) ([pthread\\_cond\\_t](#) \*cond)  
Destroy conditional variable object and make it uninitialized.
- int [pthread\\_cond\\_init](#) ([pthread\\_cond\\_t](#) \*cond, const [pthread\\_condattr\\_t](#) \*attr)

- Initialize conditional variable object `cond` with attributes given by `attr` parameter.*
- int [pthread\\_cond\\_signal](#) ([pthread\\_cond\\_t](#) \*cond)  
*Unblock at least one of the threads that are blocked on the specified condition variable `cond` (if any threads are blocked on `cond`).*
  - int [pthread\\_cond\\_broadcast](#) ([pthread\\_cond\\_t](#) \*cond)  
*Wake up all threads locked by conditional variable `cond`.*
  - int [pthread\\_cond\\_wait](#) ([pthread\\_cond\\_t](#) \*cond, [pthread\\_mutex\\_t](#) \*mutex)  
*Block on condition variable `cond`, while another thread will unblock this variable by [pthread\\_cond\\_broadcast\(\)](#) / [pthread\\_cond\\_signal\(\)](#) call.*
  - [pthread\\_t](#) [pthread\\_self](#) (void)  
*Obtain ID of the calling thread.*
  - int [pthread\\_kill](#) ([pthread\\_t](#) thread, int sig)  
*Send signal to specified thread.*

### 6.19.1 Detailed Description

POSIX compatible thread library.

Copyright

(C) 2015, Samsung Electronics Co., Ltd.

## 6.20 rpmb.h File Reference

RPMB Read/Write API.

### Functions

- TEE\_Result [TEES\\_RPMBRead](#) (uint32\_t partition, uint32\_t offset, uint8\_t \*data, uint32\_t data\_size, uint8\_t type\_flag)  
*Read data from RPMB Storage.*
- TEE\_Result [TEES\\_RPMBWrite](#) (uint32\_t partition, uint32\_t offset, uint8\_t \*data, uint32\_t data\_size, uint8\_t type\_flag)  
*Write data to RPMB Storage.*
- TEE\_Result [TEES\\_RPMBCheckEnable](#) (void)  
*Check RPMB availability.*

### 6.20.1 Detailed Description

RPMB Read/Write API.

Copyright

(C) 2018, Samsung Electronics Co., Ltd.

## 6.21 sched.h File Reference

Scheduler syscalls.

### Data Structures

- struct [cpu\\_set\\_t](#)

### Macros

- #define [BIT\\_PER\\_CPU](#) (1)
- #define [MAX\\_CPUS](#) (32)
- #define [BITS\\_TO\\_CPU\\_MASK](#)(bits) (((bits) + BITS\_PER\_LONG - 1) / BITS\_PER\_LONG)
- #define [BITMAP\\_ELT](#)(cpu) ((cpu) / BITS\_PER\_LONG)
- #define [\\_\\_CPUMASK](#)(cpu) (1L << ((cpu) % BITS\_PER\_LONG))
- #define [DECLARE\\_BITMAP](#)(name, bits) unsigned long name[[BITS\\_TO\\_CPU\\_MASK](#)(bits)]
- #define [CPU\\_ZERO](#)(cpusetp)
- #define [CPU\\_SET](#)(cpu, cpusetp)
- #define [CPU\\_CLR](#)(cpu, cpusetp)
- #define [CPU\\_ISSET](#)(cpu, cpusetp)

### Functions

- int [sched\\_yield](#) (void)  
*Causes the calling thread to relinquish the CPU. The thread is moved to the end of the queue for its static priority and a new thread gets to run.*
- int [sched\\_setaffinity](#) ([pid\\_t](#) pid, [size\\_t](#) cpusetsize, [cpu\\_set\\_t](#) \*mask)  
*Sets the CPU affinity mask of the process whose ID is pid to the value specified by mask. If pid is zero, then the calling process is used.*
- int [sched\\_getaffinity](#) ([pid\\_t](#) pid, [size\\_t](#) cpusetsize, [cpu\\_set\\_t](#) \*mask)  
*Writes the affinity mask of the process whose ID is pid into the [cpu\\_set\\_t](#) structure pointed to by mask.*

#### 6.21.1 Detailed Description

Scheduler syscalls.

Copyright

(C) 2014-2018, Samsung Electronics Co., Ltd.

## 6.22 scma.h File Reference

Secure Contiguous memory allocator API.

## Typedefs

- typedef struct \_\_TEES\_ContiguousMemoryHandle \* [TEES\\_ContiguousMemoryHandle](#)

## Enumerations

- enum [TEES\\_ContiguousAllocateFlags](#) { [TEES\\_CONTIGUOUS\\_FLAG\\_ZERO\\_ON\\_FREE](#) = (1U << 0), [TEES\\_CONTIGUOUS\\_FLAG\\_ZERO\\_ON\\_ALLOC](#) = (1U << 1) }  
*Determines the desired allocated memory handle properties.*
- enum [TEES\\_ContiguousMapFlags](#) { [TEES\\_CONTIGUOUS\\_MAP\\_READ](#) = (1U << 0), [TEES\\_CONTIGUOUS\\_MAP\\_WRITE](#) = (1U << 1), [TEES\\_CONTIGUOUS\\_MAP\\_NON\\_CACHEABLE](#) = (1U << 2), [TEES\\_CONTIGUOUS\\_MAP\\_FIXED](#) = (1U << 3), [TEES\\_CONTIGUOUS\\_MAP\\_POPULATE](#) = (1U << 4) }  
*Determines the desired mapping properties.*

## Functions

- TEE\_Result [TEES\\_AllocateContiguousMemory](#) (const char \*region, size\_t size, size\_t align, uint32\_t flags, [TEES\\_ContiguousMemoryHandle](#) \*memory)  
*Allocates physically contiguous memory buffer handle of the specified region type.*
- void [TEES\\_ReleaseContiguousMemory](#) ([TEES\\_ContiguousMemoryHandle](#) memory)  
*Releases physically contiguous memory buffer that was previously allocated by [TEES\\_AllocateContiguousMemory](#).*
- TEE\_Result [TEES\\_MapContiguousMemory](#) ([TEES\\_ContiguousMemoryHandle](#) memory, void \*\*addr, uint32\_t flags)  
*Maps physically contiguous memory buffer previously allocated by [TEES\\_AllocateContiguousMemory](#) into the address space of current process.*
- void [TEES\\_UnmapContiguousMemory](#) (void \*addr)  
*Unmaps physically contiguous memory buffer that was previously mapped by [TEES\\_MapContiguousMemory](#).*

### 6.22.1 Detailed Description

Secure Contiguous memory allocator API.

Copyright

(C) 2017, Samsung Electronics Co., Ltd.

## 6.23 stdio.h File Reference

Standard I/O.

### Macros

- #define [EOF](#) (-1)

## Functions

- int **printf** (const char \*fmt,...)  
*Format and print string.*
- int **fprintf** (FILE \*\_restrict\_ stream, const char \*\_restrict\_ fmt,...)  
*The `fprintf()` is equivalent to the `printf()`, but uses a custom file handle.*
- int **vfprintf** (FILE \*\_restrict\_ stream, const char \*\_restrict\_ fmt, va\_list ap)  
*The `vfprintf()` is equivalent to the `fprintf()`, but uses an argument list.*
- int **fflush** (FILE \*stream)  
*Flush a stream.*
- int **sprintf** (char \*s, const char \*fmt,...)  
*format and string and save it to 's'.*
- int **printf\_s** (const char \*\_restrict\_ fmt,...)  
*format and print string.*
- int **vsprintf\_s** (char \*\_restrict\_ s, rsize\_t n, const char \*\_restrict\_ format, va\_list arg)  
*Write formatted data from variable length argument list to string.*
- int **vsnprintf\_s** (char \*\_restrict\_ s, rsize\_t n, const char \*\_restrict\_ format, va\_list arg)  
*Write formatted data from variable argument list to sized buffer.*
- int **sprintf\_s** (char \*\_restrict\_ s, rsize\_t n, const char \*\_restrict\_ format,...)  
*format string and save it to 's'.*
- int **snprintf\_s** (char \*\_restrict\_ s, rsize\_t n, const char \*\_restrict\_ format,...)  
*Format string and save it to 's'.*
- int **sscanf\_s** (const char \*\_restrict\_ s, const char \*\_restrict\_ format,...)  
*Reads data safely from buf and stores them according to parameter fmt into the locations given by the additional arguments.*
- int **vsscanf\_s** (const char \*\_restrict\_ s, const char \*\_restrict\_ format, va\_list arg)  
*vsscanf unformat a buffer into a list of arguments.*
- int **snprintf** (char \*s, size\_t count, const char \*fmt,...)  
*format and string and save it to 's'.*
- int **vsprintf** (char \*buffer, const char \*format, va\_list args)  
*Write formatted data from variable argument list to string.*
- int **vsnprintf** (char \*buffer, size\_t size, const char \*format, va\_list args)  
*Write formatted data from variable argument list to sized buffer.*
- int **sscanf** (const char \*buf, const char \*fmt,...)  
*Reads data from buf and stores them according to parameter fmt into the locations given by the additional arguments.*
- int **asprintf** (char \*\*strp, const char \*fmt,...)  
*print to allocated string.*
- int **vasprintf** (char \*\*strp, const char \*fmt, va\_list args)  
*print to allocated string.*
- int **vasprintf\_s** (char \*\*strp, const char \*fmt, va\_list args)  
*Print to allocated string in secure mode.*
- int **putchar** (int ch)  
*writes a character to log.*
- int **puts** (const char \*s)  
*writes the string s and a trailing newline to log(dmesg).*
- int **vsscanf** (const char \*buffer, const char \*fmt, va\_list args)  
*vsscanf unformat a buffer into a list of arguments.*

## Variables

- FILE \* [stdin](#)  
*Standard input stream (stub).*
- FILE \* [stdout](#)  
*Standard output stream.*
- FILE \* [stderr](#)  
*Standard error stream.*

### 6.23.1 Detailed Description

Standard I/O.

Copyright

(C) 2013 - 2018 Samsung Electronics Co., Ltd.

## 6.24 stdlib.h File Reference

Standard library header.

### Typedefs

- typedef void(\* [constraint\\_handler\\_t](#)) (const char \*restrict msg, void \*restrict ptr, [errno\\_t](#) error)

### Functions

- [constraint\\_handler\\_t set\\_constraint\\_handler\\_s](#) ([constraint\\_handler\\_t](#) handler)  
*Set the handler to be handler.*
- void [invoke\\_constraint\\_handler\\_s](#) (const char \*msg, const char \*file, const char \*function, uint32\_t line, [errno\\_t](#) error)  
*Print msg if constraint was caused.*
- void [abort\\_handler\\_s](#) (const char \*restrict msg, void \*restrict ptr, [errno\\_t](#) error)  
*Abort system if constraint was caused.*
- void [ignore\\_handler\\_s](#) (const char \*restrict msg, void \*restrict ptr, [errno\\_t](#) error)  
*Returns to the caller without performing any actions.*
- void [exit](#) (int status)  
*Cause normal process termination and return the value of status & 0377 to the parent.*
- void [\\_exit](#) (int status)  
*Terminate the calling process "immediately". Any open file descriptors belonging to the process are closed; process's parent is sent a SIGCHLD signal.*
- static \_\_inline\_\_ int [abs](#) (int j)  
*Compute the absolute value of the integer argument `__n`.*
- void [abort](#) (void)  
*Cause abnormal process termination.*
- long [strtol](#) (const char \*nptr, char \*\*endptr, int base)

- Convert the initial part of the string in *nptr* to a long integer value according to the given *base*, which must be between 2 and 36 inclusive, or be the special value 0.
- unsigned long **strtoul** (const char \*cp, char \*\*endp, int base)
 

Convert the initial part of the string in *nptr* to a unsigned long integer value according to the given *base*, which must be between 2 and 36 inclusive, or be the special value 0.
  - double **strtod** (const char \*nptr, char \*\*endptr)
 

the initial portion of the string pointed to by *nptr* to double.
  - long long **strtoll** (const char \*nptr, char \*\*endptr, int base)
 

Convert the initial part of the string in *nptr* to a long long integer value according to the given *base*, which must be between 2 and 36 inclusive, or be the special value 0.
  - unsigned long long **strtoull** (const char \*cp, char \*\*endp, int base)
 

Convert the initial part of the string in *nptr* to a unsigned long long integer value according to the given *base*, which must be between 2 and 36 inclusive, or be the special value 0.
  - float **strtof** (const char \*nptr, char \*\*endptr)
 

the initial portion of the string pointed to by *nptr* to float.
  - long double **strtold** (const char \*nptr, char \*\*endptr)
 

the initial portion of the string pointed to by *nptr* to long double.
  - int **atexit** (void(\*func)(void))
 

Register the given function to be called at normal process termination.
  - void \* **malloc** (size\_t size)
 

Allocate *size* bytes and return a pointer to the allocated memory.
  - void **free** (void \*ptr)
 

Free the memory space pointer to by *ptr*.
  - void \* **calloc** (size\_t nmemb, size\_t size)
 

Allocate memory for an array of *nmemb* elements of *size* bytes each and return a pointer to the allocated memory.
  - void \* **realloc** (void \*ptr, size\_t size)
 

Change the size of the memory block pointed to by *ptr* to *size* bytes.
  - void **qsort** (void \*base, size\_t nmemb, size\_t size, int(\*compar)(const void \*, const void \*))
 

Sort an array.
  - void **qsort\_r** (void \*base, size\_t nmemb, size\_t size, int(\*compar)(const void \*, const void \*, void \*), void \*arg)
 

Sort an array.

### 6.24.1 Detailed Description

Standard library header.

Copyright

(C) 2013 - 2018, Samsung Electronics Co., Ltd.

## 6.25 string.h File Reference

String manipulation functions.

## Functions

- [errno\\_t memcpy\\_s](#) (void \*restrict dest, rsize\_t dest\_max, const void \*restrict src, rsize\_t n)  
Check arguments and copy *src* sized memory area to *dest*. Memory must not be overlapped. To copy overlapped area use [memmove\\_s\(\)](#) function.
- [errno\\_t memmove\\_s](#) (void \*dest, rsize\_t dest\_max, const void \*src, rsize\_t n)  
Check arguments and copy *src* sized memory area to *dest*. Memory may be overlapped.
- [errno\\_t memset\\_s](#) (void \*block, rsize\_t block\_max, int c, rsize\_t n)  
Check arguments and fill *n* bytes of *block* sized memory with *c* constant value.
- [size\\_t strlen\\_s](#) (const char \*s, size\_t s\_max)  
Check arguments and calculate the length of the *s* fixed-size string, excluding the terminating null byte ('\0').
- [errno\\_t strcpy\\_s](#) (char \*restrict dest, rsize\_t dest\_max, const char \*restrict src)  
Check arguments and copy the *src* string, including the terminating null byte ('\0'), to the *dest* sized buffer. If *n* is bigger than size of *src* then the remaining characters (after '\0') are unspecified.
- [errno\\_t strncpy\\_s](#) (char \*restrict dest, rsize\_t dest\_max, const char \*restrict src, rsize\_t n)  
Check arguments and copy at most *n* bytes of the *src* string, including the terminating null byte ('\0') to the *dest* sized buffer.
- [errno\\_t strcat\\_s](#) (char \*restrict dest, rsize\_t dest\_max, const char \*restrict src)  
Check arguments and append the *src* string to the *dest* string, overwriting the terminating null byte ('\0') at the end of *dest*, and add a terminating null byte.
- [errno\\_t strncat\\_s](#) (char \*restrict dest, rsize\_t dest\_max, const char \*restrict src, rsize\_t n)  
Check arguments and append at most *n* bytes of *src* string to the *dest* string, overwriting the terminating null byte ('\0') at the end of *dest* and then adds a terminating null byte.
- [errno\\_t strerror\\_s](#) (char \*buf, rsize\_t bufmax, errno\_t errnum)  
Check arguments and return a pointer to a *buf* string that describes the *errnum* error code.
- void \* [memcpy](#) (void \*dest, const void \*src, size\_t n)  
Copy memory area from *src* to *dst*. The memory must not overlap. To copy overlapped area use [memmove\(\)](#) function.
- void \* [memmove](#) (void \*dest, const void \*src, size\_t n)  
Copy memory area from *src* to *dest*. The memory may overlap.
- int [memcmp](#) (const void \*s1, const void \*s2, size\_t n)  
Compare first *n* bytes of memory pointed by *s1* and *s2*.
- void \* [memset](#) (void \*block, int c, size\_t size)  
Fill *size* bytes with a constant value.
- [size\\_t strlen](#) (const char \*s)  
Calculate the length of the string *s*, excluding the terminating null byte ('\0').
- [size\\_t strlen\\_s](#) (const char \*s, size\_t n)  
Calculate the length of the fixed-size string *s*, excluding the terminating null byte ('\0').
- char \* [strcpy](#) (char \*dest, const char \*src)  
Copy the string pointed to by *src*, including the terminating null byte ('\0'), to the buffer pointed to by *dest*.
- char \* [strncpy](#) (char \*dest, const char \*src, size\_t n)  
Copy at most *n* bytes of the string pointed to by *src*, including the terminating null byte ('\0'), to the buffer pointed to by *dest*.
- char \* [strdup](#) (const char \*s)  
Return a pointer to a new string which is a duplicate of the string *s*. Memory for the new string is obtained with [malloc\(\)](#).
- char \* [strstr](#) (const char \*text, const char \*pattern)  
Find the first occurrence of the substring *pattern* in the string *text*. The terminating null bytes ('\0') are not compared.
- char \* [strncat](#) (char \*dest, const char \*src, size\_t n)  
Append the *src* string to the *dest* string, overwriting the terminating null byte ('\0') at the end of *dest*, and then adds a terminating null byte.
- [size\\_t strlcat](#) (char \*dest, const char \*src, size\_t n)

Append the NUL-terminated string *src* to the end of *dest* string, overwriting the terminating null byte ('\0') at the end of *dest*, and guarantee to NUL-terminate the result.

- char \* **strcat** (char \*dest, const char \*src)  
Append the *src* string to the *dest* string, overwriting the terminating null byte ('\0') at the end of *dest*, and then adds a terminating null byte.
- int **strcmp** (const char \*s1, const char \*s2)  
Compare the two strings *s1* and *s2*.
- int **strncmp** (const char \*s1, const char \*s2, size\_t n)  
Compare at most *n* bytes of two strings *s1* and *s2*.
- char \* **strchr** (const char \*s, int c)  
Find last occurrence of character *c* in string *s*.
- const char \* **strerror** (int errnum)  
Return a pointer to a string that describes the error code passed in the argument *errnum*.
- void \* **memchr** (const void \*s, int c, size\_t n)  
Find a character in an area of memory.
- char \* **strchr** (const char \*s, int c)  
Find first occurrence of character *c* in string *s*.
- char \* **strchrnul** (const char \*s, int c)  
Find first occurrence of character *c* in string *s*.
- size\_t **strspn** (const char \*s, const char \*accept)  
Calculate the length (in bytes) of the initial segment of *s* which consists entirely of bytes in *accept*.
- size\_t **strcspn** (const char \*s, const char \*reject)  
Calculate the length of the initial segment of *s* which consists entirely of bytes not in *reject*.
- char \* **strtok** (char \*str, const char \*delim)  
Function breaks a string into a sequence of zero or more nonempty tokens.
- char \* **strtok\_r** (char \*\_restrict\_ s, const char \*\_restrict\_ sep, char \*\*\_restrict\_ p)  
Function breaks a string into a sequence of zero or more nonempty tokens.
- int **strerror\_r** (int errnum, char \*buf, size\_t buflen)  
Returns the error string in the user-supplied *buf* of length *buflen*. XSI-compliant version.

### 6.25.1 Detailed Description

String manipulation functions.

Copyright

(C) 2012-2018, Samsung Electronics Co., Ltd.

## 6.26 sys/credentials.h File Reference

Definitions for credentials.

### Functions

- int **cred\_compare** (const struct cred \*as\_is, const struct cred \*to\_be)  
Compare credentials.

## 6.26.1 Detailed Description

Definitions for credentials.

Copyright

(C) 2012-2017, Samsung Electronics Co., Ltd.

## 6.26.2 Function Documentation

### 6.26.2.1 `int cred_compare ( const struct cred * as_is, const struct cred * to_be )`

Compare credentials.

Parameters

|    |                    |                                          |
|----|--------------------|------------------------------------------|
| in | <code>as_is</code> | struct of credentials which user has got |
| in | <code>to_be</code> | struct of credentials which need to have |

Returns

0 on success  
-1 on error

## 6.27 `sys/ioctl.h` File Reference

ioctl declaration.

### Functions

- int [ioctl](#) (int fd, int request, unsigned long data)  
*Manipulates the underlying device parameters of special files.*

### 6.27.1 Detailed Description

ioctl declaration.

Copyright

(C) 2013 Samsung Electronics Co., Ltd.

## 6.28 `sys/socket.h` File Reference

Support socket routines.

