



Samsung Internal API reference

Date	2019.11.12	Version	2.0
Organizator	Mobile Security Technologies Group, IT & Mobile Communications		

Contents

1 Overview	38
1.1 Objective	38
1.2 Scope	38
1.3 Abbreviations and Acronyms	38
1.4 API type	38
1.4.1 TA properties	38
1.4.2 Samsung Internal API	42
1.4.3 Thread support library API	42
1.4.4 Math library API	42
1.4.5 Auxiliary API	42
2 Deprecated List	43
3 Data Structure Index	44
3.1 Data Structures	44
4 Module Documentation	45
4.1 Samsung Internal API	45
4.1.1 Detailed Description	47
4.1.2 Data Structure Documentation	47
4.1.2.1 struct TEES_El2if_Args	47
4.1.2.2 struct rot_t	47
4.1.2.3 struct wrapped_wkth_rek_t	48
4.1.2.4 struct secHDCPKeyInfo_t	48
4.1.3 Macro Definition Documentation	48
4.1.3.1 KM_KW_MAX_AAD_LEN	48
4.1.3.2 KM_KW_MAX_INPUT_LEN	48
4.1.3.3 KM_KW_MAX_IV_LEN	49
4.1.3.4 KM_KW_MAX_KEY_LEN	49

4.1.3.5	KM_KW_MAX_SALT_LEN	49
4.1.3.6	KM_KW_MAX_TAG_LEN	49
4.1.3.7	SO_AC_LEN	49
4.1.3.8	SO_AUTH_ID_LEN	49
4.1.3.9	SO_HEADER_SIZE_STATIC	49
4.1.3.10	SO_IV_LEN	50
4.1.3.11	SO_MAGIC_NUMBER_LEN	50
4.1.3.12	SO_OUT_BUF_SIZE	50
4.1.3.13	SO_TA_ID_LEN	50
4.1.3.14	SO_TAG_LEN	50
4.1.4	Function Documentation	50
4.1.4.1	errno_to_tee_error(int error_code)	50
4.1.4.2	TEES_CheckSecureObjectCreator(const unsigned char *in, uint32_t in_len, SO_AccessControllInfoType *ac)	51
4.1.4.3	TEES_DeriveKeyKDF(const void *label, uint32_t labelLen, const void *context, uint32_t contextLen, uint32_t outputKeyLen, TEE_ObjectHandle object)	51
4.1.4.4	TEES_DeriveKeySetKDF(const void *label, uint32_t labelLen, const void *context, uint32_t contextLen, uint32_t outputKeyLen, TEE_ObjectHandle object)	52
4.1.4.5	TEES_El2if(struct TEES_El2if_Args *args)	53
4.1.4.6	TEES_EnterCritical(void)	53
4.1.4.7	TEES_ExitCritical(void)	53
4.1.4.8	TEES_GetRoT(ROOT_OF_TRUST *rot)	54
4.1.4.9	TEES_HDCP_SetKeyInfo(struct secHDCPKeyInfo_t *SecureHDCPKey_info, unsigned int *data)	54
4.1.4.10	TEES_LockHWCryptoBuf(void)	55
4.1.4.11	TEES_SECCAM_GetStatus(unsigned int *data)	55
4.1.4.12	TEES_SECCAM_IsProtected(uint64_t paddr, size_t size, unsigned int *data)	56
4.1.4.13	TEES_SECCAM_Protect(uint64_t paddr, size_t size, unsigned int *data)	57
4.1.4.14	TEES_SECCAM_Unprotect(uint64_t paddr, size_t size, unsigned int *data)	58
4.1.4.15	TEES_TUI_Protect(uint64_t paddr, size_t size, unsigned int *data)	58

4.1.4.16	TEES_TUI_SetInfo(uint32_t cmd, uint64_t paddr, size_t size, unsigned int *data)	59
4.1.4.17	TEES_TUI_Unprotect(uint64_t paddr, size_t size, unsigned int *data)	60
4.1.4.18	TEES_UnlockHWCryptoBuf(void)	61
4.1.4.19	TEES_UnwrapSecureObject(const unsigned char *in, uint32_t in_len, unsigned char *out, uint32_t *out_len)	61
4.1.4.20	TEES_WrappedWithREK(WRAP_REK *data)	62
4.1.4.21	TEES_WrapSecureObject(const unsigned char *in, uint32_t in_len, unsigned char *out, uint32_t *out_len, SO_AccessControllInfoType *ac)	62
4.2	Loadable driver API	64
4.2.1	Detailed Description	65
4.2.2	Data Structure Documentation	65
4.2.2.1	struct intrinfo	65
4.2.3	Typedef Documentation	65
4.2.3.1	intruhandler	65
4.2.3.2	TEES_DriverDestructor_t	65
4.2.3.3	TEES_InterruptHandle	65
4.2.4	Function Documentation	66
4.2.4.1	TEES_AcquireUserBuffer(struct drv_info *filp, uint64_t addr, const size_t size, int prot)	66
4.2.4.2	TEES_AllocateInterrupt(int nr, intruhandler handler, TEES_InterruptHandle *handle)	66
4.2.4.3	TEES_CompleteInterrupt(TEES_InterruptHandle handle)	67
4.2.4.4	TEES_CompleteRequest(struct drv_info *filp, long ret)	68
4.2.4.5	TEES_DcacheClean(const void *addr, size_t nbytes)	69
4.2.4.6	TEES_DcacheFlush(const void *addr, size_t nbytes)	69
4.2.4.7	TEES_FiniDriver(struct usr_drv_info *info)	70
4.2.4.8	TEES_GenerateInterrupt(TEES_InterruptHandle handle)	70
4.2.4.9	TEES_InitDriver(char *name, struct fops *fops, unsigned int drvid, struct usr_drv_info **info)	71
4.2.4.10	TEES_RegisterDriver(char *name, struct fops *fops, unsigned int drvid, struct usr_drv_info **info) _deprecated_	72
4.2.4.11	TEES_RegisterDriverDestructor(TEES_DriverDestructor_t destr)	73

4.2.4.12	TEES_RegisterIoctlDesc(struct usr_drv_info *info, unsigned int cmd, struct ioctl_desc *desc)	73
4.2.4.13	TEES_ReleaseDriver(struct usr_drv_info **info) _deprecated_	74
4.2.4.14	TEES_ReleaseInterrupt(TEES_InterruptHandle handle)	74
4.2.4.15	TEES_ReleaseUserBuffer(const void *addr, const size_t size)	75
4.2.4.16	TEES_WaitForInterrupt(TEES_InterruptHandle handle, uint32_t timeout)	75
4.3	Custom handler API	77
4.4	Contiguous memory API	78
4.4.1	Detailed Description	78
4.4.2	Typedef Documentation	78
4.4.2.1	TEES_ContiguousMemoryHandle	78
4.4.3	Enumeration Type Documentation	79
4.4.3.1	TEES_ContiguousAllocateFlags	79
4.4.3.2	TEES_ContiguousMapFlags	79
4.4.4	Function Documentation	79
4.4.4.1	TEES_AllocateContiguousMemory(const char *region, size_t size, size_t align, uint32_t flags, TEES_ContiguousMemoryHandle *memory)	79
4.4.4.2	TEES_GetContiguousMemoryPhysaddr(TEES_ContiguousMemoryHandle memory, uint64_t *physaddr)	80
4.4.4.3	TEES_MapContiguousMemory(TEES_ContiguousMemoryHandle memory, void **addr, uint32_t flags)	81
4.4.4.4	TEES_ReleaseContiguousMemory(TEES_ContiguousMemoryHandle memory)	82
4.4.4.5	TEES_UnmapContiguousMemory(void *addr)	82
4.5	SPI API	83
4.5.1	Detailed Description	83
4.5.2	Data Structure Documentation	84
4.5.2.1	struct TEES_SPIConfig	84
4.5.2.2	struct TEES_SPITransfer	84
4.5.3	Typedef Documentation	84
4.5.3.1	TEES_SPIHandler	84
4.5.4	Function Documentation	84
4.5.4.1	TEES_SPIClockDisable(TEES_SPIHandler handler)	84

4.5.4.2	TEES_SPIClockEnable(TEES_SPIHandler handler)	85
4.5.4.3	TEES_SPIDMAInit(uintptr_t address)	85
4.5.4.4	TEES_SPIExit(TEES_SPIHandler handler)	85
4.5.4.5	TEES_SPIInit(uint32_t port, const TEES_SPIConfig *cfg, TEES_SPIHandler *handler)	86
4.5.4.6	TEES_SPIRead(TEES_SPIHandler handler, TEES_SPITransfer *rx)	87
4.5.4.7	TEES_SPISetBitsPerWord(TEES_SPIHandler handler, uint8_t bitsPerWord)	88
4.5.4.8	TEES_SPISetClockSpeed(TEES_SPIHandler handler, uint32_t speedHz)	88
4.5.4.9	TEES_SPISetConfig(TEES_SPIHandler handler, const TEES_SPIConfig *cfg)	89
4.5.4.10	TEES_SPISetCPHA(TEES_SPIHandler handler, uint8_t cpha)	89
4.5.4.11	TEES_SPISetCPOL(TEES_SPIHandler handler, uint8_t cpol)	90
4.5.4.12	TEES_SPISetDMAMode(TEES_SPIHandler handler, bool isEnabled)	90
4.5.4.13	TEES_SPIWrite(TEES_SPIHandler handler, TEES_SPITransfer *tx)	91
4.5.4.14	TEES_SPIWriteRead(TEES_SPIHandler handler, TEES_SPITransfer *tx, TEES_SPITransfer *rx)	91
4.6	I2C API	93
4.6.1	Detailed Description	93
4.6.2	Data Structure Documentation	93
4.6.2.1	struct TEES_I2CTransfer	93
4.6.3	Typedef Documentation	94
4.6.3.1	TEES_I2CHandler	94
4.6.4	Function Documentation	94
4.6.4.1	TEES_I2CExit(TEES_I2CHandler handler)	94
4.6.4.2	TEES_I2CInit(const char *name, uint32_t transac_len, TEES_I2CHandler *handler)	94
4.6.4.3	TEES_I2CRead(TEES_I2CHandler handler, uint8_t chip, TEES_I2CTransfer *rx)	95
4.6.4.4	TEES_I2CWrite(TEES_I2CHandler handler, uint8_t chip, TEES_I2CTransfer *tx)	95
4.6.4.5	TEES_I2CWriteRead(TEES_I2CHandler handler, uint8_t chip, TEES_I2CTransfer *tx, TEES_I2CTransfer *rx)	96
4.7	Trusted user interface	97
4.7.1	Detailed Description	99

4.7.2	Data Structure Documentation	99
4.7.2.1	struct TEES_TUIScreenInfo	99
4.7.2.2	struct TEES_TUITouchInfo	99
4.7.2.3	struct TEES_TUIMTInfo	100
4.7.2.4	struct TEES_TUI_MT_Info	100
4.7.2.5	struct TEES_TUIImage	100
4.7.2.6	struct TEE_TUIImage	100
4.7.2.7	union TEE_TUIImage.__unnamed__	101
4.7.2.8	struct TEE_TUIImage.__unnamed__.ref	101
4.7.2.9	struct TEE_TUIImage.__unnamed__.object	101
4.7.2.10	struct TEE_TUIScreenLabel	101
4.7.2.11	struct TEE_TUIButton	102
4.7.2.12	struct TEE_TUIScreenConfiguration	102
4.7.2.13	struct TEE_TUIScreenButtonInfo	102
4.7.2.14	struct TEE_TUIScreenInfo	102
4.7.2.15	struct TEE_TUIEntryField	103
4.7.3	Macro Definition Documentation	103
4.7.3.1	MAX_FONT_SIZE	103
4.7.3.2	MIN_FONT_SIZE	104
4.7.3.3	MT_INFO_MAX_EVENT	104
4.7.3.4	TEE_TUI_NUMBER_BUTTON_TYPES	104
4.7.4	Enumeration Type Documentation	104
4.7.4.1	TEE_TUIButtonType	104
4.7.4.2	TEE_TUIEntryFieldMode	104
4.7.4.3	TEE_TUIEntryFieldType	105
4.7.4.4	TEE_TUIImageSource	105
4.7.4.5	TEE_TUIScreenOrientation	105
4.7.4.6	TEES_bool	105
4.7.4.7	TEES_Result	106
4.7.4.8	TEES_TUITouchTypes	106

4.7.5	Function Documentation	106
4.7.5.1	TEE_TUICheckTextFormat(char *text, uint32_t *width, uint32_t *height, uint32_t *lastIndex)	106
4.7.5.2	TEE_TUICloseSession(void)	107
4.7.5.3	TEE_TUIDisplayScreen(TEE_TUIScreenConfiguration *screenConfiguration, bool closeTUISession, TEE_TUIEntryField *entryFields, uint32_t entryFieldCount, TEE_TUIButtonType *selectedButton)	107
4.7.5.4	TEE_TUIGetScreenInfo(TEE_TUIScreenOrientation screenOrientation, uint32_t nbEntryFields, TEE_TUIScreenInfo *screenInfo)	109
4.7.5.5	TEE_TUIInitSession(void)	110
4.7.5.6	TEES_TUICheckTextFormat(char *inputText, uint32_t textSize, uint32_t *width, uint32_t *height, uint32_t *lastIndex)	110
4.7.5.7	TEES_TUICloseSession(void)	111
4.7.5.8	TEES_TUIDrawBuffer(uint32_t *buffer, uint32_t posX, uint32_t posY, uint32_t W, uint32_t H)	111
4.7.5.9	TEES_TUIDrawImage(TEES_TUIImage *image, uint32_t posX, uint32_t posY)	112
4.7.5.10	TEES_TUIDrawImageFromBuff(uint32_t posX, uint32_t posY, uint32_t W, uint32_t H, void *buff, uint32_t buf_size)	113
4.7.5.11	TEES_TUIDrawImageToBuff(TEES_TUIImage *image, void *buff, uint32_t *buf_size)	113
4.7.5.12	TEES_TUIGetBuffer(uint32_t *buffer, uint32_t posX, uint32_t posY, uint32_t W, uint32_t H)	114
4.7.5.13	TEES_TUIGetMTEvent(TEES_TUI_MT_Info *touchInfo, uint32_t timeout)	114
4.7.5.14	TEES_TUIGetScreenInfo(TEES_TUIScreenInfo *screenInfo)	115
4.7.5.15	TEES_TUIGetTouchEvent(TEES_TUITouchInfo *touchInfo, uint32_t timeout)	115
4.7.5.16	TEES_TUIOpenSession(void)	116
4.7.5.17	TEES_TUIPrintString(char *inputText, uint32_t textSize, uint32_t posX, uint32_t posY, uint8_t redColorValue, uint8_t greenColorValue, uint8_t blueColorValue, bool rotation90)	116
4.7.5.18	TEES_TUIRefreshScreen(void)	117
4.8	Integrity Report System API	118
4.8.1	Detailed Description	119
4.8.2	Macro Definition Documentation	119
4.8.2.1	IRS_DENY_DELETE_FROM_SMC_TZ	119

4.8.2.2	IRS_DENY_READ_FROM_SMC_TZ	119
4.8.2.3	IRS_DENY_WRITE_FROM_SMC_TZ	119
4.8.2.4	IRS_FAIL_TZ	119
4.8.2.5	IRS_INCORRECT_CHECKSUM_TZ	119
4.8.2.6	IRS_INCORRECT_FLAG_TYPE_TZ	119
4.8.2.7	IRS_INCORRECT_RT_ID_TZ	119
4.8.2.8	IRS_MAX_VAL_COUNTER_RT_TZ	120
4.8.2.9	IRS_MAX_VAL_COUNTER_TZ	120
4.8.2.10	IRS_RT_FLAGS_EMPTY_TZ	120
4.8.2.11	IRS_RT_FLAGS_FULL_TZ	120
4.8.2.12	IRS_SUCCESS_TZ	120
4.8.2.13	IRS_UNKNOWN_ERROR_TZ	120
4.8.2.14	IRS_UNKNOWN_ID_TZ	120
4.8.2.15	IRS_UNKNOWN_INT_CMD_TZ	121
4.8.3	Enumeration Type Documentation	121
4.8.3.1	IRS_INTERNAL_CMD	121
4.8.3.2	IRS_PARAM	121
4.8.4	Function Documentation	122
4.8.4.1	TEES_AddIrsFlag(unsigned int *flag_id, unsigned int param)	122
4.8.4.2	TEES_DelIrsFlag(unsigned int *flag_id)	122
4.8.4.3	TEES_GetIrsFlagValue(unsigned int *flag_id, unsigned int *value)	123
4.8.4.4	TEES_IncIrsFlag(unsigned int *flag_id)	123
4.8.4.5	TEES_SetIrsFlag(unsigned int *flag_id)	123
4.8.4.6	TEES_SetIrsFlagValue(unsigned int *flag_id, unsigned int value)	124
4.9	Miscellaneous extensions	125
4.9.1	Detailed Description	125
4.9.2	Data Structure Documentation	125
4.9.2.1	struct TEES_ClientCredentials	125
4.9.3	Function Documentation	126
4.9.3.1	TEES_DupIwshmem(void *address, uint32_t size)	126

4.9.3.2	TEES_GetClientCredentials(struct TEES_ClientCredentials *credentials)	126
4.9.3.3	TEES_GetTaskDataSize(size_t *data_size)	127
4.9.3.4	TEES_IsREESharedMemory(uint32_t accessFlags, const void *buffer, size_t size)	127
4.10	RPMB API	128
4.10.1	Detailed Description	128
4.10.2	Enumeration Type Documentation	128
4.10.2.1	anonymous enum	128
4.10.3	Function Documentation	128
4.10.3.1	TEES_RPMBCheckEnable(void)	128
4.10.3.2	TEES_RPMBRead(uint32_t partition, uint32_t offset, uint8_t *data, uint32_t size, uint8_t type)	129
4.10.3.3	TEES_RPMBWrite(uint32_t partition, uint32_t offset, const uint8_t *data, uint32_t size, uint8_t type)	129
4.11	Thread support library API	130
4.11.1	Detailed Description	133
4.11.2	Data Structure Documentation	133
4.11.2.1	struct __pthread_once_t	133
4.11.2.2	struct __pthread_attr_t	133
4.11.2.3	struct __pthread_mutex_t	133
4.11.2.4	struct __pthread_cond_t	133
4.11.2.5	struct __pthread_condattr_t	134
4.11.3	Macro Definition Documentation	134
4.11.3.1	MUTEX_STATE_LOCKED	134
4.11.3.2	MUTEX_STATE_UNLOCKED	134
4.11.3.3	PTHREAD_ERRORCHECK_MUTEX_INITIALIZER_NP	134
4.11.3.4	PTHREAD_GUARD_MAX	134
4.11.3.5	PTHREAD_GUARD_MIN	134
4.11.3.6	PTHREAD_MUTEX_INITIALIZER	134
4.11.3.7	PTHREAD_ONCE_INIT	135
4.11.3.8	PTHREAD_RECURSIVE_MUTEX_INITIALIZER_NP	135
4.11.3.9	pthread_sigmask	135

4.11.3.10 PTHREAD_STACK_MIN	135
4.11.4 Typedef Documentation	136
4.11.4.1 pthread_attr_t	136
4.11.4.2 pthread_cond_t	136
4.11.4.3 pthread_condattr_t	136
4.11.4.4 pthread_impl_t	136
4.11.4.5 pthread_key_t	136
4.11.4.6 pthread_mutex_t	136
4.11.4.7 pthread_mutexattr_t	136
4.11.4.8 pthread_once_t	137
4.11.4.9 pthread_t	137
4.11.5 Enumeration Type Documentation	137
4.11.5.1 anonymous enum	137
4.11.5.2 anonymous enum	137
4.11.6 Function Documentation	138
4.11.6.1 pthread_attr_destroy(pthread_attr_t *attr)	138
4.11.6.2 pthread_attr_getdetachstate(const pthread_attr_t *attr, int *detachstate)	138
4.11.6.3 pthread_attr_getguardsize(const pthread_attr_t *attr, size_t *guardsize)	138
4.11.6.4 pthread_attr_getstack(const pthread_attr_t *attr, void **stackaddr, size_t *stacksize)	139
4.11.6.5 pthread_attr_getstackaddr(const pthread_attr_t *attr, void **stackaddr)	139
4.11.6.6 pthread_attr_getstacksize(const pthread_attr_t *attr, size_t *stacksize)	139
4.11.6.7 pthread_attr_init(pthread_attr_t *attr)	140
4.11.6.8 pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate)	140
4.11.6.9 pthread_attr_setguardsize(pthread_attr_t *attr, size_t guardsize)	140
4.11.6.10 pthread_attr_setstack(pthread_attr_t *attr, void *stackaddr, size_t stacksize)	141
4.11.6.11 pthread_attr_setstackaddr(pthread_attr_t *attr, void *stackaddr)	141
4.11.6.12 pthread_attr_setstacksize(pthread_attr_t *attr, size_t stacksize)	141
4.11.6.13 pthread_cond_broadcast(pthread_cond_t *cond)	142
4.11.6.14 pthread_cond_destroy(pthread_cond_t *cond)	142

4.11.6.15 pthread_cond_init(pthread_cond_t *cond, const pthread_condattr_t *attr)	143
4.11.6.16 pthread_cond_signal(pthread_cond_t *cond)	144
4.11.6.17 pthread_cond_timedwait(pthread_cond_t *cond, pthread_mutex_t *mutex, const struct timespec *timeout)	144
4.11.6.18 pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex)	145
4.11.6.19 pthread_condattr_destroy(pthread_condattr_t *attr)	146
4.11.6.20 pthread_condattr_init(pthread_condattr_t *attr)	146
4.11.6.21 pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine)(void *), void *arg)	147
4.11.6.22 pthread_detach(pthread_t thread)	147
4.11.6.23 pthread_equal(pthread_t t1, pthread_t t2)	148
4.11.6.24 pthread_exit(void *retval)	148
4.11.6.25 pthread_getattr_np(pthread_t thread, pthread_attr_t *attr)	149
4.11.6.26 pthread_getspecific(pthread_key_t key)	149
4.11.6.27 pthread_join(pthread_t thread, void **retval)	150
4.11.6.28 pthread_key_create(pthread_key_t *key, void(*destructor)(void *))	151
4.11.6.29 pthread_key_delete(pthread_key_t key)	152
4.11.6.30 pthread_kill(pthread_t thread, int sig)	153
4.11.6.31 pthread_mutex_destroy(pthread_mutex_t *mutex)	154
4.11.6.32 pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr)	154
4.11.6.33 pthread_mutex_lock(pthread_mutex_t *mutex)	155
4.11.6.34 pthread_mutex_trylock(pthread_mutex_t *mutex)	156
4.11.6.35 pthread_mutex_unlock(pthread_mutex_t *mutex)	156
4.11.6.36 pthread_mutexattr_destroy(pthread_mutexattr_t *attr)	157
4.11.6.37 pthread_mutexattr_gettype(const pthread_mutexattr_t *attr, int *type)	157
4.11.6.38 pthread_mutexattr_init(pthread_mutexattr_t *attr)	158
4.11.6.39 pthread_mutexattr_settype(pthread_mutexattr_t *attr, int type)	159
4.11.6.40 pthread_once(pthread_once_t *once_control, void(*init_routine)(void))	159
4.11.6.41 pthread_self(void)	160
4.11.6.42 pthread_setspecific(pthread_key_t key, const void *value)	161
4.12 Math library API	162

4.12.1 Detailed Description	164
4.12.2 Macro Definition Documentation	164
4.12.2.1 FP_ILOGB0	164
4.12.2.2 FP_ILOGBNAN	164
4.12.2.3 INFINITY	164
4.12.2.4 isfinite	164
4.12.2.5 isnan	165
4.12.2.6 M_E	165
4.12.2.7 M_PI	165
4.12.2.8 M_PI_2	165
4.12.2.9 M_PI_4	165
4.12.2.10 NAN	165
4.12.3 Function Documentation	166
4.12.3.1 atan(double x)	166
4.12.3.2 atan2(double y, double x)	166
4.12.3.3 atan2f(float y, float x)	166
4.12.3.4 atanf(float x)	167
4.12.3.5 ceil(double x)	167
4.12.3.6 ceilf(float x)	167
4.12.3.7 copysign(double x, double y)	168
4.12.3.8 copysignf(float x, float y)	168
4.12.3.9 cos(double x)	168
4.12.3.10 cosf(float x)	169
4.12.3.11 exp(double x)	169
4.12.3.12 expf(float x)	169
4.12.3.13 fabs(double x)	170
4.12.3.14 fabsf(float x)	170
4.12.3.15 floor(double x)	170
4.12.3.16 floorf(float x)	171
4.12.3.17 fmax(double x, double y)	171

4.12.3.18 fmaxf(float x, float y)	171
4.12.3.19 fmaxl(long double x, long double y)	172
4.12.3.20 fmin(double x, double y)	172
4.12.3.21 fminf(float x, float y)	172
4.12.3.22 hypot(double x, double y)	173
4.12.3.23 hypotf(float x, float y)	173
4.12.3.24 ilogb(double x)	173
4.12.3.25 ilogbf(float x)	174
4.12.3.26 ilogbl(long double x)	174
4.12.3.27 log(double x)	174
4.12.3.28 logb(double x)	175
4.12.3.29 logbf(float x)	175
4.12.3.30 logbl(long double x)	175
4.12.3.31 logf(float x)	176
4.12.3.32 modf(double x, double *iptr)	176
4.12.3.33 modff(float x, float *iptr)	176
4.12.3.34 pow(double base, double exp)	177
4.12.3.35 powf(float base, float exp)	177
4.12.3.36 round(double x)	177
4.12.3.37 roundf(float x)	178
4.12.3.38 scalbn(double x, int exp)	178
4.12.3.39 scalbnf(float x, int exp)	178
4.12.3.40 scalbnl(long double x, int exp)	179
4.12.3.41 sin(double x)	179
4.12.3.42 sinf(float x)	179
4.12.3.43 sqrt(double x)	180
4.12.3.44 sqrtf(float x)	180
4.13 Message queue library API	181
4.13.1 Detailed Description	181
4.13.2 Macro Definition Documentation	181

4.13.2.1 MQ_MAX_NAME	181
4.13.3 Typedef Documentation	181
4.13.3.1 mqd_t	181
4.13.4 Function Documentation	182
4.13.4.1 mq_close(mqd_t fd)	182
4.13.4.2 mq_open(const char *pathname, int flags,...)	182
4.13.4.3 mq_receive(mqd_t fd, char *msg_ptr, size_t msg_len, unsigned *msg_prio)	182
4.13.4.4 mq_send(mqd_t fd, const char *msg_ptr, size_t msg_len, unsigned msg_prio)	183
4.13.4.5 mq_unlink(const char *pathname)	183
4.14 Socket library API	184
4.14.1 Detailed Description	184
4.14.2 Function Documentation	184
4.14.2.1 accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen)	184
4.14.2.2 bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen)	185
4.14.2.3 connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen)	185
4.14.2.4 getsockopt(int sockfd, int level, int optname, void *optval, socklen_t *optlen)	186
4.14.2.5 listen(int sockfd, int backlog)	186
4.14.2.6 recv(int sockfd, void *buf, size_t len, int flags)	186
4.14.2.7 recvmsg(int sockfd, struct msghdr *msg, int flags)	187
4.14.2.8 send(int sockfd, const void *buf, size_t len, int flags)	187
4.14.2.9 sendmsg(int sockfd, struct msghdr *msg, int flags)	187
4.14.2.10 setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen)	188
4.14.2.11 socket(int socket_family, int socket_type, int protocol)	188
4.14.2.12 socketpair(int socket_family, int socket_type, int protocol, int sv[2])	188
4.15 Auxiliary API	190
4.15.1 Detailed Description	201
4.15.2 Data Structure Documentation	201
4.15.2.1 struct cpu_set_t	201
4.15.2.2 struct tm	201

4.15.2.3 struct __uuid_t	201
4.15.3 Macro Definition Documentation	202
4.15.3.1 __CPUMASK	202
4.15.3.2 _POSIX_THREAD_KEYS_MAX	202
4.15.3.3 _POSIX_THREAD_THREADS_MAX	202
4.15.3.4 _POSIX_THREADS	202
4.15.3.5 alloca	202
4.15.3.6 assert	203
4.15.3.7 AUTO_BUFFER_SIZE	203
4.15.3.8 BIT_PER_CPU	203
4.15.3.9 BITMAP_ELT	203
4.15.3.10 BITS_TO_CPU_MASK	203
4.15.3.11 CHAR_BIT	203
4.15.3.12 CPU_CLR	204
4.15.3.13 CPU_ISSET	204
4.15.3.14 CPU_SET	204
4.15.3.15 CPU_ZERO	205
4.15.3.16 DECLARE_BITMAP	205
4.15.3.17 E2BIG	205
4.15.3.18 EACCES	205
4.15.3.19 EADDRINUSE	205
4.15.3.20 EADDRNOTAVAIL	205
4.15.3.21 EADV	206
4.15.3.22 EAFNOSUPPORT	206
4.15.3.23 EAGAIN	206
4.15.3.24 EALREADY	206
4.15.3.25 EBADE	206
4.15.3.26 EBADF	206
4.15.3.27 EBADFD	206
4.15.3.28 EBADMSG	207

4.15.3.29EBADR	207
4.15.3.30EBADRQC	207
4.15.3.31 EBADSLT	207
4.15.3.32EBFONT	207
4.15.3.33EBUSY	207
4.15.3.34ECANCELED	207
4.15.3.35ECHILD	208
4.15.3.36ECHRNG	208
4.15.3.37ECOMM	208
4.15.3.38ECONNABORTED	208
4.15.3.39ECONNREFUSED	208
4.15.3.40ECONNRESET	208
4.15.3.41 EDEADLK	208
4.15.3.42EDEADLOCK	209
4.15.3.43EDESTADDRREQ	209
4.15.3.44EDOM	209
4.15.3.45EDOTDOT	209
4.15.3.46EDQUOT	209
4.15.3.47EEXIST	209
4.15.3.48EFAULT	209
4.15.3.49EFBIG	210
4.15.3.50EHOSTDOWN	210
4.15.3.51 EHOSTUNREACH	210
4.15.3.52EIDRM	210
4.15.3.53EILSEQ	210
4.15.3.54EINPROGRESS	210
4.15.3.55EINTR	210
4.15.3.56EINVAL	211
4.15.3.57EIO	211
4.15.3.58EISCONN	211

4.15.3.59EISDIR	211
4.15.3.60EISNAM	211
4.15.3.61 EKEYREJECTED	211
4.15.3.62EL2HLT	211
4.15.3.63EL2NSYNC	212
4.15.3.64EL3HLT	212
4.15.3.65EL3RST	212
4.15.3.66ELIBACC	212
4.15.3.67ELIBBAD	212
4.15.3.68ELIBEXEC	212
4.15.3.69ELIBMAX	212
4.15.3.70ELIBSCN	213
4.15.3.71 ELNRNG	213
4.15.3.72ELOOP	213
4.15.3.73EMFILE	213
4.15.3.74EMLINK	213
4.15.3.75EMSGSIZE	213
4.15.3.76EMULTIHOP	213
4.15.3.77 ENAMETOOLONG	214
4.15.3.78 ENAVAIL	214
4.15.3.79 ENETDOWN	214
4.15.3.80ENETRESET	214
4.15.3.81 ENETUNREACH	214
4.15.3.82ENFILE	214
4.15.3.83ENOANO	214
4.15.3.84ENOBUFFS	215
4.15.3.85ENOCSI	215
4.15.3.86ENODATA	215
4.15.3.87ENODEV	215
4.15.3.88ENOENT	215

4.15.3.89ENOEXEC	215
4.15.3.90ENOKEY	215
4.15.3.91ENOLCK	216
4.15.3.92ENOLINK	216
4.15.3.93ENOMEM	216
4.15.3.94ENOMSG	216
4.15.3.95ENONET	216
4.15.3.96ENOPKG	216
4.15.3.97ENOPROTOOPT	216
4.15.3.98ENOSPC	217
4.15.3.99ENOSR	217
4.15.3.100ENOSTR	217
4.15.3.101ENOSYS	217
4.15.3.102ENOTBLK	217
4.15.3.103ENOTCONN	217
4.15.3.104ENOTDIR	217
4.15.3.105NOTEMPTY	218
4.15.3.106NOTNAM	218
4.15.3.107NOTRECOVERABLE	218
4.15.3.108NOTSOCK	218
4.15.3.109NOTTTY	218
4.15.3.110NOTUNIQ	218
4.15.3.111ENXIO	218
4.15.3.112EOF	219
4.15.3.113EOPNOTSUPP	219
4.15.3.114EOVERFLOW	219
4.15.3.115EPERM	219
4.15.3.116EPFNOSUPPORT	219
4.15.3.117EPIPE	219
4.15.3.118EPROTO	219

4.15.3.11 E PROTONOSUPPORT	220
4.15.3.12 E PROTOTYPE	220
4.15.3.12 E RANGE	220
4.15.3.12 E REMCHG	220
4.15.3.12 E REMOTE	220
4.15.3.12 E REMOTEIO	220
4.15.3.12 E RESTART	220
4.15.3.12 E ROFS	221
4.15.3.12 E rrno	221
4.15.3.12 E SHUTDOWN	221
4.15.3.12 E SOCKTNOSUPPORT	221
4.15.3.13 E SPIPE	221
4.15.3.13 E SRCH	221
4.15.3.13 E SRMNT	221
4.15.3.13 E STALE	222
4.15.3.13 E STRPIPE	222
4.15.3.13 E TIME	222
4.15.3.13 E TIMEDOUT	222
4.15.3.13 E TOOMANYREFS	222
4.15.3.13 E TXTBSY	222
4.15.3.13 E UCLEAN	222
4.15.3.14 E UNATCH	223
4.15.3.14 E USERS	223
4.15.3.14 E WOULDBLOCK	223
4.15.3.14 E XDEV	223
4.15.3.14 E XFULL	223
4.15.3.14 E NT_MAX	223
4.15.3.14 E NT_MIN	223
4.15.3.14 E LONG_MAX	224
4.15.3.14 E LONG_MIN	224

4.15.3.14	LONG_MAX	224
4.15.3.15	LONG_MIN	224
4.15.3.15	M_CACHE_PAGES	224
4.15.3.15	MAP_ANONYMOUS	224
4.15.3.15	MAP_FAILED	224
4.15.3.15	MAP_FIXED	225
4.15.3.15	MAP_GROWSDOWN	225
4.15.3.15	MAP_PHYS_NON_CACHED	225
4.15.3.15	MAP_PHYS_NON_SECURE	225
4.15.3.15	MAP_POPULATE	225
4.15.3.15	MAP_PRIVATE	225
4.15.3.16	MAP_SHARED	225
4.15.3.16	MAP_STACK	226
4.15.3.16	MAX_CPUS	226
4.15.3.16	NUM_MILLIS_IN_SEC	226
4.15.3.16	NUM_NANOS_IN_MILLI	226
4.15.3.16	NUM_NANOS_IN_SEC	226
4.15.3.16	NUM_NANOS_IN_USEC	226
4.15.3.16	NUM_SECONDS_IN_MIN	226
4.15.3.16	PGOFF_SHIFT	227
4.15.3.16	PRid16	227
4.15.3.17	PRid32	227
4.15.3.17	PRid64	227
4.15.3.17	PRlu16	227
4.15.3.17	PRlu32	227
4.15.3.17	PRlu64	227
4.15.3.17	PRix16	228
4.15.3.17	PRix32	228
4.15.3.17	PRix64	228
4.15.3.17	PROT_EXEC	228

4.15.3.17	P ROT_NONE	228
4.15.3.18	P ROT_READ	228
4.15.3.18	P ROT_WRITE	228
4.15.3.18	P THREAD_KEYS_MAX	229
4.15.3.18	S CHAR_MAX	229
4.15.3.18	S CHAR_MIN	229
4.15.3.18	S CNd16	229
4.15.3.18	S CNd32	229
4.15.3.18	S CNd64	229
4.15.3.18	S CNu16	229
4.15.3.18	S CNu32	230
4.15.3.19	S CNu64	230
4.15.3.19	S CNx16	230
4.15.3.19	S CNx32	230
4.15.3.19	S CNx64	230
4.15.3.19	S HRT_MAX	230
4.15.3.19	S HRT_MIN	230
4.15.3.19	T EMP_FAILURE_RETRY	231
4.15.3.19	T CHAR_MAX	231
4.15.3.19	T INT_MAX	231
4.15.3.19	T LLONG_MAX	231
4.15.3.20	T LONG_MAX	231
4.15.3.20	T SHRT_MAX	232
4.15.3.20	T uid_generate_time	232
4.15.3.20	T UID_STRING_LEN	232
4.15.3.20	T uid_unparse_lower	232
4.15.4	Typedef Documentation	232
4.15.4.1	T _uid_t	232
4.15.4.2	T constraint_handler_t	232
4.15.4.3	T errno_t	232

4.15.4.4	gid_t	233
4.15.4.5	mode_t	233
4.15.4.6	off_t	233
4.15.4.7	pid_t	233
4.15.4.8	ssize_t	233
4.15.4.9	time_t	233
4.15.4.10	uid_t	233
4.15.4.11	uuid_t	234
4.15.5	Enumeration Type Documentation	234
4.15.5.1	anonymous enum	234
4.15.6	Function Documentation	234
4.15.6.1	_exit(int status)	234
4.15.6.2	abort_handler_s(const char *restrict msg, void *restrict ptr, errno_t error)	235
4.15.6.3	abs(int j)	235
4.15.6.4	arm_timer_get_frequency(void)	235
4.15.6.5	asprintf(char **strp, const char *fmt,...)	236
4.15.6.6	atexit(void(*func)(void))	236
4.15.6.7	atof(const char *nptr)	237
4.15.6.8	atoi(const char *nptr)	237
4.15.6.9	calloc(size_t nmemb, size_t size)	237
4.15.6.10	clock_gettime(clockid_t clk_id, struct timespec *ts)	238
4.15.6.11	close(int fd)	238
4.15.6.12	exit(int status)	239
4.15.6.13	fflush(FILE *stream)	239
4.15.6.14	fprintf(FILE *_restrict_stream, const char *_restrict_fmt,...)	240
4.15.6.15	free(void *ptr)	240
4.15.6.16	fstat(int fd, struct stat *buf)	241
4.15.6.17	fsync(int fd)	241
4.15.6.18	ftruncate(int fd, int size)	241
4.15.6.19	get_errno_addr(void)	242

