

TCNOpen TRDP Light
V2.0.2

Generated by Doxygen 1.8.13

Contents

1	The TRDP Light Library API Specification	1
1.1	General Information	1
1.1.1	Purpose	1
1.1.2	Scope	1
1.1.3	Related documents	1
1.1.4	Abbreviations and Definitions	1
1.2	Terminology	2
1.3	Use Cases	2
1.4	Conventions of the API	5
2	Data Structure Index	7
2.1	Data Structures	7
3	File Index	9
3.1	File List	9
4	Data Structure Documentation	13
4.1	DNS_HEADER Struct Reference	13
4.1.1	Detailed Description	13
4.2	GNU_PACKED Struct Reference	13
4.2.1	Detailed Description	21
4.2.2	Field Documentation	22
4.2.2.1	callBack	22
4.2.2.2	comId	22
4.2.2.3	confVehCnt	23

4.2.2.4	confVehList	23
4.2.2.5	cstList	23
4.2.2.6	cstUUID	23
4.2.2.7	datasetLength	23
4.2.2.8	defQos	24
4.2.2.9	defTtl	24
4.2.2.10	destAddr	24
4.2.2.11	deviceName	24
4.2.2.12	etbld	24
4.2.2.13	etbTopoCnt	24
4.2.2.14	filterAddr	25
4.2.2.15	inhibit	25
4.2.2.16	isLead	25
4.2.2.17	leadDir	25
4.2.2.18	leadVehOfCst	25
4.2.2.19	lifesign	25
4.2.2.20	msgType	26
4.2.2.21	numCrcErr	26
4.2.2.22	numMissed	26
4.2.2.23	numProtErr	26
4.2.2.24	numRcv	26
4.2.2.25	numRecv	26
4.2.2.26	numSend	27
4.2.2.27	numTopoErr	27
4.2.2.28	opCstList	27
4.2.2.29	opTrnDirState	27
4.2.2.30	opTrnTopoCnt	27
4.2.2.31	opVehList	28
4.2.2.32	ownOpCstNo	28
4.2.2.33	protocolVersion	28

4.2.2.34	reserved01 [1/2]	28
4.2.2.35	reserved01 [2/2]	28
4.2.2.36	reserved02 [1/2]	28
4.2.2.37	reserved02 [2/2]	29
4.2.2.38	reserved03	29
4.2.2.39	reserved04	29
4.2.2.40	reserved06	29
4.2.2.41	safetyTrail	29
4.2.2.42	serviceEntry	29
4.2.2.43	timeout	30
4.2.2.44	toBehav	30
4.2.2.45	trnCstNo	30
4.2.2.46	trnDirState	30
4.2.2.47	trnId	30
4.2.2.48	trnNetDir	30
4.2.2.49	trnOperator	31
4.2.2.50	trnTopoCnt	31
4.2.2.51	trnVehNo	31
4.2.2.52	vehId	31
4.2.2.53	vehOrient	31
4.2.2.54	version	32
4.3	hp_slot Struct Reference	32
4.3.1	Detailed Description	33
4.4	hp_slots Struct Reference	33
4.4.1	Detailed Description	34
4.5	PD_ELE Struct Reference	34
4.5.1	Detailed Description	35
4.5.2	Field Documentation	35
4.5.2.1	pFrame	36
4.6	service_info Struct Reference	36

4.6.1	Detailed Description	36
4.6.2	Field Documentation	37
4.6.2.1	fctDev	37
4.7	srv_info_req Struct Reference	37
4.7.1	Detailed Description	37
4.8	TAU_MARSHALL_INFO_T Struct Reference	38
4.8.1	Detailed Description	38
4.9	TCN_URI Struct Reference	38
4.9.1	Detailed Description	39
4.10	TRDP_CLTR_CST_INFO_T Struct Reference	39
4.10.1	Detailed Description	39
4.11	TRDP_COM_PARAM_T Struct Reference	39
4.11.1	Detailed Description	40
4.12	TRDP_COMID_DSID_MAP_T Struct Reference	40
4.12.1	Detailed Description	40
4.13	TRDP_CONSIST_INFO_T Struct Reference	41
4.13.1	Detailed Description	42
4.13.2	Field Documentation	42
4.13.2.1	cstId	42
4.13.2.2	cstOwner	42
4.14	TRDP_DATASET Struct Reference	43
4.14.1	Detailed Description	43
4.15	TRDP_DATASET_ELEMENT_T Struct Reference	44
4.15.1	Detailed Description	44
4.16	TRDP_DBG_CONFIG_T Struct Reference	45
4.16.1	Detailed Description	45
4.17	TRDP_DNS_REPLY Struct Reference	45
4.17.1	Detailed Description	46
4.17.2	Field Documentation	46
4.17.2.1	tcnUriCnt	46

4.18 TRDP_DNS_REQUEST Struct Reference	47
4.18.1 Detailed Description	47
4.18.2 Field Documentation	47
4.18.2.1 tcnUriCnt	48
4.19 TRDP_ETB_INFO_T Struct Reference	48
4.19.1 Detailed Description	48
4.19.2 Field Documentation	48
4.19.2.1 cnCnt	48
4.20 TRDP_FUNCTION_INFO_T Struct Reference	49
4.20.1 Detailed Description	49
4.20.2 Field Documentation	49
4.20.2.1 cnId	49
4.20.2.2 cstVehNo	49
4.20.2.3 etbId	50
4.20.2.4 fctId	50
4.21 TRDP_HANDLE Struct Reference	50
4.21.1 Detailed Description	51
4.22 TRDP_IDX_TABLE_T Struct Reference	51
4.22.1 Detailed Description	51
4.22.2 Field Documentation	51
4.22.2.1 maxNoOfExtPublishers	52
4.22.2.2 maxNoOfHighCatPublishers	52
4.22.2.3 maxNoOfHighCatSubscriptions	52
4.22.2.4 maxNoOfLowCatPublishers	52
4.22.2.5 maxNoOfLowCatSubscriptions	52
4.22.2.6 maxNoOfMidCatPublishers	52
4.22.2.7 maxNoOfMidCatSubscriptions	53
4.23 TRDP_MARSHALL_CONFIG_T Struct Reference	53
4.23.1 Detailed Description	53
4.24 TRDP_MD_CONFIG_T Struct Reference	54

4.24.1 Detailed Description	54
4.25 TRDP_MD_INFO_T Struct Reference	55
4.25.1 Detailed Description	56
4.26 TRDP_MEM_CONFIG_T Struct Reference	56
4.26.1 Detailed Description	56
4.27 TRDP_PD_CONFIG_T Struct Reference	57
4.27.1 Detailed Description	57
4.28 TRDP_PD_INFO_T Struct Reference	57
4.28.1 Detailed Description	58
4.29 TRDP_PROCESS_CONFIG_T Struct Reference	58
4.29.1 Detailed Description	59
4.30 TRDP_PROP_T Struct Reference	59
4.30.1 Detailed Description	59
4.31 TRDP_SDT_PAR_T Struct Reference	60
4.31.1 Detailed Description	60
4.32 TRDP_SEQ_CNT_ENTRY_T Struct Reference	60
4.32.1 Detailed Description	61
4.33 TRDP_SESSION Struct Reference	61
4.33.1 Detailed Description	62
4.34 TRDP_SOCKET_TCP Struct Reference	62
4.34.1 Detailed Description	63
4.35 TRDP_SOCKETS Struct Reference	63
4.35.1 Detailed Description	64
4.35.2 Field Documentation	64
4.35.2.1 usage	64
4.36 TRDP_VEHICLE_INFO_T Struct Reference	65
4.36.1 Detailed Description	65
4.36.2 Field Documentation	65
4.36.2.1 vehId	66
4.37 TRDP_XML_DOC_HANDLE_T Struct Reference	66
4.37.1 Detailed Description	66
4.38 VOS SOCK_OPT_T Struct Reference	66
4.38.1 Detailed Description	67
4.39 VOS_VERSION_T Struct Reference	67
4.39.1 Detailed Description	67

5	File Documentation	69
5.1	iec61375-2-3.h File Reference	69
5.1.1	Detailed Description	73
5.1.2	Macro Definition Documentation	73
5.1.2.1	ETB_CTRL_COMID	73
5.1.2.2	TRDP_ETBCTRL_DSID	74
5.1.2.3	TRDP_MAX_FILE_NAME_LEN	74
5.1.2.4	TRDP_MAX_LABEL_LEN	74
5.1.2.5	TRDP_MAX_MD_DATA_SIZE	74
5.1.2.6	TRDP_MAX_URI_HOST_LEN	74
5.1.2.7	TRDP_MAX_URI_LEN	74
5.1.2.8	TRDP_MAX_URI_USER_LEN	75
5.1.2.9	TRDP_MD_DEFAULT_REPLY_TIMEOUT	75
5.1.2.10	TRDP_MD_INFINITE_TIME	75
5.1.2.11	TRDP_MIN_PD_HEADER_SIZE	75
5.1.2.12	TRDP_MSG_PD	75
5.1.2.13	TRDP_PD_UDP_PORT	75
5.1.2.14	TRDP_PROCESS_DEFAULT_CYCLE_TIME	76
5.1.2.15	TRDP_USR_URI_SIZE	76
5.1.2.16	TTDB_NET_DIR_REQ_COMID	76
5.1.2.17	TTDB_OP_DIR_INFO_COMID	76
5.1.2.18	TTDB_STAT_CST_REQ_COMID	76
5.1.2.19	TTDB_TRN_DIR_REQ_COMID	76
5.2	tau_cstinfo.c File Reference	77
5.2.1	Detailed Description	78
5.2.2	Function Documentation	78
5.2.2.1	cstInfoGetPropSize()	78
5.3	tau_ctrl.c File Reference	79
5.3.1	Detailed Description	80
5.3.2	Function Documentation	81

5.3.2.1	tau_getEcspStat()	81
5.3.2.2	tau_initEcspCtrl()	81
5.3.2.3	tau_requestEcspConfirm()	82
5.3.2.4	tau_setEcspCtrl()	82
5.3.2.5	tau_terminateEcspCtrl()	83
5.4	tau_ctrl.h File Reference	83
5.4.1	Detailed Description	85
5.4.2	Function Documentation	86
5.4.2.1	tau_getEcspStat()	86
5.4.2.2	tau_initEcspCtrl()	86
5.4.2.3	tau_requestEcspConfirm()	87
5.4.2.4	tau_setEcspCtrl()	87
5.4.2.5	tau_terminateEcspCtrl()	88
5.5	tau_ctrl_types.h File Reference	88
5.5.1	Detailed Description	90
5.6	tau_dnr.c File Reference	91
5.6.1	Detailed Description	92
5.6.2	Function Documentation	93
5.6.2.1	tau_addr2Uri()	93
5.6.2.2	tau_delInitDnr()	93
5.6.2.3	tau_DNRstatus()	94
5.6.2.4	tau_getOwnAddr()	94
5.6.2.5	tau_initDnr()	94
5.6.2.6	tau_uri2Addr()	95
5.7	tau_dnr.h File Reference	96
5.7.1	Detailed Description	98
5.7.2	Enumeration Type Documentation	98
5.7.2.1	TRDP_DNR_OPTS	98
5.7.3	Function Documentation	98
5.7.3.1	tau_addr2Uri()	98

5.7.3.2	tau_delInitDnr()	99
5.7.3.3	tau_DNRstatus()	100
5.7.3.4	tau_getOwnAddr()	100
5.7.3.5	tau_initDnr()	101
5.7.3.6	tau_uri2Addr()	102
5.8	tau_dnr_types.h File Reference	103
5.8.1	Detailed Description	104
5.9	tau_marshall.c File Reference	105
5.9.1	Detailed Description	106
5.9.2	Function Documentation	106
5.9.2.1	tau_calcDatasetSize()	106
5.9.2.2	tau_calcDatasetSizeByComId()	107
5.9.2.3	tau_initMarshall()	108
5.9.2.4	tau_marshall()	108
5.9.2.5	tau_marshallDs()	109
5.9.2.6	tau_unmarshall()	110
5.9.2.7	tau_unmarshallDs()	110
5.10	tau_marshall.h File Reference	111
5.10.1	Detailed Description	113
5.10.2	Function Documentation	113
5.10.2.1	tau_calcDatasetSize()	113
5.10.2.2	tau_calcDatasetSizeByComId()	114
5.10.2.3	tau_initMarshall()	115
5.10.2.4	tau_marshall()	116
5.10.2.5	tau_marshallDs()	117
5.10.2.6	tau_unmarshall()	118
5.10.2.7	tau_unmarshallDs()	120
5.11	tau_so_if.c File Reference	121
5.11.1	Detailed Description	123
5.11.2	Function Documentation	123