## Functions

- int [accept](#) (int sockfd, struct sockaddr \*addr, int \*addrlen)  
*Accept a connection on a socket.*
- int [bind](#) (int sockfd, const struct sockaddr \*addr, int addrlen)  
*Bind a name to a socket.*
- int [connect](#) (int sockfd, const struct sockaddr \*addr, int addrlen)  
*Initiate a connection on a socket.*
- int [getsockopt](#) (int sockfd, int level, int optname, void \*optval, int \*optlen)  
*Get options on socket.*
- int [setsockopt](#) (int sockfd, int level, int optname, void \*optval, int optlen)  
*Get options on socket.*
- int [listen](#) (int sockfd, int backlog)  
*Listen for connections on a socket.*
- int [socket](#) (int socket\_family, int socket\_type, int protocol)  
*Create endpoint for communication.*
- [ssize\\_t](#) [recv](#) (int sockfd, void \*buf, size\_t len, int flags)  
*Receive a message from a socket.*
- [ssize\\_t](#) [send](#) (int sockfd, const void \*buf, size\_t len, int flags)  
*Send a message to a socket.*
- int [socketpair](#) (int socket\_family, int socket\_type, int protocol, int sv[2])  
*Create a pair of connected sockets.*
- [ssize\\_t](#) [recvmsg](#) (int sockfd, struct msghdr \*msg, int flags)  
*Receive multiple message on a socket.*
- [ssize\\_t](#) [sendmsg](#) (int sockfd, struct msghdr \*msg, int flags)  
*Send multiple message on a socket.*

### 6.28.1 Detailed Description

Support socket routines.

Copyright

(C) 2016 - 2018, Samsung Electronics Co., Ltd.

## 6.29 sys/types.h File Reference

Wrapper for kernel [types.h](#) header.

### Typedefs

- typedef int [ssize\\_t](#)
- typedef int [pid\\_t](#)
- typedef int [uid\\_t](#)
- typedef int [gid\\_t](#)
- typedef long long [time\\_t](#)
- typedef int64\_t [off\\_t](#)
- typedef unsigned int [mode\\_t](#)

### 6.29.1 Detailed Description

Wrapper for kernel [types.h](#) header.

Copyright

(C) 2013-2016, Samsung Electronics Co., Ltd.

## 6.30 sys/un.h File Reference

Definitions for UNIX domain sockets.

### 6.30.1 Detailed Description

Definitions for UNIX domain sockets.

Copyright

(C) 2012-2016, Samsung Electronics Co., Ltd.

## 6.31 tee\_error.h File Reference

Manipulation with errno and TEE error.

### Functions

- TEE\_Result [errno\\_to\\_tee\\_error](#) (int error\_code)  
*Translate errno to GP TEE errors code.*

### 6.31.1 Detailed Description

Manipulation with errno and TEE error.

Copyright

(C) 2016, Samsung Electronics Co., Ltd.

## 6.32 tee\_i2c.h File Reference

TEE I2C public API.

## Data Structures

- struct [TEES\\_I2CTransfer](#)  
*I2C data transfer buffer. [More...](#)*

## Typedefs

- typedef struct \_\_TEES\_I2CHandlerImpl \* [TEES\\_I2CHandler](#)

## Functions

- TEE\_Result [TEES\\_I2CInit](#) (const char \*name, uint32\_t transac\_len, [TEES\\_I2CHandler](#) \*handler)  
*Initialize I2C device and set transfer parameters to `handler`.*
- TEE\_Result [TEES\\_I2CExit](#) ([TEES\\_I2CHandler](#) handler)  
*Terminate connection to I2C device.*
- TEE\_Result [TEES\\_I2CWrite](#) ([TEES\\_I2CHandler](#) handler, uint8\_t chip, [TEES\\_I2CTransfer](#) \*tx)  
*Write data buffer `tx` to initialized I2C device.*
- TEE\_Result [TEES\\_I2CRead](#) ([TEES\\_I2CHandler](#) handler, uint8\_t chip, [TEES\\_I2CTransfer](#) \*rx)  
*Read data buffer `rx` from initialized I2C device.*
- TEE\_Result [TEES\\_I2CWriteRead](#) ([TEES\\_I2CHandler](#) handler, uint8\_t chip, [TEES\\_I2CTransfer](#) \*tx, [TEES\\_I2CTransfer](#) \*rx)  
*Write data buffer `tx` and read buffer `rx` from initialized I2C device at the same time.*

### 6.32.1 Detailed Description

TEE I2C public API.

Copyright

(C) 2016, Samsung Electronics Co., Ltd.

## 6.33 tee\_interrupt.h File Reference

Interrupt syscalls.

### Typedefs

- typedef int [TEES\\_InterruptHandle](#)

## Functions

- TEE\_Result [TEES\\_AllocateInterrupt](#) (int nr, intruhandler handler, [TEES\\_InterruptHandle](#) \*handle)  
*This function is used to allocate interrupt and register user handler.*
- TEE\_Result [TEES\\_ReleaseInterrupt](#) ([TEES\\_InterruptHandle](#) handle)  
*This function is used to release interrupt.*
- TEE\_Result [TEES\\_GenerateInterrupt](#) ([TEES\\_InterruptHandle](#) handle)  
*This function is used to generate interrupt.*
- TEE\_Result [TEES\\_WaitForInterrupt](#) ([TEES\\_InterruptHandle](#) handle, uint32\_t timeout)  
*This function is used to wait for interrupt.*
- TEE\_Result [TEES\\_CompleteInterrupt](#) ([TEES\\_InterruptHandle](#) handle)  
*This function is used to notify about interrupt arrival.*

### 6.33.1 Detailed Description

Interrupt syscalls.

Copyright

(C) 2017, Samsung Electronics Co., Ltd.

## 6.34 tee\_isocket.h File Reference

GP iSockets interface (GPD\_SPE\_100)

### Data Structures

- struct [TEE\\_iSocket\\_s](#)  
*iSocket instance Please refer to GPD\_SPE\_100 specification for detailed description. Basic rules are following: [More...](#)*

### Typedefs

- typedef struct \_\_TEE\_iSocketHandle \* [TEE\\_iSocketHandle](#)  
*iSocket context handle*
- typedef const struct [TEE\\_iSocket\\_s](#) [TEE\\_iSocket](#)  
*iSocket instance Please refer to GPD\_SPE\_100 specification for detailed description. Basic rules are following:*
- typedef enum [TEE\\_ipSocket\\_ipVersion\\_e](#) [TEE\\_ipSocket\\_ipVersion](#)  
*IP version.*

## Enumerations

- enum {  
[TEE\\_ISOCKET\\_ERROR\\_PROTOCOL](#) = 0xF1007001, [TEE\\_ISOCKET\\_ERROR\\_REMOTE\\_CLOSED](#) = 0xF1007002, [TEE\\_ISOCKET\\_ERROR\\_TIMEOUT](#) = 0xF1007003, [TEE\\_ISOCKET\\_ERROR\\_OUT\\_OF\\_RESOURCES](#) = 0xF1007004,  
[TEE\\_ISOCKET\\_ERROR\\_LARGE\\_BUFFER](#) = 0xF1007005, [TEE\\_ISOCKET\\_WARNING\\_PROTOCOL](#) = 0xF1007006, [TEE\\_ISOCKET\\_ERROR\\_HOSTNAME](#) = 0xF1007007 }  
*iSocket common errors*
- enum { [TEE\\_ISOCKET\\_SERVER\\_NAME\\_MAX\\_LENGTH](#) = 255 }  
*Maximum server IPv4 address string length.*
- enum [TEE\\_ipSocket\\_ipVersion\\_e](#) { [TEE\\_IP\\_VERSION\\_DC](#) = 0, [TEE\\_IP\\_VERSION\\_4](#) = 1, [TEE\\_IP\\_VERSION\\_6](#) = 2 }  
*IP version.*

### 6.34.1 Detailed Description

GP iSockets interface (GPD\_SPE\_100)

Copyright

(C) 2016-2017, Samsung Electronics Co., Ltd.

## 6.35 tee\_smc.h File Reference

TEE SMC public API.

### Macros

- #define [U\(n\)](#) ((unsigned int)(n))  
*Defined for compatibility now.*

### Typedefs

- typedef int [TEES\\_SMCHandle](#)  
*SMC interface handle.*
- typedef struct [smc\\_data](#) [TEES\\_SMCData](#)  
*SMC command data.*

### Functions

- TEE\_Result [TEES\\_SMCInit](#) ([TEES\\_SMCHandle](#) \*handle)  
*Initialize handler fields.*
- TEE\_Result [TEES\\_SMCFin](#) ([TEES\\_SMCHandle](#) handle)  
*release SMC interface handle.*
- TEE\_Result [TEES\\_SMCCommand](#) ([TEES\\_SMCHandle](#) handle, [TEES\\_SMCData](#) \*data)  
*Send SMC command to EL3.*

### 6.35.1 Detailed Description

TEE SMC public API.

Copyright

(C) 2018, Samsung Electronics Co., Ltd.

## 6.36 tee\_spi.h File Reference

TEE SPI public API.

### Data Structures

- struct [TEES\\_SPIConfig](#)  
*Configuration of SPI device. [More...](#)*
- struct [TEES\\_SPITransfer](#)  
*Descriptor to transfer data over SPI. [More...](#)*

### Typedefs

- typedef struct [\\_\\_TEES\\_SPIHandlerImpl](#) \* [TEES\\_SPIHandler](#)

### Functions

- TEE\_Result [TEES\\_SPIDMAInit](#) (uintptr\_t address)  
*Initialize DMA for working with SPI device.*
- TEE\_Result [TEES\\_SPIInit](#) (uint32\_t port, const [TEES\\_SPIConfig](#) \*cfg, [TEES\\_SPIHandler](#) \*handler)  
*Initialize handler fields.*
- TEE\_Result [TEES\\_SPIExit](#) ([TEES\\_SPIHandler](#) handler)  
*Free handler resource.*
- TEE\_Result [TEES\\_SPISetConfig](#) ([TEES\\_SPIHandler](#) handler, const [TEES\\_SPIConfig](#) \*cfg)  
*Initialize handler with configuration values.*
- TEE\_Result [TEES\\_SPISetClockSpeed](#) ([TEES\\_SPIHandler](#) handler, uint32\_t speedHz)  
*Set clock frequency rate.*
- TEE\_Result [TEES\\_SPISetBitsPerWord](#) ([TEES\\_SPIHandler](#) handler, uint8\_t bitsPerWord)  
*Set the number of bits that will be transferred per SPI rate.*
- TEE\_Result [TEES\\_SPISetDMAMode](#) ([TEES\\_SPIHandler](#) handler, bool isEnabled)  
*Enable or disable DMA mode.*
- TEE\_Result [TEES\\_SPISetCPOL](#) ([TEES\\_SPIHandler](#) handler, uint8\_t cpol)  
*Set SPI clock polarity bit.*
- TEE\_Result [TEES\\_SPISetCPHA](#) ([TEES\\_SPIHandler](#) handler, uint8\_t cpha)  
*Set SPI clock phase bit.*
- TEE\_Result [TEES\\_SPIClockEnable](#) ([TEES\\_SPIHandler](#) handler)  
*Not implemented.*
- TEE\_Result [TEES\\_SPIClockDisable](#) ([TEES\\_SPIHandler](#) handler)  
*Not implemented.*
- TEE\_Result [TEES\\_SPIWrite](#) ([TEES\\_SPIHandler](#) handler, [TEES\\_SPITransfer](#) \*tx)  
*Transfer data from buffer bus to SPI.*
- TEE\_Result [TEES\\_SPIRead](#) ([TEES\\_SPIHandler](#) handler, [TEES\\_SPITransfer](#) \*rx)  
*Transfer data from SPI bus to buffer.*
- TEE\_Result [TEES\\_SPIWriteRead](#) ([TEES\\_SPIHandler](#) handler, [TEES\\_SPITransfer](#) \*tx, [TEES\\_SPITransfer](#) \*rx)  
*Transfer data from buffer to SPI bus.*

### 6.36.1 Detailed Description

TEE SPI public API.

Copyright

(C) 2015, Samsung Electronics Co., Ltd.

## 6.37 tee\_ta\_destructor.h File Reference

Driver destructor registration header.

### Typedefs

- typedef void(\* [TEES\\_DriverDestructor\\_t](#)) (int)

### Functions

- TEE\_Result [TEES\\_RegisterDriverDestructor](#) ([TEES\\_DriverDestructor\\_t](#) destr)  
*Initializes driver destructor function pointer. The destructor will be called when secure kernel generates any interrupting signal.*

### 6.37.1 Detailed Description

Driver destructor registration header.

Copyright

(C) 2017-2017, Samsung Electronics Co., Ltd.

## 6.38 tee\_tcpsocket.h File Reference

GP iSockets TCP API header (GPD\_SPE\_101)

### Data Structures

- struct [TEE\\_tcpSocket\\_Setup\\_s](#)  
*TCP Setup structure. [More...](#)*

### Typedefs

- typedef struct [TEE\\_tcpSocket\\_Setup\\_s](#) [TEE\\_tcpSocket\\_Setup](#)  
*TCP Setup structure.*

## Enumerations

- enum { [TEE\\_ISOCKET\\_TCP\\_API\\_VERSION](#) = 0x01010000 }  
*TCP iSocket API version. Used to ensure API structures matching.*
- enum { [TEE\\_ISOCKET\\_PROTOCOLID\\_TCP](#) = 0x65 }  
*TCP Protocol identifier.*
- enum { [TEE\\_ISOCKET\\_TCP\\_WARNING\\_UNKNOWN\\_OUT\\_OF\\_BAND](#) = 0xF1010002 }  
*TCP Instance specific errors.*

## Variables

- const [TEE\\_iSocket](#) \*const [TEE\\_tcpSocket](#)  
*Public TCP instance pointer.*

### 6.38.1 Detailed Description

GP iSockets TCP API header (GPD\_SPE\_101)

Copyright

(C) 2016-2017, Samsung Electronics Co., Ltd.

## 6.39 tee\_tlssocket.h File Reference

GP iSockets TLS API (GPD\_SPE\_103)

### Data Structures

- struct [TEE\\_tlsSocket\\_PSK\\_Info\\_s](#)  
*Pre-Shared Key (PSK). When PSK is used, the TA needs to provide the key and a key identity to the TLS implementation. This structure holds that information Not supported. [More...](#)*
- struct [TEE\\_tlsSocket\\_SRP\\_Info\\_s](#)  
*Secure Remote Password (SRP). When SRP is used, the TA needs to provide the password and the user identity to the TLS implementation. This structure holds that information. Not supported. [More...](#)*
- struct [TEE\\_tlsSocket\\_ClientPDC\\_s](#)  
*This structure holds the opaque client certificate for the TA as well as the corresponding private key. This is used to provide pre-installed certificates for the TA authentication on Server. [More...](#)*
- struct [TEE\\_tlsSocket\\_ServerPDC\\_s](#)  
*If the server Root public key has been pre-distributed to the TA, this structure holds the TEE\_ObjectHandle to that key. If desirable, Server Root credentials could be provided as bulkCertChain - this is GP specs extension. publicKey is used by default. [More...](#)*
- struct [TEE\\_tlsSocket\\_CertStorageCred\\_s](#)  
*Void type for future usage. Applications SHALL pass a NULL pointer. The intention is to have this structure hold handles or references to either trusted root certificates or a proper client certificate inside a future certificate storage of the TEE. [More...](#)*
- struct [TEE\\_tlsSocket\\_Credentials\\_s](#)  
*Structure holding server and client credentials. [More...](#)*
- struct [TEE\\_tlsSocket\\_CallbackInfo\\_s](#)

- *Callback description structure. [More...](#)*
- struct [TEE\\_tlsSocket\\_Setup\\_s](#)  
*TLS Setup structure. [More...](#)*
- struct [TEE\\_tlsSocket\\_CB\\_Data\\_s](#)  
*IOCTL definitions. [More...](#)*
- union [TEE\\_tlsSocket\\_Credentials\\_s.\\_\\_unnamed\\_\\_](#)
- union [TEE\\_tlsSocket\\_Credentials\\_s.\\_\\_unnamed\\_\\_](#)
- union [TEE\\_tlsSocket\\_Setup\\_s.\\_\\_unnamed\\_\\_](#)

## Typedefs

- typedef enum [TEE\\_tlsSocket\\_tlsVersion\\_e](#) [TEE\\_tlsSocket\\_tlsVersion](#)  
*TLS protocol version to use.*
- typedef enum [TEE\\_tlsSocket\\_CipherSuites\\_e](#) [TEE\\_tlsSocket\\_CipherSuites](#)  
*Cryptosuite ID definitions.*
- typedef struct [TEE\\_tlsSocket\\_PSK\\_Info\\_s](#) [TEE\\_tlsSocket\\_PSK\\_Info](#)  
*Pre-Shared Key (PSK). When PSK is used, the TA needs to provide the key and a key identity to the TLS implementation. This structure holds that information Not supported.*
- typedef struct [TEE\\_tlsSocket\\_SRP\\_Info\\_s](#) [TEE\\_tlsSocket\\_SRP\\_Info](#)  
*Secure Remote Password (SRP). When SRP is used, the TA needs to provide the password and the user identity to the TLS implementation. This structure holds that information. Not supported.*
- typedef struct [TEE\\_tlsSocket\\_ClientPDC\\_s](#) [TEE\\_tlsSocket\\_ClientPDC](#)  
*This structure holds the opaque client certificate for the TA as well as the corresponding private key. This is used to provide pre-installed certificates for the TA authentication on Server.*
- typedef struct [TEE\\_tlsSocket\\_ServerPDC\\_s](#) [TEE\\_tlsSocket\\_ServerPDC](#)  
*If the server Root public key has been pre-distributed to the TA, this structure holds the TEE\_ObjectHandle to that key. If desirable, Server Root credentials could be provided as bulkCertChain - this is GP specs extension. publicKey is used by default.*
- typedef struct [TEE\\_tlsSocket\\_CertStorageCred\\_s](#) [TEE\\_tlsSocket\\_CertStorageCred](#)  
*Void type for future usage. Applications SHALL pass a NULL pointer. The intention is to have this structure hold handles or references to either trusted root certificates or a proper client certificate inside a future certificate storage of the TEE.*
- typedef enum [TEE\\_tlsSocket\\_ClientCredentialType\\_e](#) [TEE\\_tlsSocket\\_ClientCredentialType](#)  
*This specifies what kind of client credentials the TA has.*
- typedef enum [TEE\\_tlsSocket\\_ServerCredentialType\\_e](#) [TEE\\_tlsSocket\\_ServerCredentialType](#)  
*This specifies what kind of server credentials a remote node has.*
- typedef struct [TEE\\_tlsSocket\\_Credentials\\_s](#) [TEE\\_tlsSocket\\_Credentials](#)  
*Structure holding server and client credentials.*
- typedef enum [TEE\\_tlsSocket\\_CallbackReasonType\\_e](#) [TEE\\_tlsSocket\\_CallbackReasonType](#)  
*Callback types.*
- typedef struct [TEE\\_tlsSocket\\_CallbackInfo\\_s](#) [TEE\\_tlsSocket\\_CallbackInfo](#)  
*Callback description structure.*
- typedef [TEE\\_Result](#)(\* [TEE\\_tlsCallback](#)) ([TEE\\_iSocketHandle](#) ctx, [TEE\\_tlsSocket\\_CallbackInfo](#) \*cbInfo, void \*cbData, uint32\_t \*cbDataLength)  
*Callback function. This is specification extension. Used to allow client perform custom checks of certificate chain, OCSP response. etc. cbData buffer is valid only in the callback context.*
- typedef enum [TEE\\_tlsSocket\\_StatusRequestType\\_e](#) [TEE\\_tlsSocket\\_StatusRequestType](#)  
*OCSP stapling certificate status request type.*
- typedef enum [TEE\\_tlsSocket\\_ExtensionFlags\\_e](#) [TEE\\_tlsSocket\\_ExtensionFlags](#)  
*Certificate/OCSP validation mode and callback control flags.*
- typedef struct [TEE\\_tlsSocket\\_Setup\\_s](#) [TEE\\_tlsSocket\\_Setup](#)  
*TLS Setup structure.*
- typedef struct [TEE\\_tlsSocket\\_CB\\_Data\\_s](#) [TEE\\_tlsSocket\\_CB\\_Data](#)  
*IOCTL definitions.*