4.15.6.20getcluster(void)	242
4.15.6.21 getcpu(void)	242
4.15.6.22getenv(const char *name)	242
4.15.6.23getpid(void)	243
4.15.6.24gettid(void)	243
4.15.6.25ignore_handler_s(const char *restrict msg, void *restrict ptr, errno_t error)	243
4.15.6.26invoke_constraint_handler_s(const char *msg, const char *file, const char *function, uint32_t line, errno_t error)	243
4.15.6.27ioctl(int fd, int request, unsigned long data)	244
4.15.6.28isalnum(int c)	244
4.15.6.29isalpha(int c)	245
4.15.6.30isascii(int c)	245
4.15.6.31 isblank(int c)	245
4.15.6.32isctrl(int c)	246
4.15.6.33isdigit(int c)	246
4.15.6.34isgraph(int c)	246
4.15.6.35islower(int c)	247
4.15.6.36isprint(int c)	247
4.15.6.37ispunct(int c)	247
4.15.6.38isspace(int c)	248
4.15.6.39isupper(int c)	248
4.15.6.40isxdigit(int c)	248
4.15.6.41 lseek(int fd, int offset, int whence)	249
4.15.6.42malloc(size_t size)	249
4.15.6.43memchr(const void *s, int c, size_t n)	250
4.15.6.44memcmp(const void *s1, const void *s2, size_t n)	250
4.15.6.45memcpy(void *dest, const void *src, size_t n)	250
4.15.6.46memcpy_s(void *restrict dest, rsize_t dest_max, const void *restrict src, rsize_t n)	251
4.15.6.47memmove(void *dest, const void *src, size_t n)	251
4.15.6.48memmove_s(void *dest, rsize_t dest_max, const void *src, rsize_t n) . . .	252

4.15.6.49	<code>memrchr(const void *s, int c, size_t n)</code>	252
4.15.6.50	<code>memset(void *block, int c, size_t size)</code>	252
4.15.6.51	<code>memset_s(void *block, rsize_t block_max, int c, rsize_t n)</code>	253
4.15.6.52	<code>mmap(void *addr, size_t len, int prot, int flags, int fd, off_t offset)</code>	253
4.15.6.53	<code>mprotect(void *addr, size_t len, int prot)</code>	255
4.15.6.54	<code>ms_to_timespec(int64_t t, struct timespec *a)</code>	255
4.15.6.55	<code>munmap(void *addr, size_t length)</code>	256
4.15.6.56	<code>nanosleep(const struct timespec *req, struct timespec *rem)</code>	256
4.15.6.57	<code>open(const char *pathname, int flags,...)</code>	257
4.15.6.58	<code>printf(const char *fmt,...)</code>	257
4.15.6.59	<code>printf_no_alloc(const char *fmt,...)</code>	258
4.15.6.60	<code>printf_s(const char *_restrict_ fmt,...)</code>	259
4.15.6.61	<code>profil(unsigned short *buf, size_t bufsiz, size_t offset, unsigned int scale)</code>	259
4.15.6.62	<code>putchar(int ch)</code>	260
4.15.6.63	<code>puts(const char *s)</code>	260
4.15.6.64	<code>qsort(void *base, size_t nmemb, size_t size, int(*compar)(const void *, const void *))</code>	260
4.15.6.65	<code>qsort_r(void *base, size_t nmemb, size_t size, int(*compar)(const void *, const void *, void *), void *arg)</code>	261
4.15.6.66	<code>read(int fd, void *buf, size_t count)</code>	261
4.15.6.67	<code>realloc(void *ptr, size_t size)</code>	261
4.15.6.68	<code>rename(const char *pathname, const char *new_pathname)</code>	262
4.15.6.69	<code>rmdir(char *dir_name)</code>	263
4.15.6.70	<code>sched_getaffinity(pid_t pid, size_t cpusetsize, cpu_set_t *mask)</code>	263
4.15.6.71	<code>sched_setaffinity(pid_t pid, size_t cpusetsize, cpu_set_t *mask)</code>	263
4.15.6.72	<code>sched_yield(void)</code>	264
4.15.6.73	<code>set_constraint_handler_s(constraint_handler_t handler)</code>	264
4.15.6.74	<code>setenv(const char *name, const char *value, int overwrite)</code>	265
4.15.6.75	<code>snprintf(char *s, size_t count, const char *fmt,...)</code>	265
4.15.6.76	<code>snprintf_s(char *_restrict_ s, rsize_t n, const char *_restrict_ format,...)</code>	266
4.15.6.77	<code>sprintf(char *s, const char *fmt,...)</code>	266

4.15.6.78	<code>sprintf_s(char *_restrict_s, rsize_t n, const char *_restrict_format,...)</code>	267
4.15.6.79	<code>sscanf(const char *buf, const char *fmt,...)</code>	267
4.15.6.80	<code>sscanf_s(const char *_restrict_s, const char *_restrict_format,...)</code>	268
4.15.6.81	<code>stat(const char *pathname, struct stat *buf)</code>	268
4.15.6.82	<code>strcat(char *dest, const char *src)</code>	269
4.15.6.83	<code>strcat_s(char *restrict dest, rsize_t dest_max, const char *restrict src)</code>	269
4.15.6.84	<code>strchr(const char *s, int c)</code>	270
4.15.6.85	<code>strchrnul(const char *s, int c)</code>	270
4.15.6.86	<code>strcmp(const char *s1, const char *s2)</code>	271
4.15.6.87	<code>strcpy(char *dest, const char *src)</code>	271
4.15.6.88	<code>strcpy_s(char *restrict dest, rsize_t dest_max, const char *restrict src)</code>	272
4.15.6.89	<code>strcspn(const char *s, const char *reject)</code>	272
4.15.6.90	<code>strdup(const char *s)</code>	273
4.15.6.91	<code>strerror(int errnum)</code>	273
4.15.6.92	<code>strerror_r(int errnum, char *buf, size_t buflen)</code>	273
4.15.6.93	<code>strerror_s(char *buf, rsize_t bufmax, errno_t errnum)</code>	274
4.15.6.94	<code>strlcat(char *dest, const char *src, size_t n)</code>	274
4.15.6.95	<code>strlen(const char *s)</code>	275
4.15.6.96	<code>strncat(char *dest, const char *src, size_t n)</code>	275
4.15.6.97	<code>strncat_s(char *restrict dest, rsize_t dest_max, const char *restrict src, rsize_t n)</code>	276
4.15.6.98	<code>strncmp(const char *s1, const char *s2, size_t n)</code>	276
4.15.6.99	<code>strncpy(char *dest, const char *src, size_t n)</code>	277
4.15.6.100	<code>strncpy_s(char *restrict dest, rsize_t dest_max, const char *restrict src, rsize_t n)</code>	278
4.15.6.101	<code>strnlen(const char *s, size_t n)</code>	278
4.15.6.102	<code>strnlen_s(const char *s, size_t s_max)</code>	279
4.15.6.103	<code>strrchr(const char *s, int c)</code>	279
4.15.6.104	<code>strspn(const char *s, const char *accept)</code>	280
4.15.6.105	<code>strstr(const char *text, const char *pattern)</code>	280
4.15.6.106	<code>strtod(const char *nptr, char **endptr)</code>	281

4.15.6.103	strtof(const char *nptr, char **endptr)	281
4.15.6.108	strtok(char *str, const char *delim)	282
4.15.6.109	strtok_r(char *_restrict_ s, const char *_restrict_ sep, char **_restrict_ p)	283
4.15.6.110	strtol(const char *nptr, char **endptr, int base)	283
4.15.6.111	strtold(const char *nptr, char **endptr)	284
4.15.6.112	strtoll(const char *nptr, char **endptr, int base)	284
4.15.6.113	strtoul(const char *cp, char **endp, int base)	285
4.15.6.114	strtoull(const char *cp, char **endp, int base)	286
4.15.6.115	sysconf(int name)	286
4.15.6.116	time(time_t *time)	287
4.15.6.117	timeadd(const struct timespec *a, const struct timespec *b, struct time- spec *res)	287
4.15.6.118	timespec_to_ms(const struct timespec *a)	287
4.15.6.119	timespec_to_nsec(const struct timespec *a)	288
4.15.6.120	timesub(const struct timespec *a, const struct timespec *b, struct time- spec *res)	288
4.15.6.121	tolower(int c)	288
4.15.6.122	toupper(int c)	289
4.15.6.123	unlink(const char *pathname)	289
4.15.6.124	unsetenv(const char *name)	289
4.15.6.125	uuid_clear(uuid_t *uu)	290
4.15.6.126	uuid_compare(const uuid_t *uu1, const uuid_t *uu2)	290
4.15.6.127	uuid_generate(uuid_t *out)	290
4.15.6.128	uuid_is_null(const uuid_t *uu)	291
4.15.6.129	uuid_parse(const char *in, uuid_t *uu)	291
4.15.6.130	uuid_unparse(const uuid_t *uu, char *out)	291
4.15.6.131	uuid_unparse_upper(const uuid_t *uu, char *out)	292
4.15.6.132	vasprintf(char **strp, const char *fmt, va_list args)	292
4.15.6.133	vasprintf_s(char **strp, const char *fmt, va_list args)	292
4.15.6.134	vfprintf(FILE *_restrict_ stream, const char *_restrict_ fmt, va_list ap)	293
4.15.6.135	vsprintf(char *buffer, size_t size, const char *format, va_list args)	293

4.15.6.136	vsprintf_s(char *_restrict_ s, rsize_t n, const char *_restrict_ format, va_list arg)	294
4.15.6.137	vsprintf(char *buffer, const char *format, va_list args)	295
4.15.6.138	vsprintf_s(char *_restrict_ s, rsize_t n, const char *_restrict_ format, va_list arg)	295
4.15.6.139	vsscanf(const char *buffer, const char *fmt, va_list args)	296
4.15.6.140	vsscanf_s(const char *_restrict_ s, const char *_restrict_ format, va_list arg)	296
4.15.6.141	write(int fd, const void *buf, size_t count)	297
4.15.7	Variable Documentation	297
4.15.7.1	stdin	297
4.16	Stat	298
4.16.1	Detailed Description	298
4.16.2	Macro Definition Documentation	298
4.16.2.1	S_ACCPERM	298
4.16.3	Function Documentation	298
4.16.3.1	fstat(int fd, struct stat *buf)	298
4.16.3.2	mkdir(const char *pathname, mode_t mode)	299
4.16.3.3	stat(const char *pathname, struct stat *buf)	299
4.17	TeesL_NFC	300
4.17.1	Detailed Description	300
4.17.2	Typedef Documentation	300
4.17.2.1	TEES_NFCHandler	300
4.17.3	Function Documentation	300
4.17.3.1	TEES_NFCExit(TEES_NFCHandler handler)	300
4.17.3.2	TEES_NFCInit(TEES_NFCHandler *handler)	301
4.17.3.3	TEES_NFCRead(TEES_NFCHandler handler, char *rx_buf, size_t rx_buf_len, size_t *rsp_len)	301
4.17.3.4	TEES_NFCReadWait(TEES_NFCHandler handler, char *rx_buf, size_t rx_buf_len, size_t *rsp_len, int ms_timeout)	301
4.17.3.5	TEES_NFCSetMode(TEES_NFCHandler handler, unsigned long mode)	302
4.17.3.6	TEES_NFCSleep(TEES_NFCHandler handler)	302
4.17.3.7	TEES_NFCWakeUp(TEES_NFCHandler handler)	303

4.17.3.8	TEES_NFCWrite(TEES_NFCHandler handler, void *tx_buf, size_t len) . . .	303
4.18	Teesl_smc	304
4.18.1	Detailed Description	304
4.18.2	Function Documentation	304
4.18.2.1	TEES_SMCCCommand(TEES_SMCHandle handle, TEES_SMCData *data) .	304
4.18.2.2	TEES_SMCFini(TEES_SMCHandle handle)	305
4.18.2.3	TEES_SMCInit(TEES_SMCHandle *handle)	305
4.19	Teesl_udf	306
4.19.1	Detailed Description	306
4.19.2	Function Documentation	306
4.19.2.1	TEES_UDF_FiniDriver(struct usr_drv_info *info)	306
4.19.2.2	TEES_UDF_Init_FS_Driver(char *name, struct fops *fops, unsigned int drvid, struct usr_drv_info **info)	306
4.19.2.3	TEES_UDF_InitDriver(char *name, struct fops *fops, unsigned int drvid, struct usr_drv_info **info)	307
4.19.2.4	TEES_UDF_RegisterIoctlDesc(struct usr_drv_info *info, unsigned int cmd, const struct ioctl_desc *desc)	307
4.20	Tee_sockets	309
4.20.1	Detailed Description	314
4.20.2	Data Structure Documentation	314
4.20.2.1	struct TEE_iSocket_s	314
4.20.2.2	struct TEE_tcpSocket_Setup_s	316
4.20.2.3	struct TEE_udpSocket_Setup_s	316
4.20.2.4	struct TEE_udpSocket_Change_s	317
4.20.2.5	struct TEE_tlsSocket_PSK_Info_s	317
4.20.2.6	struct TEE_tlsSocket_SRP_Info_s	317
4.20.2.7	struct TEE_tlsSocket_ClientPDC_s	317
4.20.2.8	struct TEE_tlsSocket_ServerPDC_s	318
4.20.2.9	struct TEE_tlsSocket_CertStorageCred_s	318
4.20.2.10	struct TEE_tlsSocket_Credentials_s	318
4.20.2.11	union TEE_tlsSocket_Credentials_s.__unnamed__	318
4.20.2.12	struct TEE_tlsSocket_CallbackInfo_s	319

4.20.2.13	struct TEE_tlsSocket_Setup_s	319
4.20.2.14	union TEE_tlsSocket_Setup_s.__unnamed__	321
4.20.2.15	struct TEE_tlsSocket_CB_Data_s	321
4.20.3	Typedef Documentation	321
4.20.3.1	TEE_iSocket	321
4.20.3.2	TEE_tlsSocket_CB_Data	322
4.20.4	Enumeration Type Documentation	322
4.20.4.1	anonymous enum	322
4.20.4.2	anonymous enum	322
4.20.4.3	anonymous enum	322
4.20.4.4	anonymous enum	323
4.20.4.5	anonymous enum	323
4.20.4.6	anonymous enum	323
4.20.4.7	anonymous enum	323
4.20.4.8	anonymous enum	323
4.20.4.9	anonymous enum	324
4.20.4.10	anonymous enum	324
4.20.4.11	anonymous enum	324
4.20.4.12	anonymous enum	324
4.20.4.13	protocol_error_code	325
4.20.4.14	TEE_ipSocket_ipVersion_e	327
4.20.4.15	TEE_tlsSocket_CallbackReasonType_e	327
4.20.4.16	TEE_tlsSocket_ClientCredentialType_e	327
4.20.4.17	TEE_tlsSocket_ExtensionFlags_e	327
4.20.4.18	TEE_tlsSocket_ServerCredentialType_e	328
4.20.4.19	TEE_tlsSocket_StatusRequestType_e	328
4.20.4.20	TEE_tlsSocket_tlsVersion_e	328
4.20.5	Function Documentation	328
4.20.5.1	TEES_lwtCloseChannel(TEES_lwtHandle iwtCtx)	328
4.20.5.2	TEES_lwtOpenChannel(const char *listenerName, TEES_lwtHandle *iwtCtx)	329
4.20.5.3	TEES_lwtRead(TEES_lwtHandle iwtCtx, void *buf, uint32_t *length)	329
4.20.5.4	TEES_lwtWrite(TEES_lwtHandle iwtCtx, const void *buf, uint32_t *length)	330

5	Data Structure Documentation	331
5.1	__TEE_SC_CardKeyRef	331
5.1.1	Detailed Description	331
5.1.2	Field Documentation	331
5.1.2.1	scKeyID	331
5.1.2.2	scKeyVersion	331
5.2	__TEE_SC_DeviceKeyRef	331
5.2.1	Detailed Description	331
5.2.2	Field Documentation	332
5.2.2.1	"@8	332
5.2.2.2	scKeyType	332
5.3	__TEE_SC_DeviceKeyRef.__unnamed__	332
5.3.1	Field Documentation	332
5.3.1.1	scBaseKeyHandle	332
5.3.1.2	scKeySetRef	332
5.4	__TEE_SC_KeySetRef	332
5.4.1	Detailed Description	332
5.4.2	Field Documentation	333
5.4.2.1	scKeyEncHandle	333
5.4.2.2	scKeyMacHandle	333
5.5	__TEE_SC_OID	333
5.5.1	Detailed Description	333
5.5.2	Field Documentation	333
5.5.2.1	buffer	333
5.5.2.2	bufferLen	333
5.6	__TEE_SC_Params	333
5.6.1	Detailed Description	334
5.6.2	Field Documentation	334
5.6.2.1	scCardKeyRef	334
5.6.2.2	scDeviceKeyRef	334

5.6.2.3	scOID	334
5.6.2.4	scSecurityLevel	334
5.6.2.5	scType	334
5.7	__TEE_SEAID	334
5.7.1	Detailed Description	334
5.7.2	Field Documentation	335
5.7.2.1	buffer	335
5.7.2.2	bufferLen	335
5.8	__TEE_SEReaderProperties	335
5.8.1	Detailed Description	335
5.8.2	Field Documentation	335
5.8.2.1	selectResponseEnable	335
5.8.2.2	sePresent	335
5.8.2.3	teeOnly	335
5.9	desc_atom	335
5.9.1	Detailed Description	336
5.9.2	Field Documentation	336
5.9.2.1	len	336
5.9.2.2	type	336
5.10	fops	336
5.10.1	Detailed Description	337
5.10.2	Field Documentation	337
5.10.2.1	close	337
5.10.2.2	fsync	337
5.10.2.3	ioctl	337
5.10.2.4	lookup	337
5.10.2.5	lseek	337
5.10.2.6	mkdir	337
5.10.2.7	mmap	337
5.10.2.8	open	337

5.10.2.9 probe	338
5.10.2.10 read	338
5.10.2.11 readdir	338
5.10.2.12 rename	338
5.10.2.13 rmdir	338
5.10.2.14 stat	338
5.10.2.15 truncate	338
5.10.2.16 unlink	338
5.10.2.17 write	338
5.11 ioctl_desc	338
5.11.1 Detailed Description	339
5.11.2 Field Documentation	339
5.11.2.1 cnt	339
5.11.2.2 tpl	339
5.11.2.3 type	339
5.12 smc_data	339
5.12.1 Detailed Description	339
5.12.2 Field Documentation	339
5.12.2.1 arg_types	339
5.12.2.2 args	340
5.13 stat	340
5.13.1 Detailed Description	340
5.13.2 Field Documentation	340
5.13.2.1 st_gid	340
5.13.2.2 st_mode	340
5.13.2.3 st_refcount	340
5.13.2.4 st_size	340
5.13.2.5 st_uid	340
5.14 usr_drv_info	341
5.14.1 Detailed Description	341
5.14.2 Field Documentation	341
5.14.2.1 fops	341
5.14.2.2 handle	341
5.14.2.3 priv	341

6 File Documentation	342
6.1 <code>alloca.h</code> File Reference	342
6.1.1 Detailed Description	342
6.2 <code>assert.h</code> File Reference	342
6.2.1 Detailed Description	342
6.3 <code>cache.h</code> File Reference	342
6.3.1 Detailed Description	343
6.4 <code>core/error.h</code> File Reference	343
6.5 <code>core/mman.h</code> File Reference	345
6.6 <code>sys/mman.h</code> File Reference	346
6.6.1 Detailed Description	346
6.7 <code>sys/stat.h</code> File Reference	346
6.7.1 Detailed Description	347
6.8 <code>ctype.h</code> File Reference	347
6.8.1 Detailed Description	348
6.9 <code>driver.h</code> File Reference	348
6.9.1 Detailed Description	348
6.10 <code>driver/mem/phys.h</code> File Reference	349
6.11 <code>errno.h</code> File Reference	349
6.11.1 Detailed Description	349
6.12 <code>fcntl.h</code> File Reference	349
6.12.1 Detailed Description	350
6.13 <code>inttypes.h</code> File Reference	350
6.13.1 Detailed Description	350
6.14 <code>irs.h</code> File Reference	351
6.14.1 Detailed Description	352
6.15 <code>limits.h</code> File Reference	352
6.15.1 Detailed Description	352
6.16 <code>malloc.h</code> File Reference	353
6.16.1 Detailed Description	353

6.17 math.h File Reference	353
6.17.1 Detailed Description	355
6.18 mqueue.h File Reference	355
6.18.1 Detailed Description	356
6.19 print_no_alloc.h File Reference	356
6.19.1 Detailed Description	356
6.20 protocol_errors.h File Reference	356
6.20.1 Detailed Description	358
6.21 pthread.h File Reference	358
6.21.1 Detailed Description	361
6.22 rpmb.h File Reference	361
6.22.1 Detailed Description	362
6.23 sched.h File Reference	362
6.23.1 Detailed Description	362
6.24 scma.h File Reference	363
6.24.1 Detailed Description	363
6.25 stdio.h File Reference	364
6.25.1 Detailed Description	365
6.26 stdlib.h File Reference	365
6.26.1 Detailed Description	367
6.27 string.h File Reference	367
6.27.1 Detailed Description	369
6.28 sys/credentials.h File Reference	369
6.28.1 Detailed Description	369
6.28.2 Function Documentation	369
6.28.2.1 cred_compare(const struct cred *as_is, const struct cred *to_be)	369
6.29 sys/ioctl.h File Reference	369
6.29.1 Detailed Description	370
6.30 sys/socket.h File Reference	370
6.30.1 Detailed Description	371

6.31	sys/types.h File Reference	371
6.31.1	Detailed Description	371
6.32	sys/un.h File Reference	371
6.32.1	Detailed Description	371
6.33	tee_error.h File Reference	372
6.33.1	Detailed Description	372
6.34	tee_i2c.h File Reference	372
6.34.1	Detailed Description	373
6.35	tee_interrupt.h File Reference	373
6.35.1	Detailed Description	373
6.36	tee_isocket.h File Reference	373
6.36.1	Detailed Description	374
6.37	tee_nfc.h File Reference	374
6.37.1	Detailed Description	375
6.38	tee_smc.h File Reference	375
6.38.1	Detailed Description	376
6.39	tee_spi.h File Reference	376
6.39.1	Detailed Description	377
6.40	tee_ta_destructor.h File Reference	377
6.40.1	Detailed Description	377
6.41	tee_tcpsocket.h File Reference	378
6.41.1	Detailed Description	378
6.42	tee_tlssocket.h File Reference	378
6.42.1	Detailed Description	382
6.43	tee_udpsocket.h File Reference	382
6.43.1	Detailed Description	383
6.44	tees_critical.h File Reference	383
6.44.1	Detailed Description	384
6.45	tees_el2if.h File Reference	384
6.45.1	Detailed Description	384

6.46	tees_extension.h File Reference	384
6.46.1	Detailed Description	385
6.47	tees_hwcrypto_buf.h File Reference	385
6.47.1	Detailed Description	385
6.48	tees_iwt.h File Reference	386
6.48.1	Detailed Description	386
6.49	tees_kdf.h File Reference	386
6.49.1	Detailed Description	387
6.50	tees_rot.h File Reference	387
6.50.1	Detailed Description	387
6.51	tees_secure_object.h File Reference	387
6.51.1	Detailed Description	388
6.52	tees_ssapi.h File Reference	389
6.52.1	Detailed Description	390
6.53	tees_tui.h File Reference	390
6.53.1	Detailed Description	392
6.54	tees_wrapped_with_rek.h File Reference	392
6.54.1	Detailed Description	393
6.55	time.h File Reference	393
6.55.1	Detailed Description	394
6.56	tui.h File Reference	394
6.56.1	Detailed Description	395
6.57	udf.h File Reference	395
6.57.1	Detailed Description	396
6.58	unistd.h File Reference	396
6.58.1	Detailed Description	397
6.59	uuid/uuid.h File Reference	397
6.59.1	Detailed Description	398
Bibliography		399
Index		399

Version	Date:	Change Contents	Author	Approver
1.0		Initial version	D.Mazaeva	
2.0	2016.06.20	Autobuild documentation	V.Dybala	

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_KEEPA_LIVE	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.

GPD.TEE.DBG_DLM.DATA_AVAILABLE - states what sort of DLM data can be retrieved, according to each TEE.

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.

GPD.TEE.DBG_PMR.DATA_AVAILABLE - states what sort of PMR data shall be retrievable, according to each TEE.

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".

gpd.ta.instanceKeepAlive - type 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 `gpd.ta.singleInstance` 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.

`gpd.ta.multiSession` - type Boolean. Whether the Trusted Application instance supports multiple sessions. This property is ignored when `gpd.ta.singleInstance` is set to false.

`gpd.ta.privilege.ta_authority` - type String. Group name.

`gpd.ta.singleInstance` - type 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).

`gpd.ta.stackSize` - type 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.

`gpd.ta.version` - type String. Version number of this Trusted Application.

`gpd.tee.debug.debug_unlock_properties` - type Boolean. If true, then modifiable Debug Rules Properties may be changed. If false or not present, then modifiable Debug Rules Properties shall not be changeable. This property is itself a modifiable Debug Rules Property and so setting this false prevents resetting it to true.

`gpd.tee.debug.DLM_available` - type Boolean. If true, then the DLM extension has been installed on this device. If false or not present, then the DLM extension has not been installed on this device. Note that just because the DLM extension is installed does not mean that debug log messages shall be transferred through the system, as they may be ignored due to other Debug Rules Properties of TAs and the TEE.

`samsung.client.FIPS_mode_enable` - type Boolean. If true FIPS mode is enable.

`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 $\text{sys.ta.rlim_max} / 2$.

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

Global **TEES_TUIGetTouchEvent** (**TEES_TUITouchInfo** *touchInfo, uint32_t timeout)

Use **TEES_TUIGetMTEvent()** instead 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.

3 Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

__TEE_SC_CardKeyRef	331
__TEE_SC_DeviceKeyRef	331
__TEE_SC_DeviceKeyRef.__unnamed__	332
__TEE_SC_KeySetRef	332
__TEE_SC_OID	333
__TEE_SC_Params	333
__TEE_SEAID	334
__TEE_SEReaderProperties	335
desc_atom	
Structure that contains description of single data item	335
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	336
ioctl_desc	
Structure that template for ioctl() parameters parsing	338
smc_data	
SMC command description	339
stat	340
usr_drv_info	
Structure that contains information describing the driver	341

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 [TEES_El2if_Args](#)
Arguments for EL2 SMC call. [More...](#)
- struct [rot_t](#)
Structure to handle Root of Trust information. [More...](#)
- struct [wrapped_wkth_rek_t](#)
Structure for wrapping with REK. [More...](#)
- struct [secHDCPKeyInfo_t](#)
Key information for HDCP. [More...](#)

Macros

- `#define EL2IF_DEV_NAME "/dev/el2if"`
Driver name for EL2IF.
- `#define EL2IF_ARGS_NUM 3`
Number of EL2IF arguments.
- `#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_STATIC ((SO_TAG_LEN) + (SO_IV_LEN) + (SO_AC_LEN) + (SO_MAGIC_NUMBER_LEN))`
- `#define SO_OUT_BUF_SIZE(in_len, delegated) ((in_len) + SO_HEADER_SIZE_STATIC + (((delegated) ? (SO_TA_ID_LEN + SO_AUTH_ID_LEN) : 0))`
- `#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`
- `#define SECCAM_SECURE 0x0000`
Successful return of SMC for SECCAM_GetStatus.
- `#define SECCAM_NORMAL 0x9101`
Unsuccessful return of SMC for SECCAM_GetStatus.

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.
- enum { [TUI_SET_INFO](#) = 1, [TUI_CLEAR_INFO](#) }
TUI_SETINFO commands.

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.
- int [TEES_El2if](#) (struct [TEES_El2if_Args](#) *args)
Send SMC call to EL2.
- TEE_Result [TEES_LockHWCryptoBuf](#) (void)
Lock HW crypto buffer.
- TEE_Result [TEES_UnlockHWCryptoBuf](#) (void)
Unock HW crypto buffer.
- 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_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.

- TEE_Result [TEES_SECCAM_Protect](#) (uint64_t paddr, size_t size, unsigned int *data)
Protect the memory used by secure camera.
- TEE_Result [TEES_SECCAM_Unprotect](#) (uint64_t paddr, size_t size, unsigned int *data)
Unprotect the memory used by secure camera.
- TEE_Result [TEES_SECCAM_IsProtected](#) (uint64_t paddr, size_t size, unsigned int *data)
Check the memory used by secure camera whether it is protected or not.
- TEE_Result [TEES_HDCP_SetKeyInfo](#) (struct [sechHDCPKeyInfo_t](#) *SecureHDCPKey_info, unsigned int *data)
Set HDCP key information.
- TEE_Result [TEES_TUI_SetInfo](#) (uint32_t cmd, uint64_t paddr, size_t size, unsigned int *data)
Set or clear TUI information.
- TEE_Result [TEES_TUI_Protect](#) (uint64_t paddr, size_t size, unsigned int *data)
Protect the memory used by TUI.
- TEE_Result [TEES_TUI_Unprotect](#) (uint64_t paddr, size_t size, unsigned int *data)
Unprotect the memory used by TUI.

4.1.1 Detailed Description

4.1.2 Data Structure Documentation

4.1.2.1 struct TEES_El2if_Args

Arguments for EL2 SMC call.

Data Fields

unsigned long	args[3]	
---------------	---------	--

4.1.2.2 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	

4.1.2.3 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	

4.1.2.4 struct secHDCPKeyInfo_t

Key information for HDCP.

Data Fields

uint32_t *	riv	
uint32_t *	session_key	

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_STATIC ((SO_TAG_LEN) + (SO_IV_LEN) + (SO_AC_LEN) + (SO_MAGIC_NUMBER_LEN))

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

Get the mandatory 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_STATIC + ((delegated) ? (SO_TA_ID_LEN + SO_AUTH_ID_LEN) : 0))

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

Get the size of output buffer for Secure Object, accounting Header size. Use delegated = true if any delegated flag is set

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	<i>error_code</i>	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

TEE_SUCCESS	successfully checked.
TEE_ERROR_XXXX	- unsuccessfully checked : <ul style="list-style-type: none"> • TEE_ERROR_BAD_PARAMETERS - <i>ac</i> is NULL or object size less than SO header length; • TEE_ERROR_BAD_FORMAT - SO magic number does not match or Not supported access flag on wrapped data; • TEE_ERROR_SECURITY - SO information(TA_ID, AUTH_ID, access_flags) 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 int TEES_El2if (struct TEES_El2if_Args * args)

```
#include <tees_el2if.h>
```

Send SMC call to EL2.

Parameters

in	*args	- arguments when it uses SMC call to EL2.
----	-------	---

Return values

0x0	on success or LSI defined error values otherwise.
-----	---

4.1.4.6 TEE_Result TEES_EnterCritical (void)

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

Disable routing and handling of normal world interrupts.

Return values

TEE_SUCCESS	on success or error otherwise.
-------------	--------------------------------

Example:

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

4.1.4.7 TEE_Result TEES_ExitCritical (void)

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

Enable routing and handling of normal world interrupts.

Return values

TEE_SUCCESS	on success or error otherwise.
-------------	--------------------------------

Example:

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

4.1.4.8 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);
```

4.1.4.9 TEE_Result TEES_HDCP_SetKeyInfo (struct secHDCPKeyInfo_t * *SecureHDCPKey_info*, unsigned int * *data*)

```
#include <tees_ssapi.h>
```

Set HDCP key information.

This function is used to set HDCP key information

Parameters

in	<i>SecureHDCPKey_info</i>	HDCP key information
out	<i>data</i>	Pointer to data for HDCP. If data is NULL, it does not write the return value from SMC.

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> - Unable to send SSAPI IOCTL; • <i>TEE_ERROR_COMMUNICATION</i> - Unable to open ssapi driver; • <i>TEE_ERROR_BAD_PARAMETERS</i> - <i>SecureHDCP_info</i> or one of it's members is NULL;

Example:

```

...
TEE_Result res = TEE_SUCCESS;
unsigned int data;
struct secHDCPKeyInfo_t hdcp;
hdcp.session_key = (uint32_t *)malloc(sizeof(uint32_t) * SESSION_KEY_SIZE);
hdcp.riv_key = (uint32_t *)malloc(sizeof(uint32_t) * RIV_SIZE);
// setting hdcp.session_key and hdcp.riv
res = TEES_HDCP_SetKeyInfo(hdcp, &data);
if( res != TEE_SUCCESS ){
    printf("ERROR : TEES_HDCP_SetKeyInfo = %#x\n", res);
    return res;
}
printf("HDCP SetKeyInfo is successful.\n");

```

4.1.4.10 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_LockHWCryptoBuf() is failed. res = %d\n", res);
    return res;
}
...

```

4.1.4.11 TEE_Result TEES_SECCAM_GetStatus (unsigned int * data)

```
#include <tees_ssapi.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> - SMC returns an unexpected value; • <i>TEE_ERROR_BAD_PARAMETERS</i> - data is NULL; • <i>TEE_ERROR_COMMUNICATION</i> - Unable to open ssapi 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("Secure mode.\n");
}
else {
    printf("No secure mode\n");
}
...

```

4.1.4.12 TEE_Result TEES_SECCAM_IsProtected (uint64_t *paddr*, size_t *size*, unsigned int * *data*)

```
#include <tees_ssapi.h>
```

Check the memory used by secure camera whether it is protected or not.

This function is used to check the physical memory region whether it is protected or not.

Parameters

in	<i>paddr</i>	A physical address pointing to the buffer to be examined.
in	<i>size</i>	Size of the buffer to be examined.
out	<i>data</i>	Pointer to data for secure camera. If data is NULL, it does not write the return value from SMC.

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> - The memory is not protected(data = 0x9102) or SMC returns an unexpected value; • <i>TEE_ERROR_BAD_PARAMETERS</i> - size is 0; • <i>TEE_ERROR_COMMUNICATION</i> - Unable to open ssapi driver;

Example:

```

...
TEE_Result res = TEE_SUCCESS;
unsigned int data;
uint64_t paddr = 0xaaaaaaaa;    // just example...
size_t size = 1024;
res = TEES_SECCAM_IsProtected( paddr, size, &data );
if( res != TEE_SUCCESS ){
    printf("ERROR : TEES_SECCAM_IsProtected = %#x\n", res);
    return res;
}
if( data == 0x00 ){
    printf("The memory region is protected.\n");
}
else {
    printf("The memory region is NOT protected.\n");
}

```

4.1.4.13 TEE_Result TEES_SECCAM_Protect (uint64_t paddr, size_t size, unsigned int * data)

```
#include <tees_ssapi.h>
```

Protect the memory used by secure camera.

This function is used to protect the physical memory region from paddr to paddr+size.

Parameters

in	<i>paddr</i>	A physical address pointing to the buffer to be examined.
in	<i>size</i>	Size of the buffer to be examined.
out	<i>data</i>	Pointer to data for secure camera. If data is NULL, it does not write the return value from SMC.

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> - Protection is failed if data equals 0x9102, otherwise SMC returns an unexpected value; • <i>TEE_ERRIR_BAD_PARAMETERS</i> - size is 0; • <i>TEE_ERROR_COMMUNICATION</i> - Unable to open ssapi driver;

Example:

```

...
TEE_Result res = TEE_SUCCESS;
unsigned int data;
uint64_t paddr = 0xaaaaaaaa;    // just example...
size_t size = 1024;
res = TEES_SECCAM_Protect( paddr, size, &data );
if( res != TEE_SUCCESS ){
    printf("ERROR : TEES_SECCAM_Protect = %#x\n", res);
    return res;
}
printf("Protection is successful.\n");

```

4.1.4.14 TEE_Result TEES_SECCAM_Unprotect (uint64_t *paddr*, size_t *size*, unsigned int * *data*)

```
#include <tees_ssapi.h>
```

Unprotect the memory used by secure camera.

This function is used to unprotect the physical memory region within from *paddr* to *paddr*+*size*

Parameters

in	<i>paddr</i>	A physical address pointing to the buffer to be examined.
in	<i>size</i>	Size of the buffer to be examined.
out	<i>data</i>	Pointer to data for secure camera. If data is NULL, it does not write the return value from SMC.

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> - Unprotection is failed(<i>data</i> = 0x9102) or SMC returns an unexpected value; • <i>TEE_ERROR_BAD_PARAMETERS</i> - <i>size</i> is 0; • <i>TEE_ERROR_COMMUNICATION</i> - Unable to open ssapi driver;

Example:

```
...
TEE_Result res = TEE_SUCCESS;
unsigned int data;
uint64_t paddr = 0xaaaaaaaa;      // just example...
size_t size = 1024;
res = TEES_SECCAM_Unprotect( paddr, size, &data );
if( res != TEE_SUCCESS ){
    printf("ERROR : TEES_SECCAM_Unprotect = %#x\n", res);
    return res;
}
printf("Unprotection is successful.\n");
```

4.1.4.15 TEE_Result TEES_TUI_Protect (uint64_t *paddr*, size_t *size*, unsigned int * *data*)

```
#include <tees_ssapi.h>
```

Protect the memory used by TUI.

This function is used to protect the physical memory region from *paddr* to *paddr*+*size*

Parameters

in	<i>paddr</i>	A physical address pointing to the buffer to be examined.
in	<i>size</i>	Size of the buffer to be examined.
out	<i>data</i>	Pointer to data for TUI. If data is NULL, it does not write the return value from SMC.

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> - Unable to send SSAPI IOCTL; • <i>TEE_ERROR_COMMUNICATION</i> - Unable to open ssapi driver; • <i>TEE_ERROR_BAD_PARAMETERS</i> - size is 0;

Example:

```

...
TEE_Result res = TEE_SUCCESS;
unsigned int data;
uint64_t paddr = 0xaaaaaaaa;    // just example...
size_t size = 1024;
res = TEES_TUI_Protect(paddr, size, &data);
if( res != TEE_SUCCESS ){
    printf("ERROR : TEES_TUI_Protect = %#x\n", res);
    return res;
}
printf("TUI Protect is successful.\n");

```

4.1.4.16 TEE_Result TEES_TUI_SetInfo (uint32_t cmd, uint64_t paddr, size_t size, unsigned int * data)

```
#include <tees_ssapi.h>
```

Set or clear TUI information.