5.11.2.1	tau_addService()	123
5.11.2.2	tau_delService()	124
5.11.2.3	tau_freeServicesList()	124
5.11.2.4	tau_getServicesList()	124
5.11.2.5	tau_updService()	125
5.12	tau_so_if.h File Reference	126
5.12.1	Detailed Description	127
5.12.2	Function Documentation	128
5.12.2.1	tau_addService()	128
5.12.2.2	tau_delService()	128
5.12.2.3	tau_freeServicesList()	129
5.12.2.4	tau_getServicesList()	129
5.12.2.5	tau_updService()	130
5.13	tau_tti.c File Reference	130
5.13.1	Detailed Description	132
5.13.2	Macro Definition Documentation	133
5.13.2.1	TTI_CACHED_CONSISTS	133
5.13.3	Function Documentation	133
5.13.3.1	tau_delInitTTI()	133
5.13.3.2	tau_getCstFctCnt()	133
5.13.3.3	tau_getCstFctInfo()	134
5.13.3.4	tau_getCstInfo()	134
5.13.3.5	tau_getCstVehCnt()	135
5.13.3.6	tau_getOpTrDirectory()	135
5.13.3.7	tau_getOpTrnDirectoryStatusInfo()	136
5.13.3.8	tau_getOwnIds()	136
5.13.3.9	tau_getOwnOpCstNo()	137
5.13.3.10	tau_getOwnTrnCstNo()	137
5.13.3.11	tau_getStaticCstInfo()	137
5.13.3.12	tau_getTrDirectory()	138

5.13.3.13 tau_getTrnCstCnt()	138
5.13.3.14 tau_getTrnVehCnt()	139
5.13.3.15 tau_getTTI()	139
5.13.3.16 tau_getVehInfo()	139
5.13.3.17 tau_getVehOrient()	140
5.13.3.18 tau_initTTIaccess()	140
5.14 tau_tti.h File Reference	141
5.14.1 Detailed Description	144
5.14.2 Function Documentation	144
5.14.2.1 tau_delInitTTI()	144
5.14.2.2 tau_getCstFctCnt()	145
5.14.2.3 tau_getCstFctInfo()	145
5.14.2.4 tau_getCstInfo()	146
5.14.2.5 tau_getCstVehCnt()	146
5.14.2.6 tau_getOpTrDirectory()	147
5.14.2.7 tau_getOpTrnDirectoryStatusInfo()	148
5.14.2.8 tau_getOwnIds()	148
5.14.2.9 tau_getOwnOpCstNo()	149
5.14.2.10 tau_getOwnTrnCstNo()	149
5.14.2.11 tau_getStaticCstInfo()	150
5.14.2.12 tau_getTrDirectory()	150
5.14.2.13 tau_getTrnCstCnt()	151
5.14.2.14 tau_getTrnVehCnt()	151
5.14.2.15 tau_getTTI()	152
5.14.2.16 tau_getVehInfo()	152
5.14.2.17 tau_getVehOrient()	153
5.14.2.18 tau_initTTIaccess()	154
5.15 tau_tti_types.h File Reference	155
5.15.1 Detailed Description	157
5.16 tau_xml.c File Reference	158

5.16.1 Detailed Description	159
5.16.2 Macro Definition Documentation	160
5.16.2.1 TRDP_SDT_DEFAULT_CMTHR	160
5.16.2.2 TRDP_SDT_DEFAULT_LMIMAX	160
5.16.3 Function Documentation	160
5.16.3.1 tau_freeTelegrams()	160
5.16.3.2 tau_freeXmlDatasetConfig()	160
5.16.3.3 tau_freeXmlDoc()	161
5.16.3.4 tau_prepareXmlDoc()	161
5.16.3.5 tau_prepareXmlMem()	162
5.16.3.6 tau_readXmlDatasetConfig()	162
5.16.3.7 tau_readXmlDeviceConfig()	163
5.16.3.8 tau_readXmlInterfaceConfig()	163
5.16.3.9 tau_readXmlMappedDeviceConfig()	164
5.16.3.10 tau_readXmlMappedDevices()	164
5.16.3.11 tau_readXmlMappedInterfaceConfig()	165
5.16.3.12 tau_readXmlServiceConfig()	165
5.17 tau_xml.h File Reference	166
5.17.1 Detailed Description	168
5.17.2 Macro Definition Documentation	169
5.17.2.1 TRDP_DBG_DEFAULT	169
5.17.3 Enumeration Type Documentation	169
5.17.3.1 TRDP_EXCHG_OPTION_T	169
5.17.4 Function Documentation	169
5.17.4.1 tau_freeTelegrams()	170
5.17.4.2 tau_freeXmlDatasetConfig()	170
5.17.4.3 tau_freeXmlDoc()	170
5.17.4.4 tau_prepareXmlDoc()	171
5.17.4.5 tau_prepareXmlMem()	171
5.17.4.6 tau_readXmlDatasetConfig()	172

5.17.4.7	tau_readXmlDeviceConfig()	172
5.17.4.8	tau_readXmlInterfaceConfig()	173
5.17.4.9	tau_readXmlMappedDeviceConfig()	174
5.17.4.10	tau_readXmlMappedDevices()	174
5.17.4.11	tau_readXmlMappedInterfaceConfig()	176
5.17.4.12	tau_readXmlServiceConfig()	176
5.18	tlc_if.c File Reference	178
5.18.1	Detailed Description	180
5.18.2	Function Documentation	181
5.18.2.1	tlc_closeSession()	181
5.18.2.2	tlc_configSession()	181
5.18.2.3	tlc_getETBTopoCount()	182
5.18.2.4	tlc_getInterval()	182
5.18.2.5	tlc_getOpTrainTopoCount()	183
5.18.2.6	tlc_getOwnIpAddress()	183
5.18.2.7	tlc_getVersion()	183
5.18.2.8	tlc_getVersionString()	184
5.18.2.9	tlc_init()	184
5.18.2.10	tlc_openSession()	185
5.18.2.11	tlc_presetIndexSession()	185
5.18.2.12	tlc_process()	186
5.18.2.13	tlc_reinitSession()	186
5.18.2.14	tlc_setETBTopoCount()	187
5.18.2.15	tlc_setOpTrainTopoCount()	187
5.18.2.16	tlc_terminate()	188
5.18.2.17	tlc_updateSession()	188
5.18.2.18	trdp_getAccess()	189
5.18.2.19	trdp_isValidSession()	189
5.18.2.20	trdp_releaseAccess()	189
5.18.2.21	trdp_sessionQueue()	190

5.19	tlc_if.h File Reference	190
5.19.1	Detailed Description	192
5.19.2	Function Documentation	192
5.19.2.1	trdp_isValidSession()	192
5.19.2.2	trdp_sessionQueue()	192
5.20	tlm_if.c File Reference	193
5.20.1	Detailed Description	194
5.20.2	Function Documentation	195
5.20.2.1	tlm_abortSession()	195
5.20.2.2	tlm_addListener()	195
5.20.2.3	tlm_confirm()	196
5.20.2.4	tlm_delListener()	197
5.20.2.5	tlm_getInterval()	197
5.20.2.6	tlm_notify()	198
5.20.2.7	tlm_process()	199
5.20.2.8	tlm_readdListener()	199
5.20.2.9	tlm_reply()	200
5.20.2.10	tlm_replyQuery()	201
5.20.2.11	tlm_request()	202
5.21	tlp_if.c File Reference	203
5.21.1	Detailed Description	205
5.21.2	Function Documentation	205
5.21.2.1	tlp_get()	206
5.21.2.2	tlp_getInterval()	206
5.21.2.3	tlp_getRedundant()	207
5.21.2.4	tlp_processReceive()	207
5.21.2.5	tlp_processSend()	208
5.21.2.6	tlp_publish()	208
5.21.2.7	tlp_put()	209
5.21.2.8	tlp_putImmediate()	210

5.21.2.9	tlp_republish()	210
5.21.2.10	tlp_request()	211
5.21.2.11	tlp_resubscribe()	212
5.21.2.12	tlp_setRedundant()	213
5.21.2.13	tlp_subscribe()	213
5.21.2.14	tlp_unpublish()	214
5.21.2.15	tlp_unsubscribe()	215
5.22	trdp_dllmain.c File Reference	215
5.22.1	Detailed Description	215
5.23	trdp_if_light.h File Reference	216
5.23.1	Detailed Description	220
5.23.2	Function Documentation	220
5.23.2.1	tlc_closeSession()	220
5.23.2.2	tlc_configSession()	220
5.23.2.3	tlc_getETBTopoCount()	221
5.23.2.4	tlc_getInterval()	221
5.23.2.5	tlc_getJoinStatistics()	222
5.23.2.6	tlc_getOpTrainTopoCount()	223
5.23.2.7	tlc_getOwnIpAddress()	223
5.23.2.8	tlc_getPubStatistics()	223
5.23.2.9	tlc_getRedStatistics()	224
5.23.2.10	tlc_getStatistics()	224
5.23.2.11	tlc_getSubsStatistics()	225
5.23.2.12	tlc_getVersion()	225
5.23.2.13	tlc_getVersionString()	226
5.23.2.14	tlc_init()	226
5.23.2.15	tlc_openSession()	226
5.23.2.16	tlc_presetIndexSession()	227
5.23.2.17	tlc_process()	228
5.23.2.18	tlc_reinitSession()	228

5.23.2.19 tlc_resetStatistics()	229
5.23.2.20 tlc_setETBTopoCount()	229
5.23.2.21 tlc_setOpTrainTopoCount()	230
5.23.2.22 tlc_terminate()	230
5.23.2.23 tlc_updateSession()	230
5.23.2.24 tlm_abortSession()	231
5.23.2.25 tlm_addListener()	231
5.23.2.26 tlm_confirm()	232
5.23.2.27 tlm_delListener()	233
5.23.2.28 tlm_getInterval()	233
5.23.2.29 tlm_notify()	234
5.23.2.30 tlm_process()	235
5.23.2.31 tlm_readdListener()	235
5.23.2.32 tlm_reply()	236
5.23.2.33 tlm_replyQuery()	237
5.23.2.34 tlm_request()	238
5.23.2.35 tlp_get()	239
5.23.2.36 tlp_getInterval()	240
5.23.2.37 tlp_getRedundant()	240
5.23.2.38 tlp_processReceive()	241
5.23.2.39 tlp_processSend()	241
5.23.2.40 tlp_publish()	242
5.23.2.41 tlp_put()	243
5.23.2.42 tlp_putImmediate()	243
5.23.2.43 tlp_republish()	244
5.23.2.44 tlp_request()	245
5.23.2.45 tlp_resubscribe()	246
5.23.2.46 tlp_setRedundant()	246
5.23.2.47 tlp_subscribe()	248
5.23.2.48 tlp_unpublish()	249

5.23.2.49	tlp_unsubscribe()	249
5.24	trdp_mdcom.c File Reference	250
5.24.1	Detailed Description	251
5.24.2	Function Documentation	251
5.24.2.1	trdp_mdCall()	252
5.24.2.2	trdp_mdCheckListenSocks()	253
5.24.2.3	trdp_mdCheckPending()	253
5.24.2.4	trdp_mdCheckTimeouts()	253
5.24.2.5	trdp_mdConfirm()	254
5.24.2.6	trdp_mdFreeSession()	254
5.24.2.7	trdp_mdGetTCPSocket()	255
5.24.2.8	trdp_mdReply()	255
5.24.2.9	trdp_mdSend()	256
5.25	trdp_mdcom.h File Reference	256
5.25.1	Detailed Description	258
5.25.2	Function Documentation	258
5.25.2.1	trdp_mdCall()	259
5.25.2.2	trdp_mdCheckListenSocks()	260
5.25.2.3	trdp_mdCheckPending()	260
5.25.2.4	trdp_mdCheckTimeouts()	260
5.25.2.5	trdp_mdConfirm()	261
5.25.2.6	trdp_mdFreeSession()	261
5.25.2.7	trdp_mdGetTCPSocket()	262
5.25.2.8	trdp_mdReply()	262
5.25.2.9	trdp_mdSend()	263
5.26	trdp_pdcom.c File Reference	263
5.26.1	Detailed Description	265
5.26.2	Function Documentation	265
5.26.2.1	trdp_handleTimeout()	265
5.26.2.2	trdp_pdCheck()	266

5.26.2.3	<code>trdp_pdCheckListenSocks()</code>	266
5.26.2.4	<code>trdp_pdCheckPending()</code>	267
5.26.2.5	<code>trdp_pdDistribute()</code>	267
5.26.2.6	<code>trdp_pdHandleTimeOuts()</code>	268
5.26.2.7	<code>trdp_pdInit()</code>	269
5.26.2.8	<code>trdp_pdPut()</code>	269
5.26.2.9	<code>trdp_pdReceive()</code>	270
5.26.2.10	<code>trdp_pdSend()</code>	270
5.26.2.11	<code>trdp_pdSendElement()</code>	271
5.26.2.12	<code>trdp_pdSendImmediate()</code>	272
5.26.2.13	<code>trdp_pdSendQueued()</code>	272
5.26.2.14	<code>trdp_pdUpdate()</code>	273
5.27	<code>trdp_pdcom.h</code> File Reference	273
5.27.1	Detailed Description	275
5.27.2	Function Documentation	276
5.27.2.1	<code>trdp_handleTimeout()</code>	276
5.27.2.2	<code>trdp_pdCheck()</code>	276
5.27.2.3	<code>trdp_pdCheckListenSocks()</code>	277
5.27.2.4	<code>trdp_pdCheckPending()</code>	277
5.27.2.5	<code>trdp_pdDistribute()</code>	278
5.27.2.6	<code>trdp_pdHandleTimeOuts()</code>	278
5.27.2.7	<code>trdp_pdInit()</code>	279
5.27.2.8	<code>trdp_pdPut()</code>	280
5.27.2.9	<code>trdp_pdReceive()</code>	280
5.27.2.10	<code>trdp_pdSend()</code>	281
5.27.2.11	<code>trdp_pdSendElement()</code>	281
5.27.2.12	<code>trdp_pdSendImmediate()</code>	282
5.27.2.13	<code>trdp_pdSendQueued()</code>	283
5.27.2.14	<code>trdp_pdUpdate()</code>	284
5.28	<code>trdp_pdindex.c</code> File Reference	284