## Enumerations

- enum { **TEE\_ISOCKET\_TLS\_API\_VERSION** = 0x01030000 }  
*TEE iSocket API version. Used to ensure API structures matching.*
- enum { **TEE\_ISOCKET\_PROTOCOLID\_TLS** = 0x67 }  
*TEE Protocol identifier.*
- enum {  
**TEE\_ISOCKET\_TLS\_ERROR\_REJECTED\_SUITE** = 0xF1030001, **TEE\_ISOCKET\_TLS\_ERROR\_VERSION**  
= 0xF1030002, **TEE\_ISOCKET\_TLS\_ERROR\_UNSUPPORTED\_SUITE** = 0xF1030003, **TEE\_ISOCKET\_TLS\_ERROR\_HANDSHAKE\_FAILED**  
= 0xF1030004,  
**TEE\_ISOCKET\_TLS\_ERROR\_AUTHENTICATION** = 0xF1030005, **TEE\_ISOCKET\_TLS\_ERROR\_DATA**  
= 0xF1030006 }  
*TEE Instance specific errors.*
- enum **TEE\_tlsSocket\_tlsVersion\_e** { **TEE\_TLS\_VERSION\_ALL** = 0, **TEE\_TLS\_VERSION\_1v2** = 1 }  
*TEE TLS protocol version to use.*
- enum **TEE\_tlsSocket\_CipherSuites\_e** {  
**TLS\_NULL\_WITH\_NULL\_NULL** = 0x0000, **TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA** = 0x000A,  
**TLS\_DHE\_DSS\_WITH\_3DES\_EDE\_CBC\_SHA** = 0x0013, **TLS\_DHE\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA**  
= 0x0016,  
**TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA** = 0x002F, **TLS\_DHE\_DSS\_WITH\_AES\_128\_CBC\_SHA** =  
0x0032, **TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA** = 0x0033, **TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA**  
= 0x0035,  
**TLS\_DHE\_DSS\_WITH\_AES\_256\_CBC\_SHA** = 0x0038, **TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA** =  
0x0039, **TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256** = 0x003C, **TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256**  
= 0x003D,  
**TLS\_DHE\_DSS\_WITH\_AES\_128\_CBC\_SHA256** = 0x0040, **TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256**  
= 0x0067, **TLS\_DHE\_DSS\_WITH\_AES\_256\_CBC\_SHA256** = 0x006A, **TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA256**  
= 0x006B,  
**TLS\_PSK\_WITH\_3DES\_EDE\_CBC\_SHA** = 0x008B, **TLS\_PSK\_WITH\_AES\_128\_CBC\_SHA** = 0x008C,  
**TLS\_PSK\_WITH\_AES\_256\_CBC\_SHA** = 0x008D, **TLS\_DHE\_PSK\_WITH\_3DES\_EDE\_CBC\_SHA** =  
0x008F,  
**TLS\_DHE\_PSK\_WITH\_AES\_128\_CBC\_SHA** = 0x0090, **TLS\_DHE\_PSK\_WITH\_AES\_256\_CBC\_SHA** =  
0x0091, **TLS\_RSA\_PSK\_WITH\_3DES\_EDE\_CBC\_SHA** = 0x0093, **TLS\_RSA\_PSK\_WITH\_AES\_128\_CBC\_SHA**  
= 0x0094,  
**TLS\_RSA\_PSK\_WITH\_AES\_256\_CBC\_SHA** = 0x0095, **TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256** =  
0x009C, **TLS\_RSA\_WITH\_AES\_256\_GCM\_SHA384** = 0x009D, **TLS\_DHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256**  
= 0x009E,  
**TLS\_DHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384** = 0x009F, **TLS\_DHE\_DSS\_WITH\_AES\_128\_GCM\_SHA256**  
= 0x00A2, **TLS\_DHE\_DSS\_WITH\_AES\_256\_GCM\_SHA384** = 0x00A3, **TLS\_PSK\_WITH\_AES\_128\_GCM\_SHA256**  
= 0x00A8,  
**TLS\_PSK\_WITH\_AES\_256\_GCM\_SHA384** = 0x00A9, **TLS\_DHE\_PSK\_WITH\_AES\_128\_GCM\_SHA256**  
= 0x00AA, **TLS\_DHE\_PSK\_WITH\_AES\_256\_GCM\_SHA384** = 0x00AB, **TLS\_RSA\_PSK\_WITH\_AES\_128\_GCM\_SHA256**  
= 0x00AC,  
**TLS\_RSA\_PSK\_WITH\_AES\_256\_GCM\_SHA384** = 0x00AD, **TLS\_PSK\_WITH\_AES\_128\_CBC\_SHA256**  
= 0x00AE, **TLS\_PSK\_WITH\_AES\_256\_CBC\_SHA384** = 0x00AF, **TLS\_DHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256**  
= 0x00B2,  
**TLS\_DHE\_PSK\_WITH\_AES\_256\_CBC\_SHA384** = 0x00B3, **TLS\_RSA\_PSK\_WITH\_AES\_128\_CBC\_SHA256**  
= 0x00B6, **TLS\_RSA\_PSK\_WITH\_AES\_256\_CBC\_SHA384** = 0x00B7, **TLS\_ECDHE\_ECDSA\_WITH\_3DES\_EDE\_CBC\_SHA**  
= 0xC008,  
**TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA** = 0xC009, **TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA**  
= 0xC00A, **TLS\_ECDHE\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA** = 0xC012, **TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA**  
= 0xC013,  
**TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA** = 0xC014, **TLS\_SRP\_SHA\_WITH\_3DES\_EDE\_CBC\_SHA**  
= 0xC01A, **TLS\_SRP\_SHA\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA** = 0xC01B, **TLS\_SRP\_SHA\_DSS\_WITH\_3DES\_EDE\_CBC\_SHA**  
= 0xC01C,  
**TLS\_SRP\_SHA\_WITH\_AES\_128\_CBC\_SHA** = 0xC01D, **TLS\_SRP\_SHA\_RSA\_WITH\_AES\_128\_CBC\_SHA**  
= 0xC01E, **TLS\_SRP\_SHA\_DSS\_WITH\_AES\_128\_CBC\_SHA** = 0xC01F, **TLS\_SRP\_SHA\_WITH\_AES\_256\_CBC\_SHA**

```

= 0xC020,
TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA = 0xC021, TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA
= 0xC022, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 = 0xC023, TLS_ECDHE_ECDSA_WITH_AES_256_CB
= 0xC024,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 = 0xC027, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
= 0xC028, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 = 0xC02B, TLS_ECDHE_ECDSA_WITH_AES_256_G
= 0xC02C,
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 = 0xC02F, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
= 0xC030, TLS_ECDHE_PSK_WITH_3DES_EDE_CBC_SHA = 0xC034, TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA
= 0xC035,
TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA = 0xC036, TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256
= 0xC037, TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384 = 0xC038, TLS_RSA_WITH_AES_128_CCM
= 0xC09C,
TLS_RSA_WITH_AES_256_CCM = 0xC09D, TLS_DHE_RSA_WITH_AES_128_CCM = 0xC09E,
TLS_DHE_RSA_WITH_AES_256_CCM = 0xC09F, TLS_PSK_WITH_AES_128_CCM = 0xC0A4,
TLS_PSK_WITH_AES_256_CCM = 0xC0A5, TLS_DHE_PSK_WITH_AES_128_CCM = 0xC0A6,
TLS_DHE_PSK_WITH_AES_256_CCM = 0xC0A7 }

```

*Cryptosuite ID definitions.*

- enum `TEE_tlsSocket_ClientCredentialType_e` { `TEE_TLS_CLIENT_CRED_NONE` = 0, `TEE_TLS_CLIENT_CRED_PDC` = 1, `TEE_TLS_CLIENT_CRED_CSC` = 2 }

*This specifies what kind of client credentials the TA has.*

- enum `TEE_tlsSocket_ServerCredentialType_e` { `TEE_TLS_SERVER_CRED_PDC` = 0, `TEE_TLS_SERVER_CRED_CSC` = 1 }

*This specifies what kind of server credentials a remote node has.*

- enum `TEE_tlsSocket_CallbackReasonType_e` {  
`TEE_ISOCKET_TLS_CB_CHECK_CERT_CHAIN` = 1, `TEE_ISOCKET_TLS_CB_BAD_CERT_CHAIN` = 2,  
`TEE_ISOCKET_TLS_CB_CHECK_OCSP_STATUS` = 11, `TEE_ISOCKET_TLS_CB_UNKNOWN_OCSP_STATUS`  
= 12,  
`TEE_ISOCKET_TLS_CB_REVOKED_OCSP_STATUS` = 13 }

*Callback types.*

- enum `TEE_tlsSocket_StatusRequestType_e` { `TEE_ISOCKET_TLS_OCSP_STATUS_REQUEST_NO` = 0, `TEE_ISOCKET_TLS_OCSP_STATUS_REQUEST` = 1 }

*OCSP stapling certificate status request type.*

- enum `TEE_tlsSocket_ExtensionFlags_e` {  
`TEE_ISOCKET_TLS_CERT_NAME_CHECK_CLIENT` = 0x00000001, `TEE_ISOCKET_TLS_CERT_KEYUSAGE_CHECK_C`  
= 0x00000002, `TEE_ISOCKET_TLS_CERT_NOTIFY_CLIENT` = 0x00000004, `TEE_ISOCKET_TLS_OCSP_CHECK_CLI`  
= 0x00010000,  
`TEE_ISOCKET_TLS_OCSP_CHECK_ADVISORY` = 0x00020000, `TEE_ISOCKET_TLS_OCSP_CHECK_MANDATORY`  
= 0x00040000 }

*Certificate/OCSP validation mode and callback control flags.*

- enum { `TEE_ISOCKET_TLS_MAX_ALPN_LIST_LENGTH` = 16 }
- enum { `TEE_TLS_BINDING_INFO` = 0x67000001 }

*IOCTL codes.*

## Variables

- const `TEE_iSocket` \*const `TEE_tlsSocket`

*Public TLS instance pointer.*

### 6.39.1 Detailed Description

GP iSockets TLS API (GPD\_SPE\_103)

Copyright

(C) 2016-2017, Samsung Electronics Co., Ltd.

## 6.40 tee\_udpsocket.h File Reference

GP iSockets UDP API header (GPD\_SPE\_102)

### Data Structures

- struct [TEE\\_udpSocket\\_Setup\\_s](#)  
*UDP Setup structure. [More...](#)*
- struct [TEE\\_udpSocket\\_Change\\_s](#)  
*UDP change addr and port IOCTL structure. TEE\_UDP\_CHANGE\* functions are implementation as synonyms. Both server\_addr and server\_port must be provided for either call. In case of error returned Client should try to open new socket as usual. [More...](#)*

### Typedefs

- typedef struct [TEE\\_udpSocket\\_Setup\\_s](#) [TEE\\_udpSocket\\_Setup](#)  
*UDP Setup structure.*
- typedef struct [TEE\\_udpSocket\\_Change\\_s](#) [TEE\\_udpSocket\\_Change](#)  
*UDP change addr and port IOCTL structure. TEE\_UDP\_CHANGE\* functions are implementation as synonyms. Both server\_addr and server\_port must be provided for either call. In case of error returned Client should try to open new socket as usual.*

### Enumerations

- enum { [TEE\\_ISOCKET\\_UDP\\_API\\_VERSION](#) = 0x01000000 }  
*UDP iSocket API version. Used to ensure API structures matching.*
- enum { [TEE\\_ISOCKET\\_PROTOCOLID\\_UDP](#) = 0x66 }  
*UDP Protocol identifier.*
- enum { [TEE\\_ISOCKET\\_UDP\\_WARNING\\_UNKNOWN\\_OUT\\_OF\\_BAND](#) = 0xF1020002 }  
*UDP Instance specific errors.*
- enum { [TEE\\_UDP\\_CHANGEADDR](#) = 0x66000001, [TEE\\_UDP\\_CHANGEPORT](#) = 0x66000002 }  
*UDP IOCTL codes.*

### Variables

- const [TEE\\_iSocket](#) \*const [TEE\\_udpSocket](#)  
*Public UDP instance pointer.*

#### 6.40.1 Detailed Description

GP iSockets UDP API header (GPD\_SPE\_102)

Copyright

(C) 2017, Samsung Electronics Co., Ltd.

## 6.41 tees\_critical.h File Reference

Critical sections support.

### Functions

- TEE\_Result [TEES\\_EnterCritical](#) (void)  
*Disable routing and handling of normal world interrupts.*
- TEE\_Result [TEES\\_ExitCritical](#) (void)  
*Enable routing and handling of normal world interrupts.*

### 6.41.1 Detailed Description

Critical sections support.

Copyright

(C) 2018, Samsung Electronics Co., Ltd.

## 6.42 tees\_extension.h File Reference

Samsung's extension of TEE internal API.

### Functions

- int [TEES\\_GetTaskDataSize](#) (size\_t \*data\_size)  
*Get used size of data memory of the current Trusted Application instance.*
- void \* [TEES\\_DuplWshmem](#) (void \*address, uint32\_t size)  
*Make long-life duplicate of an Interworld Shared memory buffer.*
- TEE\_Result [TEES\\_IsREESharedMemory](#) (uint32\_t accessFlags, const void \*buffer, size\_t size)  
*Check if buffer is shared with REE.*
- TEE\_Result [TEES\\_IsPhysMemoryProtected](#) (uint32\_t flags, uint64\_t base, size\_t size)  
*Check if physical memory is protected.*

### 6.42.1 Detailed Description

Samsung's extension of TEE internal API.

Copyright

(C) 2012-2017, Samsung Electronics Co., Ltd.

## 6.43 tees\_hwcrypto\_buf.h File Reference

Lock or Unlock HW crypto buffer.

### Functions

- TEE\_Result [TEES\\_LockHWCryptoBuf](#) (void)  
*Lock HW crypto buffer.*
- TEE\_Result [TEES\\_UnlockHWCryptoBuf](#) (void)  
*Unock HW crypto buffer.*

### 6.43.1 Detailed Description

Lock or Unlock HW crypto buffer.

Copyright

(C) 2018, Samsung Electronics Co., Ltd.

## 6.44 tees\_iwt.h File Reference

Samsung IWT API header.

### Typedefs

- typedef struct \_\_TEES\_iwtHandle \* [TEES\\_iwtHandle](#)  
*Handle representing active IWT connection. Must be treated as opaque structure.*

### Enumerations

- enum { [TEES\\_IWT\\_LISTENER\\_NAME\\_MAX\\_LENGTH](#) = 91 }  
*TEES\_IWT\_LISTENER\_NAME\_MAX\_LENGTH.*

### Functions

- TEE\_Result [TEES\\_iwtOpenChannel](#) (const char \*listenerName, [TEES\\_iwtHandle](#) \*iwtCtx)  
*Open interworld transport (IWT) connection (channel) to the NWd Listener "listenerName".*
- TEE\_Result [TEES\\_iwtWrite](#) ([TEES\\_iwtHandle](#) iwtCtx, const void \*buf, uint32\_t \*length)  
*Write length bytes from buf to the active IWT connection iwtCtx.*
- TEE\_Result [TEES\\_iwtRead](#) ([TEES\\_iwtHandle](#) iwtCtx, void \*buf, uint32\_t \*length)  
*Read length bytes to buf from the active IWT connection iwtCtx.*
- TEE\_Result [TEES\\_iwtCloseChannel](#) ([TEES\\_iwtHandle](#) iwtCtx)  
*Close active IWT connection iwtCtx.*

### 6.44.1 Detailed Description

Samsung IWT API header.

Copyright

(C) 2016-2017, Samsung Electronics Co., Ltd.

## 6.45 tees\_kdf.h File Reference

KDF API.

### Functions

- TEE\_Result [TEES\\_DeriveKeyKDF](#) (const void \*label, uint32\_t labelLen, const void \*context, uint32\_t contextLen, uint32\_t outputKeyLen, TEE\_ObjectHandle object)  
*Key Derivation Function(KDF) based on device key. Internal implementation of KDF depends on the chipset.*
- TEE\_Result [TEES\\_DeriveKeySetKDF](#) (const void \*label, uint32\_t labelLen, const void \*context, uint32\_t contextLen, uint32\_t outputKeyLen, TEE\_ObjectHandle object)  
*Key Derivation Function(KDF) based on device key. This function returns the same key for the set of TAs of the same authority. Internal implementation of KDF depends on the chipset.*

### 6.45.1 Detailed Description

KDF API.

Copyright

(C) 2013, Samsung Electronics Co., Ltd.

## 6.46 tees\_rot.h File Reference

ROT API.

### Data Structures

- struct [rot\\_t](#)  
*Structure to handle Root of Trust information. [More...](#)*

### Macros

- #define [SHA256\\_DIGEST\\_LEN](#) 32  
*SHA256\_DIGEST\_LEN is defined to set size for verified\_boot\_key of ROOT\_OF\_TRUST.*

## Typedefs

- typedef struct [rot\\_t](#) [ROOT\\_OF\\_TRUST](#)  
*Structure to handle Root of Trust information.*

## Functions

- TEE\_Result [TEES\\_GetRoT](#) ([ROOT\\_OF\\_TRUST](#) \*rot)  
*Get RoT information.*

### 6.46.1 Detailed Description

ROT API.

Copyright

(C) 2018, Samsung Electronics Co., Ltd.

## 6.47 tees\_seccam.h File Reference

API for secure camera using the special SMC.

### Functions

- TEE\_Result [TEES\\_SECCAM\\_GetStatus](#) (unsigned int \*data)  
*Get a status of secure camera.*

### 6.47.1 Detailed Description

API for secure camera using the special SMC.

Copyright

(C) 2018, Samsung Electronics Co., Ltd.

## 6.48 tees\_secure\_object.h File Reference

Secure Objects API.

## Macros

- #define [SO\\_TAG\\_LEN](#) (16)
- #define [SO\\_IV\\_LEN](#) (16)
- #define [SO\\_AC\\_LEN](#) (4)
- #define [SO\\_MAGIC\\_NUMBER\\_LEN](#) (4)
- #define [SO\\_TA\\_ID\\_LEN](#) (16)
- #define [SO\\_AUTH\\_ID\\_LEN](#) (16)
- #define [SO\\_HEADER\\_SIZE](#)(delegated)
- #define [SO\\_OUT\\_BUF\\_SIZE](#)(in\_len, delegated) ((in\_len) + ([SO\\_HEADER\\_SIZE](#)(delegated)))

## Functions

- TEE\_Result [TEES\\_WrapSecureObject](#) (const unsigned char \*in, uint32\_t in\_len, unsigned char \*out, uint32\_t \*out\_len, SO\_AccessControllInfoType \*ac)  
*Encrypt and sign input data.*
- TEE\_Result [TEES\\_UnwrapSecureObject](#) (const unsigned char \*in, uint32\_t in\_len, unsigned char \*out, uint32\_t \*out\_len)  
*Decrypt and verify wrapped data.*
- TEE\_Result [TEES\\_CheckSecureObjectCreator](#) (const unsigned char \*in, uint32\_t in\_len, SO\_AccessControllInfoType \*ac)  
*Check UUID and AUTH\_ID of creator on wrapped data.*

### 6.48.1 Detailed Description

Secure Objects API.

Copyright

(C) 2013, Samsung Electronics Co., Ltd.

Secure Object APIs provide a TEE service for protecting TA sensitive data.

Secure Object is a wrapped and protected data that only authorized access holder can read. Data is protected by encrypting it with a key that only authorized trusted application can retrieve from the TEE. This process is called wrapping.

Secure Object consists of:

1. Encrypted Data
2. 4 byte Access Control
3. 16 Byte Initialization Vector
4. 16 Byte TAG
5. 16 Byte TA UUID of the creator TA (optional)
6. 16 Byte TA Authority of the creator TA (optional)

Structure of wrapped object:

| Header                  |     |     |          |           |           |         |         |          |          | Data       |
|-------------------------|-----|-----|----------|-----------|-----------|---------|---------|----------|----------|------------|
| Access Control (4 byte) |     |     |          |           |           |         |         |          |          | Data       |
| MAGIC                   | TAG | IV  | reserved | delegated | delegated | Auth ID | TA UUID | optional | optional | encrypted  |
| 4b                      | 16b | 16b | 28bit    | 1bit      | 1bit      | 1bit    | 1bit    | 16b      | 16b      | image size |

The following data structures are used for implementing Access Control, based on KDF function:

```
typedef struct {
 uint32_t access_flags;
 UUID ta_id;
 AUTHORITY auth_id;
} SO_AccessControlInfoType;
```

`access_flags` - is a bit-mask indicating the access control restrictions.

- if bit-0 (TA\_ID\_AC) is set => Current TA Id and Authority are used as input to KDF, therefore Secure Object has to be valid only for this TA.
- if bit-1 (AUTH\_ID\_AC) is set => Current TA Authority is used as input to KDF, therefore Secure Object has to be valid only for the group of TA Authority.
- if bit-2 (DELEGATED\_TA\_ID\_AC) is set => `ta_id` is used as input to KDF. `ta_id` is considered as delegated. Trusted application passes through `ta_id` the UUID of another TA if it wants to share the SO with that TA.
- if bit-3 (DELEGATED\_AUTH\_ID\_AC) set => deprecated
- if bit-2 (DELEGATED\_TA\_ID\_AC) and bit-3 (DELEGATED\_AUTH\_ID\_AC) set => `auth_id` and `ta_id` are used as input to KDF. `auth_id` is considered as delegated. Trusted application passes through `auth_id` the ID of another TA Authority if it wants to share the SO with that TA.

UUID - A Universally Unique Identifier of another Trusted Application with which the Secure Object needs to be shared. User needs to define this only if the Secure Object is to be shared with another TA and TA\_ID\_AC is set.

AUTHORITY - This is the name of the TA Authority access group with which the SO may be shared. User needs to define this only if Secure Object needs to be shared with another TA Authority and TA access flag AUTH\_ID\_AC is set.