This function is used to set or clear TUI information

Parameters

in	<i>cmd</i>	Command of TUI_SETINFO
in	<i>paddr</i>	A physical address pointing to the buffer to be examined.
in	<i>size</i>	Size of the buffer to be examined.
out	<i>data</i>	Pointer to data for TUI. If data is NULL, it does not write the return value from SMC.

Return values

<i>TEE_SUCCESS</i>	no error.
<i>TEE_ERROR_*</i>	on failure <ul style="list-style-type: none"> • TEE_ERROR_GENERIC - Unable to send SSAPI IOCTL; • TEE_ERROR_COMMUNICATION - Unable to open ssapi driver; • TEE_ERROR_BAD_PARAMETERS - size is 0 or cmd is not valid;

Example:

```
...
TEE_Result res = TEE_SUCCESS;
unsigned int data;
uint32_t cmd = 0x01;
uint64_t paddr = 0xaaaaaaaa;    // just example...
size_t size = 1024;
res = TEES_TUI_SetInfo(cmd, paddr, size, &data);
if( res != TEE_SUCCESS ){
    printf("ERROR : TEES_TUI_SetInfo = %#x\n", res);
    return res;
}
printf("TUI SetInfo is successful.\n");
```

4.1.4.17 TEE_Result TEES_TUI_Unprotect (uint64_t paddr, size_t size, unsigned int * data)

```
#include <tees_ssapi.h>
```

Unprotect the memory used by TUI.

This function is used to unprotect the physical memory region from paddr to paddr+size

Parameters

in	<i>paddr</i>	A physical address pointing to the buffer to be examined.
in	<i>size</i>	Size of the buffer to be examined.
out	<i>data</i>	Pointer to data for TUI. If data is NULL, it does not write the return value from SMC.

Return values

<i>TEE_SUCCESS</i>	no error.
<i>TEE_ERROR_*</i>	on failure <ul style="list-style-type: none"> • TEE_ERROR_GENERIC - Unable to send SSAPI IOCTL; • TEE_ERROR_COMMUNICATION - Unable to open ssapi driver; • TEE_ERROR_BAD_PARAMETERS - size is 0;

Example:

```
...
TEE_Result res = TEE_SUCCESS;
unsigned int data;
uint64_t paddr = 0xaaaaaaaa;    // just example...
size_t size = 1024;
res = TEES_TUI_Unprotect(paddr, size, &data);
if( res != TEE_SUCCESS ){
    printf("ERROR : TEES_TUI_Protect = %#x\n", res);
    return res;
}
printf("TUI Unprotect is successful.\n");
```

4.1.4.18 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 <code>open()</code> or <code>ioctl()</code> 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;
}
...
```

4.1.4.19 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);
```

4.1.4.20 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);
```

4.1.4.21 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_AccessControlInfoType. 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 [intrinfo](#)

Typedefs

- typedef struct __TEES_InterruptHandle * [TEES_InterruptHandle](#)
- typedef void(* [intruhandler](#)) (struct [intrinfo](#) *)
- 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 in /dev/ directory under name, using mask of file operations fops and drvid of served asset.
- int [TEES_RegisterIoctlDesc](#) (struct [usr_drv_info](#) *info, unsigned int cmd, struct [ioctl_desc](#) *desc)
Register an ioctl() cmd for driver.
- 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 in /dev/ directory 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. this API can not be used in read()/write() cmd for driver.
- 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 intrinfo

interrupt information structure

Data Fields

unsigned int	cpu	
unsigned int	nr	

4.2.3 Typedef Documentation

4.2.3.1 intruhandler

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

interrupt handler abstraction

4.2.3.2 TEES_DriverDestructor_t

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

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

4.2.3.3 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. this API can not be used in `read()`/`write()` cmd for driver.

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 <code>filp->share_fd</code>
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_ioctl(struct drv_info *info, int ioctl_cmd, unsigned long ioctl_data)
{
    (void) ioctl_cmd;
    char *vaddr = NULL;

    // Always return "Ioctl".
    vaddr = TEES_AcquireUserBuffer(info, (uint64_t)ioctl_data, (ROUND_PGUP(ioctl_data)
        ) - ioctl_data), PROT_READ | PROT_WRITE);
    strcpy(vaddr, "Ioctl");
    TEES_ReleaseUserBuffer(vaddr, (ROUND_PGUP(ioctl_data) - ioctl_data));

    return 0;
}
```

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);
        TEES_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 = TEES_AllocateInterrupt(IRQ, &handle, handler);
    if (ret != TEE_SUCCESS) {
        return -1;
    }
    return 0;
}
```

4.2.4.3 TEE_Result TEES_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 TEES_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);
        TEES_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) {
        TEES_GenerateInterrupt(handle);
        if (TEES_WaitForInterrupt(handle, 0)) {
            printf("Failed test TEES_WaitForInterrupt() call at line %d.\n"
                "errno = %d\n", __LINE__, errno);
            return -1;
        }
        i++;
    }
    return 0;
}

```

4.2.4.4 int TEES_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. errno 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 usr_drv_info that contains information describing the driver.
----	-------------	---

Return values

<i>non-negative</i>	driver handle.
-1	on failure. errno 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 in /dev/ directory 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>	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[] = "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 in /dev/ directory 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 <code>usr_drv_info</code> that contains information describing the driver

Return values

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

Example:

```
int ret;
char my_driver_name[] = "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_RegisterIoctlDesc (struct usr_drv_info * info, unsigned int cmd, struct ioctl_desc * desc)

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

Register an `ioctl()` cmd for driver.

Parameters

in	<i>info</i>	A pointer to struct <code>usr_drv_info</code> that contains information describing the driver.
in	<i>cmd</i>	An <code>ioctl()</code> command being added.
in	<i>desc</i>	An <code>ioctl()</code> arguments descriptor, see below for details.

Return values

0	on success, -1 otherwise. <code>errno</code> variable is set appropriately.
---	---

Example:

```
struct example { uint32_t a; char *b; uint32_t items_cnt; uint32_t items[]; };
```

```
struct example data = { .a = 0xa5a5a5a5, .b = "0123456789", .items_cnt = 3, .items = {0, 1, 2}, };
```

```
struct ioctl_desc example_desc = { .cnt = 5, .tpl = { { .type = DESC_ATOM_VAL, .len = sizeof(uint32_t) },
{ .type = DESC_ATOM_REF, .len = 10 }, { .type = DESC_ATOM_ARR_CNT_0, .len = sizeof(uint32_t) }, { .type =
DESC_ATOM_ARR_ITEM_0, .len = 1 }, { .type = DESC_ATOM_VAL, .len = sizeof(uint32_t) }, }, };
```

```
ret = TEES_RegisterIoctlDesc(my_driver_info, EXAMPLE_IOCTL_ID, &example_desc);
if (ret) {
    printf("register ioctl() desc failed, ret = %d\n", ret);
}
```

4.2.4.13 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.
-1	on failure. errno variable is set appropriately.

Example:

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

4.2.4.14 TEE_Result TEES_ReleaseInterrupt (TEE_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 TEE_InterruptHandle handle;

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

4.2.4.15 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_ioctl(struct drv_info *info, int ioctl_cmd, unsigned long ioctl_data)
{
    (void) ioctl_cmd;
    char *vaddr = NULL;

    // Always return "Ioctl".
    vaddr = TEES_AcquireUserBuffer(info, (uint64_t)ioctl_data, (ROUND_PGUP(ioctl_data)
        ) - ioctl_data), PROT_READ | PROT_WRITE);
    strcpy(vaddr, "Ioctl");
    TEES_ReleaseUserBuffer(vaddr, (ROUND_PGUP(ioctl_data) - ioctl_data));

    return 0;
}
```

4.2.4.16 TEE_Result TEES_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

TEE_SUCCESS	on success.
TEE_ERROR	on failure.

Example:

```
#define NUM_IRQS_GENERATED (10)
static TEES_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);
        TEES_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) {
        TEES_GenerateInterrupt(handle);
        if (TEES_WaitForInterrupt(handle, NULL) != TEE_SUCCESS) {
            printf("Failed test TEES_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](#).
- TEE_Result [TEES_GetContiguousMemoryPhysaddr](#) ([TEES_ContiguousMemoryHandle](#) memory, uint64_t *physaddr)
Retrieves physical address of physically contiguous memory buffer that was previously allocated by [TEES_AllocateContiguousMemory](#).

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 TEES_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 = TEES_CONTIGUOUS_FLAG_ZERO_ON_ALLOC;

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

4.4.4.2 TEE_Result TEES_GetContiguousMemoryPhysaddr (TEES_ContiguousMemoryHandle memory, uint64_t* physaddr)

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

Retrieves physical address of physically contiguous memory buffer that was previously allocated by TEES_AllocateContiguousMemory.

Parameters

in	<i>memory</i>	A handle on the previously allocated memory.
out	<i>physaddr</i>	Filled with the physical address of the allocated memory assigned to handle.

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.

Example:

```
static TEE_ContiguousMemoryHandle handle;
static uint64_t physaddr;

static int test_get_contiguous_memory_physaddr(void)
{
    TEE_Result ret;

    ret = TEE_GetContiguousMemoryPhysaddr(memory, &physaddr);
    if (ret != TEE_SUCCESS) {
        return -1;
    }
    return 0;
}
```

4.4.4.3 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 TEE_CONTIGUOUS_MAP_READ or TEE_CONTIGUOUS_MAP_WRITE and the bitwise OR of zero or more of the following flags: TEE_CONTIGUOUS_MAP_NON_CACHEABLE: map memory as non-cacheable. TEE_CONTIGUOUS_MAP_READ: map memory with read permission. TEE_CONTIGUOUS_MAP_WRITE: map memory with write permission. TEE_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. TEE_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 TEE_CONTIGUOUS_MAP_READ or TEE_CONTIGUOUS_MAP_WRITE or contains unknown bits set.

Example:

```
static TEE_ContiguousMemoryHandle handle;
static void *addr;

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

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

4.4.4.4 void TEE_ReleaseContiguousMemory (TEE_ContiguousMemoryHandle *memory*)

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

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

Parameters

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

Example:

```
static TEE_ContiguousMemoryHandle handle;

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

4.4.4.5 void TEE_UnmapContiguousMemory (void * *addr*)

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

Unmaps physically contiguous memory buffer that was previously mapped by TEE_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)
{
    TEE_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_TUIMTInfo](#)
Structure that represents information of touch event (touch position, touch type). [More...](#)
- struct [TEES_TUI_MT_Info](#)
Structure that represents all the information of touch events. [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 [MT_INFO_MAX_EVENT](#) 10
- #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_CANCEL](#) = 0xFFFF0002, [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_ERROR_SHORT_BUFFER](#) = 0xFFFF0010, [TEES_ERROR_TUI_TIMEOUT](#) = 0xFFFF3001 }
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_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

- `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_TUIGetMTEvent` (`TEES_TUI_MT_Info` *touchInfo, uint32_t timeout)
Get Multi touch event events.
- `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.
- `TEES_Result TEES_TUIDrawImageToBuff` (`TEES_TUIImage` *image, void *buff, uint32_t *buf_size)
Unpack png picture to bitmap in buffer.
- `TEES_Result TEES_TUIDrawImageFromBuff` (uint32_t posX, uint32_t posY, uint32_t W, uint32_t H, void *buff, uint32_t buf_size)

- Draw unpacked bitmap from buffer on display.*
- 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_TUIMTInfo

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

Data Fields

uint8_t	finger	the finger id of touch event.
TEES_TUITouchTypes	touchType	indicates touch type.
uint32_t	xPosition	the x-coordinate of touch event.
uint32_t	yPosition	the y-coordinate of touch event.

4.7.2.4 struct TEES_TUI_MT_Info

Structure that represents all the information of touch events.

Data Fields

TEES_TUIMTInfo	buf[10]	the information of touch events.
uint8_t	num	the number of touch events.

4.7.2.5 struct TEES_TUIImage

Structure that represents properties of image and buffers.

Data Fields

uint32_t	height	the image height in pixels.
void *	imageBuf	buffer with the image in PNG format (RGBA).
size_t	imageLength	buffer size.
uint32_t	use_png_alpha	not used.
uint32_t	width	the image width in pixels.

4.7.2.6 struct TEE_TUIImage

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

Data Fields

union TEE_TUIImage	__unnamed__	union of image source
uint32_t	height	the number of pixels of the height of the image.
TEE_TUIImageSource	source	indicates the source of the image.
uint32_t	width	the number of pixels of the width of the image.

4.7.2.7 union TEE_TUIImage.__unnamed__

Data Fields

__unnamed__	object	TEE_TUI_OBJECT_SOURCE
__unnamed__	ref	TEE_TUI_REF_SOURCE

4.7.2.8 struct TEE_TUIImage.__unnamed__.ref

Data Fields

void *	image	buffer containing the image.
size_t	imageLength	buffer size.

4.7.2.9 struct TEE_TUIImage.__unnamed__.object

Data Fields

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

4.7.2.10 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.

4.7.2.11 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.

4.7.2.12 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.

4.7.2.13 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.

4.7.2.14 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.

4.7.2.15 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 MT_INFO_MAX_EVENT 10

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

It means maximum number of multi touch events in [TEES_TUI_MT_Info](#).

4.7.3.4 #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 Success
TEES_ERROR_TUI_GENERIC Generic error
TEES_ERROR_TUI_CANCEL Function was canceled
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
TEES_ERROR_SHORT_BUFFER Short buffer
TEES_ERROR_TUI_TIMEOUT Timeout error

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 entryFieldCount 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. If string length 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 pixels
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>	with the image in format PNG (RGBA)
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_TUIDrawImageFromBuff (uint32_t posX, uint32_t posY, uint32_t W, uint32_t H, void * buff, uint32_t buf_size)**

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

Draw unpacked bitmap from buffer on display.

The TEES_TUIDrawImageFromBuff draws bitmap from buffer on display. Bitmap shall be unpacked using TEES_TUIDrawImageToBuff

Parameters

in	<i>posX,posY</i>	- coordinates on display to draw picture
in	<i>W,H</i>	- picture width and heigth
in	<i>buff</i>	- buffer with picture
in	<i>buf_size</i>	- buffer size

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_INVALID_PARAM</i>	If wrong values were passed to function.
<i>TEES_ERROR_SHORT_BUFFER</i>	If buffer size is small

4.7.5.11 **TEES_Result TEES_TUIDrawImageToBuff (TEES_TUIImage * image, void * buff, uint32_t * buf_size)**

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

Unpack png picture to bitmap in buffer.

The TEES_TUIDrawImageToBuff unpacks png picture to bitmap in buffer. image->width, image->height returns current image size This function doesn't need TEES_TUIOpenSession call.

Parameters

in	<i>image</i>	is a pointer to properties of image and buffers
in	<i>buff</i>	is a pointer to buffer
in	<i>buf_size</i>	is a pointer to buffer size, returns required buffer size if income value is low

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_INVALID_PARAM</i>	If wrong values were passed to function.
<i>TEES_ERROR_SHORT_BUFFER</i>	If buffer size is small

4.7.5.12 **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.

4.7.5.13 **TEES_Result** **TEES_TUIGetMTEvent** (**TEES_TUI_MT_Info** * *touchInfo*, **uint32_t** *timeout*)

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

Get Multi touch event events.

At a given time, TA wait and get touch events. If there's no input events from device, an error is returned. A zero timeout means this function waits for a maximum time. Multiple touch events up to maximum 10 may be returned.

Parameters

out	<i>touchInfo</i>	is a pointer to information of MT event
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_CANCEL</i>	: session was closed by external request
<i>TEES_ERROR_TUI_BAD_STATE</i>	: all other reasons
<i>TEES_ERROR_TUI_INVALID_PARAM</i>	input pointer type parameter is null

4.7.5.14 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.

4.7.5.15 TEES_Result TEES_TUIGetTouchEvent (TEES_TUITouchInfo * *touchInfo*, uint32_t *timeout*)

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

Get a touch event.

Deprecated Use [TEES_TUIGetMTEvent\(\)](#) instead 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_CANCEL</i>	: session was closed by external request
<i>TEES_ERROR_TUI_BAD_STATE</i>	: all other reasons
<i>TEES_ERROR_TUI_INVALID_PARAM</i>	input pointer type parameter is null

4.7.5.16 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

4.7.5.17 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 pixels
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 <i>TEES_TUIOpenSession</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>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

4.7.5.18 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

Data Structures

- struct [TEES_ClientCredentials](#)
Client credentials structure. Used by [TEES_GetClientCredentials\(\)](#). [More...](#)

Functions

- int [TEES_GetTaskDataSize](#) (size_t *data_size)
Get used size of data memory of the current Trusted Application instance.
- void * [TEES_Duplwhmem](#) (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 is buffer shared with REE.
- TEE_Result [TEES_GetClientCredentials](#) (struct [TEES_ClientCredentials](#) *credentials)
Get client credentials (pid, gid, uid)

4.9.1 Detailed Description

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

4.9.2 Data Structure Documentation

4.9.2.1 struct TEES_ClientCredentials

Client credentials structure. Used by [TEES_GetClientCredentials\(\)](#).

Data Fields

int	gid	Group ID of client caller
int	pid	Process ID of client
int	uid	User ID of client caller

4.9.3 Function Documentation

4.9.3.1 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.

4.9.3.2 TEE_Result TEES_GetClientCredentials (struct TEES_ClientCredentials * *credentials*)

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

Get client credentials (pid, gid, uid)

TEES_GetClientCredentials fills struct pointed to by a parameter *credentials* with client's credentials. The client can be a CA employing TEE Client API or TA using Internal Client API. For full list of values see definition of struct [TEES_ClientCredentials](#).

Parameters

out	<i>credentials</i>	Structure to fill with values of client credentials
-----	--------------------	---

Return values

<i>TEE_SUCCESS</i>	On success.
<i>TEE_ERROR_BAD_PARAMETERS</i>	When called with <i>credentials</i> == NULL.
<i>TEE_RESULT_NOT_READY</i>	If called when client session does not exist, i.e. in TA_CreateEntryPoint() or TA_DestroyEntryPoint().

Example:

```
struct TEES_ClientCredentials creds;
TEE_Result res;
if ((res = TEES_GetClientCredentials(&creds)) == TEE_SUCCESS) {
    printf("CA pid = %d, ...\n", creds.pid);
} else {
    // handle error
}
```

4.9.3.3 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.

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

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

Check is buffer shared with REE.

The TEES_IsREESharedMemory examines memory buffer passed in a parameters buffer and size to determine whether Trusted Application instance can access buffer accordingly to requested accessFlags 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

Enumerations

- enum { **RPMB_TYPE_BLOCK** = 0, **RPMB_TYPE_BYTE** = 1 }
RPMB I/O type.

Functions

- TEE_Result **TEES_RPMBRead** (uint32_t partition, uint32_t offset, uint8_t *data, uint32_t size, uint8_t type)
Read data from RPMB Storage.
- TEE_Result **TEES_RPMBWrite** (uint32_t partition, uint32_t offset, const uint8_t *data, uint32_t size, uint8_t type)
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 Enumeration Type Documentation

4.10.2.1 anonymous enum

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

RPMB I/O type.

Enumerator

RPMB_TYPE_BLOCK block I/O
RPMB_TYPE_BYTE byte I/O

4.10.3 Function Documentation

4.10.3.1 TEE_Result TEES_RPMBCheckEnable (void)

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

Check RPMB availability.

Return values

TEE_SUCCESS	in case of success
error	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

4.10.3.2 TEE_Result TEES_RPMBRead (uint32_t *partition*, uint32_t *offset*, uint8_t * *data*, uint32_t *size*, uint8_t *type*)

```
#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>size</i>	Size of data to be read from RPMB
in	<i>type</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

4.10.3.3 TEE_Result TEES_RPMBWrite (uint32_t *partition*, uint32_t *offset*, const uint8_t * *data*, uint32_t *size*, uint8_t *type*)

```
#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>size</i>	Size of data to be write to RPMB
in	<i>type</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 `PTHREAD_GUARD_MIN` (`PAGE_SIZE`)
- #define `PTHREAD_GUARD_MAX` (`PAGE_SIZE << 2`)
- #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
- enum { `PTHREAD_CREATE_JOINABLE`, `PTHREAD_CREATE_DETACHED` }
detach state attribute settings

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_getstacksize](#) (const pthread_attr_t *attr, size_t *stacksize)
The [pthread_attr_getstacksize\(\)](#) function gets size of stack.
- int [pthread_attr_getguardsize](#) (const pthread_attr_t *attr, size_t *guardsize)
The [pthread_attr_getguardsize\(\)](#) function gets size of guard.
- int [pthread_attr_setstacksize](#) (pthread_attr_t *attr, size_t stacksize)
The [pthread_attr_setstacksize\(\)](#) function sets size of stack.
- int [pthread_attr_setguardsize](#) (pthread_attr_t *attr, size_t guardsize)
The [pthread_attr_setguardsize\(\)](#) function sets size of guard.
- int [pthread_attr_getstackaddr](#) (const pthread_attr_t *attr, void **stackaddr)
The [pthread_attr_getstackaddr\(\)](#) function gets stack address.
- int [pthread_attr_setstackaddr](#) (pthread_attr_t *attr, void *stackaddr)
The [pthread_attr_setstackaddr\(\)](#) function gets stack address.
- int [pthread_attr_getstack](#) (const pthread_attr_t *attr, void **stackaddr, size_t *stacksize)
The [pthread_attr_getstack\(\)](#) function gets stack address and size.
- int [pthread_attr_setstack](#) (pthread_attr_t *attr, void *stackaddr, size_t stacksize)
The [pthread_attr_setstack\(\)](#) function sets stack address and size.
- 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\(\)](#).
- [_noreturn_](#) void [pthread_exit](#) (void *retval)
Terminate calling thread.
- 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_condattr_init](#) ([pthread_condattr_t](#) *attr)
Initialize conditional variable attributes with default values.
- int [pthread_condattr_destroy](#) ([pthread_condattr_t](#) *attr)
Destroy conditional variable attributes.
- 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.
- int [pthread_cond_timedwait](#) ([pthread_cond_t](#) *cond, [pthread_mutex_t](#) *mutex, const struct timespec *timeout)
Block on condition variable cond, while another thread will unblock this variable by [pthread_cond_broadcast\(\)](#) / [pthread_cond_signal\(\)](#) call. If timeout reached before condition set then wait will be interrupted and error is returned.
- [pthread_t](#) [pthread_self](#) (void)
Obtain ID of the calling thread.
- int [pthread_kill](#) ([pthread_t](#) thread, int sig)
Send signal to specified thread.
- int [pthread_equal](#) ([pthread_t](#) t1, [pthread_t](#) t2)
Compare thread IDs.
- int [pthread_detach](#) ([pthread_t](#) thread)
Detach a thread.
- int [pthread_attr_setdetachstate](#) ([pthread_attr_t](#) *attr, int detachstate)
Set the detach state attribute.
- int [pthread_attr_getdetachstate](#) (const [pthread_attr_t](#) *attr, int *detachstate)
Get the detach state attribute.
- int [pthread_getattr_np](#) ([pthread_t](#) thread, [pthread_attr_t](#) *attr)
Get attributes of created 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 threatened atomically.

Data Fields

atomic_t	already_executed	
----------	------------------	--

4.11.2.2 struct __pthread_attr_t

Pthread attribute object type (not implemented yet).

Data Fields

unsigned int	flags	
unsigned int	guardsize	
void *	stackaddr	
unsigned int	stacksize	

4.11.2.3 struct __pthread_mutex_t

Pthread mutex object type.

Data Fields

unsigned	attr	
pthread_t	caller_thread	
unsigned	count	
atomic_t	state	

4.11.2.4 struct __pthread_cond_t

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

Data Fields

atomic_t	status	
----------	--------	--

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_GUARD_MAX (PAGE_SIZE << 2)

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

Maximum stack guard size.

4.11.3.5 #define PTHREAD_GUARD_MIN (PAGE_SIZE)

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

Minimum stack guard size.

4.11.3.6 #define PTHREAD_MUTEX_INITIALIZER { { MUTEX_STATE_UNLOCKED }, PTHREAD_MUTEX_DEFAULT, 0, 0 }

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

[pthread_mutex_t](#) object for static mutex initialization.

4.11.3.7 #define PTHREAD_ONCE_INIT { ATOMIC_INITIALIZER }

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

Object for static initialization of `pthread_once_t` variables.

4.11.3.8 #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.

4.11.3.9 #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(...);
}
```

4.11.3.10 #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](#).

4.11.5.2 anonymous enum

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

detach state attribute settings

Enumerator

PTHREAD_CREATE_JOINABLE Set thread to be joinable
PTHREAD_CREATE_DETACHED Set thread to be detached

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_getdetachstate (const pthread_attr_t * attr, int * detachstate)`

#include <pthread.h>

Get the detach state attribute.

Parameters

in	<i>attr</i>	Thread attributes object
out	<i>detachstate</i>	Buffer to return detach state

Return values

0	On success
---	------------

4.11.6.3 `int pthread_attr_getguardsize (const pthread_attr_t * attr, size_t * guardsize)`

#include <pthread.h>

The `pthread_attr_getguardsize()` function gets size of guard.

Parameters

in	<i>attr</i>	is attribute struct.
out	<i>guardsize</i>	is buffer to return size of guard.

Return values

0	On success.
---	-------------

4.11.6.4 `int pthread_attr_getstack (const pthread_attr_t * attr, void ** stackaddr, size_t * stacksize)`

#include <pthread.h>

The `pthread_attr_getstack()` function gets stack address and size.

Parameters

in	<i>attr</i>	is attribute struct.
out	<i>stackaddr</i>	is buffer to save stack address.
out	<i>stacksize</i>	is buffer to save stack size.

Return values

0	On success.
---	-------------

4.11.6.5 `int pthread_attr_getstackaddr (const pthread_attr_t * attr, void ** stackaddr)`

#include <pthread.h>

The `pthread_attr_getstackaddr()` function gets stack address.

Parameters

in	<i>attr</i>	is attribute struct.
out	<i>stackaddr</i>	is buffer to save stack address.

Return values

0	On success.
---	-------------

4.11.6.6 `int pthread_attr_getstacksize (const pthread_attr_t * attr, size_t * stacksize)`

#include <pthread.h>

The `pthread_attr_getstacksize()` function gets size of stack.

Parameters

in	<i>attr</i>	is attribute struct.
out	<i>stacksize</i>	is buffer to return size of stack.

Return values

0	On success.
---	-------------

4.11.6.7 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

4.11.6.8 int pthread_attr_setdetachstate (pthread_attr_t * attr, int detachstate)

#include <pthread.h>

Set the detach state attribute.

Parameters

in	<i>attr</i>	Thread attributes object
in	<i>detachstate</i>	Detach state to set in attr

Return values

<i>EINVAL</i>	Invalid value of detachstate
0	On success

4.11.6.9 int pthread_attr_setguardsize (pthread_attr_t * attr, size_t guardsize)

#include <pthread.h>

The `pthread_attr_setguardsize()` function sets size of guard.

Parameters

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

Return values

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

4.11.6.10 int pthread_attr_setstack (pthread_attr_t * attr, void * stackaddr, size_t stacksize)

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

The [pthread_attr_setstack\(\)](#) function sets stack address and size.

Parameters

in	<i>attr</i>	is attribute struct.
in	<i>stackaddr</i>	is buffer to save stack address.
in	<i>stacksize</i>	is buffer to save stack size.

Return values

0	On success.
---	-------------

4.11.6.11 int pthread_attr_setstackaddr (pthread_attr_t * attr, void * stackaddr)

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

The [pthread_attr_setstackaddr\(\)](#) function gets stack address.

Parameters

in	<i>attr</i>	is attribute struct.
in	<i>stackaddr</i>	is desired stack address.

Return values

0	On success.
---	-------------

4.11.6.12 int pthread_attr_setstacksize (pthread_attr_t * attr, size_t stacksize)

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

The [pthread_attr_setstacksize\(\)](#) function sets size of stack.

Parameters

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

Return values

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

4.11.6.13 int pthread_cond_broadcast (pthread_cond_t * cond)

#include <pthread.h>

Wake up all threads locked by conditional variable cond.

Parameters

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

Return values

0	If successful.
EINVAL	The value cond 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);
}
```

4.11.6.14 int pthread_cond_destroy (pthread_cond_t * cond)

#include <pthread.h>

Destroy conditional variable object and make it uninitialized.

Parameters

out	cond	A pointer to the pthread_cond_t object that to destroy.
-----	------	---

Return values

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

```

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

Return values

0	On success it returns 0.
EAGAIN	The system lacked the necessary resources (other than memory) to initialise another condition variable.
ENOMEM	Insufficient memory exists to initialise the condition variable.
EBUSY	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.
EINVAL	The value specified by <code>attr</code> is invalid.

Example:

```

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

```


4.11.6.16 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	<i>cond</i>	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);
}
```

4.11.6.17 int pthread_cond_timedwait (pthread_cond_t * cond, pthread_mutex_t * mutex, const struct timespec * timeout)

#include <pthread.h>

Block on condition variable `cond`, while another thread will unblock this variable by [pthread_cond_broadcast\(\)](#) / [pthread_cond_signal\(\)](#) call. If timeout reached before condition set then wait will be interrupted and error is returned.

Parameters

out	<i>cond</i>	A pointer to the pthread_cond_t object that user wants the threads to block on.
out	<i>mutex</i>	The mutex that to unlock.
out	<i>timeout</i>	Timeout to wait for condition

Return values

0	On success.
<i>EINVAL</i>	The value specified by <code>cond</code> or <code>mutex</code> or <code>timeout</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.
<i>ETIMEDOUT</i>	The timeout occurred while awaiting for condition

Example:

```
pthread_mutex_t myMutex;
pthread_cond_t cond;
pthread_attr_t attr;
const struct timespec to = {1, 0};
int cont;
void *anotherFunc(void*)
{
    printf("anotherFunc\n");
    pthread_mutex_lock(&myMutex);
    printf("waiting...\n");
    pthread_cond_timedwait(&cond, &myMutex, &to);
    cont += 10;
    printf("slot\n");
    pthread_mutex_unlock(&myMutex);
    printf("mutex unlocked anotherFunc\n");
    printf("Done anotherFunc\n");
    pthread_exit(NULL);
}
```

4.11.6.18 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	<i>cond</i>	A pointer to the <code>pthread_cond_t</code> object that user wants the threads to block on.
out	<i>mutex</i>	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);
}
```

4.11.6.19 int pthread_condattr_destroy (pthread_condattr_t * attr)

#include <pthread.h>

Destroy conditional variable attributes.

Parameters

in	attr	pointer to a pthread_condattr_t object that specifies the attributes that you want to use for the cond.
----	------	---

Return values

0	On success it returns 0.
---	--------------------------

4.11.6.20 int pthread_condattr_init (pthread_condattr_t * attr)

#include <pthread.h>

Initialize conditional variable attributes with default values.

Parameters

out	attr	pointer to a pthread_condattr_t object that specifies the attributes that you want to use for the cond.
-----	------	---

Return values

0	On success it returns 0.
---	--------------------------

4.11.6.21 `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;
}
```

4.11.6.22 `int pthread_detach (pthread_t thread)`

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

Detach a thread.

Parameters

in	<i>thread</i>	Thread to detach
----	---------------	------------------

Return values

0	On success
<i>EINVAL</i>	if thread is not a joinable thread
<i>ESRCH</i>	if no thread with the ID thread could be found

4.11.6.23 int pthread_equal (pthread_t t1, pthread_t t2)

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

Compare thread IDs.

Parameters

in	<i>t1</i>	Thread 1 id
in	<i>t2</i>	Thread 2 id

Return values

<i>Nonzero</i>	if two thread IDs are equal
0	if two thread IDs are not equal

Example:

```
int res = pthread_equal(t1, t2);
if (res) {
    do_some_work(...);
}
```

4.11.6.24 _noreturn_ void pthread_exit (void * retval)

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

Terminate calling thread.

Parameters

in	<i>retval</i>	value which will be available by pthread_join.
----	---------------	--

4.11.6.25 int pthread_getattr_np (pthread_t *thread*, pthread_attr_t * *attr*)

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

Get attributes of created thread.

Parameters

in	<i>thread</i>	thread to get attributes from
out	<i>attr</i>	Buffer to return thread attributes

Return values

0	On success
---	------------

4.11.6.26 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	<i>key</i>	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;
}
```

4.11.6.27 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	<i>thread</i>	Target thread which termination is waited.
out	<i>retval</i>	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

0	If successful.
<i>EINVAL</i>	The implementation has detected that the value specified by <code>thread</code> does not refer to a joinable thread.
<i>ESRCH</i>	No thread could be found corresponding to that specified by the given thread ID.
<i>EDEADLK</i>	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);
}
```

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

Return values

0	On success.
EINVAL	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;
}
```

4.11.6.29 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	key	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;
}
```

4.11.6.30 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(...);
}
```

4.11.6.31 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.
EBUSY	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.
EINVAL	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);
```

4.11.6.32 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 mutex, a previously initialised, but not yet destroyed, mutex.
<i>EINVAL</i>	The value specified by attr 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);
```

4.11.6.33 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 pthread_mutex_t object to lock.
-----	--------------	--

Return values

0	On success.
<i>EINVAL</i>	The value specified by mutex 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);
}
```

4.11.6.34 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.
EBUSY	The mutex could not be acquired because it was already locked.
EINVAL	The value specified by <i>mutex</i> does not refer to an initialized mutex object.
EAGAIN	The mutex could not be acquired because the maximum number of recursive locks for mutex 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);
```

4.11.6.35 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.
EPERM	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);
```

4.11.6.36 int pthread_mutexattr_destroy (pthread_mutexattr_t * attr)

#include <pthread.h>

Destroy attributes object and make all attribute values are uninitialized.

Parameters

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

Return values

0	On success.
EINVAL	The value specified by attr 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;
}
```

4.11.6.37 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);
```

4.11.6.38 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;
}
```

4.11.6.39 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);
```

4.11.6.40 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

0	Upon successful completion.
<i>EINVAL</i>	The implementation has detected that the value specified by <code>thread</code> does not refer to a joinable thread.
<i>ESRCH</i>	No thread could be found corresponding to that specified by the given thread ID.
<i>EDEADLK</i>	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...
}
```