5.28.1 Detailed Description	285
5.29 trdp_pdindex.h File Reference	285
5.29.1 Detailed Description	287
5.29.2 Macro Definition Documentation	287
5.29.2.1 TRDP_DEFAULT_INDEX_SIZES	287
5.30 trdp_private.h File Reference	288
5.30.1 Detailed Description	290
5.30.2 Macro Definition Documentation	291
5.30.2.1 TRDP_MAX_PD_SOCKET_CNT	291
5.30.3 Enumeration Type Documentation	291
5.30.3.1 TRDP_MD_ELE_ST_T	291
5.30.3.2 TRDP SOCK_TYPE_T	291
5.31 trdp_serviceRegistry.h File Reference	292
5.31.1 Detailed Description	295
5.31.2 Macro Definition Documentation	295
5.31.2.1 SOA_SAME_SERVICEID	295
5.31.2.2 SRM_SERVICE_READ_REQ_COMID	295
5.31.2.3 SRM_SRVINFO_NOTIFY_COMID	295
5.32 trdp_stats.c File Reference	296
5.32.1 Detailed Description	297
5.32.2 Function Documentation	297
5.32.2.1 tlc_getJoinStatistics()	297
5.32.2.2 tlc_getPubStatistics()	298
5.32.2.3 tlc_getRedStatistics()	298
5.32.2.4 tlc_getStatistics()	299
5.32.2.5 tlc_getSubsStatistics()	299
5.32.2.6 tlc_resetStatistics()	300
5.32.2.7 trdp_initStats()	300
5.32.2.8 trdp_pdPrepareStats()	301
5.32.2.9 trdp_UpdateStats()	302

5.33 trdp_stats.h File Reference	302
5.33.1 Detailed Description	303
5.33.2 Function Documentation	303
5.33.2.1 trdp_initStats()	303
5.33.2.2 trdp_pdPrepareStats()	304
5.34 trdp_tsn_def.h File Reference	305
5.34.1 Detailed Description	305
5.34.2 Macro Definition Documentation	306
5.34.2.1 TRDP_MIN_PD2_HEADER_SIZE	306
5.34.2.2 TRDP_MSG_TSN_PD	306
5.34.2.3 TRDP_PD_DEFAULT_QOS	306
5.35 trdp_types.h File Reference	306
5.35.1 Detailed Description	311
5.35.2 Macro Definition Documentation	312
5.35.2.1 TRDP_FLAGS_DEFAULT	312
5.35.3 Typedef Documentation	312
5.35.3.1 TRDP_IP_ADDR_T	312
5.35.3.2 TRDP_MARSHALL_T	312
5.35.3.3 TRDP_MD_CALLBACK_T	313
5.35.3.4 TRDP_PD_CALLBACK_T	313
5.35.3.5 TRDP_PRINT_DBG_T	313
5.35.3.6 TRDP_TIME_T	313
5.35.3.7 TRDP_UNMARSHALL_T	314
5.35.4 Enumeration Type Documentation	314
5.35.4.1 TRDP_DATA_TYPE_T	314
5.35.4.2 TRDP_ERR_T	315
5.35.4.3 TRDP_RED_STATE_T	316
5.35.4.4 TRDP_REPLY_STATUS_T	316
5.35.4.5 TRDP_TO_BEHAVIOR_T	316
5.36 trdp_utils.c File Reference	317

5.36.1 Detailed Description	318
5.36.2 Function Documentation	319
5.36.2.1 printSocketUsage()	319
5.36.2.2 trdp_checkSequenceCounter()	319
5.36.2.3 trdp_findMCjoins()	320
5.36.2.4 trdp_findSubAddr()	320
5.36.2.5 trdp_getSeqCnt()	321
5.36.2.6 trdp_initSockets()	321
5.36.2.7 trdp_isAddressed()	322
5.36.2.8 trdp_isInlPrange()	322
5.36.2.9 trdp_packetSizeMD()	322
5.36.2.10 trdp_packetSizePD()	323
5.36.2.11 trdp_queueAppLast()	323
5.36.2.12 trdp_queueDelElement()	323
5.36.2.13 trdp_queueFindComId()	324
5.36.2.14 trdp_queueFindExistingSub()	324
5.36.2.15 trdp_queueFindPubAddr()	325
5.36.2.16 trdp_queueFindSubAddr()	325
5.36.2.17 trdp_queueInsFirst()	326
5.36.2.18 trdp_releaseSocket()	326
5.36.2.19 trdp_requestSocket()	326
5.36.2.20 trdp_resetSequenceCounter()	327
5.36.2.21 trdp_SockAddJoin()	328
5.36.2.22 trdp_SockDelJoin()	328
5.36.2.23 trdp_SockIsJoined()	328
5.36.2.24 trdp_validTopoCounters()	329
5.37 trdp_utils.h File Reference	329
5.37.1 Detailed Description	331
5.37.2 Function Documentation	332
5.37.2.1 printSocketUsage()	332

5.37.2.2	<code>trdp_checkSequenceCounter()</code>	332
5.37.2.3	<code>trdp_findMCjoins()</code>	333
5.37.2.4	<code>trdp_findSubAddr()</code>	333
5.37.2.5	<code>trdp_getSeqCnt()</code>	334
5.37.2.6	<code>trdp_initSockets()</code>	334
5.37.2.7	<code>trdp_isAddressed()</code>	335
5.37.2.8	<code>trdp_isInIPrange()</code>	335
5.37.2.9	<code>trdp_packetSizeMD()</code>	335
5.37.2.10	<code>trdp_packetSizePD()</code>	336
5.37.2.11	<code>trdp_queueAppLast()</code>	336
5.37.2.12	<code>trdp_queueDelElement()</code>	336
5.37.2.13	<code>trdp_queueFindComId()</code>	337
5.37.2.14	<code>trdp_queueFindExistingSub()</code>	337
5.37.2.15	<code>trdp_queueFindPubAddr()</code>	338
5.37.2.16	<code>trdp_queueFindSubAddr()</code>	338
5.37.2.17	<code>trdp_queueInsFirst()</code>	339
5.37.2.18	<code>trdp_releaseSocket()</code>	339
5.37.2.19	<code>trdp_requestSocket()</code>	339
5.37.2.20	<code>trdp_resetSequenceCounter()</code>	340
5.37.2.21	<code>trdp_SockAddJoin()</code>	341
5.37.2.22	<code>trdp_SockDelJoin()</code>	341
5.37.2.23	<code>trdp_SockIsJoined()</code>	341
5.37.2.24	<code>trdp_validTopoCounters()</code>	342
5.38	<code>trdp_xml.c</code> File Reference	342
5.38.1	Detailed Description	344
5.38.2	Function Documentation	344
5.38.2.1	<code>trdp_XMLClose()</code>	344
5.38.2.2	<code>trdp_XMLCountStartTag()</code>	344
5.38.2.3	<code>trdp_XMLEnter()</code>	345
5.38.2.4	<code>trdp_XMLGetAttribute()</code>	345

5.38.2.5	trdp_XMLLeave()	346
5.38.2.6	trdp_XMLMemOpen()	346
5.38.2.7	trdp_XMLOpen()	347
5.38.2.8	trdp_XMLRewind()	347
5.38.2.9	trdp_XMLSeekStartTag()	347
5.38.2.10	trdp_XMLSeekStartTagAny()	348
5.39	trdp_xml.h File Reference	348
5.39.1	Detailed Description	350
5.39.2	Function Documentation	350
5.39.2.1	trdp_XMLClose()	350
5.39.2.2	trdp_XMLCountStartTag()	351
5.39.2.3	trdp_XMLEnter()	351
5.39.2.4	trdp_XMLGetAttribute()	351
5.39.2.5	trdp_XMLLeave()	352
5.39.2.6	trdp_XMLMemOpen()	352
5.39.2.7	trdp_XMLOpen()	353
5.39.2.8	trdp_XMLRewind()	353
5.39.2.9	trdp_XMLSeekStartTag()	354
5.39.2.10	trdp_XMLSeekStartTagAny()	354
5.40	vos_mem.c File Reference	354
5.40.1	Detailed Description	356
5.40.2	Function Documentation	356
5.40.2.1	vos_bsearch()	356
5.40.2.2	vos_memAlloc()	357
5.40.2.3	vos_memCount()	357
5.40.2.4	vos_memDelete()	358
5.40.2.5	vos_memFree()	358
5.40.2.6	vos_memInit()	358
5.40.2.7	vos_qsort()	359
5.40.2.8	vos_queueCreate()	359

5.40.2.9	vos_queueDestroy()	360
5.40.2.10	vos_queueReceive()	360
5.40.2.11	vos_queueSend()	361
5.40.2.12	vos_strncat()	361
5.40.2.13	vos_strncpy()	363
5.40.2.14	vos_strnicmp()	363
5.41	vos_mem.h File Reference	364
5.41.1	Detailed Description	366
5.41.2	Macro Definition Documentation	366
5.41.2.1	VOS_MEM_MAX_PREALLOCATE	366
5.41.2.2	VOS_MEM_PREALLOCATE	366
5.41.3	Function Documentation	366
5.41.3.1	vos_bsearch()	366
5.41.3.2	vos_memAlloc()	367
5.41.3.3	vos_memCount()	367
5.41.3.4	vos_memDelete()	368
5.41.3.5	vos_memFree()	368
5.41.3.6	vos_memInit()	369
5.41.3.7	vos_qsort()	369
5.41.3.8	vos_queueCreate()	370
5.41.3.9	vos_queueDestroy()	371
5.41.3.10	vos_queueReceive()	371
5.41.3.11	vos_queueSend()	372
5.41.3.12	vos_strncat()	372
5.41.3.13	vos_strncpy()	372
5.41.3.14	vos_strnicmp()	373
5.42	vos_shared_mem.h File Reference	373
5.42.1	Detailed Description	374
5.42.2	Function Documentation	375
5.42.2.1	vos_sharedClose()	375

5.42.2.2	<code>vos_sharedOpen()</code>	375
5.43	<code>vos_sock.h</code> File Reference	376
5.43.1	Detailed Description	378
5.43.2	Macro Definition Documentation	379
5.43.2.1	<code>VOS_MAX_SOCKET_CNT</code>	379
5.43.2.2	<code>VOS_TTL_MULTICAST</code>	379
5.43.3	Function Documentation	379
5.43.3.1	<code>vos_determineBindAddr()</code>	379
5.43.3.2	<code>vos_dottedIP()</code>	380
5.43.3.3	<code>vos_getInterfaces()</code>	380
5.43.3.4	<code>vos_htonl()</code>	381
5.43.3.5	<code>vos_htonll()</code>	381
5.43.3.6	<code>vos_htons()</code>	381
5.43.3.7	<code>vos_ipDotted()</code>	382
5.43.3.8	<code>vos_isMulticast()</code>	382
5.43.3.9	<code>vos_netIfUp()</code>	382
5.43.3.10	<code>vos_ntohl()</code>	383
5.43.3.11	<code>vos_ntohll()</code>	383
5.43.3.12	<code>vos_ntohs()</code>	383
5.43.3.13	<code>vos_select()</code>	384
5.43.3.14	<code>vos_sockAccept()</code>	384
5.43.3.15	<code>vos_sockBind()</code>	385
5.43.3.16	<code>vos_sockClose()</code>	385
5.43.3.17	<code>vos_sockConnect()</code>	386
5.43.3.18	<code>vos_sockGetMAC()</code>	386
5.43.3.19	<code>vos_sockInit()</code>	387
5.43.3.20	<code>vos_sockJoinMC()</code>	387
5.43.3.21	<code>vos_sockLeaveMC()</code>	387
5.43.3.22	<code>vos_sockListen()</code>	388
5.43.3.23	<code>vos_sockOpenTCP()</code>	388

5.43.3.24	<code>vos_sockOpenUDP()</code>	389
5.43.3.25	<code>vos_sockReceiveTCP()</code>	389
5.43.3.26	<code>vos_sockReceiveUDP()</code>	390
5.43.3.27	<code>vos_sockSendTCP()</code>	391
5.43.3.28	<code>vos_sockSendUDP()</code>	391
5.43.3.29	<code>vos_sockSetMulticastIf()</code>	392
5.43.3.30	<code>vos_sockSetOptions()</code>	392
5.43.3.31	<code>vos_sockTerm()</code>	393
5.44	<code>vos_thread.h</code> File Reference	393
5.44.1	Detailed Description	395
5.44.2	Function Documentation	396
5.44.2.1	<code>vos_addTime()</code>	396
5.44.2.2	<code>vos_clearTime()</code>	396
5.44.2.3	<code>vos_cmpTime()</code>	396
5.44.2.4	<code>vos_divTime()</code>	397
5.44.2.5	<code>vos_getRealTime()</code>	397
5.44.2.6	<code>vos_getTime()</code>	397
5.44.2.7	<code>vos_getTimeStamp()</code>	398
5.44.2.8	<code>vos_getUuid()</code>	398
5.44.2.9	<code>vos_mulTime()</code>	398
5.44.2.10	<code>vos_mutexCreate()</code>	399
5.44.2.11	<code>vos_mutexDelete()</code>	399
5.44.2.12	<code>vos_mutexLock()</code>	399
5.44.2.13	<code>vos_mutexTryLock()</code>	400
5.44.2.14	<code>vos_mutexUnlock()</code>	400
5.44.2.15	<code>vos_semaCreate()</code>	401
5.44.2.16	<code>vos_semaDelete()</code>	401
5.44.2.17	<code>vos_semaGive()</code>	401
5.44.2.18	<code>vos_semaTake()</code>	402
5.44.2.19	<code>vos_subTime()</code>	402