## 6.49 tees\_tui.h File Reference

The Trusted User Interface API permits the display of screens to the user and achieves three objectives:

### Data Structures

- struct [TEES\\_TUIScreenInfo](#)  
Structure that represents information on the requested screen. [More...](#)
- struct [TEES\\_TUITouchInfo](#)  
Structure that represents information of touch event (touch position, touch type). [More...](#)
- struct [TEES\\_TUIImage](#)  
Structure that represents properties of image and buffers. [More...](#)

## Macros

- #define `MIN_FONT_SIZE` 20
- #define `MAX_FONT_SIZE` 160

## Enumerations

- enum `TEES_Result` {  
`TEES_SUCCESS` = 0x00000000, `TEES_ERROR_TUI_GENERIC` = 0xFFFF0000, `TEES_ERROR_TUI_BAD_FORMAT`  
= 0xFFFF0005, `TEES_ERROR_TUI_INVALID_PARAM` = 0xFFFF0006,  
`TEES_ERROR_TUI_BAD_STATE` = 0xFFFF0007, `TEES_ERROR_NOT_IMPLEMENTED` = 0xFFFF0009,  
`TEES_ERROR_NOT_SUPPORTED` = 0xFFFF000A, `TEES_ERROR_TUI_OUT_OF_MEMORY` = 0xFFFF000C,  
`TEES_ERROR_TUI_BUSY` = 0xFFFF000D }  
*TEES result codes.*
- enum `TEES_TUITouchTypes` { `TEES_TOUCH_PRESSED`, `TEES_TOUCH_RELEASED` }  
*Internal, touch event type.*
- enum `TEES_bool` { `TEES_FALSE` = 0, `TEES_TRUE` = 1 }  
*Internal, bool type.*

## Functions

- `TEES_Result TEES_TUIGetScreenInfo` (`TEES_TUIScreenInfo` \*screenInfo)  
*Retrieves information about the screen.*
- `TEES_Result TEES_TUIOpenSession` (void)  
*Claims an exclusive access to TUI resources.*
- `TEES_Result TEES_TUICloseSession` (void)  
*Releases TUI resources.*
- `TEES_Result TEES_TUIDrawImage` (`TEES_TUIImage` \*image, uint32\_t posX, uint32\_t posY)  
*Display an image on screen.*
- `TEES_Result TEES_TUIGetTouchEvent` (`TEES_TUITouchInfo` \*touchInfo, uint32\_t timeout)  
*Get a touch event.*
- `TEES_Result TEES_TUICheckTextFormat` (char \*inputText, uint32\_t textSize, uint32\_t \*width,  
uint32\_t \*height, uint32\_t \*lastIndex)  
*Checks validity of the string and Retrieves its width and height.*
- `TEES_Result TEES_TUIPrintString` (char \*inputText, uint32\_t textSize, uint32\_t posX, uint32\_t posY,  
uint8\_t redColorValue, uint8\_t greenColorValue, uint8\_t blueColorValue, bool rotation90)  
*Print string on the screen.*
- `TEES_Result TEES_TUIRefreshScreen` (void)  
*Refresh display controller.*
- `TEES_Result TEES_TUIGetBuffer` (uint32\_t \*buffer, uint32\_t posX, uint32\_t posY, uint32\_t W,  
uint32\_t H)  
*Copy a rectangle from screen to buffer.*
- `TEES_Result TEES_TUIDrawBuffer` (uint32\_t \*buffer, uint32\_t posX, uint32\_t posY, uint32\_t W,  
uint32\_t H)  
*Copy a rectangle from buffer to screen.*

### 6.49.1 Detailed Description

The Trusted User Interface API permits the display of screens to the user and achieves three objectives:

- Secure Display – Information displayed to the user cannot be accessed, modified, or obscured by any software within the REE or by an unauthorized application in the TEE.
- Secure Input – Information entered by the user cannot be derived or modified by any software within the REE or by an unauthorized application in the TEE.
- Secure Indicator – The user can be confident that the screen displayed is actually a screen displayed by a TA.

Copyright

(C) 2016-2017, Samsung Electronics Co., Ltd.

## 6.50 tees\_wrapped\_with\_rek.h File Reference

Wrapped with REK API.

### Data Structures

- struct [wrapped\\_wkth\\_rek\\_t](#)  
*Structure for wrapping with REK. [More...](#)*

### Macros

- #define [KM\\_KW\\_MAX\\_SALT\\_LEN](#) 60
- #define [KM\\_KW\\_MAX\\_IV\\_LEN](#) 12
- #define [KM\\_KW\\_MAX\\_AAD\\_LEN](#) 32
- #define [KM\\_KW\\_MAX\\_KEY\\_LEN](#) 32
- #define [KM\\_KW\\_MAX\\_INPUT\\_LEN](#) 4096
- #define [KM\\_KW\\_MAX\\_TAG\\_LEN](#) 16

### Typedefs

- typedef struct [wrapped\\_wkth\\_rek\\_t](#) [WRAP\\_REK](#)  
*Structure for wrapping with REK.*

### Enumerations

- enum [kw\\_mode](#) { [WRAP](#), [UNWRAP](#) }  
*Wrapping mode. WRAP or UNWRAP.*

## Functions

- TEE\_Result [TEES\\_WrappedWithREK](#) ([WRAP\\_REK](#) \*data)  
*Wrapping with REK.*

### 6.50.1 Detailed Description

Wrapped with REK API.

Copyright

(C) 2018, Samsung Electronics Co., Ltd.

## 6.51 time.h File Reference

Time header.

### Data Structures

- struct [tm](#)

### Macros

- #define [NUM\\_SECONDS\\_IN\\_MIN](#) (60)
- #define [NUM\\_MILLIS\\_IN\\_SEC](#) (1000)
- #define [NUM\\_NANOS\\_IN\\_USEC](#) (1000)
- #define [NUM\\_NANOS\\_IN\\_MILLI](#) (1000000L)
- #define [NUM\\_NANOS\\_IN\\_SEC](#) (1000000000ULL)

### Functions

- int [nanosleep](#) (struct timespec \*req, struct timespec \*rem)  
*[nanosleep\(\)](#) suspends the execution of the calling thread until either at least the time specified in \*req has elapsed, or the delivery of a signal that triggers the invocation of a handler in the calling thread or that terminates the process. If the call is interrupted by a signal handler, [nanosleep\(\)](#) returns -1, sets [errno](#) to [EINTR](#), and writes the remaining time into the structure pointed to by rem unless rem is NULL. The value of \*req can then be used to call [nanosleep\(\)](#) again and complete the specified pause.*
- int [clock\\_gettime](#) (clockid\_t clk\_id, struct timespec \*ts)  
*The function retrieves the time of the specified clock [clk\\_id](#). This function is non-blocking, except [CLOCK\\_REALTIME](#) case. In this case function can sleep and can be interrupted with -1 result and [EINTR](#) [errno](#).*
- [time\\_t](#) [time](#) ([time\\_t](#) \*time)  
*Get the current calendar time as a value of type [time\\_t](#).*
- int [alarm](#) (unsigned int seconds)  
*Set the real-time timer to expire in [seconds](#) seconds.*
- int [setitimer](#) (int which, const struct itimerval \*new\_value, struct itimerval \*old\_value)  
*Set value of an interval timer.*
- int [getitimer](#) (int which, struct itimerval \*curr\_value)

- Get value of an interval timer.*

  - unsigned int [arm\\_timer\\_get\\_frequency](#) (void)
 

*The function retrieves the frequency of ARM timer.*
  - void [timeadd](#) (const struct timespec \*a, const struct timespec \*b, struct timespec \*res)
 

*The function adds two dates.*
  - void [timesub](#) (const struct timespec \*a, const struct timespec \*b, struct timespec \*res)
 

*The function subtracts two dates.*
  - int64\_t [timespec\\_to\\_ms](#) (const struct timespec \*a)
 

*The function converts time from timespec format to milliseconds.*
  - uint64\_t [timespec\\_to\\_nsec](#) (const struct timespec \*a)
 

*The function converts time from timespec format to nanoseconds.*
  - void [ms\\_to\\_timespec](#) (int64\_t t, struct timespec \*a)
 

*The function converts time in milliseconds to to timespec format.*

### 6.51.1 Detailed Description

Time header.

Copyright

(C) 2013 - 2018, Samsung Electronics Co., Ltd.

## 6.52 tui.h File Reference

The Trusted User Interface API permits the display of screens to the user and achieves three objectives:

### Data Structures

- struct [TEE\\_TUIImage](#)

*Structure that defines a way to handle an image for label area and buttons. [More...](#)*
- struct [TEE\\_TUIScreenLabel](#)

*Structure that defines the contents of the TA defined label area, which is provided to support TA branding and a TA defined message. [More...](#)*
- struct [TEE\\_TUIButton](#)

*Structure that defines the content of a button. [More...](#)*
- struct [TEE\\_TUIScreenConfiguration](#)

*Structure that enables configuration of a TUI screen. [More...](#)*
- struct [TEE\\_TUIScreenButtonInfo](#)

*Structure that represents button information associated with a TUI screen for a given orientation. [More...](#)*
- struct [TEE\\_TUIScreenInfo](#)

*Structure that represents screen information for a given orientation. [More...](#)*
- struct [TEE\\_TUIEntryField](#)

*Structure that represents an entry field which acquires user inputs. [More...](#)*
- union [TEE\\_TUIImage.\\_\\_unnamed\\_\\_](#)
- struct [TEE\\_TUIImage.\\_\\_unnamed\\_\\_.ref](#)
- struct [TEE\\_TUIImage.\\_\\_unnamed\\_\\_.object](#)

## Macros

- #define `TEE_TUI_NUMBER_BUTTON_TYPES` 6

## Enumerations

- enum `TEE_TUIEntryFieldMode` { `TEE_TUI_HIDDEN_MODE` = 0, `TEE_TUI_CLEAR_MODE`, `TEE_TEMPRARY_CLEAR_MO` }  
*Entry fields mode.*
- enum `TEE_TUIEntryFieldType` { `TEE_TUI_NUMERICAL` = 0, `TEE_TUI_ALPHANUMERICAL` }  
*Entry fields type.*
- enum `TEE_TUIScreenOrientation` { `TEE_TUI_PORTRAIT` = 0, `TEE_TUI_LANDSCAPE` }  
*Screen orientation.*
- enum `TEE_TUIButtonType` { `TEE_TUI_CORRECTION` = 0, `TEE_TUI_OK`, `TEE_TUI_CANCEL`, `TEE_TUI_VALIDATE`, `TEE_TUI_PREVIOUS`, `TEE_TUI_NEXT` }  
*Button type.*
- enum `TEE_TUIImageSource` { `TEE_TUI_NO_SOURCE` = 0, `TEE_TUI_REF_SOURCE`, `TEE_TUI_OBJECT_SOURCE` }  
*Image source.*

## Functions

- `TEE_Result` `TEE_TUICheckTextFormat` (`char *text`, `uint32_t *width`, `uint32_t *height`, `uint32_t *lastIndex`)  
*Check whether a given text can be rendered and retrieves information.*
- `TEE_Result` `TEE_TUIGetScreenInfo` (`TEE_TUIScreenOrientation` `screenOrientation`, `uint32_t nbEntryFields`, `TEE_TUIScreenInfo *screenInfo`)  
*Retrieves information about the screen.*
- `TEE_Result` `TEE_TUIInitSession` (`void`)  
*Claims an exclusive access to TUI resources.*
- `TEE_Result` `TEE_TUICloseSession` (`void`)  
*Releases TUI resources.*
- `TEE_Result` `TEE_TUIDisplayScreen` (`TEE_TUIScreenConfiguration *screenConfiguration`, `bool closeTUISession`, `TEE_TUIEntryField *entryFields`, `uint32_t entryFieldCount`, `TEE_TUIButtonType *selectedButton`)  
*Displays a TUI screen.*

### 6.52.1 Detailed Description

The Trusted User Interface API permits the display of screens to the user and achieves three objectives:

- Secure Display – Information displayed to the user cannot be accessed, modified, or obscured by any software within the REE or by an unauthorized application in the TEE.
- Secure Input – Information entered by the user cannot be derived or modified by any software within the REE or by an unauthorized application in the TEE.
- Secure Indicator – The user can be confident that the screen displayed is actually a screen displayed by a TA.

Copyright

(C) 2016-2017, Samsung Electronics Co., Ltd.

## 6.53 unistd.h File Reference

Unistd header.

### Macros

- #define **TEMP\_FAILURE\_RETRY**(expression)  
Recall function if it was interrupted by signal.

### Functions

- void **\_exit** (int status)  
Terminate the calling process "immediately". Any open file descriptors belonging to the process are closed; process's parent is sent a SIGCHLD signal.
- void **\_exit\_thread** (unsigned long status)  
Terminate the calling thread "immediately".
- int **profil** (unsigned short \*buf, size\_t bufsiz, size\_t offset, unsigned int scale)  
Provide a means to find out in what areas your program spends most of its time. The argument *buf* points to *bufsiz* bytes of core. Every virtual 10 milliseconds, the user's program counter (PC) is examined: *offset* is subtracted and the result is multiplied by *scale* and divided by 65536. If the resulting value is less than *bufsiz*, then the corresponding entry in *buf* is incremented. If *buf* is NULL, profiling is disabled.
- **pid\_t getpid** (void)  
Return the process ID of the calling process.
- **pid\_t gettid** (void)  
Return the thread ID of the calling thread.
- int **getcpu** (void)  
Return the number of CPU on which current thread is performed.
- int **getcluster** (void)  
Return the cluster id on which current thread is performed.
- int **unlink** (const char \*pathname)  
Remove a link to a file.
- int **ftruncate** (int fd, int size)  
Cause file referenced by *fd* to be truncated to a *size* of precisely length bytes.
- int **close** (int fd)  
Deallocate the file descriptor indicated by *fd*. To deallocate means to make the file descriptor available for return by subsequent calls to **open()** or other functions that allocate file descriptors. All outstanding record locks owned by the process on the file associated with the file descriptor shall be removed (that is, unlocked).
- **ssize\_t read** (int fd, void \*buf, size\_t count)  
Attempt to read *count* bytes from the file associated with the open file descriptor, *fd*, into the buffer pointed to by *buf*.
- **ssize\_t write** (int fd, const void \*buf, size\_t count)  
Attempt to write *count* bytes from buffer pointed to by *buf* to the file associated with the open file descriptor, *fd*.
- int **fstat** (int fd, struct **stat** \*buf)  
Get status of a file with a descriptor *fd*.
- int **stat** (const char \*pathname, struct **stat** \*buf)  
Get status of a file *pathname*.
- int **lseek** (int fd, int offset, int whence)  
Reposition read/write file offset.

### 6.53.1 Detailed Description

Unistd header.

Copyright

(C) 2013 - 2018, Samsung Electronics Co., Ltd.

## 6.54 uuid/uuid.h File Reference

UUID management helpers.

### Data Structures

- struct `__uuid_t`  
wrapper for `uuid` type. [More...](#)

### Macros

- #define `UUID_STRING_LEN` 37
- #define `uuid_unparse_lower`(uu, out) `uuid_unparse`((uu), (out))
- #define `uuid_generate_time`(x) `uuid_generate`(x)

### Typedefs

- typedef struct `__uuid_t` `__uuid_t`
- typedef `__uuid_t` `uuid_t`

### Functions

- void `uuid_unparse` (const `uuid_t` \*uu, char \*out)  
Convert binary representation of UUID to string.
- void `uuid_unparse_upper` (const `uuid_t` \*uu, char \*out)  
Convert binary representation of UUID to string.
- int `uuid_parse` (const char \*in, `uuid_t` \*uu)  
Convert an input UUID string into binary representation.
- void `uuid_generate` (`uuid_t` \*out)  
The `uuid_generate` function creates a new universally unique identifier (UUID).
- int `uuid_is_null` (const `uuid_t` \*uu)  
Check if UUID is null.
- void `uuid_clear` (`uuid_t` \*uu)  
set value to zero UUID.
- int `uuid_compare` (const `uuid_t` \*uu1, const `uuid_t` \*uu2)  
Compare the two supplied uuid variables `uu1` and `uu2` to each other.

### 6.54.1 Detailed Description

UUID management helpers.

Copyright

(C) 2016 - 2018, Samsung Electronics Co., Ltd.

## 6 Index

- \_\_POSIX\_THREAD\_KEYS\_MAX
  - Auxiliary API, 174
- \_\_CPUMASK
  - Auxiliary API, 174
- \_\_TEE\_SC\_CardKeyRef, 290
  - scKeyID, 290
  - scKeyVersion, 290
- \_\_TEE\_SC\_DeviceKeyRef, 290
  - scKeyType, 291
- \_\_TEE\_SC\_DeviceKeyRef.\_\_unnamed\_\_, 291
  - scBaseKeyHandle, 291
  - scKeySetRef, 291
- \_\_TEE\_SC\_KeySetRef, 291
  - scKeyEncHandle, 292
  - scKeyMacHandle, 292
- \_\_TEE\_SC\_OID, 292
  - buffer, 292
  - bufferLen, 292
- \_\_TEE\_SC\_Params, 292
  - scCardKeyRef, 293
  - scDeviceKeyRef, 293
  - scOID, 293
  - scSecurityLevel, 293
  - scType, 293
- \_\_TEE\_SEAID, 293
  - buffer, 294
  - bufferLen, 294
- \_\_TEE\_SEReaderProperties, 294
  - sePresent, 294
  - selectResponseEnable, 294
  - teeOnly, 294
- \_\_pthread\_attr\_t, 117
- \_\_pthread\_cond\_t, 117
- \_\_pthread\_condattr\_t, 118
- \_\_pthread\_mutex\_t, 117
- \_\_pthread\_once\_t, 117
- \_\_uuid\_t, 173
  - Auxiliary API, 203
- \_exit
  - Auxiliary API, 205
- \_exit\_thread
  - Auxiliary API, 205
- AUTO\_BUFFER\_SIZE
  - Auxiliary API, 174
- abort\_handler\_s
  - Auxiliary API, 205
- abs
  - Auxiliary API, 206
- accept
  - Socket library API, 158
- alarm
  - Auxiliary API, 206
- arg\_types
  - smc\_data, 295
- args
  - smc\_data, 295
- arm\_timer\_get\_frequency
  - Auxiliary API, 206
- asprintf
  - Auxiliary API, 207
- assert
  - Auxiliary API, 174
- assert.h, 296
- atan
  - Math library API, 141
- atan2
  - Math library API, 141
- atan2f
  - Math library API, 142
- atanf
  - Math library API, 142
- atexit
  - Auxiliary API, 207
- Auxiliary API, 163
  - \_\_POSIX\_THREAD\_KEYS\_MAX, 174
  - \_\_CPUMASK, 174
  - \_\_uuid\_t, 203
  - \_exit, 205
  - \_exit\_thread, 205
  - AUTO\_BUFFER\_SIZE, 174
  - abort\_handler\_s, 205
  - abs, 206
  - alarm, 206
  - arm\_timer\_get\_frequency, 206
  - asprintf, 207
  - assert, 174
  - atexit, 207
  - BIT\_PER\_CPU, 175
  - BITMAP\_ELT, 175
  - BITS\_TO\_CPU\_MASK, 175
  - CHAR\_BIT, 175
  - CPU\_CLR, 175
  - CPU\_ISSET, 175
  - CPU\_SET, 176
  - CPU\_ZERO, 176
  - calloc, 207
  - clock\_gettime, 208
  - close, 208
  - constraint\_handler\_t, 203
  - DECLARE\_BITMAP, 176
  - E2BIG, 176
  - EACCES, 177
  - EADDRINUSE, 177
  - EADDRNOTAVAIL, 177
  - EADV, 177
  - EAFNOSUPPORT, 177
  - EAGAIN, 177
  - EALREADY, 177
  - EBADFD, 178

EBADMSG, 178  
EBADRQC, 178  
EBADSLT, 178  
EBADE, 178  
EBADF, 178  
EBADR, 178  
EBFONT, 179  
EBUSY, 179  
ECANCELED, 179  
ECHILD, 179  
ECHRNG, 179  
ECOMM, 179  
ECONNABORTED, 179  
ECONNREFUSED, 180  
ECONNRESET, 180  
EDEADLOCK, 180  
EDEADLK, 180  
EDESTADDRREQ, 180  
EDOTDOT, 180  
EDOM, 180  
EDQUOT, 181  
EEXIST, 181  
EFAULT, 181  
EFBIG, 181  
EHOSTDOWN, 181  
EHOSTUNREACH, 181  
EIDRM, 181  
EILSEQ, 182  
EINPROGRESS, 182  
EINTR, 182  
EINVAL, 182  
EISCONN, 182  
EISDIR, 182  
EISNAM, 183  
EIO, 182  
EL2HLT, 183  
EL2NSYNC, 183  
EL3HLT, 183  
EL3RST, 183  
ELIBACC, 183  
ELIBBAD, 183  
ELIBEXEC, 184  
ELIBMAX, 184  
ELIBSCN, 184  
ELNRNG, 184  
ELOOP, 184  
EMFILE, 184  
EMLINK, 184  
EMSGSIZE, 185  
EMULTIHOP, 185  
ENAMETOOLONG, 185  
ENAVAIL, 185  
ENETDOWN, 185  
ENETRESET, 185  
ENETUNREACH, 185  
ENFILE, 186  
ENOANO, 186  
ENOBUFS, 186  
ENOCCSI, 186  
ENODATA, 186  
ENODEV, 186  
ENOENT, 186  
ENOEXEC, 187  
ENOKEY, 187  
ENOLCK, 187  
ENOLINK, 187  
ENOMEM, 187  
ENOMSG, 187  
ENONET, 187  
ENOPKG, 188  
ENOPROTOPT, 188  
ENOSPC, 188  
ENOSTR, 188  
ENOSYS, 188  
ENOSR, 188  
ENOTBLK, 188  
ENOTCONN, 189  
ENOTDIR, 189  
ENOTEMPTY, 189  
ENOTNAM, 189  
ENOTSOCK, 189  
ENOTTY, 189  
ENOTUNIQ, 189  
ENXIO, 190  
EOPNOTSUPP, 190  
EOVERFLOW, 190  
EOF, 190  
EPERM, 190  
EPFNOSUPPORT, 190  
EPIPE, 190  
EPROTONOSUPPORT, 191  
EPROTOTYPE, 191  
EPROTO, 191  
ERANGE, 191  
EREMCHG, 191  
EREMOTEIO, 191  
EREMOTE, 191  
ERESTART, 192  
EROFS, 192  
ESHUTDOWN, 192  
ESOCKTNOSUPPORT, 192  
ESPIPE, 192  
ESRCH, 192  
ESRMNT, 193  
ESTALE, 193  
ESTRPIPE, 193  
ETIMEDOUT, 193  
ETIME, 193  
ETOOMANYREFS, 193  
ETXTBSY, 193  
EUCLEAN, 194  
EUNATCH, 194  
EUSERS, 194  
EWOULDBLOCK, 194  
EXDEV, 194  
EXFULL, 194

errno, 192  
 errno\_t, 204  
 exit, 209  
 fflush, 209  
 fprintf, 209  
 free, 210  
 fstat, 210  
 ftruncate, 211  
 get\_errno\_addr, 211  
 getcluster, 211  
 getcpu, 212  
 getitimer, 212  
 getpid, 212  
 gettid, 213  
 gid\_t, 204  
 INT\_MAX, 194  
 INT\_MIN, 195  
 ignore\_handler\_s, 213  
 invoke\_constraint\_handler\_s, 213  
 ioctl, 213  
 isalnum, 214  
 isalpha, 214  
 isascii, 215  
 isblank, 215  
 iscntrl, 215  
 isdigit, 216  
 isgraph, 216  
 islower, 216  
 isprint, 217  
 ispunct, 217  
 isspace, 217  
 isupper, 218  
 isxdigit, 218  
 LLONG\_MAX, 195  
 LLONG\_MIN, 195  
 LONG\_MAX, 195  
 LONG\_MIN, 195  
 lseek, 218  
 M\_CACHE\_PAGES, 195  
 MAP\_ANONYMOUS, 195  
 MAP\_FAILED, 196  
 MAP\_FIXED, 196  
 MAP\_PHYS\_NON\_CACHED, 196  
 MAP\_PHYS\_NON\_SECURE, 196  
 MAP\_POPULATE, 196  
 MAP\_PRIVATE, 196  
 MAP\_SHARED, 196  
 MAX\_CPUS, 197  
 malloc, 219  
 memchr, 219  
 memcmp, 220  
 memcpy, 220  
 memcpy\_s, 221  
 memmove, 221  
 memmove\_s, 221  
 memset, 222  
 memset\_s, 222  
 mmap, 223  
 mode\_t, 204  
 ms\_to\_timespec, 224  
 munmap, 225  
 NUM\_MILLIS\_IN\_SEC, 197  
 NUM\_NANOS\_IN\_MILLI, 197  
 NUM\_NANOS\_IN\_SEC, 197  
 NUM\_NANOS\_IN\_USEC, 197  
 NUM\_SECONDS\_IN\_MIN, 197  
 nanosleep, 225  
 off\_t, 204  
 open, 226  
 PGOFF\_SHIFT, 197  
 PHYS\_DEV\_NAME\_LEN, 198  
 PHYS\_DEV\_NAME, 198  
 PRId16, 198  
 PRId32, 198  
 PRId64, 198  
 PRlU16, 198  
 PRlU32, 198  
 PRlU64, 199  
 PRlX16, 199  
 PRlX32, 199  
 PRlX64, 199  
 PROT\_EXEC, 199  
 PROT\_NONE, 199  
 PROT\_READ, 199  
 PROT\_WRITE, 200  
 PTHREAD\_KEYS\_MAX, 200  
 pid\_t, 204  
 printf, 227  
 printf\_no\_alloc, 227  
 printf\_s, 228  
 profil, 228  
 putchar, 229  
 puts, 229  
 qsort, 230  
 qsort\_r, 230  
 read, 230  
 realloc, 231  
 SCHAR\_MAX, 200  
 SCHAR\_MIN, 200  
 SCNd16, 200  
 SCNd32, 200  
 SCNd64, 200  
 SCNu16, 201  
 SCNu32, 201  
 SCNu64, 201  
 SCNx16, 201  
 SCNx32, 201  
 SCNx64, 201  
 SHRT\_MAX, 201  
 SHRT\_MIN, 202  
 sched\_getaffinity, 231  
 sched\_setaffinity, 232  
 sched\_yield, 232  
 set\_constraint\_handler\_s, 232  
 setitimer, 233  
 snprintf, 233