4.11.6.41 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) {
    ...
}
```

4.11.6.42 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 **NAN** __builtin_nan("")
- #define **INFINITY** __builtin_inf()
- #define **isnan**(x) _isnan(x)
- #define **isfinite**(x) _isfinite(x)
- #define **M_E** 2.7182818284590452354
- #define **M_PI** 3.14159265358979323846
- #define **M_PI_2** 1.57079632679489661923
- #define **M_PI_4** 0.78539816339744830962
- #define **FP_ILOGBNAN** (-1-(int)((((unsigned)-1)>>1))
- #define **FP_ILOGB0** **FP_ILOGBNAN**

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
- base-e exponential function*

 - `_const_ float expf` (float x)
- base-e exponential function*

 - `_const_ int ilogb` (double x)

Calculate integer exponent of a floating-point value.
- Calculate integer exponent of a floating-point value.*

 - `_const_ int ilogbf` (float x)
- Calculate integer exponent of a floating-point value.*

 - `_const_ int ilogbl` (long double x)
- Calculate integer exponent of a floating-point value.*

 - `_const_ double log` (double x)

Calculate the natural logarithm.
- 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.
- Calculate exponent of a floating-point value.*

 - `_const_ float logbf` (float x)
- Calculate exponent of a floating-point value.*

 - `_const_ long double logbl` (long double x)

Calculate exponent of a floating-point value.
- Calculate exponent of a floating-point value.*

 - `_const_ double fmax` (double x, double y)

Determine maximum of two floating-point numbers.
- Determine maximum of two floating-point numbers.*

 - `_const_ float fmaxf` (float x, float y)
- Determine maximum of two floating-point numbers.*

 - `_const_ long double fmaxl` (long double x, long double y)

Determine maximum of two floating-point numbers.
- Determine maximum of two floating-point numbers.*

 - `_const_ double fmin` (double x, double y)

Determine minimum of two floating-point numbers.
- 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.
- Multiply floating-point number by integral power of radix.*

 - `_const_ double scalbnl` (long 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.
- Multiply floating-point number by integral power of radix.*

 - `_const_ double copysign` (double x, double y)

Copy sign of a number.
- 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.
- 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.
- 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 FP_ILOGB0 FP_ILOGBNAN

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

ilogb return value if x is zero

4.12.2.2 #define FP_ILOGBNAN (-1-(int)((((unsigned)-1)>>1))

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

ilogb return value if x is NAN

4.12.2.3 #define INFINITY __builtin_inf()

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

Infinity number

4.12.2.4 #define isfinite(x) _isfinite(x)

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

Parameters

in	x	- value to check for finite
----	---	-----------------------------

Returns

!0 - is number, 0 - NaN or infinite

4.12.2.5 #define isnan(x) _isnan(x)

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

Parameters

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

Returns

0 - is number, !0 - NaN

4.12.2.6 #define M_E 2.7182818284590452354

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

e - base of the natural logarithm

4.12.2.7 #define M_PI 3.14159265358979323846

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

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

4.12.2.8 #define M_PI_2 1.57079632679489661923

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

pi/2

4.12.2.9 #define M_PI_4 0.78539816339744830962

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

pi/4

4.12.2.10 #define NAN __builtin_nan("")

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

Not A Number

4.12.3 Function Documentation

4.12.3.1 `_const_double atan (double x)`

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

Arc tangent function.

Parameters

in	<code>x</code>	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	<code>y</code>	function argument
in	<code>x</code>	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	<code>y</code>	function argument
in	<code>x</code>	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	<i>x</i>	function argument
in	<i>y</i>	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	<i>x</i>	function argument
in	<i>y</i>	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	<i>x</i>	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	<i>x</i>	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_ long double fmaxl (long double x, long double y)

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

Determine maximum of two floating-point numbers.

Parameters

in	x	function argument
in	y	function argument

Returns

maximal value of x or y

4.12.3.20 _const_ double fmin (double x, double y)

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

Determine minimum of two floating-point numbers.

Parameters

in	x	function argument
in	y	function argument

Returns

minimal value of x or y

4.12.3.21 _const_ float fminf (float x, float y)

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

Determine minimum of two floating-point numbers.

Parameters

in	x	function argument
in	y	function argument

Returns

minimal value of x or y

4.12.3.22 `_const_double hypot (double x, double y)`

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

Euclidean distance function.

Parameters

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

Returns

length of a right-angled triangle with sides of length `x` and `y`

4.12.3.23 `_const_float hypotf (float x, float y)`

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

Euclidean distance function.

Parameters

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

Returns

length of a right-angled triangle with sides of length `x` and `y`

4.12.3.24 `_const_int ilogb (double x)`

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

Calculate integer exponent of a floating-point value.

Parameters

in	<code>x</code>	function argument
----	----------------	-------------------

Returns

exponent of `x`, as a signed integer

4.12.3.25 `_const_int ilogbf (float x)`

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

Calculate integer exponent of a floating-point value.

Parameters

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

Returns

exponent of x, as a signed integer

4.12.3.26 `_const_int ilogbl (long double x)`

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

Calculate integer exponent of a floating-point value.

Parameters

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

Returns

exponent of x, as a signed integer

4.12.3.27 `_const_double log (double x)`

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

Calculate the natural logarithm.

Parameters

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

Returns

the result of calculation

4.12.3.28 `_const_double logb (double x)`

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

Calculate exponent of a floating-point value.

Parameters

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

Returns

the result of calculation

4.12.3.29 `_const_float logbf (float x)`

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

Calculate exponent of a floating-point value.

Parameters

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

Returns

the result of calculation

4.12.3.30 `_const_long_double logbl (long double x)`

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

Calculate exponent of a floating-point value.

Parameters

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

Returns

the result of calculation

4.12.3.31 `_const_float logf (float x)`

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

Calculate the natural logarithm.

Parameters

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

Returns

the result of calculation

4.12.3.32 `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

4.12.3.33 `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

4.12.3.34 _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

4.12.3.35 _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

4.12.3.36 _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

4.12.3.37 _const_float roundf (float 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

4.12.3.38 _const_double scalbn (double x, int exp)

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

Multiply floating-point number by integral power of radix.

Parameters

in	<i>x</i>	function argument
in	<i>exp</i>	exponent

Returns

result of calculation

4.12.3.39 _const_float scalbnf (float x, int exp)

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

Multiply floating-point number by integral power of radix.

Parameters

in	<i>x</i>	function argument
in	<i>exp</i>	exponent

Returns

result of calculation

4.12.3.40 _const_double scalbnl (long double x, int exp)

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

Multiply floating-point number by integral power of radix.

Parameters

in	x	function argument
in	exp	exponent

Returns

result of calculation

4.12.3.41 _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

4.12.3.42 _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

4.12.3.43 `_const_double sqrt (double x)`

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

Calculate the square root of x.

Parameters

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

Returns

the result of calculation

4.12.3.44 `_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 <mqqueue.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 <mqqueue.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

Typedefs

- typedef int [socklen_t](#)
A type of width of at least 32 bits representing lengths in socket subsystem.

Functions

- int [accept](#) (int sockfd, struct sockaddr *addr, [socklen_t](#) *addrlen)
Accept a connection on a socket.
- int [bind](#) (int sockfd, const struct sockaddr *addr, [socklen_t](#) addrlen)
Bind a name to a socket.
- int [connect](#) (int sockfd, const struct sockaddr *addr, [socklen_t](#) addrlen)
Initiate a connection on a socket.
- int [getsockopt](#) (int sockfd, int level, int optname, void *optval, [socklen_t](#) *optlen)
Get options on socket.
- int [setsockopt](#) (int sockfd, int level, int optname, const void *optval, [socklen_t](#) optlen)
Set 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, socklen_t * 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, socklen_t 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, socklen_t 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, socklen_t * 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 optval

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 msghdr 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, const void * optval, socklen_t optlen)`

#include <sys/socket.h>

Set options on socket.

Parameters

in	<i>sockfd</i>	socket descriptor
in	<i>level</i>	socket API level
in	<i>optname</i>	specified options
in	<i>optval</i>	buffer in which the new value of the requested options is stored
in	<i>optlen</i>	the size of the buffer pointed to by optval

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 alloca _alloca`
Allocate memory that is automatically freed.
- `#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 **EPROTOPTYPE** 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 **EKEYREJECTED** 127 /* Key was rejected by service */
- #define **ENOTRECOVERABLE** 131 /* State not recoverable */
- #define **errno** (*get_errno_addr())
- #define **PRId16** __16_PREFIX "d"
- #define **PRId32** "d"
- #define **PRId64** __64_PREFIX "d"
- #define **PRId16** __16_PREFIX "u"
- #define **PRId32** "u"
- #define **PRId64** __64_PREFIX "u"
- #define **PRId16** __16_PREFIX "x"
- #define **PRId32** "x"
- #define **PRId64** __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 _POSIX_THREAD_THREADS_MAX 64
- #define _POSIX_THREADS 1
- #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 MAP_STACK (1 << 5)
- #define MAP_GROWSDOWN (1 << 6)
- #define PROT_NONE 0
- #define PROT_READ 1
- #define PROT_WRITE 2
- #define PROT_EXEC 4
- #define PGOFF_SHIFT 12
- #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 intptr_t `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)

Enumerations

- enum {
`_SC_OPEN_MAX`, `_SC_PAGESIZE`, `_SC_PHYS_PAGES`, `_SC_AVPHYS_PAGES`,
`_SC_NPROCESSORS_CONF`, `_SC_NPROCESSORS_ONLN`, `_SC_THREADS`, `_SC_THREAD_KEYS_MAX`,
`_SC_THREAD_THREADS_MAX`, `_SC_THREAD_STACK_MIN` }
sysconf system resources' names.

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.
- 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 [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.
- [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 dest. 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 **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 **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 * **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.
- void * **memrchr** (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 **mprotect** (void *addr, size_t len, int prot)
 - int **nanosleep** (const 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.
 - 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 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.
 - 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 **rename** (const char *pathname, const char *new_pathname)
Rename file.
 - 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 **rmdir** (char *dir_name)
Remove a directory at file system.
- 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 **fsync** (int fd)
Synchronize a file's in-core state with storage device.
- int **lseek** (int fd, int offset, int whence)
Reposition read/write file offset.
- long **sysconf** (int name)
Get configuration information at run time.
- 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 _j.
- 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 *np* to double.
 - long long **strtoll** (const char *nptr, char **endptr, int base)
Convert the initial part of the string in *np* 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 *np* 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 *np* to float.
 - long double **strtold** (const char *nptr, char **endptr)
the initial portion of the string pointed to by *np* 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.
 - int **atoi** (const char *nptr)
Convert a string to an integer.
 - double **atof** (const char *nptr)
Convert a string to a double.
 - char * **getenv** (const char *name)
get an environment variable function stub
 - int **unsetenv** (const char *name)
delete environment variable function stub
 - int **setenv** (const char *name, const char *value, int overwrite)
change or add environment variable function stub

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 _POSIX_THREAD_THREADS_MAX** 64

```
#include <limits.h>
```

The number of threads per process.

4.15.3.4 **#define _POSIX_THREADS** 1

```
#include <limits.h>
```

Threads are supported.

4.15.3.5 **#define alloca _alloca**

```
#include <alloca.h>
```

Allocate memory that is automatically freed.

Parameters

in	size	of requested memory
----	------	---------------------

Returns

pointer to allocated memory

4.15.3.6 **#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

4.15.3.7 **#define AUTO_BUFFER_SIZE 1024**

```
#include <print_no_alloc.h>
```

Buffer size.

4.15.3.8 **#define BIT_PER_CPU (1)**

```
#include <sched.h>
```

The number cpu bits

4.15.3.9 **#define BITMAP_ELT(*cpu*) ((*cpu*) / BITS_PER_LONG)**

```
#include <sched.h>
```

The bit map element

4.15.3.10 **#define BITS_TO_CPU_MASK(*bits*) (((*bits*) + BITS_PER_LONG - 1) / BITS_PER_LONG)**

```
#include <sched.h>
```

The number bits of cpu mask

4.15.3.11 **#define CHAR_BIT 8**

```
#include <limits.h>
```

Number of bits in a type char

4.15.3.12 #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

4.15.3.13 #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

4.15.3.14 #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

4.15.3.15 #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

4.15.3.16 #define DECLARE_BITMAP(*name*, *bits*) unsigned long *name*[BITS_TO_CPU_MASK(*bits*)]

```
#include <sched.h>
```

The short type of bitmap

4.15.3.17 #define E2BIG 7 /* Argument list too long */

```
#include <core/error.h>
```

Argument list too long.

4.15.3.18 #define EACCES 13 /* Permission denied */

```
#include <core/error.h>
```

Permission denied.

4.15.3.19 #define EADDRINUSE 98 /* Address already in use */

```
#include <core/error.h>
```

Address already in use.

4.15.3.20 #define EADDRNOTAVAIL 99 /* Cannot assign requested address */

```
#include <core/error.h>
```

Cannot assign requested address.

4.15.3.21 #define EADV 68 /* Advertise error */

```
#include <core/error.h>
```

Advertise error.

4.15.3.22 #define EAFNOSUPPORT 97 /* Address family not supported by protocol */

```
#include <core/error.h>
```

Address family not supported by protocol.

4.15.3.23 #define EAGAIN 11 /* Try again */

```
#include <core/error.h>
```

Try again.

4.15.3.24 #define EALREADY 114 /* Operation already in progress */

```
#include <core/error.h>
```

Operation already in progress.

4.15.3.25 #define EBADE 52 /* Invalid exchange */

```
#include <core/error.h>
```

Invalid exchange.

4.15.3.26 #define EBADF 9 /* Bad file number */

```
#include <core/error.h>
```

Bad file number.

4.15.3.27 #define EBADFD 77 /* File descriptor in bad state */

```
#include <core/error.h>
```

File descriptor in bad state.

4.15.3.28 #define EBADMSG 74 /* Not a data message */

```
#include <core/error.h>
```

Not a data message.

4.15.3.29 #define EBADR 53 /* Invalid request descriptor */

```
#include <core/error.h>
```

Invalid request descriptor.

4.15.3.30 #define EBADRQC 56 /* Invalid request code */

```
#include <core/error.h>
```

Invalid request code.

4.15.3.31 #define EBADSLT 57 /* Invalid slot */

```
#include <core/error.h>
```

Invalid slot.

4.15.3.32 #define EBFONT 59 /* Bad font file format */

```
#include <core/error.h>
```

Bad font file format.

4.15.3.33 #define EBUSY 16 /* Device or resource busy */

```
#include <core/error.h>
```

Device or resource busy.

4.15.3.34 #define ECANCELED 125 /* Operation canceled */

```
#include <core/error.h>
```

Operation canceled.

4.15.3.35 #define ECHILD 10 /* No child processes */

```
#include <core/error.h>
```

No child processes.

4.15.3.36 #define ECHRNG 44 /* Channel number out of range */

```
#include <core/error.h>
```

Channel number out of range.

4.15.3.37 #define ECOMM 70 /* Communication error on send */

```
#include <core/error.h>
```

Communication error on send.

4.15.3.38 #define ECONNABORTED 103 /* Software caused connection abort */

```
#include <core/error.h>
```

Software caused connection abort.

4.15.3.39 #define ECONNREFUSED 111 /* Connection refused */

```
#include <core/error.h>
```

Connection refused.

4.15.3.40 #define ECONNRESET 104 /* Connection reset by peer */

```
#include <core/error.h>
```

Connection reset by peer.

4.15.3.41 #define EDEADLK 35 /* Resource deadlock would occur */

```
#include <core/error.h>
```

Resource deadlock would occur.

4.15.3.42 #define EDEADLOCK EDEADLK

```
#include <core/error.h>
```

Resource deadlock would occur.

4.15.3.43 #define EDESTADDRREQ 89 /* Destination address required */

```
#include <core/error.h>
```

Destination address required.

4.15.3.44 #define EDOM 33 /* Math argument out of domain of func */

```
#include <core/error.h>
```

Math argument out of domain of func.

4.15.3.45 #define EDOTDOT 73 /* RFS specific error */

```
#include <core/error.h>
```

RFS specific error.

4.15.3.46 #define EDQUOT 122 /* Quota exceeded */

```
#include <core/error.h>
```

Quota exceeded.

4.15.3.47 #define EEXIST 17 /* File exists */

```
#include <core/error.h>
```

File exists.

4.15.3.48 #define EFAULT 14 /* Bad address */

```
#include <core/error.h>
```

Bad address.

4.15.3.49 #define EFBIG 27 /* File too large */

```
#include <core/error.h>
```

File too large.

4.15.3.50 #define EHOSTDOWN 112 /* Host is down */

```
#include <core/error.h>
```

Host is down.

4.15.3.51 #define EHOSTUNREACH 113 /* No route to host */

```
#include <core/error.h>
```

No route to host.

4.15.3.52 #define EIDRM 43 /* Identifier removed */

```
#include <core/error.h>
```

Identifier removed.

4.15.3.53 #define EILSEQ 84 /* Illegal byte sequence */

```
#include <core/error.h>
```

Illegal byte sequence.

4.15.3.54 #define EINPROGRESS 115 /* Operation now in progress */

```
#include <core/error.h>
```

Operation now in progress.

4.15.3.55 #define EINTR 4 /* Interrupted system call */

```
#include <core/error.h>
```

Interrupted system call.

4.15.3.56 #define EINVAL 22 /* Invalid argument */

```
#include <core/error.h>
```

Invalid argument.

4.15.3.57 #define EIO 5 /* I/O error */

```
#include <core/error.h>
```

I/O error.

4.15.3.58 #define EISCONN 106 /* Transport endpoint is already connected */

```
#include <core/error.h>
```

Transport endpoint is already connected.

4.15.3.59 #define EISDIR 21 /* Is a directory */

```
#include <core/error.h>
```

Is a directory.

4.15.3.60 #define EISNAM 120 /* Is a named type file */

```
#include <core/error.h>
```

Is a named type file.

4.15.3.61 #define EKEYREJECTED 127 /* Key was rejected by service */

```
#include <core/error.h>
```

Key was rejected by service.

4.15.3.62 #define EL2HLT 51 /* Level 2 halted */

```
#include <core/error.h>
```

Level 2 halted.

4.15.3.63 #define EL2NSYNC 45 /* Level 2 not synchronized */

```
#include <core/error.h>
```

Level 2 not synchronized.

4.15.3.64 #define EL3HLT 46 /* Level 3 halted */

```
#include <core/error.h>
```

Level 3 halted.

4.15.3.65 #define EL3RST 47 /* Level 3 reset */

```
#include <core/error.h>
```

Level 3 reset.

4.15.3.66 #define ELIBACC 79 /* Can not access a needed shared library */

```
#include <core/error.h>
```

Can not access a needed shared library.

4.15.3.67 #define ELIBBAD 80 /* Accessing a corrupted shared library */

```
#include <core/error.h>
```

Accessing a corrupted shared library.

4.15.3.68 #define ELIBEXEC 83 /* Cannot exec a shared library directly */

```
#include <core/error.h>
```

Cannot exec a shared library directly.

4.15.3.69 #define ELIBMAX 82 /* Attempting to link in too many shared libraries */

```
#include <core/error.h>
```

Attempting to link in too many shared libraries.

4.15.3.70 #define ELIBSCN 81 /* .lib section in a.out corrupted */

```
#include <core/error.h>
```

.lib section in a.out corrupted.

4.15.3.71 #define ELNRNG 48 /* Link number out of range */

```
#include <core/error.h>
```

Link number out of range.

4.15.3.72 #define ELOOP 40 /* Too many symbolic links encountered */

```
#include <core/error.h>
```

Too many symbolic links encountered.

4.15.3.73 #define EMFILE 24 /* Too many open files */

```
#include <core/error.h>
```

Too many open files.

4.15.3.74 #define EMLINK 31 /* Too many links */

```
#include <core/error.h>
```

Too many links.

4.15.3.75 #define EMSGSIZE 90 /* Message too long */

```
#include <core/error.h>
```

Message too long.

4.15.3.76 #define EMULTIHOP 72 /* Multihop attempted */

```
#include <core/error.h>
```

Multihop attempted.

4.15.3.77 #define ENAMETOOLONG 36 /* File name too long */

```
#include <core/error.h>
```

File name too long.

4.15.3.78 #define ENAVAIL 119 /* No XENIX semaphores available */

```
#include <core/error.h>
```

No XENIX semaphores available.

4.15.3.79 #define ENETDOWN 100 /* Network is down */

```
#include <core/error.h>
```

Network is down.

4.15.3.80 #define ENETRESET 102 /* Network dropped connection because of reset */

```
#include <core/error.h>
```

Network dropped connection because of reset.

4.15.3.81 #define ENETUNREACH 101 /* Network is unreachable */

```
#include <core/error.h>
```

Network is unreachable.

4.15.3.82 #define ENFILE 23 /* File table overflow */

```
#include <core/error.h>
```

File table overflow.

4.15.3.83 #define ENOANO 55 /* No anode */

```
#include <core/error.h>
```

No anode.

4.15.3.84 #define ENOBUFS 105 /* No buffer space available */

```
#include <core/error.h>
```

No buffer space available.

4.15.3.85 #define ENOCSI 50 /* No CSI structure available */

```
#include <core/error.h>
```

No CSI structure available.

4.15.3.86 #define ENODATA 61 /* No data available */

```
#include <core/error.h>
```

No data available.

4.15.3.87 #define ENODEV 19 /* No such device */

```
#include <core/error.h>
```

No such device.

4.15.3.88 #define ENOENT 2 /* No such file or directory */

```
#include <core/error.h>
```

No such file or directory.

4.15.3.89 #define ENOEXEC 8 /* Exec format error */

```
#include <core/error.h>
```

Exec format error.

4.15.3.90 #define ENOKEY 126 /* Required key not available */

```
#include <core/error.h>
```

Required key not available.

4.15.3.91 #define ENOLCK 37 /* No record locks available */

```
#include <core/error.h>
```

No record locks available.

4.15.3.92 #define ENOLINK 67 /* Link has been severed */

```
#include <core/error.h>
```

Link has been severed.

4.15.3.93 #define ENOMEM 12 /* Out of memory */

```
#include <core/error.h>
```

Out of memory.

4.15.3.94 #define ENOMSG 42 /* No message of desired type */

```
#include <core/error.h>
```

No message of desired type.

4.15.3.95 #define ENONET 64 /* Machine is not on the network */

```
#include <core/error.h>
```

Machine is not on the network.

4.15.3.96 #define ENOPKG 65 /* Package not installed */

```
#include <core/error.h>
```

Package not installed.

4.15.3.97 #define ENOPROTOPT 92 /* Protocol not available */

```
#include <core/error.h>
```

Protocol not available.

4.15.3.98 #define ENOSPC 28 /* No space left on device */

```
#include <core/error.h>
```

No space left on device.

4.15.3.99 #define ENOSR 63 /* Out of streams resources */

```
#include <core/error.h>
```

Out of streams resources.

4.15.3.100 #define ENOSTR 60 /* Device not a stream */

```
#include <core/error.h>
```

Device not a stream.

4.15.3.101 #define ENOSYS 38 /* Function not implemented */

```
#include <core/error.h>
```

Function not implemented.

4.15.3.102 #define ENOTBLK 15 /* Block device required */

```
#include <core/error.h>
```

Block device required.

4.15.3.103 #define ENOTCONN 107 /* Transport endpoint is not connected */

```
#include <core/error.h>
```

Transport endpoint is not connected.

4.15.3.104 #define ENOTDIR 20 /* Not a directory */

```
#include <core/error.h>
```

Not a directory.

4.15.3.105 #define ENOTEMPTY 39 /* Directory not empty */

```
#include <core/error.h>
```

Directory not empty.

4.15.3.106 #define ENOTNAM 118 /* Not a XENIX named type file */

```
#include <core/error.h>
```

Not a XENIX named type file.

4.15.3.107 #define ENOTRECOVERABLE 131 /* State not recoverable */

```
#include <core/error.h>
```

State not recoverable.

4.15.3.108 #define ENOTSOCK 88 /* Socket operation on non-socket */

```
#include <core/error.h>
```

Socket operation on non-socket.

4.15.3.109 #define ENOTTY 25 /* Not a typewriter */

```
#include <core/error.h>
```

Not a typewriter.

4.15.3.110 #define ENOTUNIQ 76 /* Name not unique on network */

```
#include <core/error.h>
```

Name not unique on network.

4.15.3.111 #define ENXIO 6 /* No such device or address */

```
#include <core/error.h>
```

No such device or address.

4.15.3.112 #define EOF (-1)

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

EOF - symbol signifying End Of File.

4.15.3.113 #define EOPNOTSUPP 95 /* Operation not supported on transport endpoint */

```
#include <core/error.h>
```

Operation not supported on transport endpoint.

4.15.3.114 #define EOVERFLOW 75 /* Value too large for defined data type */

```
#include <core/error.h>
```

Value too large for defined data type.

4.15.3.115 #define EPERM 1 /* Operation not permitted */

```
#include <core/error.h>
```

Operation not permitted.

4.15.3.116 #define EPNOSUPPORT 96 /* Protocol family not supported */

```
#include <core/error.h>
```

Protocol family not supported.

4.15.3.117 #define EPIPE 32 /* Broken pipe */

```
#include <core/error.h>
```

Broken pipe.

4.15.3.118 #define EPROTO 71 /* Protocol error */

```
#include <core/error.h>
```

Protocol error.

4.15.3.119 #define EPROTONOSUPPORT 93 /* Protocol not supported */

```
#include <core/error.h>
```

Protocol not supported.

4.15.3.120 #define EPROTOTYPE 91 /* Protocol wrong type for socket */

```
#include <core/error.h>
```

Protocol wrong type for socket.

4.15.3.121 #define ERANGE 34 /* Math result not representable */

```
#include <core/error.h>
```

Math result not representable.

4.15.3.122 #define EREMCHG 78 /* Remote address changed */

```
#include <core/error.h>
```

Remote address changed.

4.15.3.123 #define EREMOTE 66 /* Object is remote */

```
#include <core/error.h>
```

Object is remote.

4.15.3.124 #define EREMOTEIO 121 /* Remote I/O error */

```
#include <core/error.h>
```

Remote I/O error.

4.15.3.125 #define ERESTART 85 /* Interrupted system call should be restarted */

```
#include <core/error.h>
```

Interrupted system call should be restarted.

4.15.3.126 #define EROFS 30 /* Read-only file system */

```
#include <core/error.h>
```

Read-only file system.

4.15.3.127 #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.

4.15.3.128 #define ESHUTDOWN 108 /* Cannot send after transport endpoint shutdown */

```
#include <core/error.h>
```

Cannot send after transport endpoint shutdown.

4.15.3.129 #define ESOCKTNOSUPPORT 94 /* Socket type not supported */

```
#include <core/error.h>
```

Socket type not supported.

4.15.3.130 #define ESPIPE 29 /* Illegal seek */

```
#include <core/error.h>
```

Illegal seek.

4.15.3.131 #define ESRCH 3 /* No such process */

```
#include <core/error.h>
```

No such process.

4.15.3.132 #define ESRMNT 69 /* Srmount error */

```
#include <core/error.h>
```

Srmount error.

4.15.3.133 #define ESTALE 116 /* Stale file handle */

```
#include <core/error.h>
```

Stale file handle.

4.15.3.134 #define ESTRPIPE 86 /* Streams pipe error */

```
#include <core/error.h>
```

Streams pipe error.

4.15.3.135 #define ETIME 62 /* Timer expired */

```
#include <core/error.h>
```

Timer expired.

4.15.3.136 #define ETIMEDOUT 110 /* Connection timed out */

```
#include <core/error.h>
```

Connection timed out.

4.15.3.137 #define ETOOMANYREFS 109 /* Too many references: cannot splice */

```
#include <core/error.h>
```

Too many references: cannot splice.

4.15.3.138 #define ETXTBSY 26 /* Text file busy */

```
#include <core/error.h>
```

Text file busy.

4.15.3.139 #define EUCLEAN 117 /* Structure needs cleaning */

```
#include <core/error.h>
```

Structure needs cleaning.

4.15.3.140 #define EUNATCH 49 /* Protocol driver not attached */

```
#include <core/error.h>
```

Protocol driver not attached.

4.15.3.141 #define EUSERS 87 /* Too many users */

```
#include <core/error.h>
```

Too many users.

4.15.3.142 #define EWOULDBLOCK EAGAIN /* Operation would block */

```
#include <core/error.h>
```

Operation would block.

4.15.3.143 #define EXDEV 18 /* Cross-device link */

```
#include <core/error.h>
```

Cross-device link.

4.15.3.144 #define EXFULL 54 /* Exchange full */

```
#include <core/error.h>
```

Exchange full.

4.15.3.145 #define INT_MAX ((int)(~0U>>1))

```
#include <limits.h>
```

Maximum value for an object of type `int`. Minimum Acceptable Value: 2 147 483 647

4.15.3.146 #define INT_MIN (-INT_MAX - 1)

```
#include <limits.h>
```

Minimum value for an object of type `int`. Maximum Acceptable Value: -2 147 483 647

4.15.3.147 #define LLONG_MAX ((long long)(~0ULL>>1))

```
#include <limits.h>
```

Maximum value for an object of type `long long`. Minimum Acceptable Value: +9223372036854775807

4.15.3.148 #define LLONG_MIN ((long long)(-LLONG_MAX - 1))

```
#include <limits.h>
```

Minimum value for an object of type `long long`. Maximum Acceptable Value: -9223372036854775807

4.15.3.149 #define LONG_MAX ((long)(~0UL>>1))

```
#include <limits.h>
```

Maximum value for an object of type `long`. Minimum Acceptable Value: +2 147 483 647

4.15.3.150 #define LONG_MIN (-LONG_MAX - 1)

```
#include <limits.h>
```

Minimum value for an object of type `long`. Maximum Acceptable Value: -2 147 483 647

4.15.3.151 #define M_CACHE_PAGES 1

```
#include <malloc.h>
```

For setting number of pages that not to be munmapped to reuse them quickly again.

4.15.3.152 #define MAP_ANONYMOUS (1 << 0)

```
#include <core/mman.h>
```

The mapping is not backed by any file; its contents are initialized to zero.

4.15.3.153 #define MAP_FAILED ((void *)-1)

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

Error return value from `mmap()`.

4.15.3.154 #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.

4.15.3.155 #define MAP_GROWSDOWN (1 << 6)

```
#include <core/mman.h>
```

This flag may be used with `MAP_STACK`. It indicates to the kernel virtual memory system that the mapping should extend downward in memory. `mmap()` calls will return the highest address on the stack.

4.15.3.156 #define MAP_PHYS_NON_CACHED (1 << 28)

```
#include <driver/mem/phys.h>
```

Memory will be mapped as non-cached.

4.15.3.157 #define MAP_PHYS_NON_SECURE (1 << 29)

```
#include <driver/mem/phys.h>
```

Non-secure memory region will be mapped.

4.15.3.158 #define MAP_POPULATE (1 << 1)

```
#include <core/mman.h>
```

Populate (prefault) page tables for a mapping.

4.15.3.159 #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.

4.15.3.160 #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.

4.15.3.161 #define MAP_STACK (1 << 5)

```
#include <core/mman.h>
```

Allocate the mapping at an address suitable for a process or thread stack. Use for SafeStack mechanism.

4.15.3.162 #define MAX_CPUS (32)

```
#include <sched.h>
```

The maximum cpu number

4.15.3.163 #define NUM_MILLIS_IN_SEC (1000)

```
#include <time.h>
```

The number millisecond in second

4.15.3.164 #define NUM_NANOS_IN_MILLI (1000000L)

```
#include <time.h>
```

The number nanosecond in millisecond

4.15.3.165 #define NUM_NANOS_IN_SEC (1000000000ULL)

```
#include <time.h>
```

The number nanosecond in second

4.15.3.166 #define NUM_NANOS_IN_USEC (1000)

```
#include <time.h>
```

The number nanosecond in microsecond

4.15.3.167 #define NUM_SECONDS_IN_MIN (60)

```
#include <time.h>
```

The number second in minute

4.15.3.168 #define PGOFF_SHIFT 12

```
#include <core/mman.h>
```

Offset shift.

4.15.3.169 #define PRId16 __16_PREFIX "d"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the value of type int16_t

4.15.3.170 #define PRId32 "d"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the value of type int32_t

4.15.3.171 #define PRId64 __64_PREFIX "d"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the value of type int64_t

4.15.3.172 #define PRIu16 __16_PREFIX "u"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the value of type uint16_t

4.15.3.173 #define PRIu32 "u"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the value of type uint32_t

4.15.3.174 #define PRIu64 __64_PREFIX "u"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the value of type uint64_t

4.15.3.175 #define PRIx16 __16_PREFIX "x"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the 16-bit value in hex

4.15.3.176 #define PRIx32 "x"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the 32-bit value in hex

4.15.3.177 #define PRIx64 __64_PREFIX "x"

```
#include <inttypes.h>
```

Macro to be used in a format string to print the 64-bit value in hex

4.15.3.178 #define PROT_EXEC 4

```
#include <core/mman.h>
```

Pages may be executed.

4.15.3.179 #define PROT_NONE 0

```
#include <core/mman.h>
```

Pages may not be accessed.

4.15.3.180 #define PROT_READ 1

```
#include <core/mman.h>
```

Pages may be read.

4.15.3.181 #define PROT_WRITE 2

```
#include <core/mman.h>
```

Pages may be written.

4.15.3.182 #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](#)

4.15.3.183 #define SCHAR_MAX (127)

```
#include <limits.h>
```

Maximum value for an object of type `signed char`. Minimum Acceptable Value: 127

4.15.3.184 #define SCHAR_MIN (-128)

```
#include <limits.h>
```

Minimum value for an object of type `signed char`. Maximum Acceptable Value: -128

4.15.3.185 #define SCNd16 __16_PREFIX "d"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the value of type `int16_t`

4.15.3.186 #define SCNd32 "d"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the value of type `int32_t`

4.15.3.187 #define SCNd64 __64_PREFIX "d"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the value of type `int64_t`

4.15.3.188 #define SCNu16 __16_PREFIX "u"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the value of type `uint16_t`

4.15.3.189 #define SCNu32 "u"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the value of type `uint32_t`

4.15.3.190 #define SCNu64 __64_PREFIX "u"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the value of type `uint64_t`

4.15.3.191 #define SCNx16 __16_PREFIX "x"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the 16-bit value in hex

4.15.3.192 #define SCNx32 "x"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the 32-bit value in hex

4.15.3.193 #define SCNx64 __64_PREFIX "x"

```
#include <inttypes.h>
```

Macro to be used in a format string to scan the 64-bit value in hex

4.15.3.194 #define SHRT_MAX (32767)

```
#include <limits.h>
```

Maximum value for an object of type `signed short int`. Minimum Acceptable Value: 32767

4.15.3.195 #define SHRT_MIN (-32768)

```
#include <limits.h>
```

Minimum value for an object of type `signed short int`. Maximum Acceptable Value: -32768

4.15.3.196 #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.

4.15.3.197 #define UCHAR_MAX (255)

```
#include <limits.h>
```

Maximum value for an object of type `unsigned char`. Minimum Acceptable Value: 255

4.15.3.198 #define UINT_MAX (~0U)

```
#include <limits.h>
```

Maximum value for an object of type `unsigned`. Minimum Acceptable Value: 4 294 967 295

4.15.3.199 #define ULLONG_MAX (~0ULL)

```
#include <limits.h>
```

Maximum value for an object of type `unsigned long long`. Minimum Acceptable Value: 18446744073709551615

4.15.3.200 #define ULONG_MAX (~0UL)

```
#include <limits.h>
```

Maximum value for an object of type `unsigned long`. Minimum Acceptable Value: 4 294 967 295

4.15.3.201 #define USHRT_MAX (0xFFFFU)

```
#include <limits.h>
```

Maximum value for an object of type unsigned short. Minimum Acceptable Value: 65 535

4.15.3.202 #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.

4.15.3.203 #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.

4.15.3.204 #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 Enumeration Type Documentation

4.15.5.1 anonymous enum

```
#include <unistd.h>
```

sysconf system resources' names.

Enumerator

_SC_OPEN_MAX Max number of files that TA can have open.

_SC_PAGESIZE Size of a page in bytes.

_SC_PHYS_PAGES The number of pages of physical memory.

_SC_AVPHYS_PAGES The number of currently available physical pages.

_SC_NPROCESSORS_CONF The number of processors configured.

_SC_NPROCESSORS_ONLN The number of processors currently online (same as previous).

_SC_THREADS Inquire about the parameter corresponding to _POSIX_THREADS.

_SC_THREAD_KEYS_MAX Inquire about the parameter corresponding to _POSIX_THREAD_KEYS_MAX

_SC_THREAD_THREADS_MAX Inquire about the parameter corresponding to _POSIX_THREAD_THREADS_MAX.

_SC_THREAD_STACK_MIN Inquire about the parameter corresponding to _POSIX_THREAD_STACK_MIN.

4.15.6 Function Documentation

4.15.6.1 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.

4.15.6.2 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> .

4.15.6.3 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.

4.15.6.4 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.
------------------	---------------------

4.15.6.5 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 TEEGRIS 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

4.15.6.6 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

4.15.6.7 double atof (const char * *nptr*)

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

Convert a string to a double.

Parameters

in	<i>nptr</i>	pointer to the string
----	-------------	-----------------------

Return values

<i>Converted</i>	value
------------------	-------

4.15.6.8 int atoi (const char * *nptr*)

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

Convert a string to an integer.

Parameters

in	<i>nptr</i>	pointer to the string
----	-------------	-----------------------

Return values

<i>Converted</i>	value
------------------	-------

4.15.6.9 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 size is equal to zero.
<i>NULL</i>	and sets <i>errno</i> to <i>ENOMEM</i> if there is no enough memory to be allocated.

4.15.6.10 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. <i>errno</i> variable is set appropriately.

4.15.6.11 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.

4.15.6.12 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.
----	---------------	--------------------

4.15.6.13 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.

4.15.6.14 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");
```

4.15.6.15 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> .
----	------------	--

4.15.6.16 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. errno is set appropriately.

4.15.6.17 int fsync (int *fd*)

```
#include <unistd.h>
```

Synchronize a file's in-core state with storage device.

Parameters

in	<i>fd</i>	File descriptor.
----	-----------	------------------

Return values

0	on success.
-1	on error. errno is set appropriately.

4.15.6.18 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. errno is set appropriately.

4.15.6.19 int* get_errno_addr (void)

```
#include <errno.h>
```

This function should NOT be used directly, use 'errno' instead.

Returns

thread local pointer to [errno](#)

4.15.6.20 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.

4.15.6.21 int getcpu (void)

```
#include <unistd.h>
```

Return the number of CPU on which current thread is performed.

Returns

function result.

4.15.6.22 char* getenv (const char * name)

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

get an environment variable function stub

Parameters

in	<i>name</i>	environment variable name
----	-------------	---------------------------

Return values

NULL	
------	--

4.15.6.23 pid_t getpid (void)

```
#include <unistd.h>
```

Return the process ID of the calling process.

Returns

function result.

4.15.6.24 pid_t gettid (void)

```
#include <unistd.h>
```

Return the thread ID of the calling thread.

Returns

function result.

4.15.6.25 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> .

4.15.6.26 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> .

4.15.6.27 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;
```

4.15.6.28 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.

4.15.6.29 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.

4.15.6.30 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.

4.15.6.31 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.

4.15.6.32 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.

4.15.6.33 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.

4.15.6.34 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.

4.15.6.35 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.

4.15.6.36 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.

4.15.6.37 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.

4.15.6.38 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.

4.15.6.39 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.

4.15.6.40 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.

4.15.6.41 `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.

4.15.6.42 `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.
<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.

4.15.6.43 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.

4.15.6.44 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> .

4.15.6.45 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.

4.15.6.46 `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	<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.

4.15.6.47 `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	<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.

4.15.6.48 `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.

4.15.6.49 `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 last occurrence of `c`, or NULL.

4.15.6.50 `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.

4.15.6.51 `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

0	function completed work with success.
!=0	function completed work with error.

4.15.6.52 `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 /dev/phys). 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 /dev/phys). Defined in driver/mem/phys.h header. • MAP_STACK - Allocate the mapping at an address suitable for a process or thread stack. Use for SafeStack mechanism. • MAP_GROWSDOWN - Get the highest address on the stack allocated with MAP_STACK.
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);
```

4.15.6.53 int mprotect (void * *addr*, size_t *len*, int *prot*)

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

Changes permission of memory region.

Parameters

in	<i>addr</i>	address where memory was previously mapped.
in	<i>len</i>	size of the memory area.
in	<i>prot</i>	new protection flags.

Return values

0	no error.
-1	on failure. errno variable is set appropriately. <ul style="list-style-type: none"> • EINVAL - <i>addr</i> is not multiple of page size. • EACCES - <i>prot</i> argument violates protection of file mapped. • EAGAIN - no memory resources to create writable private mapping for file. • ENOMEM - specified address range is invalid or includes not mapped zones. • EOPNOTSUPP - <i>prot</i> argument specifies PROT_EXEC.

4.15.6.54 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	<i>t</i>	time in milliseconds.
out	<i>a</i>	Pointer to timespec structure to save result.

4.15.6.55 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	<i>addr</i>	address where memory was previously mapped.
in	<i>length</i>	size of the memory area.

Return values

0	no error.
-1	on failure. errno variable is set appropriately. <ul style="list-style-type: none"> • EINVAL - invalid length specified. • EPERM - memory area other than heap is specified.

Example:

```
ret = munmap(buf, 0x8000);
if (ret) {
    printf("munmap error\n");
    panic(1);
}
```

4.15.6.56 int nanosleep (const 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	<i>req</i>	Pointer to requested time.
in	<i>rem</i>	Pointer to remained time.

Return values

0	on success.
-1	on failure. errno 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);
}

```

4.15.6.57 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	<i>pathname</i>	namespace path for specific device
in	<i>flags</i>	file access mode
in	...	(optional) file mode

Return values

>0	a descriptor associated with the device.
-1	on error and sets errno to appropriated value: <ul style="list-style-type: none"> • EINVAL – pathname is invalid or device not registered; • ENOMEM – there is not enough memory for internal operations; • EMFILE – 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;
}

```

4.15.6.58 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("%.2f\n",d);
```

4.15.6.59 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");
```

4.15.6.60 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.

4.15.6.61 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.

4.15.6.62 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.

4.15.6.63 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

<i>>=0</i>	on success.
<i>EOF</i>	on error.

4.15.6.64 void qsort (void * *base*, size_t *nmem*, 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>nmem</i>	number of elements in array.
in	<i>size</i>	size of each element in array.
in	<i>compar</i>	comparison function.

4.15.6.65 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 of 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> .

4.15.6.66 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.

4.15.6.67 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 <code>NULL</code> or a pointer suitable to be passed to <code>free()</code> if <code>size</code> was equal to 0.