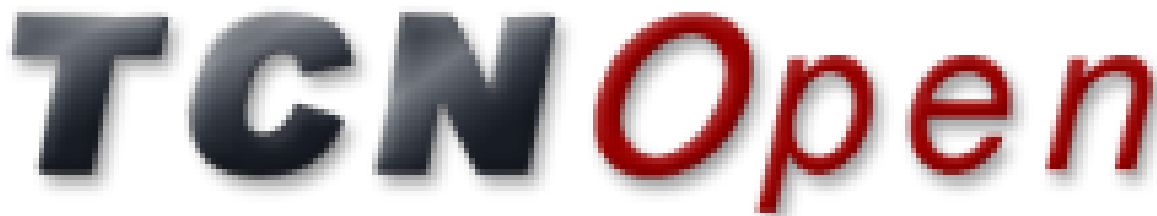
5.44.2.20	vos_threadCreate()	402
5.44.2.21	vos_threadCreateSync()	403
5.44.2.22	vos_threadDelay()	404
5.44.2.23	vos_threadInit()	404
5.44.2.24	vos_threadIsActive()	405
5.44.2.25	vos_threadSelf()	405
5.44.2.26	vos_threadTerm()	405
5.44.2.27	vos_threadTerminate()	406
5.45	vos_types.h File Reference	406
5.45.1	Detailed Description	408
5.45.2	Typedef Documentation	408
5.45.2.1	VOS_PRINT_DBG_T	408
5.45.2.2	VOS_TIMEVAL_T	408
5.45.3	Enumeration Type Documentation	409
5.45.3.1	VOS_ERR_T	409
5.45.3.2	VOS_LOG_T	409
5.46	vos_utils.c File Reference	410
5.46.1	Detailed Description	411
5.46.2	Function Documentation	411
5.46.2.1	vos_crc32()	411
5.46.2.2	vos_getErrorString()	412
5.46.2.3	vos_getVersion()	412
5.46.2.4	vos_getVersionString()	412
5.46.2.5	vos_hostIsBigEndian()	413
5.46.2.6	vos_init()	413
5.46.2.7	vos_sc32()	413
5.46.2.8	vos_terminate()	414
5.47	vos_utils.h File Reference	414
5.47.1	Detailed Description	416
5.47.2	Macro Definition Documentation	416
5.47.2.1	INITFCS	416
5.47.2.2	VOS_MAX_ERR_STR_SIZE	416
5.47.2.3	VOS_MAX_FRMT_SIZE	416
5.47.2.4	VOS_MAX_PRNT_STR_SIZE	417
5.47.3	Function Documentation	417
5.47.3.1	vos_crc32()	417
5.47.3.2	vos_getErrorString()	418
5.47.3.3	vos_getVersion()	418
5.47.3.4	vos_getVersionString()	418
5.47.3.5	vos_hostIsBigEndian()	419
5.47.3.6	vos_init()	419
5.47.3.7	vos_sc32()	420
5.47.3.8	vos_terminate()	420

[Index](#)

421

Chapter 1

The TRDP Light Library API Specification



1.1 General Information

1.1.1 Purpose

The TRDP protocol has been defined as the standard communication protocol in IP-enabled trains. It allows communication via process data (periodically transmitted data using UDP/IP) and message data (client - server messaging using UDP/IP or TCP/IP). This document describes the light API of the TRDP Library.

1.1.2 Scope

The intended audience of this document is the developers and project members of the TRDP project. TRDP Client Applications are programs using the TRDP protocol library to access the services of TRDP. Programmers developing such applications are the main target audience for this documentation.

1.1.3 Related documents

TCN-TRDP2-D-BOM-004-01 IEC61375-2-3_CD_ANNEXA Protocol definition of the TRDP standard
TCN-TRDP2-D-BOM-011-32 TRDP User's Manual

1.1.4 Abbreviations and Definitions

-*API* Application Programming Interface -*ECN* Ethernet Consist Network -*TRDP* Train Real-time Data Protocol
-*TCMS* Train Control Management System

1.2 Terminology

The API documented here is mainly concerned with three bodies of code:

- *TRDP Client Applications* (or 'client applications' for short): These are programs using the API to access the services of TRDP. Programmers developing such applications are the main target audience for this documentation.
- *TRDP Light Implementations* (or just 'TRDP implementation'): These are libraries realising the API as documented here. Programmers developing such implementations will find useful definitions about syntax and semantics of the API within this documentation.
- *VOS Subsystem* (Virtual Operating System): An OS and hardware abstraction layer which offers memory, networking, threading, queues and debug functions. The VOS API is documented here.

1.3 Use Cases

The following diagram shows how these pieces of software are interrelated. Single threaded flow:

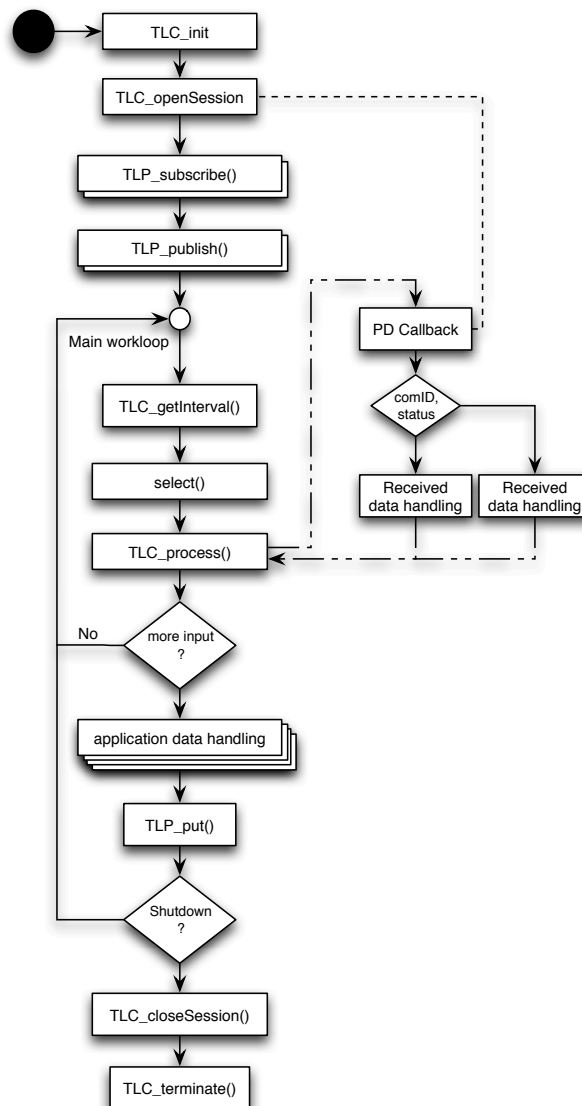


Figure 1.1 Sample client workflow (Single Thread)

Usage of the separate process handling (separate threads for PD and MD):

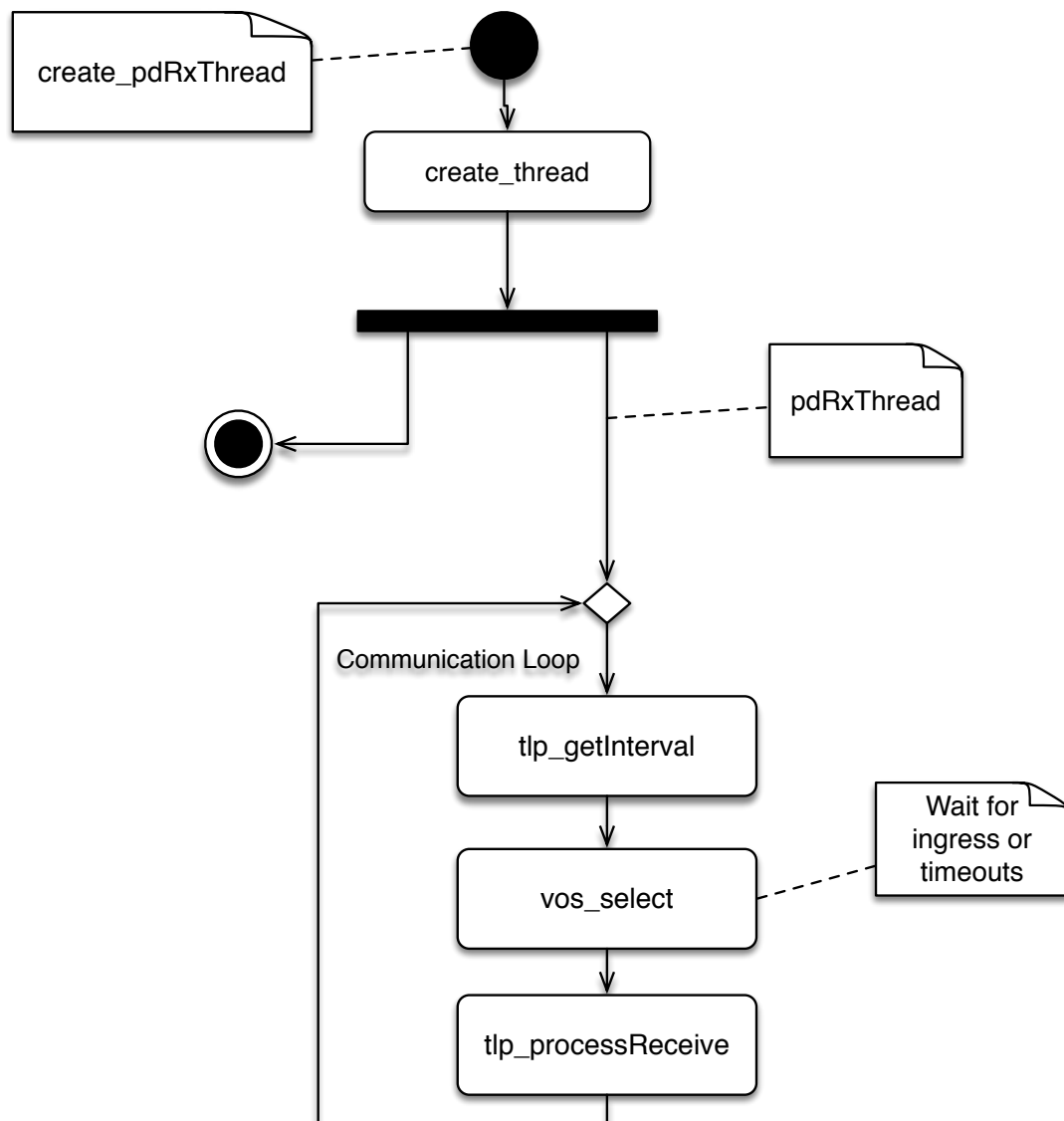


Figure 1.2 Multi-threaded processing of PD Reception

The transmit thread should be a cyclic thread. Cycle times down to 1ms are supported:

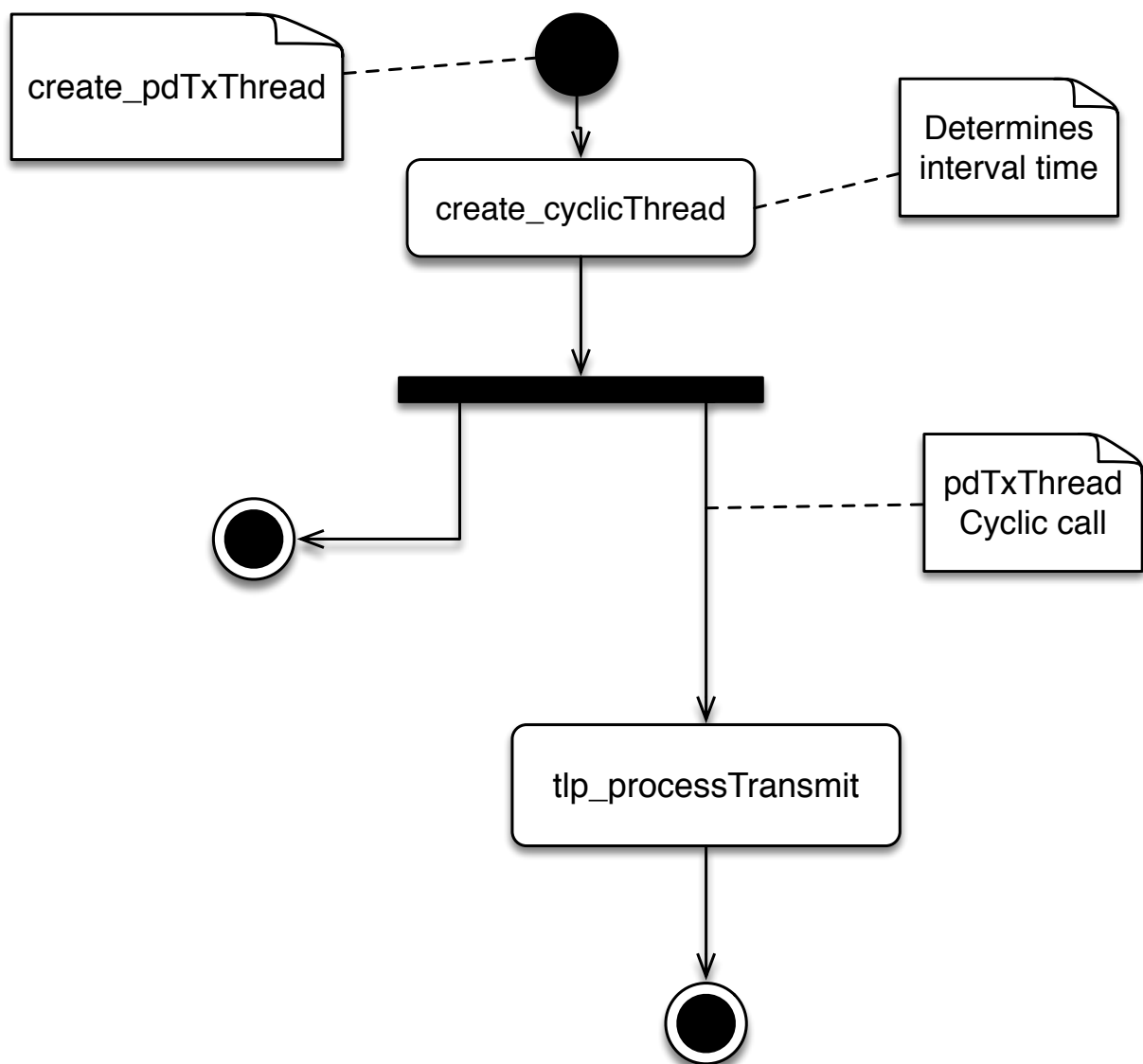


Figure 1.3 Multi-threaded processing of PD Transmit

If Message Data support is needed (MD_SUPPORT=1):

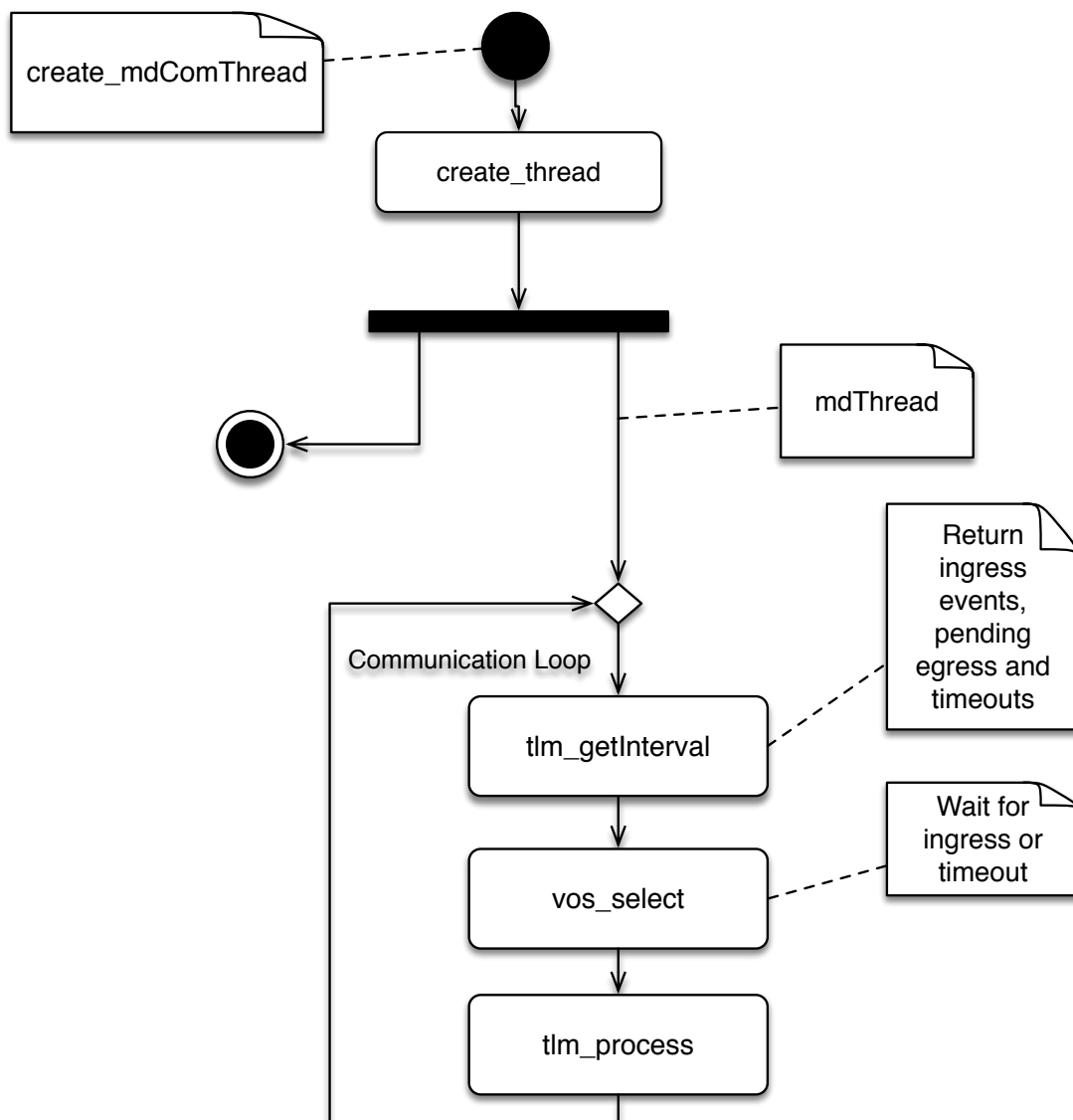


Figure 1.4 Multi-threaded processing of MD

Note: Mixed usage of the single threaded call `tlc_process()` with the multi-threaded calls `tlm_process/tlp_process`↔
Transmit/tlp_processReceive is not supported!

1.4 Conventions of the API

The API comprises a set of C header files that can also be used from client applications written in C++. These header files are contained in a directory named `trdp/api` and a subdirectory called `trdp/vos/api` with declarations not topical to TRDP but needed by the stack. Client applications shall include these header files like:

```
#include "trdp_if_light.h"
```

and, if VOS functions are needed, also the corresponding headers:

```
#include "vos_thread.h"
```

for example.

The subdirectory `trdp/doc` contains files needed for the API documentation.

Generally client application source code including API headers will only compile if the parent directory of the `trdp` directory is part of the include path of the used compiler. No other subdirectories of the API should be added to the compiler's include path.

The client API doesn't support a "catch-all" header file that includes all declarations in one step; rather the client application has to include individual headers for each feature set it wants to use.

Further description of the API and the usage of the TRDP protocol stack can be found in the TCNOpen TRDP User's Manual (at tcnopen.eu).

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

DNS_HEADER	DNS header structure	13
GNU_PACKED	Types for ETB control	13
hp_slot	Low cycle-time slots	32
hp_slots	Entry for the application session	33
PD_ELE	Queue element for PD packets to send or receive	34
service_info	Preliminary definition of a service info entry	36
srv_info_req	Preliminary definition of a service info request	37
TAU_MARSHALL_INFO_T	Marshalling info, used to and from wire	38
TCN_URI	TCN-DNS simplified header structures	38
TRDP_CLTR_CST_INFO_T	Closed train consists information	39
TRDP_COM_PARAM_T	Quality/type of service, time to live , no	39
TRDP_COMID_DSID_MAP_T	ComId - data set mapping element definition	40
TRDP_CONSIST_INFO_T	Consist information structure	41
TRDP_DATASET	Dataset definition	43
TRDP_DATASET_ELEMENT_T	Dataset element definition	44
TRDP_DBG_CONFIG_T	Control for debug output device/file on application level	45
TRDP_DNS_REPLY	TCN-DNS Reply telegram TCN_DNS_REP_DS	45
TRDP_DNS_REQUEST	TCN-DNS Request telegram TCN_DNS_REQ_DS	47

TRDP_ETB_INFO_T	
Types for train configuration information	48
TRDP_FUNCTION_INFO_T	
Function/device information structure	49
TRDP_HANDLE	
Hidden handle definition, used as unique addressing item	50
TRDP_IDX_TABLE_T	
Settings for pre-allocation of index tables for application session initialization	51
TRDP_MARSHALL_CONFIG_T	
Marshaling/unmarshalling configuration	53
TRDP_MD_CONFIG_T	
Default MD configuration	54
TRDP_MD_INFO_T	
Message data info from received telegram; allows the application to generate responses . . .	55
TRDP_MEM_CONFIG_T	
Enumeration type for memory pre-fragmentation, reuse of VOS definition	56
TRDP_PD_CONFIG_T	
Default PD configuration	57
TRDP_PD_INFO_T	
Process data info from received telegram; allows the application to generate responses	57
TRDP_PROCESS_CONFIG_T	
Various flags/general TRDP options for library initialization	58
TRDP_PROP_T	
Application defined properties	59
TRDP_SDT_PAR_T	
Types to read out the XML configuration	60
TRDP_SEQ_CNT_ENTRY_T	
Tuples of last received sequence counter per comld	60
TRDP_SESSION	
Session/application variables store	61
TRDP_SOCKET_TCP	
TCP parameters	62
TRDP_SOCKETS	
Socket item	63
TRDP_VEHICLE_INFO_T	
Vehicle information structure	65
TRDP_XML_DOC_HANDLE_T	
Parsed XML document handle	66
VOS SOCK_OPT_T	
Common socket options	66
VOS_VERSION_T	
Version information	67

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

iec61375-2-3.h	All definitions from IEC 61375-2-3	69
tau_cstinfo.c	Functions for consist information access	77
tau_ctrl.c	Functions for train switch control	79
tau_ctrl.h	TRDP utility interface definitions	83
tau_ctrl_types.h	TRDP utility interface definitions	88
tau_dnr.c	Functions for domain name resolution	91
tau_dnr.h	TRDP utility interface definitions	96
tau_dnr_types.h	TRDP utility interface definitions	103
tau_marshall.c	Marshalling functions for TRDP	105
tau_marshall.h	TRDP utility interface definitions	111
tau_so_if.c	Access to service oriented functions of the SRM	121
tau_so_if.h	Access to the Service Registry	126
tau_tti.c	Functions for train topology information access	130
tau_tti.h	TRDP utility interface definitions	141
tau_tti_types.h	TRDP utility interface definitions	155
tau_xml.c	Functions for XML file parsing	158
tau_xml.h	TRDP utility interface definitions	166
tlc_if.c	Functions for ECN communication	178

tlc_if.h	Typedefs for TRDP communication	190
tlm_if.c	Functions for Message Data Communication	193
tlp_if.c	Functions for Process Data Communication	203
trdp_dllmain.c	Windows DLL main function	215
trdp_if_light.h	TRDP Light interface functions (API)	216
trdp_mdcom.c	Functions for MD communication	250
trdp_mdcom.h	Functions for MD communication	256
trdp_pdcom.c	Functions for PD communication	263
trdp_pdcom.h	Functions for PD communication	273
trdp_pdindex.c	Functions for indexed PD communication	284
trdp_pdindex.h	Functions for indexed PD communication	285
trdp_private.h	Typedefs for TRDP communication	288
trdp_serviceRegistry.h	Additional definitions for IEC 61375-2-3 (Service Discovery) The definitions herein are preliminary and will change with the next major release of the IEC 61375-2-3 standard	292
trdp_stats.c	Statistics functions for TRDP communication	296
trdp_stats.h	Statistics for TRDP communication	302
trdp_tsn_def.h	Additional definitions for TSN	305
trdp_types.h	Typedefs for TRDP communication	306
trdp_utils.c	Helper functions for TRDP communication	317
trdp_utils.h	Common utilities for TRDP communication	329
trdp_xml.c	Simple XML parser	342
trdp_xml.h	Simple XML parser	348
vos_mem.c	Memory functions	354
vos_mem.h	Memory and queue functions for OS abstraction	364
vos_shared_mem.h	Shared Memory functions for OS abstraction	373
vos_sock.h	Typedefs for OS abstraction	376
vos_thread.h	Threading functions for OS abstraction	393
vos_types.h	Typedefs for OS abstraction	406
vos_utils.c	Common functions for VOS	410

[vos_utils.h](#)Typedefs for OS abstraction [414](#)

Chapter 4

Data Structure Documentation

4.1 DNS_HEADER Struct Reference

DNS header structure.

4.1.1 Detailed Description

DNS header structure.