- snprintf\_s, 234
- sprintf, 234
- sprintf\_s, 235
- sscanf, 235
- sscanf\_s, 236
- ssize\_t, 204
- stat, 236
- stdin, 265
- strcat, 237
- strcat\_s, 237
- strchr, 238
- strchrnul, 238
- strcmp, 239
- strcpy, 239
- strcpy\_s, 240
- strcspn, 240
- strdup, 241
- strerror, 241
- strerror\_r, 241
- strerror\_s, 242
- strlcat, 242
- strlen, 243
- strncat, 243
- strncat\_s, 244
- strncmp, 244
- strncpy, 245
- strncpy\_s, 246
- strnlen, 246
- strnlen\_s, 247
- strrchr, 247
- strspn, 248
- strstr, 248
- strtod, 249
- strtodf, 249
- strtok, 250
- strtok\_r, 251
- strtol, 251
- strtold, 252
- strtoll, 252
- strtoul, 253
- strtoull, 253
- TEMP\_FAILURE\_RETRY, 202
- time, 254
- time\_t, 204
- timeadd, 254
- timespec\_to\_ms, 255
- timespec\_to\_nsec, 255
- timesub, 255
- tolower, 256
- toupper, 256
- UCHAR\_MAX, 202
- UINT\_MAX, 202
- ULLONG\_MAX, 202
- ULONG\_MAX, 203
- USHRT\_MAX, 203
- UUID\_STRING\_LEN, 203
- uid\_t, 205
- unlink, 256
- uuid\_clear, 257
- uuid\_compare, 257
- uuid\_generate, 257
- uuid\_generate\_time, 203
- uuid\_is\_null, 258
- uuid\_parse, 258
- uuid\_t, 205
- uuid\_unparse, 258
- uuid\_unparse\_lower, 203
- uuid\_unparse\_upper, 259
- vasprintf, 259
- vasprintf\_s, 259
- vfprintf, 260
- vsnprintf, 260
- vsnprintf\_s, 261
- vsprintf, 262
- vsprintf\_s, 263
- vsscanf, 263
- vsscanf\_s, 264
- write, 264
  
- BIT\_PER\_CPU
  - Auxiliary API, 175
- BITMAP\_ELT
  - Auxiliary API, 175
- BITS\_TO\_CPU\_MASK
  - Auxiliary API, 175
- bind
  - Socket library API, 158
- buffer
  - \_\_TEE\_SC\_OID, 292
  - \_\_TEE\_SEAID, 294
- bufferLen
  - \_\_TEE\_SC\_OID, 292
  - \_\_TEE\_SEAID, 294
  
- CHAR\_BIT
  - Auxiliary API, 175
- CPU\_CLR
  - Auxiliary API, 175
- CPU\_ISSET
  - Auxiliary API, 175
- CPU\_SET
  - Auxiliary API, 176
- CPU\_ZERO
  - Auxiliary API, 176
- cache.h, 296
- calloc
  - Auxiliary API, 207
- ceil
  - Math library API, 142
- ceilf
  - Math library API, 143
- clock\_gettime
  - Auxiliary API, 208
- close
  - Auxiliary API, 208
  - fops, 52
  - TEE\_iSocket\_s, 274

- connect
  - Socket library API, [159](#)
- constraint\_handler\_t
  - Auxiliary API, [203](#)
- Contiguous memory API, [65](#)
  - TEES\_AllocateContiguousMemory, [66](#)
  - TEES\_CONTIGUOUS\_FLAG\_ZERO\_ON\_ALLOC, [66](#)
  - TEES\_CONTIGUOUS\_FLAG\_ZERO\_ON\_FREE, [66](#)
  - TEES\_CONTIGUOUS\_MAP\_FIXED, [66](#)
  - TEES\_CONTIGUOUS\_MAP\_NON\_CACHEABLE, [66](#)
  - TEES\_CONTIGUOUS\_MAP\_POPULATE, [66](#)
  - TEES\_CONTIGUOUS\_MAP\_READ, [66](#)
  - TEES\_CONTIGUOUS\_MAP\_WRITE, [66](#)
  - TEES\_ContiguousAllocateFlags, [66](#)
  - TEES\_ContiguousMapFlags, [66](#)
  - TEES\_ContiguousMemoryHandle, [65](#)
  - TEES\_MapContiguousMemory, [67](#)
  - TEES\_ReleaseContiguousMemory, [68](#)
  - TEES\_UnmapContiguousMemory, [69](#)
- copysign
  - Math library API, [143](#)
- copysignf
  - Math library API, [143](#)
- core/error.h, [297](#)
- core/mman.h, [299](#)
- cos
  - Math library API, [144](#)
- cosf
  - Math library API, [144](#)
- cpu\_set\_t, [173](#)
- cred\_compare
  - credentials.h, [321](#)
- credentials.h
  - cred\_compare, [321](#)
- ctype.h, [300](#)
- Custom handler API, [64](#)
  
- DECLARE\_BITMAP
  - Auxiliary API, [176](#)
- driver.h, [301](#)
- driver/mem/phys.h, [302](#)
  
- E2BIG
  - Auxiliary API, [176](#)
- EACCES
  - Auxiliary API, [177](#)
- EADDRINUSE
  - Auxiliary API, [177](#)
- EADDRNOTAVAIL
  - Auxiliary API, [177](#)
- EADV
  - Auxiliary API, [177](#)
- EAFNOSUPPORT
  - Auxiliary API, [177](#)
- EAGAIN
  - Auxiliary API, [177](#)
- EALREADY
  - Auxiliary API, [177](#)
- EBADFD
  - Auxiliary API, [178](#)
- EBADMSG
  - Auxiliary API, [178](#)
- EBADRQC
  - Auxiliary API, [178](#)
- EBADSLT
  - Auxiliary API, [178](#)
- EBADE
  - Auxiliary API, [178](#)
- EBADF
  - Auxiliary API, [178](#)
- EBADR
  - Auxiliary API, [178](#)
- EBFONT
  - Auxiliary API, [179](#)
- EBUSY
  - Auxiliary API, [179](#)
- ECANCELED
  - Auxiliary API, [179](#)
- ECHILD
  - Auxiliary API, [179](#)
- ECHRNG
  - Auxiliary API, [179](#)
- ECOMM
  - Auxiliary API, [179](#)
- ECONNABORTED
  - Auxiliary API, [179](#)
- ECONNREFUSED
  - Auxiliary API, [180](#)
- ECONNRESET
  - Auxiliary API, [180](#)
- EDEADLOCK
  - Auxiliary API, [180](#)
- EDEADLK
  - Auxiliary API, [180](#)
- EDESTADDRREQ
  - Auxiliary API, [180](#)
- EDOTDOT
  - Auxiliary API, [180](#)
- EDOM
  - Auxiliary API, [180](#)
- EDQUOT
  - Auxiliary API, [181](#)
- EEXIST
  - Auxiliary API, [181](#)
- EFAULT
  - Auxiliary API, [181](#)
- EFBIG
  - Auxiliary API, [181](#)
- EHOSTDOWN
  - Auxiliary API, [181](#)
- EHOSTUNREACH
  - Auxiliary API, [181](#)
- EIDRM
  - Auxiliary API, [181](#)

- EILSEQ
  - Auxiliary API, [182](#)
- EINPROGRESS
  - Auxiliary API, [182](#)
- EINTR
  - Auxiliary API, [182](#)
- EINVAL
  - Auxiliary API, [182](#)
- EISCONN
  - Auxiliary API, [182](#)
- EISDIR
  - Auxiliary API, [182](#)
- EISNAM
  - Auxiliary API, [183](#)
- EIO
  - Auxiliary API, [182](#)
- EL2HLT
  - Auxiliary API, [183](#)
- EL2NSYNC
  - Auxiliary API, [183](#)
- EL3HLT
  - Auxiliary API, [183](#)
- EL3RST
  - Auxiliary API, [183](#)
- ELIBACC
  - Auxiliary API, [183](#)
- ELIBBAD
  - Auxiliary API, [183](#)
- ELIBEXEC
  - Auxiliary API, [184](#)
- ELIBMAX
  - Auxiliary API, [184](#)
- ELIBSCN
  - Auxiliary API, [184](#)
- ELNRNG
  - Auxiliary API, [184](#)
- ELOOP
  - Auxiliary API, [184](#)
- EMFILE
  - Auxiliary API, [184](#)
- EMLINK
  - Auxiliary API, [184](#)
- EMSGSIZE
  - Auxiliary API, [185](#)
- EMULTIHOP
  - Auxiliary API, [185](#)
- ENAMETOOLONG
  - Auxiliary API, [185](#)
- ENAVAIL
  - Auxiliary API, [185](#)
- ENETDOWN
  - Auxiliary API, [185](#)
- ENETRESET
  - Auxiliary API, [185](#)
- ENETUNREACH
  - Auxiliary API, [185](#)
- ENFILE
  - Auxiliary API, [186](#)
- ENOANO
  - Auxiliary API, [186](#)
- ENOBUFFS
  - Auxiliary API, [186](#)
- ENOCSI
  - Auxiliary API, [186](#)
- ENODATA
  - Auxiliary API, [186](#)
- ENODEV
  - Auxiliary API, [186](#)
- ENOENT
  - Auxiliary API, [186](#)
- ENOEXEC
  - Auxiliary API, [187](#)
- ENOKEY
  - Auxiliary API, [187](#)
- ENOLCK
  - Auxiliary API, [187](#)
- ENOLINK
  - Auxiliary API, [187](#)
- ENOMEM
  - Auxiliary API, [187](#)
- ENOMSG
  - Auxiliary API, [187](#)
- ENONET
  - Auxiliary API, [187](#)
- ENOPKG
  - Auxiliary API, [188](#)
- ENOPROTOOPT
  - Auxiliary API, [188](#)
- ENOSPC
  - Auxiliary API, [188](#)
- ENOSTR
  - Auxiliary API, [188](#)
- ENOSYS
  - Auxiliary API, [188](#)
- ENOSR
  - Auxiliary API, [188](#)
- ENOTBLK
  - Auxiliary API, [188](#)
- ENOTCONN
  - Auxiliary API, [189](#)
- ENOTDIR
  - Auxiliary API, [189](#)
- ENOTEMPTY
  - Auxiliary API, [189](#)
- ENOTNAM
  - Auxiliary API, [189](#)
- ENOTSOCK
  - Auxiliary API, [189](#)
- ENOTTY
  - Auxiliary API, [189](#)
- ENOTUNIQ
  - Auxiliary API, [189](#)
- ENXIO
  - Auxiliary API, [190](#)
- EOPNOTSUPP
  - Auxiliary API, [190](#)

- EOVERFLOW
  - Auxiliary API, [190](#)
- EOF
  - Auxiliary API, [190](#)
- EPERM
  - Auxiliary API, [190](#)
- EPFNOSUPPORT
  - Auxiliary API, [190](#)
- EPIPE
  - Auxiliary API, [190](#)
- EPROTONOSUPPORT
  - Auxiliary API, [191](#)
- EPROTOTYPE
  - Auxiliary API, [191](#)
- EPROTO
  - Auxiliary API, [191](#)
- ERANGE
  - Auxiliary API, [191](#)
- EREMCHG
  - Auxiliary API, [191](#)
- EREMOTEIO
  - Auxiliary API, [191](#)
- EREMOTE
  - Auxiliary API, [191](#)
- ERESTART
  - Auxiliary API, [192](#)
- EROFS
  - Auxiliary API, [192](#)
- ESHUTDOWN
  - Auxiliary API, [192](#)
- ESOCKTNOSUPPORT
  - Auxiliary API, [192](#)
- ESPIPE
  - Auxiliary API, [192](#)
- ESRCH
  - Auxiliary API, [192](#)
- ESRMNT
  - Auxiliary API, [193](#)
- ESTALE
  - Auxiliary API, [193](#)
- ESTRPIPE
  - Auxiliary API, [193](#)
- ETIMEDOUT
  - Auxiliary API, [193](#)
- ETIME
  - Auxiliary API, [193](#)
- ETOOMANYREFS
  - Auxiliary API, [193](#)
- ETXTBSY
  - Auxiliary API, [193](#)
- EUCLEAN
  - Auxiliary API, [194](#)
- EUNATCH
  - Auxiliary API, [194](#)
- EUSERS
  - Auxiliary API, [194](#)
- EWOLDBLOCK
  - Auxiliary API, [194](#)
- EXDEV
  - Auxiliary API, [194](#)
- EXFULL
  - Auxiliary API, [194](#)
- errno
  - Auxiliary API, [192](#)
- errno.h, [302](#)
- errno\_t
  - Auxiliary API, [204](#)
- errno\_to\_tee\_error
  - Samsung Internal API, [43](#)
- error
  - TEE\_iSocket\_s, [274](#)
- exit
  - Auxiliary API, [209](#)
- exp
  - Math library API, [144](#)
- expf
  - Math library API, [145](#)
- fabs
  - Math library API, [145](#)
- fabsf
  - Math library API, [145](#)
- fcntl.h, [302](#)
- fflush
  - Auxiliary API, [209](#)
- floor
  - Math library API, [146](#)
- floorf
  - Math library API, [146](#)
- fmax
  - Math library API, [146](#)
- fmaxf
  - Math library API, [147](#)
- fmin
  - Math library API, [147](#)
- fminf
  - Math library API, [147](#)
- fops, [52](#)
  - close, [52](#)
  - ioctl, [52](#)
  - lseek, [52](#)
  - mmap, [52](#)
  - open, [52](#)
  - read, [52](#)
  - stat, [53](#)
  - truncate, [53](#)
  - unlink, [53](#)
  - write, [53](#)
- fprintf
  - Auxiliary API, [209](#)
- free
  - Auxiliary API, [210](#)
- fstat
  - Auxiliary API, [210](#)
- ftruncate
  - Auxiliary API, [211](#)

- get\_errno\_addr
  - Auxiliary API, [211](#)
- getcluster
  - Auxiliary API, [211](#)
- getcpu
  - Auxiliary API, [212](#)
- gettimer
  - Auxiliary API, [212](#)
- getpid
  - Auxiliary API, [212](#)
- getsockopt
  - Socket library API, [159](#)
- gettid
  - Auxiliary API, [213](#)
- gid\_t
  - Auxiliary API, [204](#)
- hypot
  - Math library API, [148](#)
- hypotf
  - Math library API, [148](#)
- I2C API, [80](#)
  - TEES\_I2CExit, [81](#)
  - TEES\_I2CHandler, [81](#)
  - TEES\_I2CInit, [81](#)
  - TEES\_I2CRead, [81](#)
  - TEES\_I2CWrite, [82](#)
  - TEES\_I2CWriteRead, [82](#)
- INT\_MAX
  - Auxiliary API, [194](#)
- INT\_MIN
  - Auxiliary API, [195](#)
- IRS\_ADD\_FLAG\_CMD
  - Integrity Report System API, [106](#)
- IRS\_DEL\_FLAG\_CMD
  - Integrity Report System API, [106](#)
- IRS\_DENY\_DELETE\_FROM\_SMC\_TZ
  - Integrity Report System API, [104](#)
- IRS\_DENY\_READ\_FROM\_SMC\_TZ
  - Integrity Report System API, [104](#)
- IRS\_DENY\_WRITE\_FROM\_SMC\_TZ
  - Integrity Report System API, [104](#)
- IRS\_FAIL\_TZ
  - Integrity Report System API, [104](#)
- IRS\_GET\_FLAG\_VALUE\_CMD
  - Integrity Report System API, [106](#)
- IRS\_INC\_FLAG\_CMD
  - Integrity Report System API, [106](#)
- IRS\_INCORRECT\_CHECKSUM\_TZ
  - Integrity Report System API, [104](#)
- IRS\_INCORRECT\_FLAG\_TYPE\_TZ
  - Integrity Report System API, [104](#)
- IRS\_INCORRECT\_RT\_ID\_TZ
  - Integrity Report System API, [104](#)
- IRS\_INTERNAL\_CMD
  - Integrity Report System API, [106](#)
- IRS\_MAX\_VAL\_COUNTER\_RT\_TZ
  - Integrity Report System API, [104](#)
- IRS\_MAX\_VAL\_COUNTER\_TZ
  - Integrity Report System API, [106](#)
- IRS\_NWD\_CTRL
  - Integrity Report System API, [106](#)
- IRS\_NWD\_RD
  - Integrity Report System API, [106](#)
- IRS\_NWD\_WR
  - Integrity Report System API, [106](#)
- IRS\_PARAM
  - Integrity Report System API, [106](#)
- IRS\_RT\_FLAGS\_EMPTY\_TZ
  - Integrity Report System API, [105](#)
- IRS\_RT\_FLAGS\_FULL\_TZ
  - Integrity Report System API, [105](#)
- IRS\_SET\_FLAG\_CMD
  - Integrity Report System API, [106](#)
- IRS\_SET\_FLAG\_VALUE\_CMD
  - Integrity Report System API, [106](#)
- IRS\_SUCCESS\_TZ
  - Integrity Report System API, [105](#)
- IRS\_SWD\_CTRL
  - Integrity Report System API, [106](#)
- IRS\_SWD\_RD
  - Integrity Report System API, [106](#)
- IRS\_SWD\_WR
  - Integrity Report System API, [106](#)
- IRS\_TYPE\_BOOLEAN
  - Integrity Report System API, [106](#)
- IRS\_TYPE\_COUNTER
  - Integrity Report System API, [106](#)
- IRS\_TYPE\_VALUE
  - Integrity Report System API, [106](#)
- IRS\_UNKNOWN\_ERROR\_TZ
  - Integrity Report System API, [105](#)
- IRS\_UNKNOWN\_ID\_TZ
  - Integrity Report System API, [105](#)
- IRS\_UNKNOWN\_INT\_CMD\_TZ
  - Integrity Report System API, [105](#)
- ignore\_handler\_s
  - Auxiliary API, [213](#)
- Integrity Report System API, [103](#)
  - IRS\_ADD\_FLAG\_CMD, [106](#)
  - IRS\_DEL\_FLAG\_CMD, [106](#)
  - IRS\_DENY\_DELETE\_FROM\_SMC\_TZ, [104](#)
  - IRS\_DENY\_READ\_FROM\_SMC\_TZ, [104](#)
  - IRS\_DENY\_WRITE\_FROM\_SMC\_TZ, [104](#)
  - IRS\_FAIL\_TZ, [104](#)
  - IRS\_GET\_FLAG\_VALUE\_CMD, [106](#)
  - IRS\_INC\_FLAG\_CMD, [106](#)
  - IRS\_INCORRECT\_CHECKSUM\_TZ, [104](#)
  - IRS\_INCORRECT\_FLAG\_TYPE\_TZ, [104](#)
  - IRS\_INCORRECT\_RT\_ID\_TZ, [104](#)
  - IRS\_INTERNAL\_CMD, [106](#)
  - IRS\_MAX\_VAL\_COUNTER\_RT\_TZ, [104](#)
  - IRS\_MAX\_VAL\_COUNTER\_TZ, [105](#)
  - IRS\_NWD\_CTRL, [106](#)
  - IRS\_NWD\_RD, [106](#)
  - IRS\_NWD\_WR, [106](#)