4.15.6.68 `int rename (const char * pathname, const char * new_pathname)`

```
#include <unistd.h>
```

Rename file.

Parameters

in	<i>pathname</i>	is name of file to rename.
in	<i>new_pathname</i>	is new name of file.

Return values

0	on success.
-1	on error. <code>errno</code> is set appropriately.

4.15.6.69 int rmdir (char * *dir_name*)

```
#include <unistd.h>
```

Remove a directory at file system.

Parameters

in	<i>dir_name</i>	is name of the directory.
----	-----------------	---------------------------

Return values

0	on success.
-1	on error. errno is set appropriately.

4.15.6.70 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.

4.15.6.71 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.

4.15.6.72 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.

4.15.6.73 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.

4.15.6.74 int setenv (const char * *name*, const char * *value*, int *overwrite*)

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

change or add environment variable function stub

Parameters

in	<i>name</i>	environment variable name
in	<i>value</i>	new value for environment variable
in	<i>overwrite</i>	replace variable's value to new if name exists

Return values

0	
---	--

4.15.6.75 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);
}

```

4.15.6.76 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	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	Number of characters printed if succeed.
-1	if error occurred.

4.15.6.77 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

<i>number</i>	of characters printed if succeed.
-1	if error occurred.

Example:

```
char str[80];
sprintf(str, "Value of Pi = %f", M_PI);
```

4.15.6.78 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	<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>	of characters printed if succeed.
-1	if encoding error occurred.
0	if constrain violation occurred.

4.15.6.79 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 <i>scanf</i> .

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);
```

4.15.6.80 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.

4.15.6.81 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. <i>errno</i> is set appropriately.

4.15.6.82 `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;
}
```

4.15.6.83 `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	<i>dest</i>	Pointer to string to append.
in	<i>dest_max</i>	max number of bytes to modify in the <code>dest</code> .
in	<i>src</i>	Pointer to source string.

Return values

0	function completed work with success.
!=0	function completed work with error.

4.15.6.84 char* strchr (const char * s, int c)

```
#include <string.h>
```

Find first occurrence of character *c* in string *s*.

Parameters

in	s	Pointer to string to search in.
in	c	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');
}
```

4.15.6.85 char* strchrnul (const char * s, int c)

```
#include <string.h>
```

Find first occurrence of character *c* in string *s*.

Parameters

in	s	Pointer to string to search in.
in	c	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.

4.15.6.86 int strcmp (const char * s1, const char * s2)

```
#include <string.h>
```

Compare the two strings s1 and s2.

Parameters

in	s1	Pointer to first string to compare.
in	s2	Pointer to second string to compare.

Return values

0	strings are equal.
<0	string s1 has lower value than s2.
>0	string s1 has greater value than s2.

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;
}
```

4.15.6.87 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	dest	pointer to destination string.
in	src	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;
}
```

4.15.6.88 `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	<i>dest</i>	pointer to destination string.
in	<i>dest_max</i>	max number of bytes to modify in the <i>dest</i> .
in	<i>src</i>	pointer to source string.

Return values

0	function completed work with success.
!=0	function completed work with error.

4.15.6.89 `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	<i>s</i>	Pointer to string to search in.
in	<i>reject</i>	String containing the characters not to match.

Returns

number of bytes in the initial segment of *s* which are not in the string *reject*.

4.15.6.90 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	<i>s</i>	pointer to source string.
----	----------	---------------------------

Returns

pointer to duplicated string.

4.15.6.91 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	<i>errnum</i>	error code.
----	---------------	-------------

Returns

pointer to description.

4.15.6.92 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 <i>errno</i> is set.

4.15.6.93 `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.
!=0	function completed work with error.

4.15.6.94 `size_t strcat (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 \0 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.");

    strcat(dst, src, sizeof(dst));

    return 0;
}
```

4.15.6.95 `size_t strlen (const char * s)`

#include <string.h>

Calculate the length of the string `s`, excluding the terminating null byte (`'\0'`).

Parameters

in	<code>s</code>	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;
}
```

4.15.6.96 `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	<code>dest</code>	Pointer to string to append.
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.

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;
}
```

4.15.6.97 `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 \0 terminator.

Return values

<code>0</code>	function completed work with success.
<code>!=0</code>	function completed work with error.

4.15.6.98 `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	<code>s1</code>	Pointer to first string to compare.
in	<code>s2</code>	Pointer to second string to compare.
in	<code>n</code>	maximal number of bytes to compare if string does not have \0 terminator.

Return values

<code>0</code>	strings are equal.
<code><0</code>	string <code>s1</code> has lower value than <code>s2</code> .
<code>>0</code>	string <code>s1</code> has greater value than <code>s2</code> .

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;
}

```

4.15.6.99 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;
}

```

4.15.6.100 `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	<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.
in	<code>n</code>	maximal number of bytes to copy if string does not have \0 terminator.

Return values

<code>0</code>	function completed work with success.
<code>!=0</code>	function completed work with error.

4.15.6.101 `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	<code>s</code>	Pointer to string.
in	<code>n</code>	maximal number of bytes to scan.

Returns

number of bytes in the string, or `n` if string is nonger than `n`.

Example:

```
int main ()
{
    char string[32] = "hello, world";
    printf("String length is %zu\n", strlen(string, 10));
    return 0;
}
```

4.15.6.102 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	<i>s</i>	pointer to string.
in	<i>s_max</i>	maximal number of bytes to scan.

Return values

0	<i>s</i> is a null pointer.
!=0	number of symbols that precede the terminating null character.

4.15.6.103 char* strrchr (const char * s, int c)

```
#include <string.h>
```

Find last 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 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');
}
```


4.15.6.104 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.

4.15.6.105 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;
}
```

4.15.6.106 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) an infinity.
- (iii) 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. 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 .

4.15.6.107 float strtodf (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 .

4.15.6.108 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;
```

4.15.6.109 `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	<i>s</i>	The string to be parsed.
in	<i>sep</i>	The argument specifies a set of bytes that delimit the tokens in the parsed string.
out	<i>p</i>	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.

4.15.6.110 `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	<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
- LONG_MIN if an underflow occurs. `errno` is set to `ERANGE`.
- LONG_MAX if an overflow occurs. `errno` is set to `ERANGE`.

4.15.6.111 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. errno is set to ERANGE .
<i>-HUGE_VAL</i>	if an underflow occurs. errno is set to ERANGE .

4.15.6.112 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](#).

4.15.6.113 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.
<i>ULONG_MAX</i>	if an overflow occurs. errno is set to ERANGE .

4.15.6.114 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.
<i>ULONG_MAX</i>	if an overflow occurs. errno is set to ERANGE .

4.15.6.115 long sysconf (int *name*)

```
#include <unistd.h>
```

Get configuration information at run time.

Allows an application to test at compile or run time whether certain options are supported, or what the value is of certain configurable constants or limits.

Parameters

in	<i>name</i>	System resource to find out.
----	-------------	------------------------------

Returns

- -1 if *name* is invalid and set [errno](#) to indicate the error.
- value of the system resource.

4.15.6.116 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.

4.15.6.117 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.

4.15.6.118 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.
-------------	------------------

4.15.6.119 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.
-------------	-----------------

4.15.6.120 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.

4.15.6.121 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.

4.15.6.122 static int toupper (int c) [static]

```
#include <ctype.h>
```

Convert lowercase letter to uppercase.

Parameters

in	c	symbol for converting
----	---	-----------------------

Returns

- the value of the converted letter,
- c if the conversion was not possible.

4.15.6.123 int unlink (const char * *pathname*)

```
#include <unistd.h>
```

Remove a link to a file.

Parameters

in	<i>pathname</i>	is name of file to remove.
----	-----------------	----------------------------

Return values

0	on success.
-1	on error. errno is set appropriately.

4.15.6.124 int unsetenv (const char * *name*)

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

delete environment variable function stub

Parameters

in	<i>name</i>	environment variable name
----	-------------	---------------------------

Return values

0	
---	--

4.15.6.125 void uuid_clear (uuid_t * uu)

```
#include <uuid/uuid.h>
```

set value to zero UUID.

Parameters

in	<i>uu</i>	value to cleanup.
----	-----------	-------------------

4.15.6.126 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	uu2 value is lexicographically greater than uu1.
>0	uu2 value is lexicographically less than uu1.

4.15.6.127 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.
-----	------------	------------------

4.15.6.128 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.

4.15.6.129 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.

4.15.6.130 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.

4.15.6.131 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.

4.15.6.132 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 [vprintf\(\)](#), 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 TEEGRIS 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 strp to NULL

4.15.6.133 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 strp to NULL

4.15.6.134 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.

4.15.6.135 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 <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 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;
}
```

4.15.6.136 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 <i>va_start</i> . <i>va_list</i> 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.

4.15.6.137 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 va_start. va_list 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;
}
```

4.15.6.138 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 <i>va_start</i> . <i>va_list</i> is a special type.

Return values

<i>number</i>	of characters printed if succeed.
0	if constrain violation occurred.
-1	if encoding error occurred.

4.15.6.139 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

<i>>=0</i>	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.

4.15.6.140 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.

4.15.6.141 `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.

4.15.7 Variable Documentation**4.15.7.1 `FILE* stdin`**

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

Standard input stream (stub).

PRIVATE

4.16 Stat

Macros

- #define **S_ACCPERM** (S_IRWXU | S_IRWXG | S_IRWXO)

Functions

- int **fstat** (int fd, struct **stat** *buf)
Get file status of a given file descriptor (system call)
- int **stat** (const char *pathname, struct **stat** *buf)
Get file status of a given path name.
- int **mkdir** (const char *pathname, **mode_t** mode)
Create directory at file system.

4.16.1 Detailed Description

4.16.2 Macro Definition Documentation

4.16.2.1 #define S_ACCPERM (S_IRWXU | S_IRWXG | S_IRWXO)

#include <sys/stat.h>

mask to extract permission from **stat.st_mode**

4.16.3 Function Documentation

4.16.3.1 int fstat (int fd, struct stat * buf)

#include <sys/stat.h>

Get file status of a given file descriptor (system call)

Parameters

in	<i>fd</i>	file descriptor
out	<i>buf</i>	pointer to struct stat buffer

Returns

Error code

4.16.3.2 int mkdir (const char * *pathname*, mode_t *mode*)

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

Create directory at file system.

Parameters

in	<i>pathname</i>	directory name
in	<i>mode</i>	specifies the permissions.

Return values

0	on success.
-1	on error. errno is set appropriately.

4.16.3.3 int stat (const char * *pathname*, struct stat * *buf*)

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

Get file status of a given path name.

Parameters

in	<i>pathname</i>	name of a file
out	<i>buf</i>	pointer to struct stat buffer

Returns

Error code

4.17 Teesl_NFC

Typedefs

- typedef struct [TEES_NFCHandler](#) * [TEES_NFCHandler](#)

Functions

- TEE_Result [TEES_NFCInit](#) ([TEES_NFCHandler](#) *handler)
Initialize NFC device and set transfer parameters to handler.
- void [TEES_NFCExit](#) ([TEES_NFCHandler](#) handler)
Terminate connection to NFC device.
- TEE_Result [TEES_NFCWrite](#) ([TEES_NFCHandler](#) handler, void *tx_buf, size_t len)
Write data buffer tx to initialized NFC device.
- TEE_Result [TEES_NFCRead](#) ([TEES_NFCHandler](#) handler, char *rx_buf, size_t rx_buf_len, size_t *rsp_len)
Read data buffer rx from initialized NFC device.
- TEE_Result [TEES_NFCReadWait](#) ([TEES_NFCHandler](#) handler, char *rx_buf, size_t rx_buf_len, size_t *rsp_len, int ms_timeout)
Read data buffer rx from initialized NFC device.
- TEE_Result [TEES_NFCSetMode](#) ([TEES_NFCHandler](#) handler, unsigned long mode)
Set NFC device mode.
- TEE_Result [TEES_NFCWakeUp](#) ([TEES_NFCHandler](#) handler)
Send wake up command to initialized NFC device.
- TEE_Result [TEES_NFCSleep](#) ([TEES_NFCHandler](#) handler)
Send sleep command to initialized NFC device.

4.17.1 Detailed Description

4.17.2 Typedef Documentation

4.17.2.1 TEES_NFCHandler

```
#include <tee_nfc.h>
```

Handler for NFC device.

4.17.3 Function Documentation

4.17.3.1 void TEES_NFCExit (TEES_NFCHandler handler)

```
#include <tee_nfc.h>
```

Terminate connection to NFC device.

Parameters

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

4.17.3.2 TEE_Result TEES_NFCInit (TEES_NFCHandler * *handler*)

```
#include <tee_nfc.h>
```

Initialize NFC device and set transfer parameters to *handler*.

Parameters

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.

4.17.3.3 TEE_Result TEES_NFCRead (TEES_NFCHandler *handler*, char * *rx_buf*, size_t *rx_buf_len*, size_t * *rsp_len*)

```
#include <tee_nfc.h>
```

Read data buffer *rx* from initialized NFC device.

Parameters

in	<i>handler</i>	initialized parameters.
out	<i>rx_buf</i>	is pointer to buffer to receive.
in	<i>rx_buf_len</i>	is length of storage buffer for receiving.
out	<i>rsp_len</i>	is pointer to store length of response data.

Return values

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

4.17.3.4 TEE_Result TEES_NFCReadWait (TEES_NFCHandler *handler*, char * *rx_buf*, size_t *rx_buf_len*, size_t * *rsp_len*, int *ms_timeout*)

```
#include <tee_nfc.h>
```

Read data buffer *rx* from initialized NFC device.

Parameters

in	<i>handler</i>	initialized parameters.
out	<i>rx_buf</i>	is pointer to buffer to receive.
in	<i>rx_buf_len</i>	is length of storage buffer for receiving.
out	<i>rsp_len</i>	is pointer to store length of response data.
in	<i>ms_timeout</i>	time in millisecond during which data from nfc is expected.

Return values

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

4.17.3.5 TEE_Result TEES_NFCSetMode (TEES_NFCHandler *handler*, unsigned long *mode*)

```
#include <tee_nfc.h>
```

Set NFC device mode.

Parameters

in	<i>handler</i>	initialized parameters.
in	<i>mode</i>	to set on NFC.

Return values

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

4.17.3.6 TEE_Result TEES_NFCSleep (TEES_NFCHandler *handler*)

```
#include <tee_nfc.h>
```

Send sleep command to initialized NFC 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.

4.17.3.7 TEE_Result TEES_NFCWakeUp (TEES_NFCHandler *handler*)

```
#include <tee_nfc.h>
```

Send wake up command to initialized NFC 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.

4.17.3.8 TEE_Result TEES_NFCWrite (TEES_NFCHandler *handler*, void * *tx_buf*, size_t *len*)

```
#include <tee_nfc.h>
```

Write data buffer *tx* to initialized NFC device.

Parameters

in	<i>handler</i>	initialized parameters.
in	<i>tx_buf</i>	is pointer to buffer to transfer.
in	<i>len</i>	of buffer.

Return values

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

4.18 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_SMCHandle (TEES_SMCHandle *handle)`
Initialize handler fields.
- `TEE_Result TEES_SMCFini (TEES_SMCHandle handle)`
release SMC interface handle.
- `TEE_Result TEES_SMCCCommand (TEES_SMCHandle handle, TEES_SMCData *data)`
Send SMC command to EL3.

4.18.1 Detailed Description

4.18.2 Function Documentation

4.18.2.1 TEE_Result TEES_SMCCCommand (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_SMCData 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);
...

```

4.18.2.2 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.
--------------------	-----------------------------------

4.18.2.3 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.
--------------------	-----------------------------------

4.19 Teesl_udf

Functions

- int **TEES_UDF_InitDriver** (char *name, struct fops *fops, unsigned int drvid, struct usr_drv_info **info)
This function is used to initialize and register UDF driver.
- int **TEES_UDF_Init_FS_Driver** (char *name, struct fops *fops, unsigned int drvid, struct usr_drv_info **info)
This function is used to initialize and register UDF FS driver.
- int **TEES_UDF_FiniDriver** (struct usr_drv_info *info)
This function is used to de-initialize and stop UDF driver.
- int **TEES_UDF_RegisterIoctlDesc** (struct usr_drv_info *info, unsigned int cmd, const struct ioctl_desc *desc)
*Register an **ioctl()** cmd for UDF driver.*

4.19.1 Detailed Description

4.19.2 Function Documentation

4.19.2.1 int TEES_UDF_FiniDriver (struct usr_drv_info * info)

```
#include <udf.h>
```

This function is used to de-initialize and stop UDF driver.

Parameters

in	<i>info</i>	driver handle.
----	-------------	----------------

Return values

0	on success.
-1	on fail and errno is set.

4.19.2.2 int TEES_UDF_Init_FS_Driver (char * name, struct fops * fops, unsigned int drvid, struct usr_drv_info ** info)

```
#include <udf.h>
```

This function is used to initialize and register UDF FS driver.

Parameters

in	<i>name</i>	name.
in	<i>fops</i>	file operations.
in	<i>drvid</i>	id.
in	<i>info</i>	driver handle.

Return values

0	on success.
-1	on fail and errno is set.

4.19.2.3 int TEES_UDF_InitDriver (char * *name*, struct fops * *fops*, unsigned int *drvid*, struct usr_drv_info ** *info*)

```
#include <udf.h>
```

This function is used to initialize and register UDF driver.

Parameters

in	<i>name</i>	name.
in	<i>fops</i>	file operations.
in	<i>drvid</i>	id.
in	<i>info</i>	driver handle.

Return values

0	on success.
-1	on fail and errno is set.

4.19.2.4 int TEES_UDF_RegisterIoctlDesc (struct usr_drv_info * *info*, unsigned int *cmd*, const struct ioctl_desc * *desc*)

```
#include <udf.h>
```

Register an `ioctl()` cmd for UDF driver.

Parameters

in	<i>info</i>	A pointer to struct usr_drv_info that contains information describing the driver.
in	<i>cmd</i>	An ioctl() command being added.
in	<i>desc</i>	An ioctl() arguments descriptor, see below for details.

Return values

0	on success, -1 otherwise. errno variable is set appropriately.
---	--

Example:

```
struct example { uint32_t a; char *b; uint32_t items_cnt; uint32_t items[]; };
```

```
struct example data = { .a = 0xa5a5a5a5, .b = "0123456789", .items_cnt = 3, .items = {0, 1, 2}, };
```

```
struct ioctl\_desc example_desc = { .cnt = 5, .tpl = { { .type = DESC_ATOM_VAL, .len = sizeof(uint32_t) },  
{ .type = DESC_ATOM_REF, .len = 10 }, { .type = DESC_ATOM_ARR_CNT_0, .len = sizeof(uint32_t) }, { .type =  
DESC_ATOM_ARR_ITEM_0, .len = 1 }, { .type = DESC_ATOM_VAL, .len = sizeof(uint32_t) }, }, };
```

```
ret = TEES\_UDF\_RegisterIoctlDesc(my_driver_info, EXAMPLE_IOCTL_ID, &example_desc)  
;  
if (ret) {  
    printf("udf register ioctl() desc failed, ret = %d\n", ret);  
}
```

4.20 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_BAND` = 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,
`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` = 0x81030010, `TEE_ISOCKET_TLS_ERROR_ECDHE_SHARED_SECRET` = 0x81030011, `TEE_ISOCKET_TLS_ERROR_ECDSA_VERIFY` = 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_CANCELLED = 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_STATUS_EXPIRED = 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_ERROR = 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_CERT = 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_EXPIRED = 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_DENIED = 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_PROTOCOL_VERSION = 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_NO_RENEGOTIATION = 0x81031100, TEE_ISOCKET_TLS_ALERT_MISSING_EXTENSION = 0x81031109, TEE_ISOCKET_TLS_ALERT_UNSUPPORTED_EXTENSION = 0x81031110, TEE_ISOCKET_TLS_ALERT_CERT_REQUIRED = 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_UNKNOWN_CA = 0x81031115,
TEE_ISOCKET_TLS_ALERT_CERT_REQUIRED = 0x81031116 }

```

Proprate 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_HANDSHAKE_FAILED** = 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 }

```

= 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_SHA
= 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_CLIE`
= 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.

4.20.1 Detailed Description

4.20.2 Data Structure Documentation

4.20.2.1 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

4.20.2.2 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

4.20.2.3 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

4.20.2.4 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

4.20.2.5 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	

4.20.2.6 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	

4.20.2.7 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	

4.20.2.8 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	

4.20.2.9 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.

4.20.2.10 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

4.20.2.11 union TEE_tlsSocket_Credentials_s.__unnamed__

Data Fields

TEE_tlsSocket_CertStorageCred *	rootCertStore	Certificate storage - for future extension
TEE_tlsSocket_ServerPDC *	serverCred	Predistributed (explicitly provided)

4.20.2.12 struct TEE_tlsSocket_CallbackInfo_s

Callback description structure.

Data Fields

uint32_t	protocolError	
TEE_tlsSocket_CallbackReasonType	reason	
TEE_Result	result	

4.20.2.13 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 no one 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

4.20.2.14 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

4.20.2.15 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	

4.20.3 Typedef Documentation

4.20.3.1 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().

4.20.3.2 typedef struct TEE_tlsSocket_CB_Data_s TEE_tlsSocket_CB_Data

```
#include <tee_tlssocket.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].

4.20.4 Enumeration Type Documentation

4.20.4.1 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

4.20.4.2 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

4.20.4.3 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

4.20.4.4 anonymous enum

```
#include <tee_tcpsocket.h>
```

TCP Protocol identifier.

Enumerator

TEE_ISOCKET_PROTOCOLID_TCP GP TCP protocol ID

4.20.4.5 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

4.20.4.6 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

4.20.4.7 anonymous enum

```
#include <tee_udpsocket.h>
```

UDP Protocol identifier.

Enumerator

TEE_ISOCKET_PROTOCOLID_UDP GP UDP protocol ID

4.20.4.8 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

4.20.4.9 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

4.20.4.10 anonymous enum

```
#include <tee_tlssocket.h>
```

TLS iSocket API version. Used to ensure API structures matching.

Enumerator

TEE_ISOCKET_TLS_API_VERSION Currently supported version

4.20.4.11 anonymous enum

```
#include <tee_tlssocket.h>
```

TLS Protocol identifier.

Enumerator

TEE_ISOCKET_PROTOCOLID_TLS GP TLS protocol ID

4.20.4.12 anonymous enum

```
#include <tee_tlssocket.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

4.20.4.13 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 Unknown, 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 received 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 received

TEE_ISOCKET_TLS_ALERT_BAD_CERTIFICATE Bad certificate received

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 vulnerable 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 Bad client certificate hash

TEE_ISOCKET_TLS_ALERT_UNKNOWN_PSK_IDENTITY Unknown PSK identity

TEE_ISOCKET_TLS_ALERT_CERT_REQUIRED Client certificate not provided

4.20.4.14 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

4.20.4.15 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

4.20.4.16 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

4.20.4.17 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

4.20.4.18 enum TEE_tlsSocket_ServerCredentialType_e

```
#include <tee_tlssocket.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

4.20.4.19 enum TEE_tlsSocket_StatusRequestType_e

```
#include <tee_tlssocket.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

4.20.4.20 enum TEE_tlsSocket_tlsVersion_e

```
#include <tee_tlssocket.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

4.20.5 Function Documentation

4.20.5.1 TEE_Result TEES_IwtCloseChannel (TEES_IwtHandle iwtCtx)

```
#include <tees_iwt.h>
```

Close active IWT connection iwtCtx.

Parameters

in	iwtCtx	Handle representing active IWT connection
----	--------	---

Return values

TEE_SUCCESS	no error
TEE_ERROR_*	on failure

4.20.5.2 TEE_Result TEES_lwtOpenChannel (const char * *listenerName*, TEES_lwtHandle * *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/" <i>listenerName</i> ". 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

4.20.5.3 TEE_Result TEES_lwtRead (TEES_lwtHandle *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

4.20.5.4 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 desc_atom

Structure that contains description of single data item.

Data Fields

- uint32_t [type](#)
- uint32_t [len](#)

5.9.1 Detailed Description

Structure that contains description of single data item.

5.9.2 Field Documentation

5.9.2.1 uint32_t desc_atom::len

value length

5.9.2.2 uint32_t desc_atom::type

value type

5.10 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

- fops_open [open](#)
- fops_close [close](#)
- fops_truncate [truncate](#)
- fops_read [read](#)
- fops_write [write](#)
- fops_ioctl [ioctl](#)
- fops_stat [stat](#)
- fops_mmap [mmap](#)
- fops_lseek [lseek](#)
- fops_fsync [fsync](#)
- fops_unlink [unlink](#)
- fops_rename [rename](#)
- fops_rmdir [rmdir](#)
- fops_mkdir [mkdir](#)
- fops_readdir [readdir](#)
- fops_lookup [lookup](#)
- fops_probe [probe](#)

5.10.1 Detailed Description

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.

5.10.2 Field Documentation

5.10.2.1 `fops_close fops::close`

callback for close operation

5.10.2.2 `fops_fsync fops::fsync`

callback for fsync operation

5.10.2.3 `fops_ioctl fops::ioctl`

callback for ioctl operation

5.10.2.4 `fops_lookup fops::lookup`

callback for lookup operation

5.10.2.5 `fops_lseek fops::lseek`

callback for seek operation

5.10.2.6 `fops_mkdir fops::mkdir`

callback for mkdir operation

5.10.2.7 `fops_mmap fops::mmap`

callback for mmap operation

5.10.2.8 `fops_open fops::open`

callback for open operation

5.10.2.9 fops_probe fops::probe

callback for probe operation

5.10.2.10 fops_read fops::read

callback for read operation

5.10.2.11 fops_readdir fops::readdir

callback for readdir operation

5.10.2.12 fops_rename fops::rename

callback for rename operation

5.10.2.13 fops_rmdir fops::rmdir

callback for rmdir operation

5.10.2.14 fops_stat fops::stat

callback for stat operation

5.10.2.15 fops_truncate fops::truncate

callback for truncate operation

5.10.2.16 fops_unlink fops::unlink

callback for unlink operation

5.10.2.17 fops_write fops::write

callback for write operation

5.11 ioctl_desc

Structure that template for `ioctl()` parameters parsing.

Data Fields

- uint32_t [cnt](#)
- uint32_t [type](#)
- struct [desc_atom](#) [tpl](#) [32]

5.11.1 Detailed Description

Structure that template for [ioctl\(\)](#) parameters parsing.

5.11.2 Field Documentation

5.11.2.1 uint32_t ioctl_desc::cnt

count of atoms in template

5.11.2.2 struct desc_atom ioctl_desc::tpl[32]

parse template

5.11.2.3 uint32_t ioctl_desc::type

type of descriptor

5.12 smc_data

SMC command description.

Data Fields

- uint64_t [arg_types](#)
- struct [smc_args](#) [args](#)

5.12.1 Detailed Description

SMC command description.

5.12.2 Field Documentation

5.12.2.1 uint64_t smc_data::arg_types

Argument types description

5.12.2.2 struct smc_args smc_data::args

Arguments array

5.13 stat

Data Fields

- uint32_t [st_mode](#)
- uint32_t [st_uid](#)
- uint32_t [st_gid](#)
- uint32_t [st_size](#)
- uint32_t [st_refcount](#)

5.13.1 Detailed Description

Stat struct

5.13.2 Field Documentation

5.13.2.1 uint32_t stat::st_gid

Group ID of file

5.13.2.2 uint32_t stat::st_mode

Mode of file

5.13.2.3 uint32_t stat::st_refcount

Reference counter of file

5.13.2.4 uint32_t stat::st_size

Size of file

5.13.2.5 uint32_t stat::st_uid

User ID of file

5.14 usr_drv_info

Structure that contains information describing the driver.

Data Fields

- const struct [fops](#) * [fops](#)
- int [handle](#)
- void * [priv](#)

5.14.1 Detailed Description

Structure that contains information describing the driver.

5.14.2 Field Documentation

5.14.2.1 const struct [fops](#)* [usr_drv_info::fops](#)

See also

[fops](#)

5.14.2.2 int [usr_drv_info::handle](#)

driver handle

5.14.2.3 void* [usr_drv_info::priv](#)

driver private part

6 File Documentation

6.1 `alloca.h` File Reference

`alloca` implementation

Macros

- #define `alloca` `_alloca`
Allocate memory that is automatically freed.

6.1.1 Detailed Description

`alloca` implementation

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.2 `assert.h` File Reference

Assertion support.

Macros

- #define `assert`(expr) `__assert__(expr, __FILE__, __LINE__)`
Abort the program if assertion is false.

6.2.1 Detailed Description

Assertion support.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.3 `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.

6.3.1 Detailed Description

cache maintenance

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.4 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 */
- #define **EKEYREJECTED** 127 /* Key was rejected by service */
- #define **ENOTRECOVERABLE** 131 /* State not recoverable */

6.5 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 **MAP_STACK** (1 << 5)
- #define **MAP_GROWSDOWN** (1 << 6)
- #define **PROT_NONE** 0
- #define **PROT_READ** 1
- #define **PROT_WRITE** 2
- #define **PROT_EXEC** 4
- #define **PGOFF_SHIFT** 12

6.6 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)
- int `mprotect` (void *addr, size_t len, int prot)

6.6.1 Detailed Description

Memory mapping declarations.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.7 sys/stat.h File Reference

File status syscalls.

Macros

- #define `S_ACCPERM` (S_IRWXU | S_IRWXG | S_IRWXO)

Functions

- int `fstat` (int fd, struct `stat` *buf)
Get file status of a given file descriptor (system call)
- int `stat` (const char *pathname, struct `stat` *buf)
Get file status of a given path name.
- int `mkdir` (const char *pathname, `mode_t` mode)
Create directory at file system.

6.7.1 Detailed Description

File status syscalls.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.8 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 an 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.

6.8.1 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.

6.9 driver.h File Reference

User-space driver API declarations.

Functions

- int [TEES_InitDriver](#) (char *name, struct [fops](#) *fops, unsigned int drvid, struct [usr_drv_info](#) **info)
Register user driver in /dev/ directory under name, using mask of file operations fops and drvid of served asset.
- int [TEES_RegisterIoctlDesc](#) (struct [usr_drv_info](#) *info, unsigned int cmd, struct [ioctl_desc](#) *desc)
Register an ioctl() cmd for driver.
- 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 in /dev/ directory 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. this API can not be used in read()/write() cmd for driver.
- int [TEES_ReleaseUserBuffer](#) (const void *addr, const size_t size)
Deletes the shared mapped area for the specified address range.

6.9.1 Detailed Description

User-space driver API declarations.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.10 driver/mem/phys.h File Reference

Macros

- #define [MAP_PHYS_NON_SECURE](#) (1 << 29)
- #define [MAP_PHYS_NON_CACHED](#) (1 << 28)

6.11 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.

6.11.1 Detailed Description

system error numbers

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.12 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.

6.12.1 Detailed Description

Fcctl declarations.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.13 inttypes.h File Reference

integer type declarations.

Macros

- #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"

6.13.1 Detailed Description

integer type declarations.

Copyright

(C) 2012-2019, 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;
}
```

6.14 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.

6.14.1 Detailed Description

Integrity Report System.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.15 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
- #define `_POSIX_THREAD_THREADS_MAX` 64
- #define `_POSIX_THREADS` 1

6.15.1 Detailed Description

Implementation-defined constants.

Copyright

(C) 2012-2019, 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.

6.16 malloc.h File Reference

Memory allocation definitions.

Macros

- #define `M_CACHE_PAGES` 1

6.16.1 Detailed Description

Memory allocation definitions.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.17 math.h File Reference

Math library.

Macros

- #define `NAN` `__builtin_nan("")`
- #define `INFINITY` `__builtin_inf()`
- #define `isnan(x)` `_isnan(x)`
- #define `isfinite(x)` `_isfinite(x)`
- #define `M_E` 2.7182818284590452354
- #define `M_PI` 3.14159265358979323846
- #define `M_PI_2` 1.57079632679489661923
- #define `M_PI_4` 0.78539816339744830962
- #define `FP_ILOGBNAN` `(-1-(int)((((unsigned)-1)>>1))`
- #define `FP_ILOGB0` `FP_ILOGBNAN`

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_ int ilogb` (double x)
Calculate integer exponent of a floating-point value.
- `_const_ int ilogbf` (float x)
Calculate integer exponent of a floating-point value.
- `_const_ int ilogbl` (long double x)
Calculate integer exponent of a floating-point value.
- `_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_ long double logbl` (long double 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_ long double fmaxl` (long double x, long double 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_ double` [scalbnl](#) (long 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*

6.17.1 Detailed Description

Math library.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.18 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.

6.18.1 Detailed Description

POSIX message queue declarations.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.19 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.

6.19.1 Detailed Description

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.20 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_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_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_TIMEOUT
    = 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_K
    = 0x81030010, TEE_ISOCKET_TLS_ERROR_ECDHE_SHARED_SECRET = 0x81030011, TEE_ISOCKET_TLS_ERROR_ECDSA_SIGNATURE
    = 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_FAILURE
    = 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_CANCELLED
    = 0x8103001A,
    TEE_ISOCKET_TLS_ERROR_CERT_UNKNOWN_CA = 0x8103001B, TEE_ISOCKET_TLS_ERROR_CERT_UNSUPPORTED_VERSION
    = 0x8103001C, TEE_ISOCKET_TLS_ERROR_CERT_REVOKED = 0x8103001D, TEE_ISOCKET_TLS_ERROR_CERT_STATUS_UNKNOWN
    = 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_ERROR
    = 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_CERT_RECEIVED
    = 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_EXPIRED
    = 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_DENIED
    = 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_PROTOCOL_VERSION
    = 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_UNSUPPORTED_VERSION
    = 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_CERTIFICATE_REQUIRED
    = 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_UNKNOWN_CERTIFICATE
    = 0x81031115,
    TEE_ISOCKET_TLS_ALERT_CERT_REQUIRED = 0x81031116 }

```

Proprate protocol specific error codes. According to GPD_SPE_010 specification, TEE error code range 0x80000000..0xFFFFFFFF 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.

6.20.1 Detailed Description

iSocket Protocol error extended codes definitions

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.21 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 [PTHREAD_GUARD_MIN](#) (PAGE_SIZE)
- #define [PTHREAD_GUARD_MAX](#) (PAGE_SIZE << 2)
- #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
- enum { PTHREAD_CREATE_JOINABLE, PTHREAD_CREATE_DETACHED }
detach state attribute settings

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_getstacksize (const pthread_attr_t *attr, size_t *stacksize)
The pthread_attr_getstacksize() function gets size of stack.
- int pthread_attr_getguardsize (const pthread_attr_t *attr, size_t *guardsize)
The pthread_attr_getguardsize() function gets size of guard.
- int pthread_attr_setstacksize (pthread_attr_t *attr, size_t stacksize)
The pthread_attr_setstacksize() function sets size of stack.
- int pthread_attr_setguardsize (pthread_attr_t *attr, size_t guardsize)
The pthread_attr_setguardsize() function sets size of guard.
- int pthread_attr_getstackaddr (const pthread_attr_t *attr, void **stackaddr)
The pthread_attr_getstackaddr() function gets stack address.
- int pthread_attr_setstackaddr (pthread_attr_t *attr, void *stackaddr)
The pthread_attr_setstackaddr() function gets stack address.
- int pthread_attr_getstack (const pthread_attr_t *attr, void **stackaddr, size_t *stacksize)
The pthread_attr_getstack() function gets stack address and size.
- int pthread_attr_setstack (pthread_attr_t *attr, void *stackaddr, size_t stacksize)
The pthread_attr_setstack() function sets stack address and size.
- 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().
- _noreturn_ void pthread_exit (void *retval)
Terminate calling thread.

- `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_condattr_init(pthread_condattr_t *attr)`
Initialize conditional variable attributes with default values.
- `int pthread_condattr_destroy(pthread_condattr_t *attr)`
Destroy conditional variable attributes.
- `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.*
- `int pthread_cond_timedwait (pthread_cond_t *cond, pthread_mutex_t *mutex, const struct timespec *timeout)`
Block on condition variable `cond`, while another thread will unblock this variable by `pthread_cond_broadcast()` / `pthread_cond_signal()` call. If timeout reached before condition set then wait will be interrupted and error is returned.
 - `pthread_t pthread_self (void)`
Obtain ID of the calling thread.
 - `int pthread_kill (pthread_t thread, int sig)`
Send signal to specified thread.
 - `int pthread_equal (pthread_t t1, pthread_t t2)`
Compare thread IDs.
 - `int pthread_detach (pthread_t thread)`
Detach a thread.
 - `int pthread_attr_setdetachstate (pthread_attr_t *attr, int detachstate)`
Set the detach state attribute.
 - `int pthread_attr_getdetachstate (const pthread_attr_t *attr, int *detachstate)`
Get the detach state attribute.
 - `int pthread_getattr_np (pthread_t thread, pthread_attr_t *attr)`
Get attributes of created thread.

6.21.1 Detailed Description

POSIX compatible thread library.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.22 rpmb.h File Reference

TEESL: RPMB API.

Enumerations

- enum { `RPMB_TYPE_BLOCK` = 0, `RPMB_TYPE_BYTE` = 1 }
RPMB I/O type.

Functions

- TEE_Result `TEES_RPMBRead` (uint32_t partition, uint32_t offset, uint8_t *data, uint32_t size, uint8_t type)
Read data from RPMB Storage.
- TEE_Result `TEES_RPMBWrite` (uint32_t partition, uint32_t offset, const uint8_t *data, uint32_t size, uint8_t type)
Write data to RPMB Storage.
- TEE_Result `TEES_RPMBCheckEnable` (void)
Check RPMB availability.

6.22.1 Detailed Description

TEESL: RPMB API.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.23 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.

6.23.1 Detailed Description

Scheduler syscalls.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.24 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](#).
- TEE_Result [TEES_GetContiguousMemoryPhysaddr](#) ([TEES_ContiguousMemoryHandle](#) memory, uint64_t *physaddr)
Retrieves physical address of physically contiguous memory buffer that was previously allocated by [TEES_AllocateContiguousMemory](#).

6.24.1 Detailed Description

Secure Contiguous memory allocator API.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.25 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 **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.

Variables

- FILE * **stdin**
Standard input stream (stub).
- FILE * **stdout**
Standard output stream.
- FILE * **stderr**
Standard error stream.