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

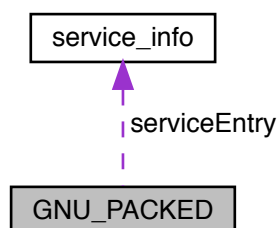
- [tau_dnr.c](#)

4.2 GNU_PACKED Struct Reference

Types for ETB control.

```
#include <trdp_private.h>
```

Collaboration diagram for GNU_PACKED:



Data Fields

- UINT8 [trnVehNo](#)
vehicle sequence number within the train with vehicle 01 being the first vehicle in ETB reference direction 1 as defined in IEC61375-2-5 value range: 0..63 a value of 0 indicates that this vehicle has been inserted by correction
- ANTIVALENT8 [isLead](#)
vehicle is leading
- UINT8 [leadDir](#)
vehicle leading direction 0 = not relevant 1 = leading direction 1 2 = leading direction 2
- UINT8 [vehOrient](#)
vehicle orientation 0 = not known (corrected vehicle) 1 = same as operational train direction 2 = inverse to operational train direction
- TRDP_SHORT_VERSION_T [version](#)
telegram version information, main_version = 1, sub_version = 0
- UINT16 [reserved01](#)
reserved (=0)
- UINT8 [trnCstNo](#)
own TCN consist number (= 1..32)
- UINT8 [reserved02](#)
reserved (=0)
- UINT8 [ownOpCstNo](#)
own operational address (= 1..32) = 0 if unknown (e.g.
- UINT8 [reserved03](#)
reserved (=0)
- UINT32 [cstTopoCount](#)
Consist topology counter.
- UINT32 [trnTopoCount](#)
Train directory topology counter.
- UINT32 [opTrnTopoCount](#)
Operational Train topology counter.
- ANTIVALENT8 [wasLead](#)
consist was leading, '01'B = false, '10'B = true
- ANTIVALENT8 [reqLead](#)
leading request, '01'B = false, '10'B = true
- UINT8 [reqLeadDir](#)
(request) leading direction, '01'B = consist direction 1, '10'B = consist direction 2
- ANTIVALENT8 [accLead](#)
accept remote leading request, '01'B = false/not accepted, '10'B = true/accepted
- ANTIVALENT8 [clearConfComp](#)
clear confirmed composition, '01'B = false, '10'B = true
- ANTIVALENT8 [corrRequest](#)
request confirmation, '01'B = false, '10'B = true
- ANTIVALENT8 [corrInfoSet](#)
correction info set, '01'B = false, '10'B = true
- ANTIVALENT8 [compStored](#)
corrected composition stored, '01'B = false, '10'B = true
- ANTIVALENT8 [sleepRequest](#)
request sleep mode, '01'B = false, '10'B = true
- UINT8 [leadVehOfCst](#)
position of leading vehicle in consist, 0..31 (1: first vehicle in consist in Direction 1, 2: second vehicle, etc.)
- UINT8 [reserved04](#)

- reserved (=0)*
- UINT16 [reserved05](#)
 - reserved (=0)*
- UINT8 [reserved06](#)
 - reserved (=0)*
- UINT8 [confVehCnt](#)
 - number of confirmed vehicles in train (1..63)*
- TRDP_CONF_VEHICLE_T [confVehList](#) [[TRDP_MAX_VEH_CNT](#)]
 - dynamic ordered list of confirmed vehicles in train, starting with vehicle at train head, see sub-clause 5.3.3.2.6*
- TRDP_ETB_CTRL_VDP_T [safetyTrail](#)
 - ETBCTRL-VDP trailer, completely set to 0 == not used.*
- UINT8 [reserved01](#)
 - reserved (=0)*
- [TRDP_NET_LABEL_T](#) [deviceName](#)
 - function device of ECSC which sends the telegram*
- UINT8 [inhibit](#)
 - inauguration inhibit 0 = no inhibit request 1 = inhibit request*
- UINT8 [leadingReq](#)
 - leading request 0 = no leading request 1 = leading request*
- UINT8 [leadingDir](#)
 - leading direction 0 = no leading request 1 = leading request direction 1 2 = leading request direction 2*
- UINT8 [sleepReq](#)
 - sleep request 0 = no sleep request 1 = sleep request*
- UINT16 [lifesign](#)
 - wrap-around counter, incremented with each produced datagram.*
- UINT8 [ecspState](#)
 - ECSP state indication 0 = ECSP not operational(initial value) 1 = ECSP in operation.*
- UINT8 [etbInhibit](#)
 - inauguration inhibit indication 0 = n/a (default) 1 = inhibit not requested on ETB 2 = inhibit set on local ETBN 3 = inhibit set on remote ETBN 4 = inhibit set on local and remote ETBN*
- UINT8 [etbLength](#)
 - indicates train lengthening in case train inauguration is inhibit 0 = no lengthening (default) 1 = lengthening detected*
- UINT8 [etbShort](#)
 - indicates train shortening in case train inauguration is inhibit 0 = no shortening (default) 1 = shortening detected*
- UINT16 [reserved02](#)
 - reserved (=0)*
- UINT8 [etbLeadState](#)
 - indication of local consist leadership 5 = consist not leading (initial value) 6 = consist is leading requesting 9 = consist is leading 10 = leading conflict other values are not allowed*
- UINT8 [etbLeadDir](#)
 - direction of the leading end car in the local consist 0 = unknown (default) 1 = TCN direction 1 2 = TCN direction 2 other values are not allowed*
- UINT8 [ttdbSrvState](#)
 - TTDB server state indication 0 = n/a (initial value) 1 = Leader (default) 2 = Follower 3 = Error.*
- UINT8 [dnsSrvState](#)
 - DNS server state indication 0 = n/a (initial value) 1 = Leader (default) 2 = Follower 3 = Error.*
- UINT8 [trnDirState](#)
 - train directory state 1 = UNCONFIRMED 2 = CONFIRMED other values are not allowed*
- UINT8 [opTrnDirState](#)
 - train directory state 1 = INVALID 2 = VALID 4 = SHARED other values are not allowed*
- UINT8 [sleepCtrlState](#)

- sleep control state (option) 0 = option not available 1 = RegularOperation 2 = WaitForSleepMode 3 = PrepareForSleepMode*
- UINT8 [sleepReqCnt](#)
number of sleep requests (option) value range: 0..63, not used = 0
 - UINT32 [opTrnTopoCnt](#)
operational train topology counter
 - UINT8 [command](#)
confirmation order 1 = confirmation/correction request 2 = un-confirmation request
 - UINT16 [confVehCnt](#)
number of confirmed vehicles in the train (1..63).
 - TRDP_OP_VEHICLE_T [confVehList](#) [TRDP_MAX_VEH_CNT]
ordered list of confirmed vehicles in the train, starting with vehicle at train head, see chapter 5.3.3.2.10.
 - UINT8 [status](#)
status of storing correction info 0 = correctly stored 1 = not stored
 - UINT32 [reqSafetyCode](#)
SC-32 value of the request message.
 - UINT8 [byPassCtrl](#)
ETBN bypass control 0 = no action (keep old state) 1 = no bypass 2 = activate bypass.
 - UINT8 [txCtrl](#)
ETBN transmission control 0 = no action (keep old state) 1 = activate sending on ETB (default) 2 = stop sending on ETB.
 - UINT8 [slCtrl](#)
sleep mode control (option) 0 = no action (keep old state) 1 = deactivate sleep mode 2 = activate sleep mode (line activity sensing)
 - UINT8 [etbnState](#)
state indication of the (active) ETBN 0 = ETBN not operational(initial value) 1 = ETBN in operation
 - UINT8 [etbnInaugState](#)
ETBN inauguration state as defined in IEC61375-2-5 0 = init 1 = not inaugurated 2 = inaugurated 3 = ready for inauguration.
 - UINT8 [etbnPosition](#)
position of the ETBN 0 = unknown (default) 1 = single node 2 = middle node 3 = end node TCN direction 1 4 = end node TCN direction 2
 - UINT8 [etbnRole](#)
ETBN node role as defined in IEC61375-2-5 0 = undefined 1 = master (redundancy leader) 2 = backup (redundancy follower) 3 = not redundant.
 - BITSET8 [etbLineState](#)
indication of ETB line status (FALSE == not trusted, TRUE == trusted) bit0 = line A ETBN direction 1 bit1 = line B ETBN direction 1 bit2 = line C ETBN direction 1 bit3 = line D ETBN direction 1 bit4 = line A ETBN direction 2 bit5 = line B ETBN direction 2 bit6 = line C ETBN direction 2 bit7 = line D ETBN direction 2
 - UINT8 [byPassState](#)
state of bypass function 0 = bypass disabled 1 = bypass enabled
 - UINT8 [slState](#)
sleep mode state (option) 0 = no sleep mode 1 = sleep mode active (line activity sensing)
 - UINT32 [etbTopoCnt](#)
ETB topography counter.
 - TRDP_TRAIN_NET_DIR_T [trnNetDir](#)
dynamic train info
 - UINT8 [ver](#)
Version - incremented for incompatible changes.
 - UINT8 [rel](#)
Release - incremented for compatible changes.
 - UINT32 [reserved01](#)

- reserved (=0)*
- TRDP_SHORT_VERSION_T [userDataVersion](#)
version of the vital ETBCTRL telegram mainVersion = 1, subVersion = 0
- UINT32 [safeSeqCount](#)
safe sequence counter, as defined in B.9
- UINT32 [safetyCode](#)
checksum, as defined in B.9
- TRDP_UUID_T [cstUUID](#)
UUID of the consist, provided by ETBN (TrainNetworkDirectory) Reference to static consist attributes 0 if not available (e.g.
- UINT32 [cstTopoCnt](#)
consist topology counter provided with the CSTINFO 0 if no CSTINFO available
- UINT8 [cstOrient](#)
consist orientation '01'B = same as train direction '10'B = inverse to train direction
- UINT8 [cstCnt](#)
number of consists in train; range: 1..63
- TRDP_CONSIST_T [cstList](#) [TRDP_MAX_CST_CNT]
consist list.
- UINT32 [trnTopoCnt](#)
trnTopoCnt value ctrlType == 0: actual value ctrlType == 1: set to 0
- UINT8 [etbld](#)
identification of the ETB the TTDB is computed for bit0: ETB0 (operational network) bit1: ETB1 (multimedia network) bit2: ETB2 (other network) bit3: ETB3 (other network)
- TRDP_NET_LABEL_T [vehId](#)
Unique vehicle identifier, application defined (e.g.
- UINT8 [opVehNo](#)
operational vehicle sequence number in train value range 1..63
- UINT8 [opCstNo](#)
operational consist number in train (1..63)
- UINT8 [opCstOrient](#)
consist orientation '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction
- TRDP_NET_LABEL_T [trnId](#)
train identifier, application defined (e.g.
- TRDP_NET_LABEL_T [trnOperator](#)
train operator, e.g.
- UINT32 [crc](#)
sc-32 computed over record (seed value: 'FFFFFFFF'H)
- UINT8 [opTrnOrient](#)
operational train orientation '00'B = unknown '01'B = same as train direction '10'B = inverse to train direction
- UINT8 [opCstCnt](#)
number of consists in train (1..63)
- TRDP_OP_CONSIST_T [opCstList](#) [TRDP_MAX_CST_CNT]
operational consist list starting with op.
- UINT8 [reserved05](#)
reserved for future use (= 0)
- UINT8 [opVehCnt](#)
number of vehicles in train (1..63)
- TRDP_OP_VEHICLE_T [opVehList](#) [TRDP_MAX_VEH_CNT]
operational vehicle list starting with op.
- TRDP_OP_TRAIN_DIR_STATE_T [state](#)

- operational state of the train*
- UINT32 [cstNetProp](#)
 - consist network properties bit0..1: consist orientation bit2..7: 0 bit8..13: ETBN Id bit14..15: 0 bit16..21: subnet Id bit24..29: CN Id bit30..31: 0*
- UINT16 [entryCnt](#)
 - number of entries in train network directory*
- TRDP_TRAIN_NET_DIR_ENTRY_T [trnNetDir](#) [[TRDP_MAX_CST_CNT](#)]
 - train network directory*
- TRDP_OP_TRAIN_DIR_T [opTrnDir](#)
 - operational directory*
- TRDP_TRAIN_DIR_T [trnDir](#)
 - train directory*
- UINT16 [noOfEntries](#)
 - number of entries in array*
- [SRM_SERVICE_INFO_T](#) [serviceEntry](#) [1]
 - var.*
- UINT32 [comId](#)
 - ComId to request: 35...41.*
- UINT32 [total](#)
 - total memory size*
- UINT32 [free](#)
 - free memory size*
- UINT32 [minFree](#)
 - minimal free memory size in statistics interval*
- UINT32 [numAllocBlocks](#)
 - allocated memory blocks*
- UINT32 [numAllocErr](#)
 - allocation errors*
- UINT32 [numFreeErr](#)
 - free errors*
- UINT32 [blockSize](#) [[VOS_MEM_NBLOCKSIZES](#)]
 - preallocated memory blocks*
- UINT32 [usedBlockSize](#) [[VOS_MEM_NBLOCKSIZES](#)]
 - used memory blocks*
- UINT32 [defQos](#)
 - default QoS for PD*
- UINT32 [defTtl](#)
 - default TTL for PD*
- UINT32 [defTimeout](#)
 - default timeout in us for PD*
- UINT32 [numSubs](#)
 - number of subscribed ComId's*
- UINT32 [numPub](#)
 - number of published ComId's*
- UINT32 [numRcv](#)
 - number of received PD packets*
- UINT32 [numCrcErr](#)
 - number of received PD packets with CRC err*
- UINT32 [numProtErr](#)
 - number of received PD packets with protocol err*
- UINT32 [numTopoErr](#)