- IRS\_PARAM, 106
- IRS\_RT\_FLAGS\_EMPTY\_TZ, 105
- IRS\_RT\_FLAGS\_FULL\_TZ, 105
- IRS\_SET\_FLAG\_CMD, 106
- IRS\_SET\_FLAG\_VALUE\_CMD, 106
- IRS\_SUCCESS\_TZ, 105
- IRS\_SWD\_CTRL, 106
- IRS\_SWD\_RD, 106
- IRS\_SWD\_WR, 106
- IRS\_TYPE\_BOOLEAN, 106
- IRS\_TYPE\_COUNTER, 106
- IRS\_TYPE\_VALUE, 106
- IRS\_UNKNOWN\_ERROR\_TZ, 105
- IRS\_UNKNOWN\_ID\_TZ, 105
- IRS\_UNKNOWN\_INT\_CMD\_TZ, 105
- TEES\_AddIrsFlag, 107
- TEES\_DelIrsFlag, 107
- TEES\_GetIrsFlagValue, 107
- TEES\_IncIrsFlag, 108
- TEES\_SetIrsFlag, 108
- TEES\_SetIrsFlagValue, 108
- inttypes.h, 303
- invoke\_constraint\_handler\_s
  - Auxiliary API, 213
- ioctl
  - Auxiliary API, 213
  - fops, 52
  - TEE\_iSocket\_s, 274
- irs.h, 304
- isalnum
  - Auxiliary API, 214
- isalpha
  - Auxiliary API, 214
- isascii
  - Auxiliary API, 215
- isblank
  - Auxiliary API, 215
- iscntrl
  - Auxiliary API, 215
- isdigit
  - Auxiliary API, 216
- isgraph
  - Auxiliary API, 216
- islower
  - Auxiliary API, 216
- isnan
  - Math library API, 140
- isprint
  - Auxiliary API, 217
- ispunct
  - Auxiliary API, 217
- isspace
  - Auxiliary API, 217
- isupper
  - Auxiliary API, 218
- isxdigit
  - Auxiliary API, 218
- KM\_KW\_MAX\_AAD\_LEN
  - Samsung Internal API, 41
- KM\_KW\_MAX\_INPUT\_LEN
  - Samsung Internal API, 41
- KM\_KW\_MAX\_IV\_LEN
  - Samsung Internal API, 41
- KM\_KW\_MAX\_KEY\_LEN
  - Samsung Internal API, 42
- KM\_KW\_MAX\_SALT\_LEN
  - Samsung Internal API, 42
- KM\_KW\_MAX\_TAG\_LEN
  - Samsung Internal API, 42
- LLONG\_MAX
  - Auxiliary API, 195
- LLONG\_MIN
  - Auxiliary API, 195
- LONG\_MAX
  - Auxiliary API, 195
- LONG\_MIN
  - Auxiliary API, 195
- limits.h, 305
- listen
  - Socket library API, 159
- Loadable driver API, 51
  - TEES\_AcquireUserBuffer, 54
  - TEES\_AllocateInterrupt, 54
  - TEES\_CompleteInterrupt, 55
  - TEES\_CompleteRequest, 56
  - TEES\_DcacheClean, 57
  - TEES\_DcacheFlush, 57
  - TEES\_DriverDestructor\_t, 53
  - TEES\_FiniDriver, 57
  - TEES\_GenerateInterrupt, 58
  - TEES\_InitDriver, 59
  - TEES\_InterruptHandle, 53
  - TEES\_RegisterDriver, 60
  - TEES\_RegisterDriverDestructor, 60
  - TEES\_ReleaseDriver, 61
  - TEES\_ReleaseInterrupt, 61
  - TEES\_ReleaseUserBuffer, 62
  - TEES\_WaitForInterrupt, 63
- log
  - Math library API, 148
- logb
  - Math library API, 149
- logbf
  - Math library API, 149
- logf
  - Math library API, 149
- lseek
  - Auxiliary API, 218
  - fops, 52
- M\_CACHE\_PAGES
  - Auxiliary API, 195
- M\_PI\_2
  - Math library API, 141
- M\_PI\_4
  - Math library API, 141

- M\_PI
  - Math library API, 141
- M\_E
  - Math library API, 141
- MAP\_ANONYMOUS
  - Auxiliary API, 195
- MAP\_FAILED
  - Auxiliary API, 196
- MAP\_FIXED
  - Auxiliary API, 196
- MAP\_PHYS\_NON\_CACHED
  - Auxiliary API, 196
- MAP\_PHYS\_NON\_SECURE
  - Auxiliary API, 196
- MAP\_POPULATE
  - Auxiliary API, 196
- MAP\_PRIVATE
  - Auxiliary API, 196
- MAP\_SHARED
  - Auxiliary API, 196
- MAX\_CPU
  - Auxiliary API, 197
- MAX\_FONT\_SIZE
  - Trusted user interface, 90
- MIN\_FONT\_SIZE
  - Trusted user interface, 90
- MQ\_MAX\_NAME
  - Message queue library API, 155
- MUTEX\_STATE\_LOCKED
  - Thread support library API, 118
- MUTEX\_STATE\_UNLOCKED
  - Thread support library API, 118
- malloc
  - Auxiliary API, 219
- malloc.h, 306
- Math library API, 139
  - atan, 141
  - atan2, 141
  - atan2f, 142
  - atanf, 142
  - ceil, 142
  - ceilf, 143
  - copysign, 143
  - copysignf, 143
  - cos, 144
  - cosf, 144
  - exp, 144
  - expf, 145
  - fabs, 145
  - fabsf, 145
  - floor, 146
  - floorf, 146
  - fmax, 146
  - fmaxf, 147
  - fmin, 147
  - fminf, 147
  - hypot, 148
  - hypotf, 148
  - isnan, 140
  - log, 148
  - logb, 149
  - logbf, 149
  - logf, 149
  - M\_PI\_2, 141
  - M\_PI\_4, 141
  - M\_PI, 141
  - M\_E, 141
  - modf, 150
  - modff, 150
  - pow, 150
  - powf, 151
  - round, 151
  - roundf, 151
  - scalbn, 152
  - scalbnf, 152
  - sin, 152
  - sinf, 153
  - sqrt, 153
  - sqrtf, 153
- math.h, 306
- memchr
  - Auxiliary API, 219
- memcpy
  - Auxiliary API, 220
- memcpy
  - Auxiliary API, 220
- memcpy\_s
  - Auxiliary API, 221
- memmove
  - Auxiliary API, 221
- memmove\_s
  - Auxiliary API, 221
- memset
  - Auxiliary API, 222
- memset\_s
  - Auxiliary API, 222
- Message queue library API, 155
  - MQ\_MAX\_NAME, 155
  - mq\_close, 156
  - mq\_open, 156
  - mq\_receive, 156
  - mq\_send, 156
  - mq\_unlink, 157
  - mqd\_t, 155
- Miscellaneous extensions, 110
  - TEES\_DuplWshmem, 110
  - TEES\_GetTaskDataSize, 110
  - TEES\_IsPhysMemoryProtected, 111
  - TEES\_IsREESharedMemory, 111
- mmap
  - Auxiliary API, 223
- fops, 52
- mode\_t
  - Auxiliary API, 204
- modf
  - Math library API, 150

- modff
  - Math library API, [150](#)
- mq\_close
  - Message queue library API, [156](#)
- mq\_open
  - Message queue library API, [156](#)
- mq\_receive
  - Message queue library API, [156](#)
- mq\_send
  - Message queue library API, [156](#)
- mq\_unlink
  - Message queue library API, [157](#)
- mqd\_t
  - Message queue library API, [155](#)
- mqueue.h, [308](#)
- ms\_to\_timespec
  - Auxiliary API, [224](#)
- munmap
  - Auxiliary API, [225](#)
- NUM\_MILLIS\_IN\_SEC
  - Auxiliary API, [197](#)
- NUM\_NANOS\_IN\_MILLI
  - Auxiliary API, [197](#)
- NUM\_NANOS\_IN\_SEC
  - Auxiliary API, [197](#)
- NUM\_NANOS\_IN\_USEC
  - Auxiliary API, [197](#)
- NUM\_SECONDS\_IN\_MIN
  - Auxiliary API, [197](#)
- nanosleep
  - Auxiliary API, [225](#)
- off\_t
  - Auxiliary API, [204](#)
- open
  - Auxiliary API, [226](#)
  - fops, [52](#)
  - TEE\_socket\_s, [274](#)
- PGOFF\_SHIFT
  - Auxiliary API, [197](#)
- PHYS\_DEV\_NAME\_LEN
  - Auxiliary API, [198](#)
- PHYS\_DEV\_NAME
  - Auxiliary API, [198](#)
- PRId16
  - Auxiliary API, [198](#)
- PRId32
  - Auxiliary API, [198](#)
- PRId64
  - Auxiliary API, [198](#)
- PRlu16
  - Auxiliary API, [198](#)
- PRlu32
  - Auxiliary API, [198](#)
- PRlu64
  - Auxiliary API, [199](#)
- PRlx16
  - Auxiliary API, [199](#)
- PRlx32
  - Auxiliary API, [199](#)
- PRlx64
  - Auxiliary API, [199](#)
- PROT\_EXEC
  - Auxiliary API, [199](#)
- PROT\_NONE
  - Auxiliary API, [199](#)
- PROT\_READ
  - Auxiliary API, [199](#)
- PROT\_WRITE
  - Auxiliary API, [200](#)
- PTHREAD\_ERRORCHECK\_MUTEX\_INITIALIZER\_NP
  - Thread support library API, [118](#)
- PTHREAD\_KEYS\_MAX
  - Auxiliary API, [200](#)
- PTHREAD\_MUTEX\_DEFAULT
  - Thread support library API, [121](#)
- PTHREAD\_MUTEX\_DESTROYED
  - Thread support library API, [121](#)
- PTHREAD\_MUTEX\_ERRORCHECK\_NP
  - Thread support library API, [121](#)
- PTHREAD\_MUTEX\_ERRORCHECK
  - Thread support library API, [121](#)
- PTHREAD\_MUTEX\_INITIALIZER
  - Thread support library API, [118](#)
- PTHREAD\_MUTEX\_NORMAL
  - Thread support library API, [121](#)
- PTHREAD\_MUTEX\_RECURSIVE\_NP
  - Thread support library API, [121](#)
- PTHREAD\_MUTEX\_RECURSIVE
  - Thread support library API, [121](#)
- PTHREAD\_ONCE\_INIT
  - Thread support library API, [118](#)
- PTHREAD\_RECURSIVE\_MUTEX\_INITIALIZER\_NP
  - Thread support library API, [118](#)
- PTHREAD\_STACK\_MIN
  - Thread support library API, [119](#)
- pid\_t
  - Auxiliary API, [204](#)
- pow
  - Math library API, [150](#)
- powf
  - Math library API, [151](#)
- print\_no\_alloc.h, [309](#)
- printf
  - Auxiliary API, [227](#)
- printf\_no\_alloc
  - Auxiliary API, [227](#)
- printf\_s
  - Auxiliary API, [228](#)
- profil
  - Auxiliary API, [228](#)
- protocol\_error\_code
  - Tee\_sockets, [283](#)
- protocol\_errors.h, [309](#)
- protocolID

- TEE\_iSocket\_s, [274](#)
- pthread.h, [311](#)
- pthread\_attr\_destroy
  - Thread support library API, [121](#)
- pthread\_attr\_init
  - Thread support library API, [121](#)
- pthread\_attr\_setstacksize
  - Thread support library API, [122](#)
- pthread\_attr\_t
  - Thread support library API, [119](#)
- pthread\_cond\_broadcast
  - Thread support library API, [122](#)
- pthread\_cond\_destroy
  - Thread support library API, [123](#)
- pthread\_cond\_init
  - Thread support library API, [124](#)
- pthread\_cond\_signal
  - Thread support library API, [124](#)
- pthread\_cond\_t
  - Thread support library API, [119](#)
- pthread\_cond\_wait
  - Thread support library API, [125](#)
- pthread\_condattr\_t
  - Thread support library API, [120](#)
- pthread\_create
  - Thread support library API, [126](#)
- pthread\_getspecific
  - Thread support library API, [127](#)
- pthread\_impl\_t
  - Thread support library API, [120](#)
- pthread\_join
  - Thread support library API, [127](#)
- pthread\_key\_create
  - Thread support library API, [128](#)
- pthread\_key\_delete
  - Thread support library API, [129](#)
- pthread\_key\_t
  - Thread support library API, [120](#)
- pthread\_kill
  - Thread support library API, [130](#)
- pthread\_mutex\_destroy
  - Thread support library API, [130](#)
- pthread\_mutex\_init
  - Thread support library API, [131](#)
- pthread\_mutex\_lock
  - Thread support library API, [132](#)
- pthread\_mutex\_t
  - Thread support library API, [120](#)
- pthread\_mutex\_trylock
  - Thread support library API, [132](#)
- pthread\_mutex\_unlock
  - Thread support library API, [133](#)
- pthread\_mutexattr\_destroy
  - Thread support library API, [134](#)
- pthread\_mutexattr\_gettype
  - Thread support library API, [134](#)
- pthread\_mutexattr\_init
  - Thread support library API, [135](#)
- pthread\_mutexattr\_settype
  - Thread support library API, [135](#)
- pthread\_mutexattr\_t
  - Thread support library API, [120](#)
- pthread\_once
  - Thread support library API, [136](#)
- pthread\_once\_t
  - Thread support library API, [120](#)
- pthread\_self
  - Thread support library API, [137](#)
- pthread\_setspecific
  - Thread support library API, [137](#)
- pthread\_sigmask
  - Thread support library API, [119](#)
- pthread\_t
  - Thread support library API, [120](#)
- putchar
  - Auxiliary API, [229](#)
- puts
  - Auxiliary API, [229](#)
- qsort
  - Auxiliary API, [230](#)
- qsort\_r
  - Auxiliary API, [230](#)
- RPMB API, [113](#)
  - TEES\_RPMBCheckEnable, [113](#)
  - TEES\_RPMBRead, [113](#)
  - TEES\_RPMBWrite, [114](#)
- read
  - Auxiliary API, [230](#)
  - fops, [52](#)
- realloc
  - Auxiliary API, [231](#)
- recv
  - Socket library API, [160](#)
  - TEE\_iSocket\_s, [274](#)
- recvmsg
  - Socket library API, [160](#)
- rot\_t, [40](#)
- round
  - Math library API, [151](#)
- roundf
  - Math library API, [151](#)
- rpm.h, [313](#)
- SCHAR\_MAX
  - Auxiliary API, [200](#)
- SCHAR\_MIN
  - Auxiliary API, [200](#)
- SCNd16
  - Auxiliary API, [200](#)
- SCNd32
  - Auxiliary API, [200](#)
- SCNd64
  - Auxiliary API, [200](#)
- SCNu16
  - Auxiliary API, [201](#)

- SCNu32
  - Auxiliary API, 201
- SCNu64
  - Auxiliary API, 201
- SCNx16
  - Auxiliary API, 201
- SCNx32
  - Auxiliary API, 201
- SCNx64
  - Auxiliary API, 201
- SHRT\_MAX
  - Auxiliary API, 201
- SHRT\_MIN
  - Auxiliary API, 202
- SO\_AC\_LEN
  - Samsung Internal API, 42
- SO\_AUTH\_ID\_LEN
  - Samsung Internal API, 42
- SO\_HEADER\_SIZE
  - Samsung Internal API, 42
- SO\_IV\_LEN
  - Samsung Internal API, 42
- SO\_MAGIC\_NUMBER\_LEN
  - Samsung Internal API, 43
- SO\_OUT\_BUF\_SIZE
  - Samsung Internal API, 43
- SO\_TA\_ID\_LEN
  - Samsung Internal API, 43
- SO\_TAG\_LEN
  - Samsung Internal API, 43
- SPI API, 70
  - TEES\_SPIClockDisable, 71
  - TEES\_SPIClockEnable, 71
  - TEES\_SPIDMAInit, 72
  - TEES\_SPIExit, 72
  - TEES\_SPIHandler, 71
  - TEES\_SPIInit, 73
  - TEES\_SPIRead, 74
  - TEES\_SPISetBitsPerWord, 74
  - TEES\_SPISetCPHA, 76
  - TEES\_SPISetCPOL, 76
  - TEES\_SPISetClockSpeed, 75
  - TEES\_SPISetConfig, 75
  - TEES\_SPISetDMAMode, 77
  - TEES\_SPIWrite, 77
  - TEES\_SPIWriteRead, 78
- Samsung Internal API, 39
  - errno\_to\_tee\_error, 43
  - KM\_KW\_MAX\_AAD\_LEN, 41
  - KM\_KW\_MAX\_INPUT\_LEN, 41
  - KM\_KW\_MAX\_IV\_LEN, 41
  - KM\_KW\_MAX\_KEY\_LEN, 42
  - KM\_KW\_MAX\_SALT\_LEN, 42
  - KM\_KW\_MAX\_TAG\_LEN, 42
  - SO\_AC\_LEN, 42
  - SO\_AUTH\_ID\_LEN, 42
  - SO\_HEADER\_SIZE, 42
  - SO\_IV\_LEN, 42
  - SO\_MAGIC\_NUMBER\_LEN, 43
  - SO\_OUT\_BUF\_SIZE, 43
  - SO\_TA\_ID\_LEN, 43
  - SO\_TAG\_LEN, 43
  - TEES\_CheckSecureObjectCreator, 43
  - TEES\_DeriveKeyKDF, 44
  - TEES\_DeriveKeySetKDF, 45
  - TEES\_EnterCritical, 45
  - TEES\_ExitCritical, 46
  - TEES\_GetRoT, 46
  - TEES\_LockHWCryptoBuf, 46
  - TEES\_SECCAM\_GetStatus, 47
  - TEES\_UnlockHWCryptoBuf, 48
  - TEES\_UnwrapSecureObject, 48
  - TEES\_WrapSecureObject, 49
  - TEES\_WrappedWithREK, 49
- scBaseKeyHandle
  - \_\_TEE\_SC\_DeviceKeyRef.\_\_unnamed\_\_, 291
- scCardKeyRef
  - \_\_TEE\_SC\_Params, 293
- scDeviceKeyRef
  - \_\_TEE\_SC\_Params, 293
- scKeyEncHandle
  - \_\_TEE\_SC\_KeySetRef, 292
- scKeyID
  - \_\_TEE\_SC\_CardKeyRef, 290
- scKeyMacHandle
  - \_\_TEE\_SC\_KeySetRef, 292
- scKeySetRef
  - \_\_TEE\_SC\_DeviceKeyRef.\_\_unnamed\_\_, 291
- scKeyType
  - \_\_TEE\_SC\_DeviceKeyRef, 291
- scKeyVersion
  - \_\_TEE\_SC\_CardKeyRef, 290
- scOID
  - \_\_TEE\_SC\_Params, 293
- scSecurityLevel
  - \_\_TEE\_SC\_Params, 293
- scType
  - \_\_TEE\_SC\_Params, 293
- scalbn
  - Math library API, 152
- scalbnf
  - Math library API, 152
- sched.h, 314
- sched\_getaffinity
  - Auxiliary API, 231
- sched\_setaffinity
  - Auxiliary API, 232
- sched\_yield
  - Auxiliary API, 232
- scma.h, 314
- sePresent
  - \_\_TEE\_SEReaderProperties, 294
- selectResponseEnable
  - \_\_TEE\_SEReaderProperties, 294
- send
  - Socket library API, 160

- TEE\_iSocket\_s, 275
- sendmsg
  - Socket library API, 161
- set\_constraint\_handler\_s
  - Auxiliary API, 232
- setitimer
  - Auxiliary API, 233
- setsockopt
  - Socket library API, 161
- sin
  - Math library API, 152
- sinf
  - Math library API, 153
- smc\_data, 294
  - arg\_types, 295
  - args, 295
- snprintf
  - Auxiliary API, 233
- snprintf\_s
  - Auxiliary API, 234
- socket
  - Socket library API, 162
- Socket library API, 158
  - accept, 158
  - bind, 158
  - connect, 159
  - getsockopt, 159
  - listen, 159
  - recv, 160
  - recvmsg, 160
  - send, 160
  - sendmsg, 161
  - setsockopt, 161
  - socket, 162
  - socketpair, 162
- socketpair
  - Socket library API, 162
- sprintf
  - Auxiliary API, 234
- sprintf\_s
  - Auxiliary API, 235
- sqrt
  - Math library API, 153
- sqrtf
  - Math library API, 153
- sscanf
  - Auxiliary API, 235
- sscanf\_s
  - Auxiliary API, 236
- ssize\_t
  - Auxiliary API, 204
- st\_gid
  - stat, 295
- st\_mode
  - stat, 295
- st\_size
  - stat, 295
- st\_uid
  - stat, 295
- stat, 295
  - Auxiliary API, 236
  - fops, 53
  - st\_gid, 295
  - st\_mode, 295
  - st\_size, 295
  - st\_uid, 295
- stdin
  - Auxiliary API, 265
- stdio.h, 315
- stdlib.h, 317
- strcat
  - Auxiliary API, 237
- strcat\_s
  - Auxiliary API, 237
- strchr
  - Auxiliary API, 238
- strchrnul
  - Auxiliary API, 238
- strcmp
  - Auxiliary API, 239
- strcpy
  - Auxiliary API, 239
- strcpy\_s
  - Auxiliary API, 240
- strcspn
  - Auxiliary API, 240
- strdup
  - Auxiliary API, 241
- strerror
  - Auxiliary API, 241
- strerror\_r
  - Auxiliary API, 241
- strerror\_s
  - Auxiliary API, 242
- string.h, 318
- strlcat
  - Auxiliary API, 242
- strlen
  - Auxiliary API, 243
- strncat
  - Auxiliary API, 243
- strncat\_s
  - Auxiliary API, 244
- strncmp
  - Auxiliary API, 244
- strncpy
  - Auxiliary API, 245
- strncpy\_s
  - Auxiliary API, 246
- strnlen
  - Auxiliary API, 246
- strnlen\_s
  - Auxiliary API, 247
- strrchr
  - Auxiliary API, 247
- strspn