6.25.1 Detailed Description

Standard I/O.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.26 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.
 - int **atoi** (const char *nptr)

Convert a string to an integer.
 - double **atof** (const char *nptr)

Convert a string to a double.
 - char * **getenv** (const char *name)

get an environment variable function stub
 - int **unsetenv** (const char *name)

delete environment variable function stub
 - int **setenv** (const char *name, const char *value, int overwrite)

change or add environment variable function stub

6.26.1 Detailed Description

Standard library header.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.27 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](#) (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.
- void * **memrchr** (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.

6.27.1 Detailed Description

String manipulation functions.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.28 sys/credentials.h File Reference

Definitions for credentials.

Functions

- int [cred_compare](#) (const struct cred *as_is, const struct cred *to_be)
Compare credentials.

6.28.1 Detailed Description

Definitions for credentials.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.28.2 Function Documentation

6.28.2.1 int cred_compare (const struct cred * as_is, const struct cred * to_be)

Compare credentials.

Parameters

in	<i>as_is</i>	struct of credentials which user has got
in	<i>to_be</i>	struct of credentials which need to have

Returns

0 on success
-1 on error

6.29 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.

6.29.1 Detailed Description

ioctl declaration.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.30 sys/socket.h File Reference

Support socket routines.

Typedefs

- typedef int [socklen_t](#)
A type of width of at least 32 bits representing lengths in socket subsystem.

Functions

- int [accept](#) (int sockfd, struct sockaddr *addr, [socklen_t](#) *addrlen)
Accept a connection on a socket.
- int [bind](#) (int sockfd, const struct sockaddr *addr, [socklen_t](#) addrlen)
Bind a name to a socket.
- int [connect](#) (int sockfd, const struct sockaddr *addr, [socklen_t](#) addrlen)
Initiate a connection on a socket.
- int [getsockopt](#) (int sockfd, int level, int optname, void *optval, [socklen_t](#) *optlen)
Get options on socket.
- int [setsockopt](#) (int sockfd, int level, int optname, const void *optval, [socklen_t](#) optlen)
Set 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.

6.30.1 Detailed Description

Support socket routines.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.31 sys/types.h File Reference

Wrapper for kernel [types.h](#) header.

Typedefs

- typedef intptr_t [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](#)

6.31.1 Detailed Description

Wrapper for kernel [types.h](#) header.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.32 sys/un.h File Reference

Definitions for UNIX domain sockets.

6.32.1 Detailed Description

Definitions for UNIX domain sockets.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.33 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.

6.33.1 Detailed Description

Manipulation with errno and TEE error.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.34 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.

6.34.1 Detailed Description

TEE I2C public API.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.35 tee_interrupt.h File Reference

Interrupt syscalls.

Data Structures

- struct [intrinfo](#)

Typedefs

- typedef struct __TEES_InterruptHandle * [TEES_InterruptHandle](#)
- typedef void(* [intruhandler](#)) (struct [intrinfo](#) *)

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.

6.35.1 Detailed Description

Interrupt syscalls.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.36 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.

6.36.1 Detailed Description

GP iSockets interface (GPD_SPE_100)

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.37 tee_nfc.h File Reference

TEE NFC public API.

Typedefs

- typedef struct [TEES_NFCHandler](#) * [TEES_NFCHandler](#)

Functions

- TEE_Result [TEES_NFCInit](#) ([TEES_NFCHandler](#) *handler)
Initialize NFC device and set transfer parameters to handler.
- void [TEES_NFCExit](#) ([TEES_NFCHandler](#) handler)
Terminate connection to NFC device.
- TEE_Result [TEES_NFCWrite](#) ([TEES_NFCHandler](#) handler, void *tx_buf, size_t len)
Write data buffer tx to initialized NFC device.
- TEE_Result [TEES_NFCRead](#) ([TEES_NFCHandler](#) handler, char *rx_buf, size_t rx_buf_len, size_t *rsp_len)
Read data buffer rx from initialized NFC device.
- TEE_Result [TEES_NFCReadWait](#) ([TEES_NFCHandler](#) handler, char *rx_buf, size_t rx_buf_len, size_t *rsp_len, int ms_timeout)
Read data buffer rx from initialized NFC device.
- TEE_Result [TEES_NFCSetMode](#) ([TEES_NFCHandler](#) handler, unsigned long mode)
Set NFC device mode.
- TEE_Result [TEES_NFCWakeUp](#) ([TEES_NFCHandler](#) handler)
Send wake up command to initialized NFC device.
- TEE_Result [TEES_NFCSleep](#) ([TEES_NFCHandler](#) handler)
Send sleep command to initialized NFC device.

6.37.1 Detailed Description

TEE NFC public API.

Copyright

(C) 2019, Samsung Electronics Co., Ltd.

6.38 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.

6.38.1 Detailed Description

TEE SMC public API.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.39 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.*

6.39.1 Detailed Description

TEE SPI public API.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.40 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.*

6.40.1 Detailed Description

Driver destructor registration header.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.41 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.

6.41.1 Detailed Description

GP iSockets TCP API header (GPD_SPE_101)

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.42 tee_tlsocket.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 }
TLS 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_HANDSHAKE_FAILURE](#) = 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 Cipher Suites.

```

= 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 }

```
= 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_CLIE`
= 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.

6.42.1 Detailed Description

GP iSockets TLS API (GPD_SPE_103)

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.43 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.

6.43.1 Detailed Description

GP iSockets UDP API header (GPD_SPE_102)

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.44 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.

6.44.1 Detailed Description

Critical sections support.

Copyright

(C) 2018, Samsung Electronics Co., Ltd.

6.45 tees_el2if.h File Reference

EL2 interface support.

Data Structures

- struct [TEES_EL2if_Args](#)
Arguments for EL2 SMC call. [More...](#)

Macros

- #define [EL2IF_DEV_NAME](#) "/dev/el2if"
Driver name for EL2IF.
- #define [EL2IF_ARGS_NUM](#) 3
Number of EL2IF arguments.

Functions

- int [TEES_EL2if](#) (struct [TEES_EL2if_Args](#) *args)
Send SMC call to EL2.

6.45.1 Detailed Description

EL2 interface support.

Copyright

(C) 2019, Samsung Electronics Co., Ltd.

6.46 tees_extension.h File Reference

Samsung's extension of TEE internal API.

Data Structures

- struct [TEES_ClientCredentials](#)

Client credentials structure. Used by [TEES_GetClientCredentials\(\)](#). [More...](#)

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 is buffer shared with REE.
- TEE_Result [TEES_GetClientCredentials](#) (struct [TEES_ClientCredentials](#) *credentials)
Get client credentials (pid, gid, uid)

6.46.1 Detailed Description

Samsung's extension of TEE internal API.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.47 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.

6.47.1 Detailed Description

Lock or Unlock HW crypto buffer.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.48 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.

6.48.1 Detailed Description

Samsung IWT API header.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.49 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.

6.49.1 Detailed Description

KDF API.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.50 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.

6.50.1 Detailed Description

ROT API.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.51 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_STATIC` ((`SO_TAG_LEN`) + (`SO_IV_LEN`) + (`SO_AC_LEN`) + (`SO_MAGIC_NUMBER_LEN`))
- #define `SO_OUT_BUF_SIZE`(in_len, delegated) ((in_len) + `SO_HEADER_SIZE_STATIC` + ((delegated) ? (`SO_TA_ID_LEN` + `SO_AUTH_ID_LEN`) : 0))

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.

6.51.1 Detailed Description

Secure Objects API.

Copyright

(C) 2012-2019, 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										<-Encrypted	
<----- Access Control (4 byte) ----->										Data--->	
MAGIC	TAG	IV	reserved	AUTH ID	TA UUID	AC bit	AC bit	TA UUID	Auth ID	encrypted	
4b	16b	16b	28bit	1bit	1bit	1bit	1bit	16b	16b	image	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) set => `auth_id` and `ta_id` and Current TA Id and Current TA Authority are used as input to KDF. `auth_id` and `ta_id` are considered as delegated.
- if bit-3 (DELEGATED_AUTH_ID_AC) set => `auth_id` and Current TA Id and Current TA Authority are used as input to KDF. `auth_id` is considered as delegated.

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.

6.52 tees_ssapi.h File Reference

Samsung SMC APIs.

Data Structures

- struct [secHDCPKeyInfo_t](#)
Key information for HDCP. [More...](#)

Macros

- #define [SECCAM_SECURE](#) 0x0000
Successful return of SMC for `SECCAM_GetStatus`.
- #define [SECCAM_NORMAL](#) 0x9101
Unsuccessful return of SMC for `SECCAM_GetStatus`.

Enumerations

- enum { **TUI_SET_INFO** = 1, **TUI_CLEAR_INFO** }
TUI_SETINFO commands.

Functions

- TEE_Result [TEES_SECCAM_GetStatus](#) (unsigned int *data)
Get a status of secure camera.
- TEE_Result [TEES_SECCAM_Protect](#) (uint64_t paddr, size_t size, unsigned int *data)
Protect the memory used by secure camera.
- TEE_Result [TEES_SECCAM_Unprotect](#) (uint64_t paddr, size_t size, unsigned int *data)
Unprotect the memory used by secure camera.
- TEE_Result [TEES_SECCAM_IsProtected](#) (uint64_t paddr, size_t size, unsigned int *data)
Check the memory used by secure camera whether it is protected or not.
- TEE_Result [TEES_HDCP_SetKeyInfo](#) (struct [sechDCPKeyInfo_t](#) *SecureHDCPKey_info, unsigned int *data)
Set HDCP key information.
- TEE_Result [TEES_TUI_SetInfo](#) (uint32_t cmd, uint64_t paddr, size_t size, unsigned int *data)
Set or clear TUI information.
- TEE_Result [TEES_TUI_Protect](#) (uint64_t paddr, size_t size, unsigned int *data)
Protect the memory used by TUI.
- TEE_Result [TEES_TUI_Unprotect](#) (uint64_t paddr, size_t size, unsigned int *data)
Unprotect the memory used by TUI.

6.52.1 Detailed Description

Samsung SMC APIs.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.53 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_TUIMTInfo](#)
Structure that represents information of touch event (touch position, touch type). [More...](#)
- struct [TEES_TUI_MT_Info](#)
Structure that represents all the information of touch events. [More...](#)
- struct [TEES_TUIImage](#)
Structure that represents properties of image and buffers. [More...](#)

Macros

- #define **MT_INFO_MAX_EVENT** 10
- #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_CANCEL** = 0xFFFF0002, **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_ERROR_SHORT_BUFFER = 0xFFFF0010, **TEES_ERROR_TUI_TIMEOUT** = 0xFFFF3001 }
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_TUIGetMTEvent** (**TEES_TUI_MT_Info** *touchInfo, uint32_t timeout)
Get Multi touch event events.
- **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.
- **TEES_Result** **TEES_TUIDrawImageToBuff** (**TEES_TUIImage** *image, void *buff, uint32_t *buf_size)
Unpack png picture to bitmap in buffer.
- **TEES_Result** **TEES_TUIDrawImageFromBuff** (uint32_t posX, uint32_t posY, uint32_t W, uint32_t H, void *buff, uint32_t buf_size)
Draw unpacked bitmap from buffer on display.

6.53.1 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) 2012-2019, Samsung Electronics Co., Ltd.

6.54 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.

6.54.1 Detailed Description

Wrapped with REK API.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.55 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](#) (const 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](#).
- 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 timespec format.

6.55.1 Detailed Description

Time header.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.56 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_TEMPORARY_CLEAR_MODE](#) }
- 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.

6.56.1 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) 2012-2019, Samsung Electronics Co., Ltd.

6.57 udf.h File Reference

User-space Driver Framework declarations.

Functions

- int [TEES_UDF_InitDriver](#) (char *name, struct [fops](#) *fops, unsigned int drvid, struct [usr_drv_info](#) **info)
This function is used to initialize and register UDF driver.
- int [TEES_UDF_Init_FS_Driver](#) (char *name, struct [fops](#) *fops, unsigned int drvid, struct [usr_drv_info](#) **info)
This function is used to initialize and register UDF FS driver.
- int [TEES_UDF_FiniDriver](#) (struct [usr_drv_info](#) *info)
This function is used to de-initialize and stop UDF driver.
- int [TEES_UDF_RegisterIoctlDesc](#) (struct [usr_drv_info](#) *info, unsigned int cmd, const struct [ioctl_desc](#) *desc)
Register an [ioctl\(\)](#) cmd for UDF driver.

6.57.1 Detailed Description

User-space Driver Framework declarations.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.58 unistd.h File Reference

Unistd header.

Macros

- #define `TEMP_FAILURE_RETRY`(expression)
Recall function if it was interrupted by signal.

Enumerations

- enum {
 `_SC_OPEN_MAX`, `_SC_PAGESIZE`, `_SC_PHYS_PAGES`, `_SC_AVPHYS_PAGES`,
 `_SC_NPROCESSORS_CONF`, `_SC_NPROCESSORS_ONLN`, `_SC_THREADS`, `_SC_THREAD_KEYS_MAX`,
 `_SC_THREAD_THREADS_MAX`, `_SC_THREAD_STACK_MIN` }
sysconf system resources' names.

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.
- 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 `rename` (const char *pathname, const char *new_pathname)
Rename file.
- 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 [rmdir](#) (char *dir_name)
Remove a directory at file system.
- 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 [fsync](#) (int fd)
Synchronize a file's in-core state with storage device.
- int [lseek](#) (int fd, int offset, int whence)
Reposition read/write file offset.
- long [sysconf](#) (int name)
Get configuration information at run time.

6.58.1 Detailed Description

Unistd header.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6.59 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.

6.59.1 Detailed Description

UUID management helpers.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

6 Index

- _POSIX_THREADS
 - Auxiliary API, [202](#)
- _POSIX_THREAD_KEYS_MAX
 - Auxiliary API, [202](#)
- _POSIX_THREAD_THREADS_MAX
 - Auxiliary API, [202](#)
- _SC_AVPHYS_PAGES
 - Auxiliary API, [234](#)
- _SC_NPROCESSORS_CONF
 - Auxiliary API, [234](#)
- _SC_NPROCESSORS_ONLN
 - Auxiliary API, [234](#)
- _SC_OPEN_MAX
 - Auxiliary API, [234](#)
- _SC_PAGESIZE
 - Auxiliary API, [234](#)
- _SC_PHYS_PAGES
 - Auxiliary API, [234](#)
- _SC_THREADS
 - Auxiliary API, [234](#)
- _SC_THREAD_KEYS_MAX
 - Auxiliary API, [234](#)
- _SC_THREAD_STACK_MIN
 - Auxiliary API, [234](#)
- _SC_THREAD_THREADS_MAX
 - Auxiliary API, [234](#)
- __CPUMASK
 - Auxiliary API, [202](#)
- __TEE_SC_CardKeyRef, [331](#)
 - scKeyID, [331](#)
 - scKeyVersion, [331](#)
- __TEE_SC_DeviceKeyRef, [331](#)
 - scKeyType, [332](#)
- __TEE_SC_DeviceKeyRef.__unnamed__, [332](#)
 - scBaseKeyHandle, [332](#)
 - scKeySetRef, [332](#)
- __TEE_SC_KeySetRef, [332](#)
 - scKeyEncHandle, [333](#)
 - scKeyMacHandle, [333](#)
- __TEE_SC_OID, [333](#)
 - buffer, [333](#)
 - bufferLen, [333](#)
- __TEE_SC_Params, [333](#)
 - scCardKeyRef, [334](#)
 - scDeviceKeyRef, [334](#)
 - scOID, [334](#)
 - scSecurityLevel, [334](#)
 - scType, [334](#)
- __TEE_SEAID, [334](#)
 - buffer, [335](#)
 - bufferLen, [335](#)
- __TEE_SEReaderProperties, [335](#)
 - sePresent, [335](#)
 - selectResponseEnable, [335](#)
 - teeOnly, [335](#)
- __pthread_attr_t, [133](#)
- __pthread_cond_t, [133](#)
- __pthread_condattr_t, [133](#)
- __pthread_mutex_t, [133](#)
- __pthread_once_t, [133](#)
- __uuid_t, [201](#)
 - Auxiliary API, [232](#)
- _exit
 - Auxiliary API, [234](#)
- AUTO_BUFFER_SIZE
 - Auxiliary API, [203](#)
- abort_handler_s
 - Auxiliary API, [234](#)
- abs
 - Auxiliary API, [235](#)
- accept
 - Socket library API, [184](#)
- alloca
 - Auxiliary API, [202](#)
- alloca.h, [342](#)
- arg_types
 - smc_data, [339](#)
- args
 - smc_data, [339](#)
- arm_timer_get_frequency
 - Auxiliary API, [235](#)
- asprintf
 - Auxiliary API, [235](#)
- assert
 - Auxiliary API, [202](#)
- assert.h, [342](#)
- atan
 - Math library API, [166](#)
- atan2
 - Math library API, [166](#)
- atan2f
 - Math library API, [166](#)
- atanf
 - Math library API, [166](#)
- atexit
 - Auxiliary API, [236](#)
- atof
 - Auxiliary API, [236](#)
- atoi
 - Auxiliary API, [237](#)
- Auxiliary API, [190](#)
 - _POSIX_THREADS, [202](#)
 - _POSIX_THREAD_KEYS_MAX, [202](#)
 - _POSIX_THREAD_THREADS_MAX, [202](#)
 - _SC_AVPHYS_PAGES, [234](#)
 - _SC_NPROCESSORS_CONF, [234](#)
 - _SC_NPROCESSORS_ONLN, [234](#)
 - _SC_OPEN_MAX, [234](#)
 - _SC_PAGESIZE, [234](#)

[_SC_PHYS_PAGES, 234](#)
[_SC_THREADS, 234](#)
[_SC_THREAD_KEYS_MAX, 234](#)
[_SC_THREAD_STACK_MIN, 234](#)
[_SC_THREAD_THREADS_MAX, 234](#)
[__CPUMASK, 202](#)
[__uuid_t, 232](#)
[_exit, 234](#)
[AUTO_BUFFER_SIZE, 203](#)
[abort_handler_s, 234](#)
[abs, 235](#)
[alloca, 202](#)
[arm_timer_get_frequency, 235](#)
[asprintf, 235](#)
[assert, 202](#)
[atexit, 236](#)
[atof, 236](#)
[atoi, 237](#)
[BIT_PER_CPU, 203](#)
[BITMAP_ELT, 203](#)
[BITS_TO_CPU_MASK, 203](#)
[CHAR_BIT, 203](#)
[CPU_CLR, 203](#)
[CPU_ISSET, 204](#)
[CPU_SET, 204](#)
[CPU_ZERO, 204](#)
[calloc, 237](#)
[clock_gettime, 238](#)
[close, 238](#)
[constraint_handler_t, 232](#)
[DECLARE_BITMAP, 205](#)
[E2BIG, 205](#)
[EACCES, 205](#)
[EADDRINUSE, 205](#)
[EADDRNOTAVAIL, 205](#)
[EADV, 205](#)
[EAFNOSUPPORT, 206](#)
[EAGAIN, 206](#)
[EALREADY, 206](#)
[EBADFD, 206](#)
[EBADMSG, 206](#)
[EBADRQC, 207](#)
[EBADSLT, 207](#)
[EBADE, 206](#)
[EBADF, 206](#)
[EBADR, 207](#)
[EBFONT, 207](#)
[EBUSY, 207](#)
[ECANCELED, 207](#)
[ECHILD, 207](#)
[ECHRNG, 208](#)
[ECOMM, 208](#)
[ECONNABORTED, 208](#)
[ECONNREFUSED, 208](#)
[ECONNRESET, 208](#)
[EDEADLOCK, 208](#)
[EDEADLK, 208](#)
[EDESTADDRREQ, 209](#)
[EDOTDOT, 209](#)
[EDOM, 209](#)
[EDQUOT, 209](#)
[EEXIST, 209](#)
[EFAULT, 209](#)
[EFBIG, 209](#)
[EHOSTDOWN, 210](#)
[EHOSTUNREACH, 210](#)
[EIDRM, 210](#)
[EILSEQ, 210](#)
[EINPROGRESS, 210](#)
[EINTR, 210](#)
[EINVAL, 210](#)
[EISCONN, 211](#)
[EISDIR, 211](#)
[EISNAM, 211](#)
[EIO, 211](#)
[EKEYREJECTED, 211](#)
[EL2HLT, 211](#)
[EL2NSYNC, 211](#)
[EL3HLT, 212](#)
[EL3RST, 212](#)
[ELIBACC, 212](#)
[ELIBBAD, 212](#)
[ELIBEXEC, 212](#)
[ELIBMAX, 212](#)
[ELIBSCN, 212](#)
[ELNRNG, 213](#)
[ELOOP, 213](#)
[EMFILE, 213](#)
[EMLINK, 213](#)
[EMSGSIZE, 213](#)
[EMULTIHOP, 213](#)
[ENAMETOOLONG, 213](#)
[ENAVAIL, 214](#)
[ENETDOWN, 214](#)
[ENETRESET, 214](#)
[ENETUNREACH, 214](#)
[ENFILE, 214](#)
[ENOANO, 214](#)
[ENOBUFS, 214](#)
[ENOCSI, 215](#)
[ENODATA, 215](#)
[ENODEV, 215](#)
[ENOENT, 215](#)
[ENOEXEC, 215](#)
[ENOKEY, 215](#)
[ENOLCK, 215](#)
[ENOLINK, 216](#)
[ENOMEM, 216](#)
[ENOMSG, 216](#)
[ENONET, 216](#)
[ENOPKG, 216](#)
[ENOPROTOPT, 216](#)
[ENOSPC, 216](#)
[ENOSTR, 217](#)
[ENOSYS, 217](#)
[ENOSR, 217](#)

ENOTBLK, 217
ENOTCONN, 217
ENOTDIR, 217
ENOTEMPTY, 217
ENOTNAM, 218
ENOTRECOVERABLE, 218
ENOTSOCK, 218
ENOTTY, 218
ENOTUNIQ, 218
ENXIO, 218
EOPNOTSUPP, 219
EOVERFLOW, 219
EOF, 218
EPERM, 219
EPFNOSUPPORT, 219
EPIPE, 219
EPROTONOSUPPORT, 219
EPROTOTYPE, 220
EPROTO, 219
ERANGE, 220
EREMCHG, 220
EREMOTEIO, 220
EREMOTE, 220
ERESTART, 220
EROFS, 220
ESHUTDOWN, 221
ESOCKTNOSUPPORT, 221
ESPIPE, 221
ESRCH, 221
ESRMNT, 221
ESTALE, 221
ESTRPIPE, 222
ETIMEDOUT, 222
ETIME, 222
ETOOMANYREFS, 222
ETXTBSY, 222
EUCLEAN, 222
EUNATCH, 222
EUSERS, 223
EWOULDBLOCK, 223
EXDEV, 223
EXFULL, 223
errno, 221
errno_t, 232
exit, 239
fflush, 239
fprintf, 239
free, 240
fstat, 240
fsync, 241
ftruncate, 241
get_errno_addr, 241
getcluster, 242
getcpu, 242
getenv, 242
getpid, 242
gettid, 243
gid_t, 232
INT_MAX, 223
INT_MIN, 223
ignore_handler_s, 243
invoke_constraint_handler_s, 243
ioctl, 243
isalnum, 244
isalpha, 244
isascii, 245
isblank, 245
iscntrl, 245
isdigit, 246
isgraph, 246
islower, 246
isprint, 247
ispunct, 247
isspace, 247
isupper, 248
isxdigit, 248
LLONG_MAX, 223
LLONG_MIN, 224
LONG_MAX, 224
LONG_MIN, 224
lseek, 248
M_CACHE_PAGES, 224
MAP_ANONYMOUS, 224
MAP_FAILED, 224
MAP_FIXED, 224
MAP_GROWSDOWN, 225
MAP_PHYS_NON_CACHED, 225
MAP_PHYS_NON_SECURE, 225
MAP_POPULATE, 225
MAP_PRIVATE, 225
MAP_SHARED, 225
MAP_STACK, 225
MAX_CPUS, 226
malloc, 249
memchr, 249
memcmp, 250
memcpy, 250
memcpy_s, 251
memmove, 251
memmove_s, 251
memrchr, 252
memset, 252
memset_s, 252
mmap, 253
mode_t, 233
mprotect, 255
ms_to_timespec, 255
munmap, 255
NUM_MILLIS_IN_SEC, 226
NUM_NANOS_IN_MILLI, 226
NUM_NANOS_IN_SEC, 226
NUM_NANOS_IN_USEC, 226
NUM_SECONDS_IN_MIN, 226
nanosleep, 256
off_t, 233
open, 257

PGOFF_SHIFT, 226
PRId16, 227
PRId32, 227
PRId64, 227
PRIu16, 227
PRIu32, 227
PRIu64, 227
PRIx16, 227
PRIx32, 228
PRIx64, 228
PROT_EXEC, 228
PROT_NONE, 228
PROT_READ, 228
PROT_WRITE, 228
PTHREAD_KEYS_MAX, 228
pid_t, 233
printf, 257
printf_no_alloc, 258
printf_s, 258
profil, 259
putchar, 259
puts, 260
qsort, 260
qsort_r, 260
read, 261
realloc, 261
rename, 262
rmdir, 262
SCHAR_MAX, 229
SCHAR_MIN, 229
SCNd16, 229
SCNd32, 229
SCNd64, 229
SCNu16, 229
SCNu32, 229
SCNu64, 230
SCNx16, 230
SCNx32, 230
SCNx64, 230
SHRT_MAX, 230
SHRT_MIN, 230
sched_getaffinity, 263
sched_setaffinity, 263
sched_yield, 264
set_constraint_handler_s, 264
setenv, 264
snprintf, 265
snprintf_s, 266
sprintf, 266
sprintf_s, 267
sscanf, 267
sscanf_s, 268
ssize_t, 233
stat, 268
stdin, 297
strcat, 268
strcat_s, 269
strchr, 269
strchrnul, 270
strcmp, 270
strcpy, 271
strcpy_s, 272
strcspn, 272
strdup, 272
strerror, 273
strerror_r, 273
strerror_s, 273
strlcat, 274
strlen, 274
strncat, 275
strncat_s, 275
strncmp, 276
strncpy, 277
strncpy_s, 277
strnlen, 278
strnlen_s, 278
strrchr, 279
strspn, 279
strstr, 280
strtod, 280
strtof, 281
strtok, 282
strtok_r, 282
strtol, 283
strtold, 283
strtoll, 284
strtoul, 285
strtoull, 285
sysconf, 286
TEMP_FAILURE_RETRY, 230
time, 286
time_t, 233
timeadd, 287
timespec_to_ms, 287
timespec_to_nsec, 287
timesub, 288
tolower, 288
toupper, 288
UCHAR_MAX, 231
UINT_MAX, 231
ULLONG_MAX, 231
ULONG_MAX, 231
USHRT_MAX, 231
UUID_STRING_LEN, 232
uid_t, 233
unlink, 289
unsetenv, 289
uuid_clear, 289
uuid_compare, 290
uuid_generate, 290
uuid_generate_time, 232
uuid_is_null, 290
uuid_parse, 291
uuid_t, 233
uuid_unparse, 291
uuid_unparse_lower, 232

- uuid_unparse_upper, 291
- vasprintf, 292
- vasprintf_s, 292
- vfprintf, 293
- vsnprintf, 293
- vsnprintf_s, 294
- vsprintf, 294
- vsprintf_s, 295
- vsscanf, 296
- vsscanf_s, 296
- write, 297
- BIT_PER_CPU
 - Auxiliary API, 203
- BITMAP_ELT
 - Auxiliary API, 203
- BITS_TO_CPU_MASK
 - Auxiliary API, 203
- bind
 - Socket library API, 185
- buffer
 - __TEE_SC_OID, 333
 - __TEE_SEAID, 335
- bufferLen
 - __TEE_SC_OID, 333
 - __TEE_SEAID, 335
- CHAR_BIT
 - Auxiliary API, 203
- CPU_CLR
 - Auxiliary API, 203
- CPU_ISSET
 - Auxiliary API, 204
- CPU_SET
 - Auxiliary API, 204
- CPU_ZERO
 - Auxiliary API, 204
- cache.h, 342
- calloc
 - Auxiliary API, 237
- ceil
 - Math library API, 167
- ceilf
 - Math library API, 167
- clock_gettime
 - Auxiliary API, 238
- close
 - Auxiliary API, 238
 - fops, 337
 - TEE_iSocket_s, 315
- cnt
 - ioctl_desc, 339
- connect
 - Socket library API, 185
- constraint_handler_t
 - Auxiliary API, 232
- Contiguous memory API, 78
 - TEES_AllocateContiguousMemory, 79
- TEES_CONTIGUOUS_FLAG_ZERO_ON_ALLOC, 79
- TEES_CONTIGUOUS_FLAG_ZERO_ON_FREE, 79
- TEES_CONTIGUOUS_MAP_FIXED, 79
- TEES_CONTIGUOUS_MAP_NON_CACHEABLE, 79
- TEES_CONTIGUOUS_MAP_POPULATE, 79
- TEES_CONTIGUOUS_MAP_READ, 79
- TEES_CONTIGUOUS_MAP_WRITE, 79
- TEES_ContiguousAllocateFlags, 79
- TEES_ContiguousMapFlags, 79
- TEES_ContiguousMemoryHandle, 78
- TEES_GetContiguousMemoryPhysaddr, 80
- TEES_MapContiguousMemory, 81
- TEES_ReleaseContiguousMemory, 82
- TEES_UnmapContiguousMemory, 82
- copysign
 - Math library API, 167
- copysignf
 - Math library API, 168
- core/error.h, 343
- core/mman.h, 345
- cos
 - Math library API, 168
- cosf
 - Math library API, 168
- cpu_set_t, 201
- cred_compare
 - credentials.h, 369
- credentials.h
 - cred_compare, 369
- ctype.h, 347
- Custom handler API, 77
- DECLARE_BITMAP
 - Auxiliary API, 205
- desc_atom, 335
 - len, 336
 - type, 336
- driver.h, 348
- driver/mem/phys.h, 349
- E2BIG
 - Auxiliary API, 205
- EACCES
 - Auxiliary API, 205
- EADDRINUSE
 - Auxiliary API, 205
- EADDRNOTAVAIL
 - Auxiliary API, 205
- EADV
 - Auxiliary API, 205
- EAFNOSUPPORT
 - Auxiliary API, 206
- EAGAIN
 - Auxiliary API, 206
- EALREADY
 - Auxiliary API, 206

EBADFD	Auxiliary API, 206	EINPROGRESS	Auxiliary API, 210
EBADMSG	Auxiliary API, 206	EINTR	Auxiliary API, 210
EBADRQC	Auxiliary API, 207	EINVAL	Auxiliary API, 210
EBADSLT	Auxiliary API, 207	EISCONN	Auxiliary API, 211
EBADE	Auxiliary API, 206	EISDIR	Auxiliary API, 211
EBADF	Auxiliary API, 206	EISNAM	Auxiliary API, 211
EBADR	Auxiliary API, 207	EIO	Auxiliary API, 211
EBFONT	Auxiliary API, 207	EKEYREJECTED	Auxiliary API, 211
EBUSY	Auxiliary API, 207	EL2HLT	Auxiliary API, 211
ECANCELED	Auxiliary API, 207	EL2NSYNC	Auxiliary API, 211
ECHILD	Auxiliary API, 207	EL3HLT	Auxiliary API, 212
ECHRNG	Auxiliary API, 208	EL3RST	Auxiliary API, 212
ECOMM	Auxiliary API, 208	ELIBACC	Auxiliary API, 212
ECONNABORTED	Auxiliary API, 208	ELIBBAD	Auxiliary API, 212
ECONNREFUSED	Auxiliary API, 208	ELIBEXEC	Auxiliary API, 212
ECONNRESET	Auxiliary API, 208	ELIBMAX	Auxiliary API, 212
EDEADLOCK	Auxiliary API, 208	ELIBSCN	Auxiliary API, 212
EDEADLK	Auxiliary API, 208	ELNRNG	Auxiliary API, 213
EDESTADDRREQ	Auxiliary API, 209	ELOOP	Auxiliary API, 213
EDOTDOT	Auxiliary API, 209	EMFILE	Auxiliary API, 213
EDOM	Auxiliary API, 209	EMLINK	Auxiliary API, 213
EDQUOT	Auxiliary API, 209	EMSGSIZE	Auxiliary API, 213
EEXIST	Auxiliary API, 209	EMULTIHOP	Auxiliary API, 213
EFAULT	Auxiliary API, 209	ENAMETOOLONG	Auxiliary API, 213
EFBIG	Auxiliary API, 209	ENAVAIL	Auxiliary API, 214
EHOSTDOWN	Auxiliary API, 210	ENETDOWN	Auxiliary API, 214
EHOSTUNREACH	Auxiliary API, 210	ENETRESET	Auxiliary API, 214
EIDRM	Auxiliary API, 210	ENETUNREACH	Auxiliary API, 214
EILSEQ	Auxiliary API, 210	ENFILE	Auxiliary API, 214

ENOANO	Auxiliary API, 214	EOPNOTSUPP	Auxiliary API, 219
ENOBUFFS	Auxiliary API, 214	EOVERFLOW	Auxiliary API, 219
ENOCSS	Auxiliary API, 215	EOF	Auxiliary API, 218
ENODATA	Auxiliary API, 215	EPERM	Auxiliary API, 219
ENODEV	Auxiliary API, 215	EPFNOSUPPORT	Auxiliary API, 219
ENOENT	Auxiliary API, 215	EPIPE	Auxiliary API, 219
ENOEXEC	Auxiliary API, 215	EPROTONOSUPPORT	Auxiliary API, 219
ENOKEY	Auxiliary API, 215	EPROTOTYPE	Auxiliary API, 220
ENOLCK	Auxiliary API, 215	EPROTO	Auxiliary API, 219
ENOLINK	Auxiliary API, 216	ERANGE	Auxiliary API, 220
ENOMEM	Auxiliary API, 216	EREMCHG	Auxiliary API, 220
ENOMSG	Auxiliary API, 216	EREMOTEIO	Auxiliary API, 220
ENONET	Auxiliary API, 216	EREMOTE	Auxiliary API, 220
ENOPKG	Auxiliary API, 216	ERESTART	Auxiliary API, 220
ENOPROTOPT	Auxiliary API, 216	EROFS	Auxiliary API, 220
ENOSPC	Auxiliary API, 216	ESHUTDOWN	Auxiliary API, 221
ENOSTR	Auxiliary API, 217	ESOCKTNOSUPPORT	Auxiliary API, 221
ENOSYS	Auxiliary API, 217	ESPIPE	Auxiliary API, 221
ENOSR	Auxiliary API, 217	ESRCH	Auxiliary API, 221
ENOTBLK	Auxiliary API, 217	ESRMNT	Auxiliary API, 221
ENOTCONN	Auxiliary API, 217	ESTALE	Auxiliary API, 221
ENOTDIR	Auxiliary API, 217	ESTRPIPE	Auxiliary API, 222
ENOTEMPTY	Auxiliary API, 217	ETIMEDOUT	Auxiliary API, 222
ENOTNAM	Auxiliary API, 218	ETIME	Auxiliary API, 222
ENOTRECOVERABLE	Auxiliary API, 218	ETOOMANYREFS	Auxiliary API, 222
ENOTSOCK	Auxiliary API, 218	ETXTBSY	Auxiliary API, 222
ENOTTY	Auxiliary API, 218	EUCLEAN	Auxiliary API, 222
ENOTUNIQ	Auxiliary API, 218	EUNATCH	Auxiliary API, 222
ENXIO	Auxiliary API, 218	EUSERS	Auxiliary API, 223

- EWOLDBLOCK
 - Auxiliary API, [223](#)
- EXDEV
 - Auxiliary API, [223](#)
- EXFULL
 - Auxiliary API, [223](#)
- errno
 - Auxiliary API, [221](#)
- errno.h, [349](#)
- errno_t
 - Auxiliary API, [232](#)
- errno_to_tee_error
 - Samsung Internal API, [50](#)
- error
 - TEE_iSocket_s, [315](#)
- exit
 - Auxiliary API, [239](#)
- exp
 - Math library API, [169](#)
- expf
 - Math library API, [169](#)
- FP_ILOGB0
 - Math library API, [164](#)
- FP_ILOGBNAN
 - Math library API, [164](#)
- fabs
 - Math library API, [169](#)
- fabsf
 - Math library API, [170](#)
- fcntl.h, [349](#)
- fflush
 - Auxiliary API, [239](#)
- floor
 - Math library API, [170](#)
- floorf
 - Math library API, [170](#)
- fmax
 - Math library API, [171](#)
- fmaxf
 - Math library API, [171](#)
- fmaxl
 - Math library API, [171](#)
- fmin
 - Math library API, [172](#)
- fminf
 - Math library API, [172](#)
- fops, [336](#)
 - close, [337](#)
 - fsync, [337](#)
 - ioctl, [337](#)
 - lookup, [337](#)
 - lseek, [337](#)
 - mkdir, [337](#)
 - mmap, [337](#)
 - open, [337](#)
 - probe, [337](#)
 - read, [338](#)
 - readdir, [338](#)
 - rename, [338](#)
 - rmdir, [338](#)
 - stat, [338](#)
 - truncate, [338](#)
 - unlink, [338](#)
 - usr_drv_info, [341](#)
 - write, [338](#)
- fprintf
 - Auxiliary API, [239](#)
- free
 - Auxiliary API, [240](#)
- fstat
 - Auxiliary API, [240](#)
 - Stat, [298](#)
- fsync
 - Auxiliary API, [241](#)
 - fops, [337](#)
- ftruncate
 - Auxiliary API, [241](#)
- get_errno_addr
 - Auxiliary API, [241](#)
- getcluster
 - Auxiliary API, [242](#)
- getcpu
 - Auxiliary API, [242](#)
- getenv
 - Auxiliary API, [242](#)
- getpid
 - Auxiliary API, [242](#)
- getsockopt
 - Socket library API, [185](#)
- gettid
 - Auxiliary API, [243](#)
- gid_t
 - Auxiliary API, [232](#)
- handle
 - usr_drv_info, [341](#)
- hypot
 - Math library API, [172](#)
- hypotf
 - Math library API, [173](#)
- I2C API, [93](#)
 - TEES_I2CExit, [94](#)
 - TEES_I2CHandler, [94](#)
 - TEES_I2CInit, [94](#)
 - TEES_I2CRead, [94](#)
 - TEES_I2CWrite, [95](#)
 - TEES_I2CWriteRead, [95](#)
- INFINITY
 - Math library API, [164](#)
- INT_MAX
 - Auxiliary API, [223](#)
- INT_MIN
 - Auxiliary API, [223](#)
- IRS_ADD_FLAG_CMD
 - Integrity Report System API, [121](#)