- number of received PD packets with wrong topo count*
- UINT32 [numNoSubs](#)
 - number of received PD push packets without subscription*
- UINT32 [numNoPub](#)
 - number of received PD pull packets without publisher*
- UINT32 [numTimeout](#)
 - number of PD timeouts*
- UINT32 [numSend](#)
 - number of sent PD packets*
- UINT32 [numMissed](#)
 - number of packets skipped*
- UINT32 [defReplyTimeout](#)
 - default reply timeout in us for MD*
- UINT32 [defConfirmTimeout](#)
 - default confirm timeout in us for MD*
- UINT32 [numList](#)
 - number of listeners*
- UINT32 [numNoListener](#)
 - number of received MD packets without listener*
- UINT32 [numReplyTimeout](#)
 - number of reply timeouts*
- UINT32 [numConfirmTimeout](#)
 - number of confirm timeouts*
- UINT32 [version](#)
 - TRDP version.*
- UINT64 [timeStamp](#)
 - actual time stamp*
- UINT32 [upTime](#)
 - time in sec since last initialisation*
- UINT32 [statisticTime](#)
 - time in sec since last reset of statistics*
- [TRDP_NET_LABEL_T](#) [hostName](#)
 - host name*
- [TRDP_NET_LABEL_T](#) [leaderName](#)
 - leader host name*
- [TRDP_IP_ADDR_T](#) [ownIpAddr](#)
 - own IP address*
- [TRDP_IP_ADDR_T](#) [leaderIpAddr](#)
 - leader IP address*
- UINT32 [processPrio](#)
 - priority of TRDP process*
- UINT32 [processCycle](#)
 - cycle time of TRDP process in microseconds*
- UINT32 [numJoin](#)
 - number of joins*
- UINT32 [numRed](#)
 - number of redundancy groups*
- [TRDP_MEM_STATISTICS_T](#) [mem](#)
 - memory statistics*
- [TRDP_PD_STATISTICS_T](#) [pd](#)
 - pd statistics*

- TRDP_MD_STATISTICS_T [udpMd](#)
UDP md statistics.
- TRDP_MD_STATISTICS_T [tcpMd](#)
TCP md statistics.
- TRDP_IP_ADDR_T [joinedAddr](#)
Joined IP address.
- TRDP_IP_ADDR_T [filterAddr](#)
Filter IP address, i.e IP address of the sender for this subscription, 0.0.0.0 in case all senders.
- UINT32 [callBack](#)
call back function if used
- UINT32 [userRef](#)
User reference if used.
- UINT32 [timeout](#)
Time-out value in us.
- UINT32 [status](#)
Receive status information TRDP_NO_ERR, TRDP_TIMEOUT_ERR.
- UINT32 [toBehav](#)
Behavior at time-out.
- UINT32 [numRecv](#)
Number of packets received for this subscription.
- TRDP_IP_ADDR_T [destAddr](#)
IP address of destination for this publishing.
- UINT32 [cycle](#)
Publishing cycle in us.
- UINT32 [redId](#)
Redundancy group id.
- UINT32 [redState](#)
Redundant state. Leader or Follower.
- UINT32 [numPut](#)
Number of packet updates.
- CHAR8 [uri](#) [32]
URI user part to listen to.
- UINT32 [queue](#)
Queue reference if used.
- UINT32 [id](#)
Redundant Id.
- UINT32 [state](#)
Redundant state. Leader or Follower.
- UINT32 [sequenceCounter](#)
Unique counter (autom incremented)
- UINT16 [protocolVersion](#)
fix value for compatibility (set by the API)
- UINT16 [msgType](#)
of datagram: PD Request (0x5072) or PD_MSG (0x5064)
- UINT32 [datasetLength](#)
length of the data to transmit 0...1432
- UINT32 [reserved](#)
reserved for ServiceID/InstanceID support
- UINT32 [replyComId](#)
used in PD request
- UINT32 [replyIpAddress](#)

- used for PD request*
- UINT32 [frameChecksum](#)
CRC32 of header.
- INT32 [replyStatus](#)
0 = OK
- UINT8 [sessionID](#) [16u]
UUID as a byte stream.
- UINT32 [replyTimeout](#)
in us
- UINT8 [sourceURI](#) [32u]
User part of URI.
- UINT8 [destinationURI](#) [32u]
User part of URI.
- PD_HEADER_T [frameHead](#)
Packet header in network byte order.
- UINT8 [data](#) [TRDP_MAX_PD_DATA_SIZE]
data ready to be sent or received

4.2.1 Detailed Description

Types for ETB control.

TRDP PD packet.

TRDP message data header - network order and alignment.

TRDP process data header - network order and alignment.

A table containing PD redundant group information.

Information about a particular MD listener.

Table containing particular PD publishing information.

Table containing particular PD subscription information.

Structure containing all general memory, PD and MD statistics information.

Structure containing all general MD statistics information.

Structure containing all general PD statistics information.

Structure containing all general memory statistics information.

TRDP statistics type definitions.

Complete TTDB structure.

Train network directory structure.

Train network directory entry structure acc.

Operational Train directory status info structure.

Operational train structure.

Operational train directory state.

Operational consist structure.

Operational vehicle structure.

TCN train directory.

CSTINFO Control telegram.

TCN consist structure.

Version information for communication buffers.

to IEC61375-2-5

Statistical data regarding the former info provided via SNMP the following information was left out/can be implemented additionally using MD:

- PD subscr table: ComId, sourceIpAddr, destIpAddr, cbFct?, timeout, toBehavior, counter
- PD publish table: ComId, destIpAddr, redId, redState cycle, ttl, qos, counter
- PD join table: joined MC address table
- MD listener table: ComId destIpAddr, destUri, cbFct?, counter
- Memory usageStructure containing comId for MD statistics request (ComId 32).

4.2.2 Field Documentation

4.2.2.1 callBack

```
UINT32 GNU_PACKED::callBack
```

call back function if used

Call back function if used.

4.2.2.2 comId

```
UINT32 GNU_PACKED::comId
```

ComId to request: 35...41.

set by user: unique id

ComId to listen to.

Published ComId.

Subscribed ComId.

4.2.2.3 confVehCnt

```
UINT16 GNU_PACKED::confVehCnt
```

number of confirmed vehicles in the train (1..63).

4.2.2.4 confVehList

```
TRDP_OP_VEHICLE_T GNU_PACKED::confVehList[TRDP_MAX_VEH_CNT]
```

ordered list of confirmed vehicles in the train, starting with vehicle at train head, see chapter 5.3.3.2.10.

Parameters 'isLead' and 'leadDir' to be set to 0

4.2.2.5 cstList

```
TRDP_CONSIST_T GNU_PACKED::cstList
```

consist list.

consist list ordered list starting with trnCstNo == 1 Note: This is a variable size array, only opCstCnt array elements are present on the network and for crc computation

If trnCstNo > 0 this shall be an ordered list starting with trnCstNo == 1 (exactly the same as in structure TRAIN↔_DIRECTORY). If trnCstNo == 0 it is not mandatory to list all consists (only consists which should send CSTINFO telegram). The parameters 'trnCstNo' and 'cstOrient' are optional and can be set to 0.

4.2.2.6 cstUUID

```
TRDP_UUID_T GNU_PACKED::cstUUID
```

UUID of the consist, provided by ETBN (TrainNetworkDirectory) Reference to static consist attributes 0 if not available (e.g.

unique consist identifier

Reference to static consist attributes, 0 if not available (e.g.

correction)

4.2.2.7 datasetLength

```
UINT32 GNU_PACKED::datasetLength
```

length of the data to transmit 0...1432

defined by user: length of data to transmit

4.2.2.8 defQos

UINT32 GNU_PACKED::defQos

default QoS for PD

default QoS for MD

4.2.2.9 defTtl

UINT32 GNU_PACKED::defTtl

default TTL for PD

default TTL for MD

4.2.2.10 destAddr

TRDP_IP_ADDR_T GNU_PACKED::destAddr

IP address of destination for this publishing.

4.2.2.11 deviceName

TRDP_NET_LABEL_T GNU_PACKED::deviceName

function device of ECSC which sends the telegram

function device of ED which sends the telegram

4.2.2.12 etbId

UINT8 GNU_PACKED::etbId

identification of the ETB the TTDB is computed for bit0: ETB0 (operational network) bit1: ETB1 (multimedia network) bit2: ETB2 (other network) bit3: ETB3 (other network)

identification of the ETB the TTDB is computed for 0: ETB0 (operational network) 1: ETB1 (multimedia network) 2: ETB2 (other network) 3: ETB3 (other network)

4.2.2.13 etbTopoCnt

UINT32 GNU_PACKED::etbTopoCnt

ETB topography counter.

set by user: ETB to use, '0' for consist local traffic

train network directory CRC

4.2.2.14 filterAddr

```
TRDP_IP_ADDR_T GNU_PACKED::filterAddr
```

Filter IP address, i.e IP address of the sender for this subscription, 0.0.0.0 in case all senders.

4.2.2.15 inhibit

```
UINT8 GNU_PACKED::inhibit
```

inauguration inhibit 0 = no inhibit request 1 = inhibit request

ETBN inhibit 0 = no action (keep old state) 1 = no inhibit request 2 = inhibit request.

4.2.2.16 isLead

```
ANTIVALENT8 GNU_PACKED::isLead
```

vehicle is leading

consist contains leading vehicle, '01'B = false, '10'B = true

4.2.2.17 leadDir

```
UINT8 GNU_PACKED::leadDir
```

vehicle leading direction 0 = not relevant 1 = leading direction 1 2 = leading direction 2

'vehicle leading direction 0 = not relevant 1 = leading direction 1 2 = leading direction 2

4.2.2.18 leadVehOfCst

```
UINT8 GNU_PACKED::leadVehOfCst
```

position of leading vehicle in consist, 0..31 (1: first vehicle in consist in Direction 1, 2: second vehicle, etc.)

position of leading vehicle in consist range 0...32 0 = not defined 1 = first vehicle in consist in direction 1 2 = second vehicle etc.

4.2.2.19 lifesign

```
UINT16 GNU_PACKED::lifesign
```

wrap-around counter, incremented with each produced datagram.

4.2.2.20 msgType

UINT16 GNU_PACKED::msgType

of datagram: PD Request (0x5072) or PD_MSG (0x5064)

of datagram: Mn, Mr, Mp, Mq, Mc or Me

4.2.2.21 numCrcErr

UINT32 GNU_PACKED::numCrcErr

number of received PD packets with CRC err

number of received MD packets with CRC err

4.2.2.22 numMissed

UINT32 GNU_PACKED::numMissed

number of packets skipped

number of packets skipped for this subscription

4.2.2.23 numProtErr

UINT32 GNU_PACKED::numProtErr

number of received PD packets with protocol err

number of received MD packets with protocol err

4.2.2.24 numRcv

UINT32 GNU_PACKED::numRcv

number of received PD packets

number of received MD packets

4.2.2.25 numRecv

UINT32 GNU_PACKED::numRecv

Number of packets received for this subscription.

Number of received packets.

4.2.2.26 numSend

UINT32 GNU_PACKED::numSend

number of sent PD packets

Number of packets sent out.

number of sent MD packets

4.2.2.27 numTopoErr

UINT32 GNU_PACKED::numTopoErr

number of received PD packets with wrong topo count

number of received MD packets with wrong topo count

4.2.2.28 opCstList

TRDP_OP_CONSIST_T GNU_PACKED::opCstList [TRDP_MAX_CST_CNT]

operational consist list starting with op.

consist #1 Note: This is a variable size array, only opCstCnt array elements are present

4.2.2.29 opTrnDirState

UINT8 GNU_PACKED::opTrnDirState

train directory state 1 = INVALID 2 = VALID 4 = SHARED other values are not allowed

Operational train directory status: '01'B == invalid, '10'B == valid, '100'B == shared.