- Auxiliary API, [248](#)
- strchr
  - Auxiliary API, [248](#)
- strtod
  - Auxiliary API, [249](#)
- strtof
  - Auxiliary API, [249](#)
- strtok
  - Auxiliary API, [250](#)
- strtok\_r
  - Auxiliary API, [251](#)
- strtol
  - Auxiliary API, [251](#)
- strtold
  - Auxiliary API, [252](#)
- strtoll
  - Auxiliary API, [252](#)
- strtoul
  - Auxiliary API, [253](#)
- strtoull
  - Auxiliary API, [253](#)
- sys/credentials.h, [320](#)
- sys/ioctl.h, [321](#)
- sys/mman.h, [299](#)
- sys/socket.h, [321](#)
- sys/types.h, [322](#)
- sys/un.h, [323](#)
- TEE\_IP\_VERSION\_4
  - Tee\_sockets, [286](#)
- TEE\_IP\_VERSION\_6
  - Tee\_sockets, [286](#)
- TEE\_IP\_VERSION\_DC
  - Tee\_sockets, [286](#)
- TEE\_ISOCKET\_ERROR\_HOSTNAME
  - Tee\_sockets, [281](#)
- TEE\_ISOCKET\_ERROR\_LARGE\_BUFFER
  - Tee\_sockets, [281](#)
- TEE\_ISOCKET\_ERROR\_OUT\_OF\_RESOURCES
  - Tee\_sockets, [281](#)
- TEE\_ISOCKET\_ERROR\_PROTOCOL
  - Tee\_sockets, [281](#)
- TEE\_ISOCKET\_ERROR\_REMOTE\_CLOSED
  - Tee\_sockets, [281](#)
- TEE\_ISOCKET\_ERROR\_TIMEOUT
  - Tee\_sockets, [281](#)
- TEE\_ISOCKET\_IWC\_ERROR\_CHANNEL
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_IWC\_ERROR\_INVALID\_VERSION
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_IWC\_ERROR\_NOT\_IMPLEMENTED
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_IWC\_ERROR\_SWD\_CLIENT\_AUTH\_FAILED
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_IWC\_ERROR\_TIMEOUT
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_BAD\_PARAMETERS
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_BUFFER\_TOO\_SMALL
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_COMMUNICATION
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_CONNECTION\_REFUSED
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_DATA\_REMAIN
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_GENERIC
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_HOSTNAME\_NOTRESOLVED
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_HOSTNAME\_TRYAGAIN
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_HOSTNAME\_UNKNOWN
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_LARGE\_BUFFER
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_NET\_UNREACHABLE
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_OUT\_OF\_MEMORY
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_OUT\_OF\_RESOURCES
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_REMOTE\_CLOSED
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_NET\_ERROR\_TIMEOUT
  - Tee\_sockets, [284](#)
- TEE\_ISOCKET\_PROTOCOLID\_TCP
  - Tee\_sockets, [282](#)
- TEE\_ISOCKET\_PROTOCOLID\_TLS
  - Tee\_sockets, [283](#)
- TEE\_ISOCKET\_PROTOCOLID\_UDP
  - Tee\_sockets, [282](#)
- TEE\_ISOCKET\_SERVER\_NAME\_MAX\_LENGTH
  - Tee\_sockets, [281](#)
- TEE\_ISOCKET\_TCP\_API\_VERSION
  - Tee\_sockets, [281](#)
- TEE\_ISOCKET\_TCP\_WARNING\_UNKNOWN\_OUT\_OF\_BAND
  - Tee\_sockets, [282](#)
- TEE\_ISOCKET\_TLS\_ALERT\_ACCESS\_DENIED
  - Tee\_sockets, [285](#)
- TEE\_ISOCKET\_TLS\_ALERT\_BAD\_CERT\_HASH\_VALUE
  - Tee\_sockets, [285](#)
- TEE\_ISOCKET\_TLS\_ALERT\_BAD\_CERT\_STATUS\_RESPONSE
  - Tee\_sockets, [285](#)
- TEE\_ISOCKET\_TLS\_ALERT\_BAD\_CERTIFICATE
  - Tee\_sockets, [285](#)
- TEE\_ISOCKET\_TLS\_ALERT\_BAD\_RECORD\_MAC
  - Tee\_sockets, [285](#)
- TEE\_ISOCKET\_TLS\_ALERT\_CERT\_EXPIRED
  - Tee\_sockets, [285](#)
- TEE\_ISOCKET\_TLS\_ALERT\_CERT\_REQUIRED
  - Tee\_sockets, [285](#)
- TEE\_ISOCKET\_TLS\_ALERT\_CERT\_REVOKED
  - Tee\_sockets, [285](#)
- TEE\_ISOCKET\_TLS\_ALERT\_CERT\_UNKNOWN
  - Tee\_sockets, [285](#)
- TEE\_ISOCKET\_TLS\_ALERT\_CERT\_UNOBTAINABLE
  - Tee\_sockets, [285](#)

- Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_CLOSE\_NOTIFY
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_DECODE\_ERROR
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_DECOMP\_FAILED
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_DECRYPT\_ERROR
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_DECRYPT\_FAILED
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_EXPORT\_RESTRICTED
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_HANDSHAKE\_FAILED
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_ILLEGAL\_PARAMETER
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_INAPPROPRIATE\_FALLBACK
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_INSUFFICIENT\_SECURITY
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_INTERNAL\_ERROR
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_MISSING\_EXTENSION
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_NO\_CERTIFICATE
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_NO\_RENEGOTIATION
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_PROTOCOL\_VERSION
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_RECORD\_OVERFLOW
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_UNEXPECTED\_MSG
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_UNKNOWN\_CA
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_UNKNOWN\_PSK\_IDENTITY
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_UNRECOGNIZED\_NAME
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_UNSUPPORTED\_CERT
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_UNSUPPORTED\_EXTENSION
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ALERT\_USER\_CANCELED
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_API\_VERSION
  - Tee\_sockets, 283
- TEE\_ISOCKET\_TLS\_CB\_BAD\_CERT\_CHAIN
  - Tee\_sockets, 286
- TEE\_ISOCKET\_TLS\_CB\_CHECK\_CERT\_CHAIN
  - Tee\_sockets, 286
- TEE\_ISOCKET\_TLS\_CB\_CHECK\_OCSP\_STATUS
  - Tee\_sockets, 286
- TEE\_ISOCKET\_TLS\_CB\_REVOKED\_OCSP\_STATUS
  - Tee\_sockets, 286
- TEE\_ISOCKET\_TLS\_CB\_UNKNOWN\_OCSP\_STATUS
  - Tee\_sockets, 286
- TEE\_ISOCKET\_TLS\_CERT\_KEYUSAGE\_CHECK\_CLIENT
  - Tee\_sockets, 286
- TEE\_ISOCKET\_TLS\_CERT\_NAME\_CHECK\_CLIENT
  - Tee\_sockets, 286
- TEE\_ISOCKET\_TLS\_CERT\_NOTIFY\_CLIENT
  - Tee\_sockets, 286
- TEE\_ISOCKET\_TLS\_ERROR\_ALERT\_PENDING
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ERROR\_AUTHENTICATION
  - Tee\_sockets, 283
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_COMMON\_NAME\_VERIFICATION
  - Tee\_sockets, 284
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_EXPIRED
  - Tee\_sockets, 284
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_IS\_TOO\_LONG
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_PARSING
  - Tee\_sockets, 284
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_REVOKED
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_SIGN\_VERIFICATION
  - Tee\_sockets, 284
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_STATUS\_UNKNOWN
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_UNKNOWN\_CA
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_UNSUPPORTED
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ERROR\_CRL\_PARSING
  - Tee\_sockets, 284
- TEE\_ISOCKET\_TLS\_ERROR\_DATA
  - Tee\_sockets, 283
- TEE\_ISOCKET\_TLS\_ERROR\_ECDHE\_GEN\_KEY
  - Tee\_sockets, 284
- TEE\_ISOCKET\_TLS\_ERROR\_ECDHE\_SERIALIZING
  - Tee\_sockets, 284
- TEE\_ISOCKET\_TLS\_ERROR\_ECDHE\_SHARED\_SECRET
  - Tee\_sockets, 284
- TEE\_ISOCKET\_TLS\_ERROR\_ECDHE\_UNSUPPORTED\_CURVE
  - Tee\_sockets, 284
- TEE\_ISOCKET\_TLS\_ERROR\_HANDSHAKE\_UNEXPECTED\_PARAMETER
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ERROR\_HANDSHAKE
  - Tee\_sockets, 283
- TEE\_ISOCKET\_TLS\_ERROR\_NO\_ALERT\_PRESENT
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ERROR\_REJECTED\_SUITE
  - Tee\_sockets, 283
- TEE\_ISOCKET\_TLS\_ERROR\_UNEXPECTED\_MESSAGE
  - Tee\_sockets, 284
- TEE\_ISOCKET\_TLS\_ERROR\_UNSUPPORTED\_SUITE
  - Tee\_sockets, 283
- TEE\_ISOCKET\_TLS\_ERROR\_USER\_CANCELED
  - Tee\_sockets, 285
- TEE\_ISOCKET\_TLS\_ERROR\_VERSION
  - Tee\_sockets, 283
- TEE\_ISOCKET\_TLS\_OCSP\_CHECK\_ADVISORY
  - Tee\_sockets, 286

- Tee\_sockets, 286
- TEE\_ISOCKET\_TLS\_OCSP\_CHECK\_CLIENT
  - Tee\_sockets, 286
- TEE\_ISOCKET\_TLS\_OCSP\_CHECK\_MANDATORY
  - Tee\_sockets, 286
- TEE\_ISOCKET\_TLS\_OCSP\_STATUS\_REQUEST\_NO
  - Tee\_sockets, 287
- TEE\_ISOCKET\_TLS\_OCSP\_STATUS\_REQUEST
  - Tee\_sockets, 287
- TEE\_ISOCKET\_UDP\_API\_VERSION
  - Tee\_sockets, 282
- TEE\_ISOCKET\_UDP\_WARNING\_UNKNOWN\_OUT\_OF\_BAND
  - Tee\_sockets, 282
- TEE\_ISOCKET\_WARNING\_PROTOCOL
  - Tee\_sockets, 281
- TEE\_TEMPRARY\_CLEAR\_MODE
  - Trusted user interface, 91
- TEE\_TLS\_CLIENT\_CRED\_CSC
  - Tee\_sockets, 286
- TEE\_TLS\_CLIENT\_CRED\_NONE
  - Tee\_sockets, 286
- TEE\_TLS\_CLIENT\_CRED\_PDC
  - Tee\_sockets, 286
- TEE\_TLS\_SERVER\_CRED\_CSC
  - Tee\_sockets, 287
- TEE\_TLS\_SERVER\_CRED\_PDC
  - Tee\_sockets, 287
- TEE\_TLS\_VERSION\_1v2
  - Tee\_sockets, 287
- TEE\_TLS\_VERSION\_ALL
  - Tee\_sockets, 287
- TEE\_TUI\_ALPHANUMERICAL
  - Trusted user interface, 91
- TEE\_TUI\_CANCEL
  - Trusted user interface, 90
- TEE\_TUI\_CLEAR\_MODE
  - Trusted user interface, 91
- TEE\_TUI\_CORRECTION
  - Trusted user interface, 90
- TEE\_TUI\_HIDDEN\_MODE
  - Trusted user interface, 91
- TEE\_TUI\_LANDSCAPE
  - Trusted user interface, 91
- TEE\_TUI\_NEXT
  - Trusted user interface, 90
- TEE\_TUI\_NO\_SOURCE
  - Trusted user interface, 91
- TEE\_TUI\_NUMBER\_BUTTON\_TYPES
  - Trusted user interface, 90
- TEE\_TUI\_NUMERICAL
  - Trusted user interface, 91
- TEE\_TUI\_OBJECT\_SOURCE
  - Trusted user interface, 91
- TEE\_TUI\_OK
  - Trusted user interface, 90
- TEE\_TUI\_PORTRAIT
  - Trusted user interface, 91
- TEE\_TUI\_PREVIOUS
  - Trusted user interface, 90
- TEE\_TUI\_REF\_SOURCE
  - Trusted user interface, 91
- TEE\_TUI\_VALIDATE
  - Trusted user interface, 90
- TEE\_TUIButton, 88
- TEE\_TUIButtonType
  - Trusted user interface, 90
- TEE\_TUICheckTextFormat
  - Trusted user interface, 93
- TEE\_TUICloseSession
  - Trusted user interface, 93
- TEE\_TUIDisplayScreen
  - Trusted user interface, 93
- TEE\_TUIEntryField, 89
- TEE\_TUIEntryFieldMode
  - Trusted user interface, 90
- TEE\_TUIEntryFieldType
  - Trusted user interface, 91
- TEE\_TUIGetScreenInfo
  - Trusted user interface, 95
- TEE\_TUIImage, 86
- TEE\_TUIImage.\_\_unnamed\_\_, 87
- TEE\_TUIImage.\_\_unnamed\_\_.object, 87
- TEE\_TUIImage.\_\_unnamed\_\_.ref, 87
- TEE\_TUIImageSource
  - Trusted user interface, 91
- TEE\_TUIInitSession
  - Trusted user interface, 96
- TEE\_TUIScreenButtonInfo, 88
- TEE\_TUIScreenConfiguration, 88
- TEE\_TUIScreenInfo, 89
- TEE\_TUIScreenLabel, 87
- TEE\_TUIScreenOrientation
  - Trusted user interface, 91
- TEE\_iSocket
  - Tee\_sockets, 280
- TEE\_iSocket\_s, 273
  - close, 274
  - error, 274
  - ioctl, 274
  - open, 274
  - protocolID, 274
  - recv, 274
  - send, 275
  - TEE\_iSocketVersion, 275
- TEE\_iSocketVersion
  - TEE\_iSocket\_s, 275
- TEE\_ipSocket\_ipVersion\_e
  - Tee\_sockets, 285
- TEE\_tcpSocket\_Setup\_s, 275
- TEE\_tlsSocket\_CB\_Data
  - Tee\_sockets, 280
- TEE\_tlsSocket\_CB\_Data\_s, 280
- TEE\_tlsSocket\_CallbackInfo\_s, 277
- TEE\_tlsSocket\_CallbackReasonType\_e
  - Tee\_sockets, 286
- TEE\_tlsSocket\_CertStorageCred\_s, 277

- TEE\_tlsSocket\_ClientCredentialType\_e
  - Tee\_sockets, 286
- TEE\_tlsSocket\_ClientPDC\_s, 276
- TEE\_tlsSocket\_Credentials\_s, 277
- TEE\_tlsSocket\_Credentials\_s.\_\_unnamed\_\_, 277
- TEE\_tlsSocket\_ExtensionFlags\_e
  - Tee\_sockets, 286
- TEE\_tlsSocket\_PSK\_Info\_s, 276
- TEE\_tlsSocket\_SRP\_Info\_s, 276
- TEE\_tlsSocket\_ServerCredentialType\_e
  - Tee\_sockets, 286
- TEE\_tlsSocket\_ServerPDC\_s, 276
- TEE\_tlsSocket\_Setup\_s, 278
- TEE\_tlsSocket\_Setup\_s.\_\_unnamed\_\_, 280
- TEE\_tlsSocket\_StatusRequestType\_e
  - Tee\_sockets, 287
- TEE\_tlsSocket\_tlsVersion\_e
  - Tee\_sockets, 287
- TEE\_udpSocket\_Change\_s, 275
- TEE\_udpSocket\_Setup\_s, 275
- TEES\_AcquireUserBuffer
  - Loadable driver API, 54
- TEES\_AddIrsFlag
  - Integrity Report System API, 107
- TEES\_AllocateContiguousMemory
  - Contiguous memory API, 66
- TEES\_AllocateInterrupt
  - Loadable driver API, 54
- TEES\_CONTIGUOUS\_FLAG\_ZERO\_ON\_ALLOC
  - Contiguous memory API, 66
- TEES\_CONTIGUOUS\_FLAG\_ZERO\_ON\_FREE
  - Contiguous memory API, 66
- TEES\_CONTIGUOUS\_MAP\_FIXED
  - Contiguous memory API, 66
- TEES\_CONTIGUOUS\_MAP\_NON\_CACHEABLE
  - Contiguous memory API, 66
- TEES\_CONTIGUOUS\_MAP\_POPULATE
  - Contiguous memory API, 66
- TEES\_CONTIGUOUS\_MAP\_READ
  - Contiguous memory API, 66
- TEES\_CONTIGUOUS\_MAP\_WRITE
  - Contiguous memory API, 66
- TEES\_CheckSecureObjectCreator
  - Samsung Internal API, 43
- TEES\_CompleteInterrupt
  - Loadable driver API, 55
- TEES\_CompleteRequest
  - Loadable driver API, 56
- TEES\_ContiguousAllocateFlags
  - Contiguous memory API, 66
- TEES\_ContiguousMapFlags
  - Contiguous memory API, 66
- TEES\_ContiguousMemoryHandle
  - Contiguous memory API, 65
- TEES\_DcacheClean
  - Loadable driver API, 57
- TEES\_DcacheFlush
  - Loadable driver API, 57
- TEES\_DellrsFlag
  - Integrity Report System API, 107
- TEES\_DeriveKeyKDF
  - Samsung Internal API, 44
- TEES\_DeriveKeySetKDF
  - Samsung Internal API, 45
- TEES\_DriverDestructor\_t
  - Loadable driver API, 53
- TEES\_Duplwhsmem
  - Miscellaneous extensions, 110
- TEES\_ERROR\_NOT\_IMPLEMENTED
  - Trusted user interface, 92
- TEES\_ERROR\_NOT\_SUPPORTED
  - Trusted user interface, 92
- TEES\_ERROR\_TUI\_BAD\_FORMAT
  - Trusted user interface, 92
- TEES\_ERROR\_TUI\_BAD\_STATE
  - Trusted user interface, 92
- TEES\_ERROR\_TUI\_BUSY
  - Trusted user interface, 92
- TEES\_ERROR\_TUI\_GENERIC
  - Trusted user interface, 92
- TEES\_ERROR\_TUI\_INVALID\_PARAM
  - Trusted user interface, 92
- TEES\_ERROR\_TUI\_OUT\_OF\_MEMORY
  - Trusted user interface, 92
- TEES\_EnterCritical
  - Samsung Internal API, 45
- TEES\_ExitCritical
  - Samsung Internal API, 46
- TEES\_FALSE
  - Trusted user interface, 92
- TEES\_FiniDriver
  - Loadable driver API, 57
- TEES\_GenerateInterrupt
  - Loadable driver API, 58
- TEES\_GetIrsFlagValue
  - Integrity Report System API, 107
- TEES\_GetRoT
  - Samsung Internal API, 46
- TEES\_GetTaskDataSize
  - Miscellaneous extensions, 110
- TEES\_I2CExit
  - I2C API, 81
- TEES\_I2CHandler
  - I2C API, 81
- TEES\_I2CInit
  - I2C API, 81
- TEES\_I2CRead
  - I2C API, 81
- TEES\_I2CTransfer, 80
- TEES\_I2CWrite
  - I2C API, 82
- TEES\_I2CWriteRead
  - I2C API, 82
- TEES\_IWT\_LISTENER\_NAME\_MAX\_LENGTH
  - Tee\_sockets, 283
- TEES\_InclrsFlag