- IRS_DEL_FLAG_CMD
 - Integrity Report System API, [121](#)
- IRS_DENY_DELETE_FROM_SMC_TZ
 - Integrity Report System API, [119](#)
- IRS_DENY_READ_FROM_SMC_TZ
 - Integrity Report System API, [119](#)
- IRS_DENY_WRITE_FROM_SMC_TZ
 - Integrity Report System API, [119](#)
- IRS_FAIL_TZ
 - Integrity Report System API, [119](#)
- IRS_GET_FLAG_VALUE_CMD
 - Integrity Report System API, [121](#)
- IRS_INC_FLAG_CMD
 - Integrity Report System API, [121](#)
- IRS_INCORRECT_CHECKSUM_TZ
 - Integrity Report System API, [119](#)
- IRS_INCORRECT_FLAG_TYPE_TZ
 - Integrity Report System API, [119](#)
- IRS_INCORRECT_RT_ID_TZ
 - Integrity Report System API, [119](#)
- IRS_INTERNAL_CMD
 - Integrity Report System API, [121](#)
- IRS_MAX_VAL_COUNTER_RT_TZ
 - Integrity Report System API, [119](#)
- IRS_MAX_VAL_COUNTER_TZ
 - Integrity Report System API, [120](#)
- IRS_NWD_CTRL
 - Integrity Report System API, [121](#)
- IRS_NWD_RD
 - Integrity Report System API, [121](#)
- IRS_NWD_WR
 - Integrity Report System API, [121](#)
- IRS_PARAM
 - Integrity Report System API, [121](#)
- IRS_RT_FLAGS_EMPTY_TZ
 - Integrity Report System API, [120](#)
- IRS_RT_FLAGS_FULL_TZ
 - Integrity Report System API, [120](#)
- IRS_SET_FLAG_CMD
 - Integrity Report System API, [121](#)
- IRS_SET_FLAG_VALUE_CMD
 - Integrity Report System API, [121](#)
- IRS_SUCCESS_TZ
 - Integrity Report System API, [120](#)
- IRS_SWD_CTRL
 - Integrity Report System API, [121](#)
- IRS_SWD_RD
 - Integrity Report System API, [121](#)
- IRS_SWD_WR
 - Integrity Report System API, [121](#)
- IRS_TYPE_BOOLEAN
 - Integrity Report System API, [121](#)
- IRS_TYPE_COUNTER
 - Integrity Report System API, [121](#)
- IRS_TYPE_VALUE
 - Integrity Report System API, [121](#)
- IRS_UNKNOWN_ERROR_TZ
 - Integrity Report System API, [120](#)
- IRS_UNKNOWN_ID_TZ
 - Integrity Report System API, [120](#)
- IRS_UNKNOWN_INT_CMD_TZ
 - Integrity Report System API, [120](#)
- ignore_handler_s
 - Auxiliary API, [243](#)
- ilogb
 - Math library API, [173](#)
- ilogbf
 - Math library API, [173](#)
- ilogbl
 - Math library API, [174](#)
- Integrity Report System API, [118](#)
 - IRS_ADD_FLAG_CMD, [121](#)
 - IRS_DEL_FLAG_CMD, [121](#)
 - IRS_DENY_DELETE_FROM_SMC_TZ, [119](#)
 - IRS_DENY_READ_FROM_SMC_TZ, [119](#)
 - IRS_DENY_WRITE_FROM_SMC_TZ, [119](#)
 - IRS_FAIL_TZ, [119](#)
 - IRS_GET_FLAG_VALUE_CMD, [121](#)
 - IRS_INC_FLAG_CMD, [121](#)
 - IRS_INCORRECT_CHECKSUM_TZ, [119](#)
 - IRS_INCORRECT_FLAG_TYPE_TZ, [119](#)
 - IRS_INCORRECT_RT_ID_TZ, [119](#)
 - IRS_INTERNAL_CMD, [121](#)
 - IRS_MAX_VAL_COUNTER_RT_TZ, [119](#)
 - IRS_MAX_VAL_COUNTER_TZ, [120](#)
 - IRS_NWD_CTRL, [121](#)
 - IRS_NWD_RD, [121](#)
 - IRS_NWD_WR, [121](#)
 - IRS_PARAM, [121](#)
 - IRS_RT_FLAGS_EMPTY_TZ, [120](#)
 - IRS_RT_FLAGS_FULL_TZ, [120](#)
 - IRS_SET_FLAG_CMD, [121](#)
 - IRS_SET_FLAG_VALUE_CMD, [121](#)
 - IRS_SUCCESS_TZ, [120](#)
 - IRS_SWD_CTRL, [121](#)
 - IRS_SWD_RD, [121](#)
 - IRS_SWD_WR, [121](#)
 - IRS_TYPE_BOOLEAN, [121](#)
 - IRS_TYPE_COUNTER, [121](#)
 - IRS_TYPE_VALUE, [121](#)
 - IRS_UNKNOWN_ERROR_TZ, [120](#)
 - IRS_UNKNOWN_ID_TZ, [120](#)
 - IRS_UNKNOWN_INT_CMD_TZ, [120](#)
 - TEES_AddIrsFlag, [122](#)
 - TEES_DelIrsFlag, [122](#)
 - TEES_GetIrsFlagValue, [122](#)
 - TEES_IncIrsFlag, [123](#)
 - TEES_SetIrsFlag, [123](#)
 - TEES_SetIrsFlagValue, [123](#)
- intrinfo, [65](#)
- intruhandler
 - Loadable driver API, [65](#)
- inttypes.h, [350](#)
- invoke_constraint_handler_s
 - Auxiliary API, [243](#)
- ioctl

Auxiliary API, [243](#)
 fops, [337](#)
 TEE_iSocket_s, [315](#)
 ioctl_desc, [338](#)
 cnt, [339](#)
 tpl, [339](#)
 type, [339](#)
 irs.h, [351](#)
 isalnum
 Auxiliary API, [244](#)
 isalpha
 Auxiliary API, [244](#)
 isascii
 Auxiliary API, [245](#)
 isblank
 Auxiliary API, [245](#)
 iscntrl
 Auxiliary API, [245](#)
 isdigit
 Auxiliary API, [246](#)
 isfinite
 Math library API, [164](#)
 isgraph
 Auxiliary API, [246](#)
 islower
 Auxiliary API, [246](#)
 isnan
 Math library API, [164](#)
 isprint
 Auxiliary API, [247](#)
 ispunct
 Auxiliary API, [247](#)
 isspace
 Auxiliary API, [247](#)
 isupper
 Auxiliary API, [248](#)
 isxdigit
 Auxiliary API, [248](#)

 KM_KW_MAX_AAD_LEN
 Samsung Internal API, [48](#)
 KM_KW_MAX_INPUT_LEN
 Samsung Internal API, [48](#)
 KM_KW_MAX_IV_LEN
 Samsung Internal API, [48](#)
 KM_KW_MAX_KEY_LEN
 Samsung Internal API, [49](#)
 KM_KW_MAX_SALT_LEN
 Samsung Internal API, [49](#)
 KM_KW_MAX_TAG_LEN
 Samsung Internal API, [49](#)

 LLONG_MAX
 Auxiliary API, [223](#)
 LLONG_MIN
 Auxiliary API, [224](#)
 LONG_MAX
 Auxiliary API, [224](#)
 LONG_MIN

Auxiliary API, [224](#)
 len
 desc_atom, [336](#)
 limits.h, [352](#)
 listen
 Socket library API, [186](#)
 Loadable driver API, [64](#)
 intruhandler, [65](#)
 TEES_AcquireUserBuffer, [66](#)
 TEES_AllocateInterrupt, [66](#)
 TEES_CompleteInterrupt, [67](#)
 TEES_CompleteRequest, [68](#)
 TEES_DcacheClean, [69](#)
 TEES_DcacheFlush, [69](#)
 TEES_DriverDestructor_t, [65](#)
 TEES_FiniDriver, [69](#)
 TEES_GenerateInterrupt, [70](#)
 TEES_InitDriver, [71](#)
 TEES_InterruptHandle, [65](#)
 TEES_RegisterDriver, [72](#)
 TEES_RegisterDriverDestructor, [72](#)
 TEES_RegisterIoctlDesc, [73](#)
 TEES_ReleaseDriver, [73](#)
 TEES_ReleaseInterrupt, [74](#)
 TEES_ReleaseUserBuffer, [74](#)
 TEES_WaitForInterrupt, [75](#)
 log
 Math library API, [174](#)
 logb
 Math library API, [174](#)
 logbf
 Math library API, [175](#)
 logbl
 Math library API, [175](#)
 logf
 Math library API, [175](#)
 lookup
 fops, [337](#)
 lseek
 Auxiliary API, [248](#)
 fops, [337](#)

 M_CACHE_PAGES
 Auxiliary API, [224](#)
 M_PI_2
 Math library API, [165](#)
 M_PI_4
 Math library API, [165](#)
 M_PI
 Math library API, [165](#)
 M_E
 Math library API, [165](#)
 MAP_ANONYMOUS
 Auxiliary API, [224](#)
 MAP_FAILED
 Auxiliary API, [224](#)
 MAP_FIXED
 Auxiliary API, [224](#)
 MAP_GROWSDOWN

- Auxiliary API, [225](#)
- MAP_PHYS_NON_CACHED
 - Auxiliary API, [225](#)
- MAP_PHYS_NON_SECURE
 - Auxiliary API, [225](#)
- MAP_POPULATE
 - Auxiliary API, [225](#)
- MAP_PRIVATE
 - Auxiliary API, [225](#)
- MAP_SHARED
 - Auxiliary API, [225](#)
- MAP_STACK
 - Auxiliary API, [225](#)
- MAX_CPU
 - Auxiliary API, [226](#)
- MAX_FONT_SIZE
 - Trusted user interface, [103](#)
- MIN_FONT_SIZE
 - Trusted user interface, [103](#)
- MQ_MAX_NAME
 - Message queue library API, [181](#)
- MT_INFO_MAX_EVENT
 - Trusted user interface, [104](#)
- MUTEX_STATE_LOCKED
 - Thread support library API, [134](#)
- MUTEX_STATE_UNLOCKED
 - Thread support library API, [134](#)
- malloc
 - Auxiliary API, [249](#)
- malloc.h, [353](#)
- Math library API, [162](#)
 - atan, [166](#)
 - atan2, [166](#)
 - atan2f, [166](#)
 - atanf, [166](#)
 - ceil, [167](#)
 - ceilf, [167](#)
 - copysign, [167](#)
 - copysignf, [168](#)
 - cos, [168](#)
 - cosf, [168](#)
 - exp, [169](#)
 - expf, [169](#)
 - FP_ILOGB0, [164](#)
 - FP_ILOGBNAN, [164](#)
 - fabs, [169](#)
 - fabsf, [170](#)
 - floor, [170](#)
 - floorf, [170](#)
 - fmax, [171](#)
 - fmaxf, [171](#)
 - fmaxl, [171](#)
 - fmin, [172](#)
 - fminf, [172](#)
 - hypot, [172](#)
 - hypotf, [173](#)
 - INFINITY, [164](#)
 - ilogb, [173](#)
 - ilogbf, [173](#)
 - ilogbl, [174](#)
 - isfinite, [164](#)
 - isnan, [164](#)
 - log, [174](#)
 - logb, [174](#)
 - logbf, [175](#)
 - logbl, [175](#)
 - logf, [175](#)
 - M_PI_2, [165](#)
 - M_PI_4, [165](#)
 - M_PI, [165](#)
 - M_E, [165](#)
 - modf, [176](#)
 - modff, [176](#)
 - NAN, [165](#)
 - pow, [176](#)
 - powf, [177](#)
 - round, [177](#)
 - roundf, [177](#)
 - scalbn, [178](#)
 - scalbnf, [178](#)
 - scalbnl, [178](#)
 - sin, [179](#)
 - sinf, [179](#)
 - sqrt, [179](#)
 - sqrtf, [180](#)
- math.h, [353](#)
- memchr
 - Auxiliary API, [249](#)
- memcmp
 - Auxiliary API, [250](#)
- memcpy
 - Auxiliary API, [250](#)
- memcpy_s
 - Auxiliary API, [251](#)
- memmove
 - Auxiliary API, [251](#)
- memmove_s
 - Auxiliary API, [251](#)
- memrchr
 - Auxiliary API, [252](#)
- memset
 - Auxiliary API, [252](#)
- memset_s
 - Auxiliary API, [252](#)
- Message queue library API, [181](#)
 - MQ_MAX_NAME, [181](#)
 - mq_close, [182](#)
 - mq_open, [182](#)
 - mq_receive, [182](#)
 - mq_send, [182](#)
 - mq_unlink, [183](#)
 - mqd_t, [181](#)
- Miscellaneous extensions, [125](#)
 - TEES_DuplWshmem, [126](#)
 - TEES_GetClientCredentials, [126](#)
 - TEES_GetTaskDataSize, [126](#)

- TEES_IsREESharedMemory, [127](#)
- mkdir
 - fops, [337](#)
 - Stat, [298](#)
- mmap
 - Auxiliary API, [253](#)
 - fops, [337](#)
- mode_t
 - Auxiliary API, [233](#)
- modf
 - Math library API, [176](#)
- modff
 - Math library API, [176](#)
- mprotect
 - Auxiliary API, [255](#)
- mq_close
 - Message queue library API, [182](#)
- mq_open
 - Message queue library API, [182](#)
- mq_receive
 - Message queue library API, [182](#)
- mq_send
 - Message queue library API, [182](#)
- mq_unlink
 - Message queue library API, [183](#)
- mqd_t
 - Message queue library API, [181](#)
- mqueue.h, [355](#)
- ms_to_timespec
 - Auxiliary API, [255](#)
- munmap
 - Auxiliary API, [255](#)
- NAN
 - Math library API, [165](#)
- NUM_MILLIS_IN_SEC
 - Auxiliary API, [226](#)
- NUM_NANOS_IN_MILLI
 - Auxiliary API, [226](#)
- NUM_NANOS_IN_SEC
 - Auxiliary API, [226](#)
- NUM_NANOS_IN_USEC
 - Auxiliary API, [226](#)
- NUM_SECONDS_IN_MIN
 - Auxiliary API, [226](#)
- nanosleep
 - Auxiliary API, [256](#)
- off_t
 - Auxiliary API, [233](#)
- open
 - Auxiliary API, [257](#)
 - fops, [337](#)
 - TEE_iSocket_s, [315](#)
- PGOFF_SHIFT
 - Auxiliary API, [226](#)
- PRId16
 - Auxiliary API, [227](#)
- PRId32
 - Auxiliary API, [227](#)
- PRId64
 - Auxiliary API, [227](#)
- PRlu16
 - Auxiliary API, [227](#)
- PRlu32
 - Auxiliary API, [227](#)
- PRlu64
 - Auxiliary API, [227](#)
- PRlx16
 - Auxiliary API, [227](#)
- PRlx32
 - Auxiliary API, [228](#)
- PRlx64
 - Auxiliary API, [228](#)
- PROT_EXEC
 - Auxiliary API, [228](#)
- PROT_NONE
 - Auxiliary API, [228](#)
- PROT_READ
 - Auxiliary API, [228](#)
- PROT_WRITE
 - Auxiliary API, [228](#)
- PTHREAD_CREATE_DETACHED
 - Thread support library API, [137](#)
- PTHREAD_CREATE_JOINABLE
 - Thread support library API, [137](#)
- PTHREAD_ERRORCHECK_MUTEX_INITIALIZER_NP
 - Thread support library API, [134](#)
- PTHREAD_GUARD_MAX
 - Thread support library API, [134](#)
- PTHREAD_GUARD_MIN
 - Thread support library API, [134](#)
- PTHREAD_KEYS_MAX
 - Auxiliary API, [228](#)
- PTHREAD_MUTEX_DEFAULT
 - Thread support library API, [137](#)
- PTHREAD_MUTEX_DESTROYED
 - Thread support library API, [137](#)
- PTHREAD_MUTEX_ERRORCHECK_NP
 - Thread support library API, [137](#)
- PTHREAD_MUTEX_ERRORCHECK
 - Thread support library API, [137](#)
- PTHREAD_MUTEX_INITIALIZER
 - Thread support library API, [134](#)
- PTHREAD_MUTEX_NORMAL
 - Thread support library API, [137](#)
- PTHREAD_MUTEX_RECURSIVE_NP
 - Thread support library API, [137](#)
- PTHREAD_MUTEX_RECURSIVE
 - Thread support library API, [137](#)
- PTHREAD_ONCE_INIT
 - Thread support library API, [134](#)
- PTHREAD_RECURSIVE_MUTEX_INITIALIZER_NP
 - Thread support library API, [135](#)
- PTHREAD_STACK_MIN
 - Thread support library API, [135](#)

pid_t
 Auxiliary API, [233](#)
pow
 Math library API, [176](#)
powf
 Math library API, [177](#)
print_no_alloc.h, [356](#)
printf
 Auxiliary API, [257](#)
printf_no_alloc
 Auxiliary API, [258](#)
printf_s
 Auxiliary API, [258](#)
priv
 usr_drv_info, [341](#)
probe
 fops, [337](#)
profil
 Auxiliary API, [259](#)
protocol_error_code
 Tee_sockets, [324](#)
protocol_errors.h, [356](#)
protocolID
 TEE_iSocket_s, [315](#)
pthread.h, [358](#)
pthread_attr_destroy
 Thread support library API, [138](#)
pthread_attr_getdetachstate
 Thread support library API, [138](#)
pthread_attr_getguardsize
 Thread support library API, [138](#)
pthread_attr_getstack
 Thread support library API, [138](#)
pthread_attr_getstackaddr
 Thread support library API, [139](#)
pthread_attr_getstacksize
 Thread support library API, [139](#)
pthread_attr_init
 Thread support library API, [139](#)
pthread_attr_setdetachstate
 Thread support library API, [140](#)
pthread_attr_setguardsize
 Thread support library API, [140](#)
pthread_attr_setstack
 Thread support library API, [140](#)
pthread_attr_setstackaddr
 Thread support library API, [141](#)
pthread_attr_setstacksize
 Thread support library API, [141](#)
pthread_attr_t
 Thread support library API, [136](#)
pthread_cond_broadcast
 Thread support library API, [141](#)
pthread_cond_destroy
 Thread support library API, [142](#)
pthread_cond_init
 Thread support library API, [143](#)
pthread_cond_signal
 Thread support library API, [143](#)
pthread_cond_t
 Thread support library API, [136](#)
pthread_cond_timedwait
 Thread support library API, [144](#)
pthread_cond_wait
 Thread support library API, [145](#)
pthread_condattr_destroy
 Thread support library API, [146](#)
pthread_condattr_init
 Thread support library API, [146](#)
pthread_condattr_t
 Thread support library API, [136](#)
pthread_create
 Thread support library API, [146](#)
pthread_detach
 Thread support library API, [147](#)
pthread_equal
 Thread support library API, [148](#)
pthread_exit
 Thread support library API, [148](#)
pthread_getattr_np
 Thread support library API, [148](#)
pthread_getspecific
 Thread support library API, [149](#)
pthread_impl_t
 Thread support library API, [136](#)
pthread_join
 Thread support library API, [150](#)
pthread_key_create
 Thread support library API, [151](#)
pthread_key_delete
 Thread support library API, [152](#)
pthread_key_t
 Thread support library API, [136](#)
pthread_kill
 Thread support library API, [153](#)
pthread_mutex_destroy
 Thread support library API, [153](#)
pthread_mutex_init
 Thread support library API, [154](#)
pthread_mutex_lock
 Thread support library API, [155](#)
pthread_mutex_t
 Thread support library API, [136](#)
pthread_mutex_trylock
 Thread support library API, [155](#)
pthread_mutex_unlock
 Thread support library API, [156](#)
pthread_mutexattr_destroy
 Thread support library API, [157](#)
pthread_mutexattr_gettype
 Thread support library API, [157](#)
pthread_mutexattr_init
 Thread support library API, [158](#)
pthread_mutexattr_settype
 Thread support library API, [158](#)
pthread_mutexattr_t

- Thread support library API, [136](#)
- pthread_once
 - Thread support library API, [159](#)
- pthread_once_t
 - Thread support library API, [136](#)
- pthread_self
 - Thread support library API, [160](#)
- pthread_setspecific
 - Thread support library API, [160](#)
- pthread_sigmask
 - Thread support library API, [135](#)
- pthread_t
 - Thread support library API, [137](#)
- putchar
 - Auxiliary API, [259](#)
- puts
 - Auxiliary API, [260](#)
- qsort
 - Auxiliary API, [260](#)
- qsort_r
 - Auxiliary API, [260](#)
- RPMB API, [128](#)
 - RPMB_TYPE_BLOCK, [128](#)
 - RPMB_TYPE_BYTE, [128](#)
 - TEES_RPMBCheckEnable, [128](#)
 - TEES_RPMBRead, [128](#)
 - TEES_RPMBWrite, [129](#)
- RPMB_TYPE_BLOCK
 - RPMB API, [128](#)
- RPMB_TYPE_BYTE
 - RPMB API, [128](#)
- read
 - Auxiliary API, [261](#)
 - fops, [338](#)
- readdir
 - fops, [338](#)
- realloc
 - Auxiliary API, [261](#)
- recv
 - Socket library API, [186](#)
 - TEE_iSocket_s, [315](#)
- recvmsg
 - Socket library API, [186](#)
- rename
 - Auxiliary API, [262](#)
 - fops, [338](#)
- rmdir
 - Auxiliary API, [262](#)
 - fops, [338](#)
- rot_t, [47](#)
- round
 - Math library API, [177](#)
- roundf
 - Math library API, [177](#)
- rpmb.h, [361](#)
- S_ACCPERM
 - Stat, [298](#)
- SCHAR_MAX
 - Auxiliary API, [229](#)
- SCHAR_MIN
 - Auxiliary API, [229](#)
- SCNd16
 - Auxiliary API, [229](#)
- SCNd32
 - Auxiliary API, [229](#)
- SCNd64
 - Auxiliary API, [229](#)
- SCNu16
 - Auxiliary API, [229](#)
- SCNu32
 - Auxiliary API, [229](#)
- SCNu64
 - Auxiliary API, [230](#)
- SCNx16
 - Auxiliary API, [230](#)
- SCNx32
 - Auxiliary API, [230](#)
- SCNx64
 - Auxiliary API, [230](#)
- SHRT_MAX
 - Auxiliary API, [230](#)
- SHRT_MIN
 - Auxiliary API, [230](#)
- SO_AC_LEN
 - Samsung Internal API, [49](#)
- SO_AUTH_ID_LEN
 - Samsung Internal API, [49](#)
- SO_HEADER_SIZE_STATIC
 - Samsung Internal API, [49](#)
- SO_IV_LEN
 - Samsung Internal API, [49](#)
- SO_MAGIC_NUMBER_LEN
 - Samsung Internal API, [50](#)
- SO_OUT_BUF_SIZE
 - Samsung Internal API, [50](#)
- SO_TA_ID_LEN
 - Samsung Internal API, [50](#)
- SO_TAG_LEN
 - Samsung Internal API, [50](#)
- SPI API, [83](#)
 - TEES_SPIClockDisable, [84](#)
 - TEES_SPIClockEnable, [84](#)
 - TEES_SPIDMAInit, [85](#)
 - TEES_SPIExit, [85](#)
 - TEES_SPIHandler, [84](#)
 - TEES_SPIInit, [86](#)
 - TEES_SPIRead, [87](#)
 - TEES_SPISetBitsPerWord, [87](#)
 - TEES_SPISetCPHA, [89](#)
 - TEES_SPISetCPOL, [89](#)
 - TEES_SPISetClockSpeed, [88](#)
 - TEES_SPISetConfig, [88](#)
 - TEES_SPISetDMAMode, [90](#)
 - TEES_SPIWrite, [90](#)

- TEES_SPIWriteRead, 91
- Samsung Internal API, 45
 - errno_to_tee_error, 50
 - KM_KW_MAX_AAD_LEN, 48
 - KM_KW_MAX_INPUT_LEN, 48
 - KM_KW_MAX_IV_LEN, 48
 - KM_KW_MAX_KEY_LEN, 49
 - KM_KW_MAX_SALT_LEN, 49
 - KM_KW_MAX_TAG_LEN, 49
 - SO_AC_LEN, 49
 - SO_AUTH_ID_LEN, 49
 - SO_HEADER_SIZE_STATIC, 49
 - SO_IV_LEN, 49
 - SO_MAGIC_NUMBER_LEN, 50
 - SO_OUT_BUF_SIZE, 50
 - SO_TA_ID_LEN, 50
 - SO_TAG_LEN, 50
 - TEES_CheckSecureObjectCreator, 50
 - TEES_DeriveKeyKDF, 51
 - TEES_DeriveKeySetKDF, 52
 - TEES_EI2if, 52
 - TEES_EnterCritical, 53
 - TEES_ExitCritical, 53
 - TEES_GetRoT, 53
 - TEES_HDCP_SetKeyInfo, 54
 - TEES_LockHWCryptoBuf, 55
 - TEES_SECCAM_GetStatus, 55
 - TEES_SECCAM_IsProtected, 56
 - TEES_SECCAM_Protect, 57
 - TEES_SECCAM_Unprotect, 57
 - TEES_TUI_Protect, 58
 - TEES_TUI_SetInfo, 59
 - TEES_TUI_Unprotect, 60
 - TEES_UnlockHWCryptoBuf, 61
 - TEES_UnwrapSecureObject, 61
 - TEES_WrapSecureObject, 62
 - TEES_WrappedWithREK, 62
- scBaseKeyHandle
 - __TEE_SC_DeviceKeyRef.__unnamed__, 332
- scCardKeyRef
 - __TEE_SC_Params, 334
- scDeviceKeyRef
 - __TEE_SC_Params, 334
- scKeyEncHandle
 - __TEE_SC_KeySetRef, 333
- scKeyID
 - __TEE_SC_CardKeyRef, 331
- scKeyMacHandle
 - __TEE_SC_KeySetRef, 333
- scKeySetRef
 - __TEE_SC_DeviceKeyRef.__unnamed__, 332
- scKeyType
 - __TEE_SC_DeviceKeyRef, 332
- scKeyVersion
 - __TEE_SC_CardKeyRef, 331
- scOID
 - __TEE_SC_Params, 334
- scSecurityLevel
 - __TEE_SC_Params, 334
- scType
 - __TEE_SC_Params, 334
- scalbn
 - Math library API, 178
- scalbnf
 - Math library API, 178
- scalbnl
 - Math library API, 178
- sched.h, 362
- sched_getaffinity
 - Auxiliary API, 263
- sched_setaffinity
 - Auxiliary API, 263
- sched_yield
 - Auxiliary API, 264
- scma.h, 363
- sePresent
 - __TEE_SEReaderProperties, 335
- secHDCPKeyInfo_t, 48
- selectResponseEnable
 - __TEE_SEReaderProperties, 335
- send
 - Socket library API, 187
 - TEE_iSocket_s, 316
- sendmsg
 - Socket library API, 187
- set_constraint_handler_s
 - Auxiliary API, 264
- setenv
 - Auxiliary API, 264
- setsockopt
 - Socket library API, 187
- sin
 - Math library API, 179
- sinf
 - Math library API, 179
- smc_data, 339
 - arg_types, 339
 - args, 339
- snprintf
 - Auxiliary API, 265
- snprintf_s
 - Auxiliary API, 266
- socket
 - Socket library API, 188
- Socket library API, 184
 - accept, 184
 - bind, 185
 - connect, 185
 - getsockopt, 185
 - listen, 186
 - recv, 186
 - recvmsg, 186
 - send, 187
 - sendmsg, 187
 - setsockopt, 187
 - socket, 188

socketpair, 188
socketpair
 Socket library API, 188
sprintf
 Auxiliary API, 266
sprintf_s
 Auxiliary API, 267
sqrt
 Math library API, 179
sqrtf
 Math library API, 180
sscanf
 Auxiliary API, 267
sscanf_s
 Auxiliary API, 268
ssize_t
 Auxiliary API, 233
st_gid
 stat, 340
st_mode
 stat, 340
st_refcount
 stat, 340
st_size
 stat, 340
st_uid
 stat, 340
Stat, 298
 fstat, 298
 mkdir, 298
 S_ACCPERM, 298
 stat, 299
stat, 340
 Auxiliary API, 268
 fops, 338
 st_gid, 340
 st_mode, 340
 st_refcount, 340
 st_size, 340
 st_uid, 340
 Stat, 299
stdin
 Auxiliary API, 297
stdio.h, 364
stdlib.h, 365
strcat
 Auxiliary API, 268
strcat_s
 Auxiliary API, 269
strchr
 Auxiliary API, 269
strchrnul
 Auxiliary API, 270
strcmp
 Auxiliary API, 270
strcpy
 Auxiliary API, 271
strcpy_s
 Auxiliary API, 272
strcspn
 Auxiliary API, 272
strdup
 Auxiliary API, 272
strerror
 Auxiliary API, 273
strerror_r
 Auxiliary API, 273
strerror_s
 Auxiliary API, 273
string.h, 367
strlcat
 Auxiliary API, 274
strlen
 Auxiliary API, 274
strncat
 Auxiliary API, 275
strncat_s
 Auxiliary API, 275
strncmp
 Auxiliary API, 276
strncpy
 Auxiliary API, 277
strncpy_s
 Auxiliary API, 277
strnlen
 Auxiliary API, 278
strnlen_s
 Auxiliary API, 278
strrchr
 Auxiliary API, 279
strspn
 Auxiliary API, 279
strstr
 Auxiliary API, 280
strtod
 Auxiliary API, 280
strtof
 Auxiliary API, 281
strtok
 Auxiliary API, 282
strtok_r
 Auxiliary API, 282
strtol
 Auxiliary API, 283
strtold
 Auxiliary API, 283
strtoll
 Auxiliary API, 284
strtoul
 Auxiliary API, 285
strtoull
 Auxiliary API, 285
sys/credentials.h, 369
sys/ioctl.h, 369
sys/mman.h, 346
sys/socket.h, 370

sys/stat.h, [346](#)
 sys/types.h, [371](#)
 sys/un.h, [371](#)
 sysconf
 Auxiliary API, [286](#)

 TEE_IP_VERSION_4
 Tee_sockets, [327](#)
 TEE_IP_VERSION_6
 Tee_sockets, [327](#)
 TEE_IP_VERSION_DC
 Tee_sockets, [327](#)
 TEE_ISOCKET_ERROR_HOSTNAME
 Tee_sockets, [322](#)
 TEE_ISOCKET_ERROR_LARGE_BUFFER
 Tee_sockets, [322](#)
 TEE_ISOCKET_ERROR_OUT_OF_RESOURCES
 Tee_sockets, [322](#)
 TEE_ISOCKET_ERROR_PROTOCOL
 Tee_sockets, [322](#)
 TEE_ISOCKET_ERROR_REMOTE_CLOSED
 Tee_sockets, [322](#)
 TEE_ISOCKET_ERROR_TIMEOUT
 Tee_sockets, [322](#)
 TEE_ISOCKET_IWC_ERROR_CHANNEL
 Tee_sockets, [325](#)
 TEE_ISOCKET_IWC_ERROR_INVALID_VERSION
 Tee_sockets, [325](#)
 TEE_ISOCKET_IWC_ERROR_NOT_IMPLEMENTED
 Tee_sockets, [325](#)
 TEE_ISOCKET_IWC_ERROR_SWD_CLIENT_AUTH_FAILED
 Tee_sockets, [325](#)
 TEE_ISOCKET_IWC_ERROR_TIMEOUT
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_BAD_PARAMETERS
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_BUFFER_TOO_SMALL
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_COMMUNICATION
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_CONNECTION_REFUSED
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_DATA_REMAIN
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_GENERIC
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_HOSTNAME_NOTRESOLVED
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_HOSTNAME_TRYAGAIN
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_HOSTNAME_UNKNOWN
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_LARGE_BUFFER
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_NET_UNREACHABLE
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_OUT_OF_MEMORY
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_OUT_OF_RESOURCES
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_REMOTE_CLOSED
 Tee_sockets, [325](#)
 TEE_ISOCKET_NET_ERROR_TIMEOUT
 Tee_sockets, [325](#)
 TEE_ISOCKET_PROTOCOLID_TCP
 Tee_sockets, [323](#)
 TEE_ISOCKET_PROTOCOLID_TLS
 Tee_sockets, [324](#)
 TEE_ISOCKET_PROTOCOLID_UDP
 Tee_sockets, [323](#)
 TEE_ISOCKET_SERVER_NAME_MAX_LENGTH
 Tee_sockets, [322](#)
 TEE_ISOCKET_TCP_API_VERSION
 Tee_sockets, [322](#)
 TEE_ISOCKET_TCP_WARNING_UNKNOWN_OUT_OF_BAND
 Tee_sockets, [323](#)
 TEE_ISOCKET_TLS_ALERT_ACCESS_DENIED
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_BAD_CERT_HASH_VALUE
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_BAD_CERT_STATUS_RESPONSE
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_BAD_CERTIFICATE
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_BAD_RECORD_MAC
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_CERT_EXPIRED
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_CERT_REQUIRED
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_CERT_REVOKED
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_CERT_UNKNOWN
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_CERT_UNOBTAINABLE
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_CLOSE_NOTIFY
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_DECODE_ERROR
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_DECOMP_FAILED
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_DECRYPT_ERROR
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_DECRYPT_FAILED
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_EXPORT_RESTRICTED
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_HANDSHAKE_FAILED
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_ILLEGAL_PARAMETER
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_INAPPROPRIATE_FALLBACK
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_INSUFFICIENT_SECURITY
 Tee_sockets, [326](#)
 TEE_ISOCKET_TLS_ALERT_INTERNAL_ERROR

Tee_sockets, 326
 TEE_ISOCKET_TLS_ALERT_MISSING_EXTENSION Tee_sockets, 326
 TEE_ISOCKET_TLS_ALERT_NO_CERTIFICATE Tee_sockets, 326
 TEE_ISOCKET_TLS_ALERT_NO_RENEGOTIATION Tee_sockets, 326
 TEE_ISOCKET_TLS_ALERT_PROTOCOL_VERSION Tee_sockets, 326
 TEE_ISOCKET_TLS_ALERT_RECORD_OVERFLOW Tee_sockets, 326
 TEE_ISOCKET_TLS_ALERT_UNEXPECTED_MSG Tee_sockets, 326
 TEE_ISOCKET_TLS_ALERT_UNKNOWN_CA Tee_sockets, 326
 TEE_ISOCKET_TLS_ALERT_UNKNOWN_PSK_IDENTITY Tee_sockets, 326
 TEE_ISOCKET_TLS_ALERT_UNRECOGNIZED_NAME Tee_sockets, 326
 TEE_ISOCKET_TLS_ALERT_UNSUPPORTED_CERT Tee_sockets, 326
 TEE_ISOCKET_TLS_ALERT_UNSUPPORTED_EXTENSION Tee_sockets, 326
 TEE_ISOCKET_TLS_ALERT_USER_CANCELED Tee_sockets, 326
 TEE_ISOCKET_TLS_API_VERSION Tee_sockets, 324
 TEE_ISOCKET_TLS_CB_BAD_CERT_CHAIN Tee_sockets, 327
 TEE_ISOCKET_TLS_CB_CHECK_CERT_CHAIN Tee_sockets, 327
 TEE_ISOCKET_TLS_CB_CHECK_OCSP_STATUS Tee_sockets, 327
 TEE_ISOCKET_TLS_CB_REVOKED_OCSP_STATUS Tee_sockets, 327
 TEE_ISOCKET_TLS_CB_UNKNOWN_OCSP_STATUS Tee_sockets, 327
 TEE_ISOCKET_TLS_CERT_KEYUSAGE_CHECK_CLIENT Tee_sockets, 327
 TEE_ISOCKET_TLS_CERT_NAME_CHECK_CLIENT Tee_sockets, 327
 TEE_ISOCKET_TLS_CERT_NOTIFY_CLIENT Tee_sockets, 327
 TEE_ISOCKET_TLS_ERROR_ALERT_PENDING Tee_sockets, 326
 TEE_ISOCKET_TLS_ERROR_AUTHENTICATION Tee_sockets, 324
 TEE_ISOCKET_TLS_ERROR_CERT_COMMON_NAME_MISMATCH Tee_sockets, 325
 TEE_ISOCKET_TLS_ERROR_CERT_EXPIRED Tee_sockets, 325
 TEE_ISOCKET_TLS_ERROR_CERT_IS_TOO_LONG Tee_sockets, 326
 TEE_ISOCKET_TLS_ERROR_CERT_PARSING Tee_sockets, 325
 TEE_ISOCKET_TLS_ERROR_CERT_REVOKED Tee_sockets, 326
 TEE_ISOCKET_TLS_ERROR_CERT_SIGN_VERIFICATION Tee_sockets, 325
 TEE_ISOCKET_TLS_ERROR_CERT_STATUS_UNKNOWN Tee_sockets, 326
 TEE_ISOCKET_TLS_ERROR_CERT_UNKNOWN_CA Tee_sockets, 326
 TEE_ISOCKET_TLS_ERROR_CERT_UNSUPPORTED Tee_sockets, 326
 TEE_ISOCKET_TLS_ERROR_CRL_PARSING Tee_sockets, 325
 TEE_ISOCKET_TLS_ERROR_DATA Tee_sockets, 324
 TEE_ISOCKET_TLS_ERROR_ECDHE_GEN_KEY Tee_sockets, 325
 TEE_ISOCKET_TLS_ERROR_ECDHE_SERIALIZING Tee_sockets, 325
 TEE_ISOCKET_TLS_ERROR_ECDHE_SHARED_SECRET Tee_sockets, 325
 TEE_ISOCKET_TLS_ERROR_ECDHE_UNSUPPORTED_CURVE Tee_sockets, 325
 TEE_ISOCKET_TLS_ERROR_HANDSHAKE_UNEXPECTED_PARAMETER Tee_sockets, 326
 TEE_ISOCKET_TLS_ERROR_HANDSHAKE Tee_sockets, 324
 TEE_ISOCKET_TLS_ERROR_NO_ALERT_PRESENT Tee_sockets, 326
 TEE_ISOCKET_TLS_ERROR_REJECTED_SUITE Tee_sockets, 324
 TEE_ISOCKET_TLS_ERROR_UNEXPECTED_MESSAGE Tee_sockets, 325
 TEE_ISOCKET_TLS_ERROR_UNSUPPORTED_SUITE Tee_sockets, 324
 TEE_ISOCKET_TLS_ERROR_USER_CANCELED Tee_sockets, 326
 TEE_ISOCKET_TLS_ERROR_VERSION Tee_sockets, 324
 TEE_ISOCKET_TLS_OCSP_CHECK_ADVISORY Tee_sockets, 327
 TEE_ISOCKET_TLS_OCSP_CHECK_CLIENT Tee_sockets, 327
 TEE_ISOCKET_TLS_OCSP_CHECK_MANDATORY Tee_sockets, 327
 TEE_ISOCKET_TLS_OCSP_STATUS_REQUEST_NO_RESPONSE Tee_sockets, 328
 TEE_ISOCKET_TLS_OCSP_STATUS_REQUEST Tee_sockets, 328
 TEE_ISOCKET_UDP_API_VERSION Tee_sockets, 323
 TEE_ISOCKET_UDP_WARNING_UNKNOWN_OUT_OF_BAND OPERATION Tee_sockets, 323
 TEE_ISOCKET_WARNING_PROTOCOL Tee_sockets, 322
 TEE_TEMPRARY_CLEAR_MODE Trusted user interface, 104
 TEE_TLS_CLIENT_CRED_CSC Tee_sockets, 327
 TEE_TLS_CLIENT_CRED_NONE Tee_sockets, 327
 TEE_TLS_CLIENT_CRED_PDC Tee_sockets, 327

- Tee_sockets, 327
- TEE_TLS_SERVER_CRED_CSC
 - Tee_sockets, 328
- TEE_TLS_SERVER_CRED_PDC
 - Tee_sockets, 328
- TEE_TLS_VERSION_1v2
 - Tee_sockets, 328
- TEE_TLS_VERSION_ALL
 - Tee_sockets, 328
- TEE_TUI_ALPHANUMERICAL
 - Trusted user interface, 105
- TEE_TUI_CANCEL
 - Trusted user interface, 104
- TEE_TUI_CLEAR_MODE
 - Trusted user interface, 104
- TEE_TUI_CORRECTION
 - Trusted user interface, 104
- TEE_TUI_HIDDEN_MODE
 - Trusted user interface, 104
- TEE_TUI_LANDSCAPE
 - Trusted user interface, 105
- TEE_TUI_NEXT
 - Trusted user interface, 104
- TEE_TUI_NO_SOURCE
 - Trusted user interface, 105
- TEE_TUI_NUMBER_BUTTON_TYPES
 - Trusted user interface, 104
- TEE_TUI_NUMERICAL
 - Trusted user interface, 105
- TEE_TUI_OBJECT_SOURCE
 - Trusted user interface, 105
- TEE_TUI_OK
 - Trusted user interface, 104
- TEE_TUI_PORTRAIT
 - Trusted user interface, 105
- TEE_TUI_PREVIOUS
 - Trusted user interface, 104
- TEE_TUI_REF_SOURCE
 - Trusted user interface, 105
- TEE_TUI_VALIDATE
 - Trusted user interface, 104
- TEE_TUIButton, 101
- TEE_TUIButtonType
 - Trusted user interface, 104
- TEE_TUICheckTextFormat
 - Trusted user interface, 106
- TEE_TUICloseSession
 - Trusted user interface, 107
- TEE_TUIDisplayScreen
 - Trusted user interface, 107
- TEE_TUIEntryField, 103
- TEE_TUIEntryFieldMode
 - Trusted user interface, 104
- TEE_TUIEntryFieldType
 - Trusted user interface, 104
- TEE_TUIGetScreenInfo
 - Trusted user interface, 109
- TEE_TUImage, 100
- TEE_TUImage.__unnamed__, 100
- TEE_TUImage.__unnamed__.object, 101
- TEE_TUImage.__unnamed__.ref, 101
- TEE_TUImageSource
 - Trusted user interface, 105
- TEE_TUInitSession
 - Trusted user interface, 110
- TEE_TUIScreenButtonInfo, 102
- TEE_TUIScreenConfiguration, 102
- TEE_TUIScreenInfo, 102
- TEE_TUIScreenLabel, 101
- TEE_TUIScreenOrientation
 - Trusted user interface, 105
- TEE_iSocket
 - Tee_sockets, 321
- TEE_iSocket_s, 314
 - close, 315
 - error, 315
 - ioctl, 315
 - open, 315
 - protocolID, 315
 - recv, 315
 - send, 316
 - TEE_iSocketVersion, 316
- TEE_iSocketVersion
 - TEE_iSocket_s, 316
- TEE_ipSocket_ipVersion_e
 - Tee_sockets, 326
- TEE_tcpSocket_Setup_s, 316
- TEE_tlsSocket_CB_Data
 - Tee_sockets, 321
- TEE_tlsSocket_CB_Data_s, 321
- TEE_tlsSocket_CallbackInfo_s, 318
- TEE_tlsSocket_CallbackReasonType_e
 - Tee_sockets, 327
- TEE_tlsSocket_CertStorageCred_s, 318
- TEE_tlsSocket_ClientCredentialType_e
 - Tee_sockets, 327
- TEE_tlsSocket_ClientPDC_s, 317
- TEE_tlsSocket_Credentials_s, 318
- TEE_tlsSocket_Credentials_s.__unnamed__, 318
- TEE_tlsSocket_ExtensionFlags_e
 - Tee_sockets, 327
- TEE_tlsSocket_PSK_Info_s, 317
- TEE_tlsSocket_SRP_Info_s, 317
- TEE_tlsSocket_ServerCredentialType_e
 - Tee_sockets, 327
- TEE_tlsSocket_ServerPDC_s, 317
- TEE_tlsSocket_Setup_s, 319
- TEE_tlsSocket_Setup_s.__unnamed__, 321
- TEE_tlsSocket_StatusRequestType_e
 - Tee_sockets, 328
- TEE_tlsSocket_tlsVersion_e
 - Tee_sockets, 328
- TEE_udpSocket_Change_s, 316
- TEE_udpSocket_Setup_s, 316
- TEES_AcquireUserBuffer
 - Loadable driver API, 66

- TEES_AddIrsFlag
 - Integrity Report System API, [122](#)
- TEES_AllocateContiguousMemory
 - Contiguous memory API, [79](#)
- TEES_AllocateInterrupt
 - Loadable driver API, [66](#)
- TEES_CONTIGUOUS_FLAG_ZERO_ON_ALLOC
 - Contiguous memory API, [79](#)
- TEES_CONTIGUOUS_FLAG_ZERO_ON_FREE
 - Contiguous memory API, [79](#)
- TEES_CONTIGUOUS_MAP_FIXED
 - Contiguous memory API, [79](#)
- TEES_CONTIGUOUS_MAP_NON_CACHEABLE
 - Contiguous memory API, [79](#)
- TEES_CONTIGUOUS_MAP_POPULATE
 - Contiguous memory API, [79](#)
- TEES_CONTIGUOUS_MAP_READ
 - Contiguous memory API, [79](#)
- TEES_CONTIGUOUS_MAP_WRITE
 - Contiguous memory API, [79](#)
- TEES_CheckSecureObjectCreator
 - Samsung Internal API, [50](#)
- TEES_ClientCredentials, [125](#)
- TEES_CompleteInterrupt
 - Loadable driver API, [67](#)
- TEES_CompleteRequest
 - Loadable driver API, [68](#)
- TEES_ContiguousAllocateFlags
 - Contiguous memory API, [79](#)
- TEES_ContiguousMapFlags
 - Contiguous memory API, [79](#)
- TEES_ContiguousMemoryHandle
 - Contiguous memory API, [78](#)
- TEES_DcacheClean
 - Loadable driver API, [69](#)
- TEES_DcacheFlush
 - Loadable driver API, [69](#)
- TEES_DeIrsFlag
 - Integrity Report System API, [122](#)
- TEES_DeriveKeyKDF
 - Samsung Internal API, [51](#)
- TEES_DeriveKeySetKDF
 - Samsung Internal API, [52](#)
- TEES_DriverDestructor_t
 - Loadable driver API, [65](#)
- TEES_DupIwshmem
 - Miscellaneous extensions, [126](#)
- TEES_ERROR_NOT_IMPLEMENTED
 - Trusted user interface, [106](#)
- TEES_ERROR_NOT_SUPPORTED
 - Trusted user interface, [106](#)
- TEES_ERROR_SHORT_BUFFER
 - Trusted user interface, [106](#)
- TEES_ERROR_TUI_BAD_FORMAT
 - Trusted user interface, [106](#)
- TEES_ERROR_TUI_BAD_STATE
 - Trusted user interface, [106](#)
- TEES_ERROR_TUI_BUSY
 - Trusted user interface, [106](#)
- TEES_ERROR_TUI_CANCEL
 - Trusted user interface, [106](#)
- TEES_ERROR_TUI_GENERIC
 - Trusted user interface, [106](#)
- TEES_ERROR_TUI_INVALID_PARAM
 - Trusted user interface, [106](#)
- TEES_ERROR_TUI_OUT_OF_MEMORY
 - Trusted user interface, [106](#)
- TEES_ERROR_TUI_TIMEOUT
 - Trusted user interface, [106](#)
- TEES_EI2if
 - Samsung Internal API, [52](#)
- TEES_EI2if_Args, [47](#)
- TEES_EnterCritical
 - Samsung Internal API, [53](#)
- TEES_ExitCritical
 - Samsung Internal API, [53](#)
- TEES_FALSE
 - Trusted user interface, [105](#)
- TEES_FiniDriver
 - Loadable driver API, [69](#)
- TEES_GenerateInterrupt
 - Loadable driver API, [70](#)
- TEES_GetClientCredentials
 - Miscellaneous extensions, [126](#)
- TEES_GetContiguousMemoryPhysaddr
 - Contiguous memory API, [80](#)
- TEES_GetIrsFlagValue
 - Integrity Report System API, [122](#)
- TEES_GetRoT
 - Samsung Internal API, [53](#)
- TEES_GetTaskDataSize
 - Miscellaneous extensions, [126](#)
- TEES_HDCP_SetKeyInfo
 - Samsung Internal API, [54](#)
- TEES_I2CExit
 - I2C API, [94](#)
- TEES_I2CHandler
 - I2C API, [94](#)
- TEES_I2CInit
 - I2C API, [94](#)
- TEES_I2CRead
 - I2C API, [94](#)
- TEES_I2CTransfer, [93](#)
- TEES_I2CWrite
 - I2C API, [95](#)
- TEES_I2CWriteRead
 - I2C API, [95](#)
- TEES_IWT_LISTENER_NAME_MAX_LENGTH
 - Tee_sockets, [324](#)
- TEES_IncIrsFlag
 - Integrity Report System API, [123](#)
- TEES_InitDriver
 - Loadable driver API, [71](#)
- TEES_InterruptHandle
 - Loadable driver API, [65](#)
- TEES_IsREESharedMemory

- Miscellaneous extensions, [127](#)
- TEES_lwtCloseChannel
 - Tee_sockets, [328](#)
- TEES_lwtOpenChannel
 - Tee_sockets, [328](#)
- TEES_lwtRead
 - Tee_sockets, [329](#)
- TEES_lwtWrite
 - Tee_sockets, [329](#)
- TEES_LockHWCryptoBuf
 - Samsung Internal API, [55](#)
- TEES_MapContiguousMemory
 - Contiguous memory API, [81](#)
- TEES_NFCExit
 - TeesL_NFC, [300](#)
- TEES_NFCHandler
 - TeesL_NFC, [300](#)
- TEES_NFCInit
 - TeesL_NFC, [300](#)
- TEES_NFCRead
 - TeesL_NFC, [301](#)
- TEES_NFCReadWait
 - TeesL_NFC, [301](#)
- TEES_NFCSetMode
 - TeesL_NFC, [302](#)
- TEES_NFCSleep
 - TeesL_NFC, [302](#)
- TEES_NFCWakeUp
 - TeesL_NFC, [302](#)
- TEES_NFCWrite
 - TeesL_NFC, [303](#)
- TEES_RPMBCheckEnable
 - RPMB API, [128](#)
- TEES_RPMBRead
 - RPMB API, [128](#)
- TEES_RPMBWrite
 - RPMB API, [129](#)
- TEES_RegisterDriver
 - Loadable driver API, [72](#)
- TEES_RegisterDriverDestructor
 - Loadable driver API, [72](#)
- TEES_RegisterIoctlDesc
 - Loadable driver API, [73](#)
- TEES_ReleaseContiguousMemory
 - Contiguous memory API, [82](#)
- TEES_ReleaseDriver
 - Loadable driver API, [73](#)
- TEES_ReleaseInterrupt
 - Loadable driver API, [74](#)
- TEES_ReleaseUserBuffer
 - Loadable driver API, [74](#)
- TEES_Result
 - Trusted user interface, [105](#)
- TEES_SECCAM_GetStatus
 - Samsung Internal API, [55](#)
- TEES_SECCAM_IsProtected
 - Samsung Internal API, [56](#)
- TEES_SECCAM_Protect
 - Samsung Internal API, [57](#)
- TEES_SECCAM_Unprotect
 - Samsung Internal API, [57](#)
- TEES_SMCCCommand
 - TeesL_smc, [304](#)
- TEES_SMCFini
 - TeesL_smc, [305](#)
- TEES_SMCInit
 - TeesL_smc, [305](#)
- TEES_SPIClockDisable
 - SPI API, [84](#)
- TEES_SPIClockEnable
 - SPI API, [84](#)
- TEES_SPIConfig, [84](#)
- TEES_SPIDMAInit
 - SPI API, [85](#)
- TEES_SPIExit
 - SPI API, [85](#)
- TEES_SPIHandler
 - SPI API, [84](#)
- TEES_SPIInit
 - SPI API, [86](#)
- TEES_SPIRead
 - SPI API, [87](#)
- TEES_SPISetBitsPerWord
 - SPI API, [87](#)
- TEES_SPISetCPHA
 - SPI API, [89](#)
- TEES_SPISetCPOL
 - SPI API, [89](#)
- TEES_SPISetClockSpeed
 - SPI API, [88](#)
- TEES_SPISetConfig
 - SPI API, [88](#)
- TEES_SPISetDMAMode
 - SPI API, [90](#)
- TEES_SPITransfer, [84](#)
- TEES_SPIWrite
 - SPI API, [90](#)
- TEES_SPIWriteRead
 - SPI API, [91](#)
- TEES_SUCCESS
 - Trusted user interface, [106](#)
- TEES_SetIrsFlag
 - Integrity Report System API, [123](#)
- TEES_SetIrsFlagValue
 - Integrity Report System API, [123](#)
- TEES_TOUCH_PRESSED
 - Trusted user interface, [106](#)
- TEES_TOUCH_RELEASED
 - Trusted user interface, [106](#)
- TEES_TRUE
 - Trusted user interface, [105](#)
- TEES_TUI_MT_Info, [100](#)
- TEES_TUI_Protect
 - Samsung Internal API, [58](#)
- TEES_TUI_SetInfo
 - Samsung Internal API, [59](#)

- TEES_TUI_Unprotect
 - Samsung Internal API, [60](#)
- TEES_TUICheckTextFormat
 - Trusted user interface, [110](#)
- TEES_TUICloseSession
 - Trusted user interface, [111](#)
- TEES_TUIDrawBuffer
 - Trusted user interface, [111](#)
- TEES_TUIDrawImage
 - Trusted user interface, [112](#)
- TEES_TUIDrawImageFromBuff
 - Trusted user interface, [112](#)
- TEES_TUIDrawImageToBuff
 - Trusted user interface, [113](#)
- TEES_TUIGetBuffer
 - Trusted user interface, [114](#)
- TEES_TUIGetMTEvent
 - Trusted user interface, [114](#)
- TEES_TUIGetScreenInfo
 - Trusted user interface, [115](#)
- TEES_TUIGetTouchEvent
 - Trusted user interface, [115](#)
- TEES_TUIImage, [100](#)
- TEES_TUIMTInfo, [99](#)
- TEES_TUIOpenSession
 - Trusted user interface, [116](#)
- TEES_TUIPrintString
 - Trusted user interface, [116](#)
- TEES_TUIRefreshScreen
 - Trusted user interface, [117](#)
- TEES_TUIScreenInfo, [99](#)
- TEES_TUITouchInfo, [99](#)
- TEES_TUITouchTypes
 - Trusted user interface, [106](#)
- TEES_UDF_FiniDriver
 - Tees_udf, [306](#)
- TEES_UDF_Init_FS_Driver
 - Tees_udf, [306](#)
- TEES_UDF_InitDriver
 - Tees_udf, [307](#)
- TEES_UDF_RegisterIoctlDesc
 - Tees_udf, [307](#)
- TEES_UnlockHWCryptoBuf
 - Samsung Internal API, [61](#)
- TEES_UnmapContiguousMemory
 - Contiguous memory API, [82](#)
- TEES_UnwrapSecureObject
 - Samsung Internal API, [61](#)
- TEES_WaitForInterrupt
 - Loadable driver API, [75](#)
- TEES_WrapSecureObject
 - Samsung Internal API, [62](#)
- TEES_WrappedWithREK
 - Samsung Internal API, [62](#)
- TEES_bool
 - Trusted user interface, [105](#)
- TEMP_FAILURE_RETRY
 - Auxiliary API, [230](#)
- tee_error.h, [372](#)
- tee_i2c.h, [372](#)
- tee_interrupt.h, [373](#)
- tee_isocket.h, [373](#)
- tee_nfc.h, [374](#)
- tee_smc.h, [375](#)
- Tee_sockets, [309](#)
 - protocol_error_code, [324](#)
 - TEE_IP_VERSION_4, [327](#)
 - TEE_IP_VERSION_6, [327](#)
 - TEE_IP_VERSION_DC, [327](#)
 - TEE_ISOCKET_ERROR_HOSTNAME, [322](#)
 - TEE_ISOCKET_ERROR_LARGE_BUFFER, [322](#)
 - TEE_ISOCKET_ERROR_OUT_OF_RESOURCES, [322](#)
 - TEE_ISOCKET_ERROR_PROTOCOL, [322](#)
 - TEE_ISOCKET_ERROR_REMOTE_CLOSED, [322](#)
 - TEE_ISOCKET_ERROR_TIMEOUT, [322](#)
 - TEE_ISOCKET_IWC_ERROR_CHANNEL, [325](#)
 - TEE_ISOCKET_IWC_ERROR_INVALID_VERSION, [325](#)
 - TEE_ISOCKET_IWC_ERROR_NOT_IMPLEMENTED, [325](#)
 - TEE_ISOCKET_IWC_ERROR_SWD_CLIENT_AUTH_FAILED, [325](#)
 - TEE_ISOCKET_IWC_ERROR_TIMEOUT, [325](#)
 - TEE_ISOCKET_NET_ERROR_BAD_PARAMETERS, [325](#)
 - TEE_ISOCKET_NET_ERROR_BUFFER_TOO_SMALL, [325](#)
 - TEE_ISOCKET_NET_ERROR_COMMUNICATION, [325](#)
 - TEE_ISOCKET_NET_ERROR_CONNECTION_REFUSED, [325](#)
 - TEE_ISOCKET_NET_ERROR_DATA_REMAIN, [325](#)
 - TEE_ISOCKET_NET_ERROR_GENERIC, [325](#)
 - TEE_ISOCKET_NET_ERROR_HOSTNAME_NOTRESOLVED, [325](#)
 - TEE_ISOCKET_NET_ERROR_HOSTNAME_TRYAGAIN, [325](#)
 - TEE_ISOCKET_NET_ERROR_HOSTNAME_UNKNOWN, [325](#)
 - TEE_ISOCKET_NET_ERROR_LARGE_BUFFER, [325](#)
 - TEE_ISOCKET_NET_ERROR_NET_UNREACHABLE, [325](#)
 - TEE_ISOCKET_NET_ERROR_OUT_OF_MEMORY, [325](#)
 - TEE_ISOCKET_NET_ERROR_OUT_OF_RESOURCES, [325](#)
 - TEE_ISOCKET_NET_ERROR_REMOTE_CLOSED, [325](#)
 - TEE_ISOCKET_NET_ERROR_TIMEOUT, [325](#)
 - TEE_ISOCKET_PROTOCOLID_TCP, [323](#)
 - TEE_ISOCKET_PROTOCOLID_TLS, [324](#)
 - TEE_ISOCKET_PROTOCOLID_UDP, [323](#)
 - TEE_ISOCKET_SERVER_NAME_MAX_LENGTH, [323](#)

322	TEE_ISOCKET_TLS_ALERT_UNKNOWN_CA,
TEE_ISOCKET_TCP_API_VERSION, 322	326
TEE_ISOCKET_TCP_WARNING_UNKNOWN_OUT_OF_BAND,	TEE_ISOCKET_TLS_ALERT_UNKNOWN_PSK_IDENTITY,
323	326
TEE_ISOCKET_TLS_ALERT_ACCESS_DENIED,	TEE_ISOCKET_TLS_ALERT_UNRECOGNIZED_NAME,
326	326
TEE_ISOCKET_TLS_ALERT_BAD_CERT_HASH_VALUE,	TEE_ISOCKET_TLS_ALERT_UNSUPPORTED_CERT,
326	326
TEE_ISOCKET_TLS_ALERT_BAD_CERT_STATUS_RESPONSE,	TEE_ISOCKET_TLS_ALERT_UNSUPPORTED_EXTENSION,
326	326
TEE_ISOCKET_TLS_ALERT_BAD_CERTIFICATE,	TEE_ISOCKET_TLS_ALERT_USER_CANCELED,
326	326
TEE_ISOCKET_TLS_ALERT_BAD_RECORD_MAC,	TEE_ISOCKET_TLS_API_VERSION, 324
326	TEE_ISOCKET_TLS_CB_BAD_CERT_CHAIN,
TEE_ISOCKET_TLS_ALERT_CERT_EXPIRED,	327
326	TEE_ISOCKET_TLS_CB_CHECK_CERT_CHAIN,
TEE_ISOCKET_TLS_ALERT_CERT_REQUIRED,	327
326	TEE_ISOCKET_TLS_CB_CHECK_OCSP_STATUS,
TEE_ISOCKET_TLS_ALERT_CERT_REVOKED,	327
326	TEE_ISOCKET_TLS_CB_REVOKED_OCSP_STATUS,
TEE_ISOCKET_TLS_ALERT_CERT_UNKNOWN,	327
326	TEE_ISOCKET_TLS_CB_UNKNOWN_OCSP_STATUS,
TEE_ISOCKET_TLS_ALERT_CERT_UNOBTAINABLE,	327
326	TEE_ISOCKET_TLS_CERT_KEYUSAGE_CHECK_CLIENT,
TEE_ISOCKET_TLS_ALERT_CLOSE_NOTIFY,	327
326	TEE_ISOCKET_TLS_CERT_NAME_CHECK_CLIENT,
TEE_ISOCKET_TLS_ALERT_DECODE_ERROR,	327
326	TEE_ISOCKET_TLS_CERT_NOTIFY_CLIENT,
TEE_ISOCKET_TLS_ALERT_DECOMP_FAILED,	327
326	TEE_ISOCKET_TLS_ERROR_ALERT_PENDING,
TEE_ISOCKET_TLS_ALERT_DECRYPT_ERROR,	326
326	TEE_ISOCKET_TLS_ERROR_AUTHENTICATION,
TEE_ISOCKET_TLS_ALERT_DECRYPT_FAILED,	324
326	TEE_ISOCKET_TLS_ERROR_CERT_COMMON_NAME_VERIFICATION_FAILED,
TEE_ISOCKET_TLS_ALERT_EXPORT_RESTRICTED,	325
326	TEE_ISOCKET_TLS_ERROR_CERT_EXPIRED,
TEE_ISOCKET_TLS_ALERT_HANDSHAKE_FAILED,	325
326	TEE_ISOCKET_TLS_ERROR_CERT_IS_TOO_LONG,
TEE_ISOCKET_TLS_ALERT_ILLEGAL_PARAMETER,	326
326	TEE_ISOCKET_TLS_ERROR_CERT_PARSING,
TEE_ISOCKET_TLS_ALERT_INAPPROPRIATE_FALLBACK,	325
326	TEE_ISOCKET_TLS_ERROR_CERT_REVOKED,
TEE_ISOCKET_TLS_ALERT_INSUFFICIENT_SECURITY,	326
326	TEE_ISOCKET_TLS_ERROR_CERT_SIGN_VERIFICATION,
TEE_ISOCKET_TLS_ALERT_INTERNAL_ERROR,	325
326	TEE_ISOCKET_TLS_ERROR_CERT_STATUS_UNKNOWN,
TEE_ISOCKET_TLS_ALERT_MISSING_EXTENSION,	326
326	TEE_ISOCKET_TLS_ERROR_CERT_UNKNOWN_CA,
TEE_ISOCKET_TLS_ALERT_NO_CERTIFICATE,	326
326	TEE_ISOCKET_TLS_ERROR_CERT_UNSUPPORTED,
TEE_ISOCKET_TLS_ALERT_NO_RENEGOTIATION,	326
326	TEE_ISOCKET_TLS_ERROR_CRL_PARSING,
TEE_ISOCKET_TLS_ALERT_PROTOCOL_VERSION,	325
326	TEE_ISOCKET_TLS_ERROR_DATA, 324
TEE_ISOCKET_TLS_ALERT_RECORD_OVERFLOW,	TEE_ISOCKET_TLS_ERROR_ECDHE_GEN_KEY,
326	325
TEE_ISOCKET_TLS_ALERT_UNEXPECTED_MSG,	TEE_ISOCKET_TLS_ERROR_ECDHE_SERIALIZING,
326	325

TEE_ISOCKET_TLS_ERROR_ECDHE_SHARED_SECRET, 325
 TEE_ISOCKET_TLS_ERROR_ECDHE_UNSUPPORTED_CURVE, 325
 TEE_ISOCKET_TLS_ERROR_HANDSHAKE_UNEXPECTED_PARAMETER, 326
 TEE_ISOCKET_TLS_ERROR_HANDSHAKE, 324
 TEE_ISOCKET_TLS_ERROR_NO_ALERT_PRESENT, 326
 TEE_ISOCKET_TLS_ERROR_REJECTED_SUITE, 324
 TEE_ISOCKET_TLS_ERROR_UNEXPECTED_MESSAGE, 325
 TEE_ISOCKET_TLS_ERROR_UNSUPPORTED_SUITE, 324
 TEE_ISOCKET_TLS_ERROR_USER_CANCELED, 326
 TEE_ISOCKET_TLS_ERROR_VERSION, 324
 TEE_ISOCKET_TLS_OCSP_CHECK_ADVISORY, 327
 TEE_ISOCKET_TLS_OCSP_CHECK_CLIENT, 327
 TEE_ISOCKET_TLS_OCSP_CHECK_MANDATORY, 327
 TEE_ISOCKET_TLS_OCSP_STATUS_REQUEST_NOT_ALLOWED, 328
 TEE_ISOCKET_TLS_OCSP_STATUS_REQUEST, 328
 TEE_ISOCKET_UDP_API_VERSION, 323
 TEE_ISOCKET_UDP_WARNING_UNKNOWN_OUT_OF_BOUNDS, 323
 TEE_ISOCKET_WARNING_PROTOCOL, 322
 TEE_TLS_CLIENT_CRED_CSC, 327
 TEE_TLS_CLIENT_CRED_NONE, 327
 TEE_TLS_CLIENT_CRED_PDC, 327
 TEE_TLS_SERVER_CRED_CSC, 328
 TEE_TLS_SERVER_CRED_PDC, 328
 TEE_TLS_VERSION_1v2, 328
 TEE_TLS_VERSION_ALL, 328
 TEE_iSocket, 321
 TEE_ipSocket_ipVersion_e, 326
 TEE_tlsSocket_CB_Data, 321
 TEE_tlsSocket_CallbackReasonType_e, 327
 TEE_tlsSocket_ClientCredentialType_e, 327
 TEE_tlsSocket_ExtensionFlags_e, 327
 TEE_tlsSocket_ServerCredentialType_e, 327
 TEE_tlsSocket_StatusRequestType_e, 328
 TEE_tlsSocket_tlsVersion_e, 328
 TEES_IWT_LISTENER_NAME_MAX_LENGTH, 324
 TEES_lwtCloseChannel, 328
 TEES_lwtOpenChannel, 328
 TEES_lwtRead, 329
 TEES_lwtWrite, 329
 tee_spi.h, 376
 tee_ta_destructor.h, 377
 tee_tcpsocket.h, 378
 tee_udpsocket.h, 382
 tee_rcvbuf, 382
 __TEE_SEReaderProperties, 335
 RestrictedParameter, 383
 tees_el2if.h, 384
 tees_extension.h, 384
 tees_hwcrypto_buf.h, 385
 tees_iwt.h, 386
 tees_kdf.h, 386
 tees_rot.h, 387
 tees_secure_object.h, 387
 SAGEssapi.h, 389
 tees_tui.h, 390
 tees_wrapped_with_rek.h, 392
 TeesI_NFC, 300
 TEES_NFCExit, 300
 TEES_NFCHandler, 300
 TEES_NFCInit, 300
 TEES_NFCRead, 301
 TEES_NFCReadWait, 301
 TEES_NFCSetMode, 302
 TEES_NFCSleep, 302
 TEES_NFCWakeUp, 302
 TEES_NFCWrite, 303
 TeesI_smc, 304
 TEES_SMCCommand, 304
 TEES_SMCFin, 305
 TEES_SMCInit, 305
 TeesI_udf, 306
 TEES_UDF_FiniDriver, 306
 TEES_UDF_Init_FS_Driver, 306
 TEES_UDF_InitDriver, 307
 TEES_UDF_RegisterIoctlDesc, 307
 Thread support library API, 130
 MUTEX_STATE_LOCKED, 134
 MUTEX_STATE_UNLOCKED, 134
 PTHREAD_CREATE_DETACHED, 137
 PTHREAD_CREATE_JOINABLE, 137
 PTHREAD_ERRORCHECK_MUTEX_INITIALIZER_NP, 134
 PTHREAD_GUARD_MAX, 134
 PTHREAD_GUARD_MIN, 134
 PTHREAD_MUTEX_DEFAULT, 137
 PTHREAD_MUTEX_DESTROYED, 137
 PTHREAD_MUTEX_ERRORCHECK_NP, 137
 PTHREAD_MUTEX_ERRORCHECK, 137
 PTHREAD_MUTEX_INITIALIZER, 134
 PTHREAD_MUTEX_NORMAL, 137
 PTHREAD_MUTEX_RECURSIVE_NP, 137
 PTHREAD_MUTEX_RECURSIVE, 137
 PTHREAD_ONCE_INIT, 134
 PTHREAD_RECURSIVE_MUTEX_INITIALIZER_NP, 135
 PTHREAD_STACK_MIN, 135
 pthread_attr_destroy, 138
 pthread_attr_getdetachstate, 138
 pthread_attr_getguardsize, 138

- pthread_attr_getstack, 138
- pthread_attr_getstackaddr, 139
- pthread_attr_getstacksize, 139
- pthread_attr_init, 139
- pthread_attr_setdetachstate, 140
- pthread_attr_setguardsize, 140
- pthread_attr_setstack, 140
- pthread_attr_setstackaddr, 141
- pthread_attr_setstacksize, 141
- pthread_attr_t, 136
- pthread_cond_broadcast, 141
- pthread_cond_destroy, 142
- pthread_cond_init, 143
- pthread_cond_signal, 143
- pthread_cond_t, 136
- pthread_cond_timedwait, 144
- pthread_cond_wait, 145
- pthread_condattr_destroy, 146
- pthread_condattr_init, 146
- pthread_condattr_t, 136
- pthread_create, 146
- pthread_detach, 147
- pthread_equal, 148
- pthread_exit, 148
- pthread_getattr_np, 148
- pthread_getspecific, 149
- pthread_impl_t, 136
- pthread_join, 150
- pthread_key_create, 151
- pthread_key_delete, 152
- pthread_key_t, 136
- pthread_kill, 153
- pthread_mutex_destroy, 153
- pthread_mutex_init, 154
- pthread_mutex_lock, 155
- pthread_mutex_t, 136
- pthread_mutex_trylock, 155
- pthread_mutex_unlock, 156
- pthread_mutexattr_destroy, 157
- pthread_mutexattr_gettype, 157
- pthread_mutexattr_init, 158
- pthread_mutexattr_settype, 158
- pthread_mutexattr_t, 136
- pthread_once, 159
- pthread_once_t, 136
- pthread_self, 160
- pthread_setspecific, 160
- pthread_sigmask, 135
- pthread_t, 137
- time
 - Auxiliary API, 286
- time.h, 393
- time_t
 - Auxiliary API, 233
- timeadd
 - Auxiliary API, 287
- timespec_to_ms
 - Auxiliary API, 287
- timespec_to_nsec
 - Auxiliary API, 287
- timesub
 - Auxiliary API, 288
- tm, 201
- tolower
 - Auxiliary API, 288
- toupper
 - Auxiliary API, 288
- tpl
 - ioctl_desc, 339
- truncate
 - fops, 338
- Trusted user interface, 97
 - MAX_FONT_SIZE, 103
 - MIN_FONT_SIZE, 103
 - MT_INFO_MAX_EVENT, 104
 - TEE_TEMPRARY_CLEAR_MODE, 104
 - TEE_TUI_ALPHANUMERICAL, 105
 - TEE_TUI_CANCEL, 104
 - TEE_TUI_CLEAR_MODE, 104
 - TEE_TUI_CORRECTION, 104
 - TEE_TUI_HIDDEN_MODE, 104
 - TEE_TUI_LANDSCAPE, 105
 - TEE_TUI_NEXT, 104
 - TEE_TUI_NO_SOURCE, 105
 - TEE_TUI_NUMBER_BUTTON_TYPES, 104
 - TEE_TUI_NUMERICAL, 105
 - TEE_TUI_OBJECT_SOURCE, 105
 - TEE_TUI_OK, 104
 - TEE_TUI_PORTRAIT, 105
 - TEE_TUI_PREVIOUS, 104
 - TEE_TUI_REF_SOURCE, 105
 - TEE_TUI_VALIDATE, 104
 - TEE_TUIButtonType, 104
 - TEE_TUICheckTextFormat, 106
 - TEE_TUICloseSession, 107
 - TEE_TUIDisplayScreen, 107
 - TEE_TUIEntryFieldMode, 104
 - TEE_TUIEntryFieldType, 104
 - TEE_TUIGetScreenInfo, 109
 - TEE_TUIImageSource, 105
 - TEE_TUIInitSession, 110
 - TEE_TUIScreenOrientation, 105
 - TEES_ERROR_NOT_IMPLEMENTED, 106
 - TEES_ERROR_NOT_SUPPORTED, 106
 - TEES_ERROR_SHORT_BUFFER, 106
 - TEES_ERROR_TUI_BAD_FORMAT, 106
 - TEES_ERROR_TUI_BAD_STATE, 106
 - TEES_ERROR_TUI_BUSY, 106
 - TEES_ERROR_TUI_CANCEL, 106
 - TEES_ERROR_TUI_GENERIC, 106
 - TEES_ERROR_TUI_INVALID_PARAM, 106
 - TEES_ERROR_TUI_OUT_OF_MEMORY, 106
 - TEES_ERROR_TUI_TIMEOUT, 106
 - TEES_FALSE, 105
 - TEES_Result, 105
 - TEES_SUCCESS, 106

- TEES_TOUCH_PRESSED, [106](#)
- TEES_TOUCH_RELEASED, [106](#)
- TEES_TRUE, [105](#)
- TEES_TUICheckTextFormat, [110](#)
- TEES_TUICloseSession, [111](#)
- TEES_TUIDrawBuffer, [111](#)
- TEES_TUIDrawImage, [112](#)
- TEES_TUIDrawImageFromBuff, [112](#)
- TEES_TUIDrawImageToBuff, [113](#)
- TEES_TUIGetBuffer, [114](#)
- TEES_TUIGetMTEvent, [114](#)
- TEES_TUIGetScreenInfo, [115](#)
- TEES_TUIGetTouchEvent, [115](#)
- TEES_TUIOpenSession, [116](#)
- TEES_TUIPrintString, [116](#)
- TEES_TUIRefreshScreen, [117](#)
- TEES_TUITouchTypes, [106](#)
- TEES_bool, [105](#)
- tui.h, [394](#)
- type
 - desc_atom, [336](#)
 - ioctl_desc, [339](#)
- UCHAR_MAX
 - Auxiliary API, [231](#)
- UINT_MAX
 - Auxiliary API, [231](#)
- ULLONG_MAX
 - Auxiliary API, [231](#)
- ULONG_MAX
 - Auxiliary API, [231](#)
- USHRT_MAX
 - Auxiliary API, [231](#)
- UUID_STRING_LEN
 - Auxiliary API, [232](#)
- udf.h, [395](#)
- uid_t
 - Auxiliary API, [233](#)
- unistd.h, [396](#)
- unlink
 - Auxiliary API, [289](#)
 - fops, [338](#)
- unsetenv
 - Auxiliary API, [289](#)
- usr_drv_info, [341](#)
 - fops, [341](#)
 - handle, [341](#)
 - priv, [341](#)
- uuid/uuid.h, [397](#)
- uuid_clear
 - Auxiliary API, [289](#)
- uuid_compare
 - Auxiliary API, [290](#)
- uuid_generate
 - Auxiliary API, [290](#)
- uuid_generate_time
 - Auxiliary API, [232](#)
- uuid_is_null
 - Auxiliary API, [290](#)
- uuid_parse
 - Auxiliary API, [291](#)
- uuid_t
 - Auxiliary API, [233](#)
- uuid_unparse
 - Auxiliary API, [291](#)
- uuid_unparse_lower
 - Auxiliary API, [232](#)
- uuid_unparse_upper
 - Auxiliary API, [291](#)
- vasprintf
 - Auxiliary API, [292](#)
- vasprintf_s
 - Auxiliary API, [292](#)
- vfprintf
 - Auxiliary API, [293](#)
- vsnprintf
 - Auxiliary API, [293](#)
- vsnprintf_s
 - Auxiliary API, [294](#)
- vsprintf
 - Auxiliary API, [294](#)
- vsprintf_s
 - Auxiliary API, [295](#)
- vsscanf
 - Auxiliary API, [296](#)
- vsscanf_s
 - Auxiliary API, [296](#)
- wrapped_wkth_rek_t, [47](#)
- write
 - Auxiliary API, [297](#)
 - fops, [338](#)