- Integrity Report System API, [108](#)
- TEES\_InitDriver
  - Loadable driver API, [59](#)
- TEES\_InterruptHandle
  - Loadable driver API, [53](#)
- TEES\_IsPhysMemoryProtected
  - Miscellaneous extensions, [111](#)
- TEES\_IsREESharedMemory
  - Miscellaneous extensions, [111](#)
- TEES\_lwtCloseChannel
  - Tee\_sockets, [287](#)
- TEES\_lwtOpenChannel
  - Tee\_sockets, [287](#)
- TEES\_lwtRead
  - Tee\_sockets, [288](#)
- TEES\_lwtWrite
  - Tee\_sockets, [288](#)
- TEES\_LockHWCryptoBuf
  - Samsung Internal API, [46](#)
- TEES\_MapContiguousMemory
  - Contiguous memory API, [67](#)
- TEES\_RPMBCheckEnable
  - RPMB API, [113](#)
- TEES\_RPMBRead
  - RPMB API, [113](#)
- TEES\_RPMBWrite
  - RPMB API, [114](#)
- TEES\_RegisterDriver
  - Loadable driver API, [60](#)
- TEES\_RegisterDriverDestructor
  - Loadable driver API, [60](#)
- TEES\_ReleaseContiguousMemory
  - Contiguous memory API, [68](#)
- TEES\_ReleaseDriver
  - Loadable driver API, [61](#)
- TEES\_ReleaseInterrupt
  - Loadable driver API, [61](#)
- TEES\_ReleaseUserBuffer
  - Loadable driver API, [62](#)
- TEES\_Result
  - Trusted user interface, [92](#)
- TEES\_SECCAM\_GetStatus
  - Samsung Internal API, [47](#)
- TEES\_SMCCCommand
  - Tees\_smc, [266](#)
- TEES\_SMCFini
  - Tees\_smc, [267](#)
- TEES\_SMCInit
  - Tees\_smc, [267](#)
- TEES\_SPIClockDisable
  - SPI API, [71](#)
- TEES\_SPIClockEnable
  - SPI API, [71](#)
- TEES\_SPIConfig, [71](#)
- TEES\_SPIDMAInit
  - SPI API, [72](#)
- TEES\_SPIExit
  - SPI API, [72](#)
- TEES\_SPIHandler
  - SPI API, [71](#)
- TEES\_SPIInit
  - SPI API, [73](#)
- TEES\_SPIRead
  - SPI API, [74](#)
- TEES\_SPISetBitsPerWord
  - SPI API, [74](#)
- TEES\_SPISetCPHA
  - SPI API, [76](#)
- TEES\_SPISetCPOL
  - SPI API, [76](#)
- TEES\_SPISetClockSpeed
  - SPI API, [75](#)
- TEES\_SPISetConfig
  - SPI API, [75](#)
- TEES\_SPISetDMAMode
  - SPI API, [77](#)
- TEES\_SPITransfer, [71](#)
- TEES\_SPIWrite
  - SPI API, [77](#)
- TEES\_SPIWriteRead
  - SPI API, [78](#)
- TEES\_SUCCESS
  - Trusted user interface, [92](#)
- TEES\_SetIrsFlag
  - Integrity Report System API, [108](#)
- TEES\_SetIrsFlagValue
  - Integrity Report System API, [108](#)
- TEES\_TOUCH\_PRESSED
  - Trusted user interface, [92](#)
- TEES\_TOUCH\_RELEASED
  - Trusted user interface, [92](#)
- TEES\_TRUE
  - Trusted user interface, [92](#)
- TEES\_TUICheckTextFormat
  - Trusted user interface, [96](#)
- TEES\_TUICloseSession
  - Trusted user interface, [97](#)
- TEES\_TUIDrawBuffer
  - Trusted user interface, [97](#)
- TEES\_TUIDrawImage
  - Trusted user interface, [98](#)
- TEES\_TUIGetBuffer
  - Trusted user interface, [98](#)
- TEES\_TUIGetScreenInfo
  - Trusted user interface, [99](#)
- TEES\_TUIGetTouchEvent
  - Trusted user interface, [99](#)
- TEES\_TUIImage, [86](#)
- TEES\_TUIOpenSession
  - Trusted user interface, [100](#)
- TEES\_TUIPrintString
  - Trusted user interface, [100](#)
- TEES\_TUIRefreshScreen
  - Trusted user interface, [101](#)
- TEES\_TUIScreenInfo, [86](#)
- TEES\_TUITouchInfo, [86](#)

- TEES\_TUITouchTypes
  - Trusted user interface, [92](#)
- TEES\_UnlockHWCryptoBuf
  - Samsung Internal API, [48](#)
- TEES\_UnmapContiguousMemory
  - Contiguous memory API, [69](#)
- TEES\_UnwrapSecureObject
  - Samsung Internal API, [48](#)
- TEES\_WaitForInterrupt
  - Loadable driver API, [63](#)
- TEES\_WrapSecureObject
  - Samsung Internal API, [49](#)
- TEES\_WrappedWithREK
  - Samsung Internal API, [49](#)
- TEES\_bool
  - Trusted user interface, [91](#)
- TEMP\_FAILURE\_RETRY
  - Auxiliary API, [202](#)
- tee\_error.h, [323](#)
- tee\_i2c.h, [323](#)
- tee\_interrupt.h, [324](#)
- tee\_isocket.h, [325](#)
- tee\_smc.h, [326](#)
- Tee\_sockets, [268](#)
  - protocol\_error\_code, [283](#)
  - TEE\_IP\_VERSION\_4, [286](#)
  - TEE\_IP\_VERSION\_6, [286](#)
  - TEE\_IP\_VERSION\_DC, [286](#)
  - TEE\_ISOCKET\_ERROR\_HOSTNAME, [281](#)
  - TEE\_ISOCKET\_ERROR\_LARGE\_BUFFER, [281](#)
  - TEE\_ISOCKET\_ERROR\_OUT\_OF\_RESOURCES, [281](#)
  - TEE\_ISOCKET\_ERROR\_PROTOCOL, [281](#)
  - TEE\_ISOCKET\_ERROR\_REMOTE\_CLOSED, [281](#)
  - TEE\_ISOCKET\_ERROR\_TIMEOUT, [281](#)
  - TEE\_ISOCKET\_IWC\_ERROR\_CHANNEL, [284](#)
  - TEE\_ISOCKET\_IWC\_ERROR\_INVALID\_VERSION, [284](#)
  - TEE\_ISOCKET\_IWC\_ERROR\_NOT\_IMPLEMENTED, [284](#)
  - TEE\_ISOCKET\_IWC\_ERROR\_SWD\_CLIENT\_AUTH\_FAILED, [284](#)
  - TEE\_ISOCKET\_IWC\_ERROR\_TIMEOUT, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_BAD\_PARAMETERS, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_BUFFER\_TOO\_SMALL, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_COMMUNICATION, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_CONNECTION\_REFUSED, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_DATA\_REMAIN, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_GENERIC, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_HOSTNAME\_NOTRESOLVED, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_HOSTNAME\_TRYAGAIN, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_HOSTNAME\_UNKNOWN, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_LARGE\_BUFFER, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_NET\_UNREACHABLE, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_OUT\_OF\_MEMORY, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_OUT\_OF\_RESOURCES, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_REMOTE\_CLOSED, [284](#)
  - TEE\_ISOCKET\_NET\_ERROR\_TIMEOUT, [284](#)
  - TEE\_ISOCKET\_PROTOCOLID\_TCP, [282](#)
  - TEE\_ISOCKET\_PROTOCOLID\_TLS, [283](#)
  - TEE\_ISOCKET\_PROTOCOLID\_UDP, [282](#)
  - TEE\_ISOCKET\_SERVER\_NAME\_MAX\_LENGTH, [281](#)
  - TEE\_ISOCKET\_TCP\_API\_VERSION, [281](#)
  - TEE\_ISOCKET\_TCP\_WARNING\_UNKNOWN\_OUT\_OF\_BAND, [282](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_ACCESS\_DENIED, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_BAD\_CERT\_HASH\_VALUE, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_BAD\_CERT\_STATUS\_RESPONSE, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_BAD\_CERTIFICATE, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_BAD\_RECORD\_MAC, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_CERT\_EXPIRED, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_CERT\_REQUIRED, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_CERT\_REVOKED, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_CERT\_UNKNOWN, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_CERT\_UNOBTAINABLE, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_CLOSE\_NOTIFY, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_DECODE\_ERROR, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_DECOMP\_FAILED, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_DECRYPT\_ERROR, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_DECRYPT\_FAILED, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_EXPORT\_RESTRICTED, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_HANDSHAKE\_FAILED, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_ILLEGAL\_PARAMETER, [285](#)
  - TEE\_ISOCKET\_TLS\_ALERT\_INAPPROPRIATE\_FALLBACK, [285](#)

- 285
- TEE\_ISOCKET\_TLS\_ALERT\_INSUFFICIENT\_SECURITY, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_INTERNAL\_ERROR, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_MISSING\_EXTENSION, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_NO\_CERTIFICATE, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_NO\_RENEGOTIATION, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_PROTOCOL\_VERSION, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_RECORD\_OVERFLOW, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_UNEXPECTED\_MSG, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_UNKNOWN\_CA, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_UNKNOWN\_PSK\_IDENTITY, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_UNRECOGNIZED\_NAME, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_UNSUPPORTED\_CERT, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_UNSUPPORTED\_EXTENSION, 285
- 285
- TEE\_ISOCKET\_TLS\_ALERT\_USER\_CANCELED, 285
- 285
- TEE\_ISOCKET\_TLS\_API\_VERSION, 283
- TEE\_ISOCKET\_TLS\_CB\_BAD\_CERT\_CHAIN, 286
- 286
- TEE\_ISOCKET\_TLS\_CB\_CHECK\_CERT\_CHAIN, 286
- 286
- TEE\_ISOCKET\_TLS\_CB\_CHECK\_OCSP\_STATUS, 286
- 286
- TEE\_ISOCKET\_TLS\_CB\_REVOKED\_OCSP\_STATUS, 286
- 286
- TEE\_ISOCKET\_TLS\_CB\_UNKNOWN\_OCSP\_STATUS, 286
- 286
- TEE\_ISOCKET\_TLS\_CERT\_KEYUSAGE\_CHECK\_CLIENT, 286
- 286
- TEE\_ISOCKET\_TLS\_CERT\_NAME\_CHECK\_CLIENT, 286
- 286
- TEE\_ISOCKET\_TLS\_CERT\_NOTIFY\_CLIENT, 286
- 286
- TEE\_ISOCKET\_TLS\_ERROR\_ALERT\_PENDING, 285
- 283
- TEE\_ISOCKET\_TLS\_ERROR\_AUTHENTICATION, 283
- 284
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_COMMON\_NAME\_VERIFICATION, 284
- 284
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_EXPIRED, 284
- 285
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_IS\_TOO\_LONG, 285
- 284
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_PARSING, 284
- 285
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_REVOKED, 285
- 284
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_SIGN\_VERIFICATION, 284
- 285
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_STATUS\_UNKNOWN, 285
- 285
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_UNKNOWN\_CA, 285
- 285
- TEE\_ISOCKET\_TLS\_ERROR\_CERT\_UNSUPPORTED, 285
- 284
- TEE\_ISOCKET\_TLS\_ERROR\_CRL\_PARSING, 284
- 283
- TEE\_ISOCKET\_TLS\_ERROR\_DATA, 283
- 284
- TEE\_ISOCKET\_TLS\_ERROR\_ECDHE\_GEN\_KEY, 284
- 284
- TEE\_ISOCKET\_TLS\_ERROR\_ECDHE\_SERIALIZING, 284
- 284
- TEE\_ISOCKET\_TLS\_ERROR\_ECDHE\_SHARED\_SECRET, 284
- 284
- TEE\_ISOCKET\_TLS\_ERROR\_ECDHE\_UNSUPPORTED\_CURVE, 284
- 285
- TEE\_ISOCKET\_TLS\_ERROR\_HANDSHAKE\_UNEXPECTED\_PARAMETERS, 285
- 283
- TEE\_ISOCKET\_TLS\_ERROR\_HANDSHAKE, 283
- 285
- TEE\_ISOCKET\_TLS\_ERROR\_NO\_ALERT\_PRESENT, 285
- 283
- TEE\_ISOCKET\_TLS\_ERROR\_REJECTED\_SUITE, 283
- 284
- TEE\_ISOCKET\_TLS\_ERROR\_UNEXPECTED\_MESSAGE, 284
- 283
- TEE\_ISOCKET\_TLS\_ERROR\_UNSUPPORTED\_SUITE, 283
- 285
- TEE\_ISOCKET\_TLS\_ERROR\_USER\_CANCELED, 285
- 283
- TEE\_ISOCKET\_TLS\_ERROR\_VERSION, 283
- 286
- TEE\_ISOCKET\_TLS\_OCSP\_CHECK\_ADVISORY, 286
- 286
- TEE\_ISOCKET\_TLS\_OCSP\_CHECK\_CLIENT, 286
- 286
- TEE\_ISOCKET\_TLS\_OCSP\_CHECK\_MANDATORY, 286
- 287
- TEE\_ISOCKET\_TLS\_OCSP\_STATUS\_REQUEST\_NO, 287
- 287
- TEE\_ISOCKET\_TLS\_OCSP\_STATUS\_REQUEST, 287
- 282
- TEE\_ISOCKET\_UDP\_API\_VERSION, 282
- 282
- TEE\_ISOCKET\_UDP\_WARNING\_UNKNOWN\_OUT\_OF\_BAND, 282
- 281
- TEE\_ISOCKET\_WARNING\_PROTOCOL, 281
- 286
- TEE\_TLS\_CLIENT\_CRED\_CSC, 286
- 286
- TEE\_TLS\_CLIENT\_CRED\_NONE, 286
- 287
- TEE\_TLS\_SERVER\_CRED\_CSC, 287
- 287
- TEE\_TLS\_SERVER\_CRED\_PDC, 287
- 287
- TEE\_TLS\_VERSION\_1v2, 287
- 287
- TEE\_TLS\_VERSION\_ALL, 287
- 280
- TEE\_iSocket, 280

- TEE\_ipSocket\_ipVersion\_e, 285
- TEE\_tlsSocket\_CB\_Data, 280
- TEE\_tlsSocket\_CallbackReasonType\_e, 286
- TEE\_tlsSocket\_ClientCredentialType\_e, 286
- TEE\_tlsSocket\_ExtensionFlags\_e, 286
- TEE\_tlsSocket\_ServerCredentialType\_e, 286
- TEE\_tlsSocket\_StatusRequestType\_e, 287
- TEE\_tlsSocket\_tlsVersion\_e, 287
- TEES\_IWT\_LISTENER\_NAME\_MAX\_LENGTH, 283
- TEES\_lwtCloseChannel, 287
- TEES\_lwtOpenChannel, 287
- TEES\_lwtRead, 288
- TEES\_lwtWrite, 288
- tee\_spi.h, 327
- tee\_ta\_destructor.h, 328
- tee\_tcpsocket.h, 328
- tee\_tlsocket.h, 329
- tee\_udpsocket.h, 333
- teeOnly
  - \_\_TEE\_SEReaderProperties, 294
- tees\_critical.h, 334
- tees\_extension.h, 334
- tees\_hwcrypto\_buf.h, 335
- tees\_iwt.h, 335
- tees\_kdf.h, 336
- tees\_rot.h, 336
- tees\_seccam.h, 337
- tees\_secure\_object.h, 337
- tees\_tui.h, 339
- tees\_wrapped\_with\_rek.h, 341
- Teesl\_smc, 266
  - TEES\_SMCCCommand, 266
  - TEES\_SMCFinis, 267
  - TEES\_SMCInit, 267
- Thread support library API, 115
  - MUTEX\_STATE\_LOCKED, 118
  - MUTEX\_STATE\_UNLOCKED, 118
  - PTHREAD\_ERRORCHECK\_MUTEX\_INITIALIZER\_NP, 118
    - Auxiliary API, 255
  - PTHREAD\_MUTEX\_DEFAULT, 121
  - PTHREAD\_MUTEX\_DESTROYED, 121
  - PTHREAD\_MUTEX\_ERRORCHECK\_NP, 121
  - PTHREAD\_MUTEX\_ERRORCHECK, 121
  - PTHREAD\_MUTEX\_INITIALIZER, 118
  - PTHREAD\_MUTEX\_NORMAL, 121
  - PTHREAD\_MUTEX\_RECURSIVE\_NP, 121
  - PTHREAD\_MUTEX\_RECURSIVE, 121
  - PTHREAD\_ONCE\_INIT, 118
  - PTHREAD\_RECURSIVE\_MUTEX\_INITIALIZER\_NP, 118
    - Auxiliary API, 255
  - PTHREAD\_STACK\_MIN, 119
- pthread\_attr\_destroy, 121
- pthread\_attr\_init, 121
- pthread\_attr\_setstacksize, 122
- pthread\_attr\_t, 119
- pthread\_cond\_broadcast, 122
- pthread\_cond\_destroy, 123
- pthread\_cond\_init, 124
- pthread\_cond\_signal, 124
- pthread\_cond\_t, 119
- pthread\_cond\_wait, 125
- pthread\_condattr\_t, 120
- pthread\_create, 126
- pthread\_getspecific, 127
- pthread\_impl\_t, 120
- pthread\_join, 127
- pthread\_key\_create, 128
- pthread\_key\_delete, 129
- pthread\_key\_t, 120
- pthread\_kill, 130
- pthread\_mutex\_destroy, 130
- pthread\_mutex\_init, 131
- pthread\_mutex\_lock, 132
- pthread\_mutex\_t, 120
- pthread\_mutex\_trylock, 132
- pthread\_mutex\_unlock, 133
- pthread\_mutexattr\_destroy, 134
- pthread\_mutexattr\_gettype, 134
- pthread\_mutexattr\_init, 135
- pthread\_mutexattr\_settype, 135
- pthread\_mutexattr\_t, 120
- pthread\_once, 136
- pthread\_once\_t, 120
- pthread\_self, 137
- pthread\_setspecific, 137
- pthread\_sigmask, 119
- pthread\_t, 120
- time
  - Auxiliary API, 254
- time.h, 342
- time\_t
  - Auxiliary API, 204
- timeadd
  - Auxiliary API, 254
- timespec\_to\_ms
- timespec\_to\_nsec
  - Auxiliary API, 255
- timesub
  - Auxiliary API, 255
- tm, 173
- tolower
  - Auxiliary API, 256
- toupper
  - Auxiliary API, 256
- truncate
  - fops, 53
- Trusted user interface, 84
  - MAX\_FONT\_SIZE, 90
  - MIN\_FONT\_SIZE, 90
  - TEE\_TEMPORARY\_CLEAR\_MODE, 91
  - TEE\_TUI\_ALPHANUMERICAL, 91
  - TEE\_TUI\_CANCEL, 90
  - TEE\_TUI\_CLEAR\_MODE, 91
  - TEE\_TUI\_CORRECTION, 90

- TEE\_TUI\_HIDDEN\_MODE, 91
- TEE\_TUI\_LANDSCAPE, 91
- TEE\_TUI\_NEXT, 90
- TEE\_TUI\_NO\_SOURCE, 91
- TEE\_TUI\_NUMBER\_BUTTON\_TYPES, 90
- TEE\_TUI\_NUMERICAL, 91
- TEE\_TUI\_OBJECT\_SOURCE, 91
- TEE\_TUI\_OK, 90
- TEE\_TUI\_PORTRAIT, 91
- TEE\_TUI\_PREVIOUS, 90
- TEE\_TUI\_REF\_SOURCE, 91
- TEE\_TUI\_VALIDATE, 90
- TEE\_TUIButtonType, 90
- TEE\_TUICheckTextFormat, 93
- TEE\_TUICloseSession, 93
- TEE\_TUIDisplayScreen, 93
- TEE\_TUIEntryFieldMode, 90
- TEE\_TUIEntryFieldType, 91
- TEE\_TUIGetScreenInfo, 95
- TEE\_TUIImageSource, 91
- TEE\_TUIInitSession, 96
- TEE\_TUIScreenOrientation, 91
- TEES\_ERROR\_NOT\_IMPLEMENTED, 92
- TEES\_ERROR\_NOT\_SUPPORTED, 92
- TEES\_ERROR\_TUI\_BAD\_FORMAT, 92
- TEES\_ERROR\_TUI\_BAD\_STATE, 92
- TEES\_ERROR\_TUI\_BUSY, 92
- TEES\_ERROR\_TUI\_GENERIC, 92
- TEES\_ERROR\_TUI\_INVALID\_PARAM, 92
- TEES\_ERROR\_TUI\_OUT\_OF\_MEMORY, 92
- TEES\_FALSE, 92
- TEES\_Result, 92
- TEES\_SUCCESS, 92
- TEES\_TOUCH\_PRESSED, 92
- TEES\_TOUCH\_RELEASED, 92
- TEES\_TRUE, 92
- TEES\_TUICheckTextFormat, 96
- TEES\_TUICloseSession, 97
- TEES\_TUIDrawBuffer, 97
- TEES\_TUIDrawImage, 98
- TEES\_TUIGetBuffer, 98
- TEES\_TUIGetScreenInfo, 99
- TEES\_TUIGetTouchEvent, 99
- TEES\_TUIOpenSession, 100
- TEES\_TUIPrintString, 100
- TEES\_TUIRefreshScreen, 101
- TEES\_TUITouchTypes, 92
- TEES\_bool, 91
- tui.h, 343
- UCHAR\_MAX
  - Auxiliary API, 202
- UINT\_MAX
  - Auxiliary API, 202
- ULLONG\_MAX
  - Auxiliary API, 202
- ULONG\_MAX
  - Auxiliary API, 203
- USHRT\_MAX
  - Auxiliary API, 203
- UUID\_STRING\_LEN
  - Auxiliary API, 203
- uid\_t
  - Auxiliary API, 205
- unistd.h, 345
- unlink
  - Auxiliary API, 256
  - fops, 53
- usr\_drv\_info, 53
- uuid/uuid.h, 346
- uuid\_clear
  - Auxiliary API, 257
- uuid\_compare
  - Auxiliary API, 257
- uuid\_generate
  - Auxiliary API, 257
- uuid\_generate\_time
  - Auxiliary API, 203
- uuid\_is\_null
  - Auxiliary API, 258
- uuid\_parse
  - Auxiliary API, 258
- uuid\_t
  - Auxiliary API, 205
- uuid\_unparse
  - Auxiliary API, 258
- uuid\_unparse\_lower
  - Auxiliary API, 203
- uuid\_unparse\_upper
  - Auxiliary API, 259
- vasprintf
  - Auxiliary API, 259
- vasprintf\_s
  - Auxiliary API, 259
- vfprintf
  - Auxiliary API, 260
- vsprintf
  - Auxiliary API, 260
- vsprintf\_s
  - Auxiliary API, 261
- vsprintf
  - Auxiliary API, 262
- vsprintf\_s
  - Auxiliary API, 263
- vsscanf
  - Auxiliary API, 263
- vsscanf\_s
  - Auxiliary API, 264
- wrapped\_wkth\_rek\_t, 41
- write
  - Auxiliary API, 264
  - fops, 53