4.2.2.30 opTrnTopoCnt

UINT32 GNU_PACKED::opTrnTopoCnt

operational train topology counter

set by user: direction/side critical, '0' if ignored

operational train topology counter computed as defined in 5.3.3.2.16 (seed value : trnTopoCnt)

operational train topology counter set to 0 if opTrnDirState == invalid

operational train topocounter value of the operational train directory the correction is based on

4.2.2.31 opVehList

```
TRDP_OP_VEHICLE_T GNU_PACKED::opVehList [TRDP_MAX_VEH_CNT]
```

operational vehicle list starting with op.

vehicle #1 Note: This is a variable size array, only opCstCnt array elements are present

4.2.2.32 ownOpCstNo

```
UINT8 GNU_PACKED::ownOpCstNo
```

own operational address (= 1..32) = 0 if unknown (e.g.

operational consist number the vehicle belongs to

after Inauguration)

4.2.2.33 protocolVersion

```
UINT16 GNU_PACKED::protocolVersion
```

fix value for compatibility (set by the API)

fix value for compatibility

4.2.2.34 reserved01 [1/2]

```
UINT16 GNU_PACKED::reserved01
```

reserved (=0)

reserved for future use (= 0)

4.2.2.35 reserved01 [2/2]

```
UINT8 GNU_PACKED::reserved01
```

reserved (=0)

reserved for future use (= 0)

4.2.2.36 reserved02 [1/2]

```
UINT16 GNU_PACKED::reserved02
```

reserved (=0)

reserved (= 0)

reserved for future use (= 0)

4.2.2.37 reserved02 [2/2]

UINT16 GNU_PACKED::reserved02

reserved (=0)

reserved (= 0)

4.2.2.38 reserved03

UINT8 GNU_PACKED::reserved03

reserved (=0)

reserved for future use (= 0)

4.2.2.39 reserved04

UINT8 GNU_PACKED::reserved04

reserved (=0)

reserved for future use (= 0)

4.2.2.40 reserved06

UINT8 GNU_PACKED::reserved06

reserved (=0)

reserved for future use (= 0)

4.2.2.41 safetyTrail

TRDP_ETB_CTRL_VDP_T GNU_PACKED::safetyTrail

ETBCTRL-VDP trailer, completely set to 0 == not used.

ETBCTRL-VDP trailer, parameter 'safeSequCount' == 0 completely set to 0 == not used.

ETBCTRL-VDP trailer, parameter 'safeSequCount' == 0 completely set to 0 == SDTv2 not used.

ETBCTRL-VDP trailer, completely set to 0 == SDTv2 not used.

4.2.2.42 serviceEntry

[SRM_SERVICE_INFO_T](#) GNU_PACKED::serviceEntry[1]

var.

number of entries

4.2.2.43 timeout

```
UINT32 GNU_PACKED::timeout
```

Time-out value in us.

0 = No time-out supervision

4.2.2.44 toBehav

```
UINT32 GNU_PACKED::toBehav
```

Behavior at time-out.

Set data to zero / keep last value

4.2.2.45 trnCstNo

```
UINT8 GNU_PACKED::trnCstNo
```

own TCN consist number (= 1..32)

sequence number of consist in train with vehicle 01 being the first vehicle in ETB reference direction 1 as defined in IEC61375-2-5, value range: 1..63, 0 = inserted by correction

train consist number telegram control type 0 = with trnTopoCnt tracking 1 = without trnTopoCnt tracking

Sequence number of consist in train (1..63)

4.2.2.46 trnDirState

```
UINT8 GNU_PACKED::trnDirState
```

train directory state 1 = UNCONFIRMED 2 = CONFIRMED other values are not allowed

TTDB status: '01'B == unconfirmed, '10'B == confirmed.

4.2.2.47 trnId

```
TRDP_NET_LABEL_T GNU_PACKED::trnId
```

train identifier, application defined (e.g.

'ICE75', 'IC346'), informal

4.2.2.48 trnNetDir

```
TRDP_TRAIN_NET_DIR_T GNU_PACKED::trnNetDir
```

dynamic train info

network directory

4.2.2.49 trnOperator

`TRDP_NET_LABEL_T GNU_PACKED::trnOperator`

train operator, e.g.

'trenitalia.it', informal

4.2.2.50 trnTopoCnt

`UINT32 GNU_PACKED::trnTopoCnt`

trnTopoCnt value ctrlType == 0: actual value ctrlType == 1: set to 0

computed as defined in 5.3.3.2.16 (seed value: etbTopoCnt)

4.2.2.51 trnVehNo

`UINT8 GNU_PACKED::trnVehNo`

vehicle sequence number within the train with vehicle 01 being the first vehicle in ETB reference direction 1 as defined in IEC61375-2-5 value range: 0..63 a value of 0 indicates that this vehicle has been inserted by correction

vehicle sequence number within the train with vehicle 01 being the first vehicle in ETB reference direction 1 as defined in IEC61375-2-5, value range: 1..63, a value of 0 indicates that this vehicle has been inserted by correction

4.2.2.52 vehId

`TRDP_NET_LABEL_T GNU_PACKED::vehId`

Unique vehicle identifier, application defined (e.g.

UIC Identifier)

4.2.2.53 vehOrient

`UINT8 GNU_PACKED::vehOrient`

vehicle orientation 0 = not known (corrected vehicle) 1 = same as operational train direction 2 = inverse to operational train direction

vehicle orientation, '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction

- [UINT32 updPkts](#)
Counter for updated packets (statistics)
- [UINT32 getPkts](#)
Counter for read packets (statistics)
- [UINT32 numMissed](#)
Counter for skipped sequence number (statistics)
- [TRDP_ERR_T lastErr](#)
Last error (timeout)
- [TRDP_PRIV_FLAGS_T privFlags](#)
private flags
- [TRDP_FLAGS_T pktFlags](#)
flags
- [TRDP_TIME_T interval](#)
time out value for received packets or interval for packets to send (set from ms)
- [TRDP_TIME_T timeToGo](#)
next time this packet must be sent/rcv
- [TRDP_TO_BEHAVIOR_T toBehavior](#)
timeout behavior for packets
- [UINT32 dataSize](#)
net data size
- [UINT32 grossSize](#)
complete packet size (header, data)
- [UINT32 sendSize](#)
data size sent out
- [TRDP_DATASET_T * pCachedDS](#)
Pointer to dataset element if known.
- [INT32 socketIdx](#)
index into the socket list
- [const void * pUserRef](#)
from subscribe()
- [TRDP_PD_CALLBACK_T pfCbFunction](#)
Pointer to PD callback function.
- [PD_PACKET_T * pFrame](#)
header ...

4.5.1 Detailed Description

Queue element for PD packets to send or receive.

4.5.2 Field Documentation

4.5.2.1 pFrame

```
PD_PACKET_T* PD_ELE::pFrame
```

header ...

data + FCS...

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

- [trdp_private.h](#)

4.6 service_info Struct Reference

Preliminary definition of a service info entry.

```
#include <trdp_serviceRegistry.h>
```

Data Fields

- [TRDP_NET_LABEL_T](#) [srvName](#)
– service short name as defined in X
- [UINT32](#) [serviceld](#)
– High Byte = serviceInstanceld Low 24 Bits = serviceTypeld
- [TRDP_SHORT_VERSION_T](#) [srvVers](#)
– service version
- [UINT8](#) [srvFlags](#)
– Flags Bit0: 0 = non safety related; 1 = safety related Bit1: 0 = global service 1 = local service Bit3: 0 = complete service list 1 = service list update Bit4: 0 = add service (update only) 1 = delete service (update only) Bit2-7: reserved for future use (= 0)
- [UINT8](#) [reserved01](#)
– reserved for future use (= 0)
- [TIMEDATE64](#) [srvTTL](#)
– Time to Live (us or ns?)
- [TRDP_NET_LABEL_T](#) [fctDev](#)
– host identification of the function device the service is located on.
- [UINT8](#) [cstVehNo](#)
– sequence number of the vehicle within the consist (1..32)
- [UINT8](#) [cstNo](#)
– sequence number of the consist (1..63)
- [UINT16](#) [reserved03](#)
– reserved for future use (= 0)
- [UINT32](#) [addInfo](#) [3]
– service specific information

4.6.1 Detailed Description

Preliminary definition of a service info entry.

4.6.2 Field Documentation

4.6.2.1 fctDev

[TRDP_NET_LABEL_T](#) service_info::fctDev

– host identification of the function device the service is located on.

Defined in IEC 61375-2-3.

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

- [trdp_serviceRegistry.h](#)

4.7 srv_info_req Struct Reference

Preliminary definition of a service info request.

```
#include <trdp_serviceRegistry.h>
```

Data Fields

- [TRDP_SHORT_VERSION_T](#) [version](#)
– *version of the telegram mainVersion = 1 subversion = 0*
- [UINT16](#) [reserved01](#)
– *reserved for future use (= 0)*
- [UINT32](#) [trnTopoCnt](#)
– *trnTopoCnt value*
- [UINT16](#) [reserved02](#)
– *reserved for future use (= 0)*
- [UINT8](#) [reserved03](#)
– *reserved for future use (= 0)*
- [UINT8](#) [cstCnt](#)
– *number of consists in list if set to 255 all consists are requested to resend their SRVINFO telegram if set to >0 and <64 only consists with different srvTopoCnt value are requested to resend their SRVINFO telegram*
- [UINT32](#) [srvTcList](#) []
– *list of srvTopoCnt values obtained from all consists set to 0 if unknown ordered list starting with trnCstNo = 1*

4.7.1 Detailed Description

Preliminary definition of a service info request.

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

- [trdp_serviceRegistry.h](#)

4.8 TAU_MARSHALL_INFO_T Struct Reference

Marshalling info, used to and from wire.

Data Fields

- INT32 [level](#)
track recursive level
- UINT8 * [pSrc](#)
source pointer
- UINT8 * [pSrcEnd](#)
last source
- UINT8 * [pDst](#)
destination pointer
- UINT8 * [pDstEnd](#)
last destination

4.8.1 Detailed Description

Marshalling info, used to and from wire.

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

- [tau_marshall.c](#)

4.9 TCN_URI Struct Reference

TCN-DNS simplified header structures.

```
#include <tau_dnr_types.h>
```

Data Fields

- CHAR8 [tcnUriStr](#) [80]
if != 0 use TCN DNS as resolver
- INT16 [resolvState](#)
on request: reserved (= 0), on reply: -1 unknown, 0 OK
- UINT32 [tcnUriIpAddr](#)
IP address of URI.
- UINT32 [tcnUriIpAddr2](#)
if != 0, end IP address of range

4.9.1 Detailed Description

TCN-DNS simplified header structures.

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

- [tau_dnr_types.h](#)

4.10 TRDP_CLTR_CST_INFO_T Struct Reference

Closed train consists information.

```
#include <tau_tti_types.h>
```

Data Fields

- [TRDP_UUID_T cltrCstUUID](#)
closed train consist UUID
- [UINT8 cltrCstOrient](#)
closed train consist orientation '01'B = same as closed train direction '10'B = inverse to closed train direction
- [UINT8 cltrCstNo](#)
sequence number of the consist within the closed train, value range 1..32
- [UINT16 reserved01](#)
reserved for future use (= 0)

4.10.1 Detailed Description

Closed train consists information.

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

- [tau_tti_types.h](#)

4.11 TRDP_COM_PARAM_T Struct Reference

Quality/type of service, time to live , no.

```
#include <trdp_types.h>
```

Data Fields

- [UINT8 qos](#)
Quality of service (default should be 2 for PD and 2 for MD, TSN priority ≥ 3)
- [UINT8 ttl](#)
Time to live (default should be 64)
- [UINT8 retries](#)
MD Retries from XML file.
- [BOOL8 tsn](#)
if TRUE, do not schedule packet but use TSN socket
- [UINT16 vlan](#)
VLAN Id to be used.

4.11.1 Detailed Description

Quality/type of service, time to live , no.

of retries, TSN flag and VLAN ID

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

- [trdp_types.h](#)

4.12 TRDP_COMID_DSID_MAP_T Struct Reference

ComId - data set mapping element definition.

```
#include <trdp_types.h>
```

Data Fields

- [UINT32 comId](#)
comId
- [UINT32 dataSetId](#)
corresponding dataset Id

4.12.1 Detailed Description

ComId - data set mapping element definition.

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

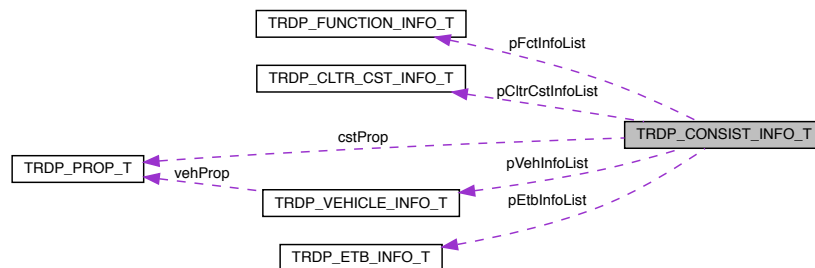
- [trdp_types.h](#)

4.13 TRDP_CONSIST_INFO_T Struct Reference

consist information structure

```
#include <tau_tti_types.h>
```

Collaboration diagram for TRDP_CONSIST_INFO_T:



Data Fields

- [TRDP_SHORT_VERSION_T](#) [version](#)
ConsistInfo data structure version, application defined mainVersion = 1, subVersion = 0.
- [UINT8](#) [cstClass](#)
consist info classification 1 = (single) consist 2 = closed train 3 = closed train consist
- [UINT8](#) [reserved01](#)
reserved for future use (= 0)
- [TRDP_NET_LABEL_T](#) [cstId](#)
application defined consist identifier, e.g.
- [TRDP_NET_LABEL_T](#) [cstType](#)
consist type, application defined
- [TRDP_NET_LABEL_T](#) [cstOwner](#)
consist owner, e.g.
- [TRDP_UUID_T](#) [cstUUID](#)
consist UUID
- [UINT32](#) [reserved02](#)
reserved for future use (= 0)
- [TRDP_PROP_T](#) [cstProp](#)
static consist properties
- [UINT16](#) [reserved03](#)
reserved for future use (= 0)
- [UINT16](#) [etbCnt](#)
number of ETB's, range: 1..4
- [TRDP_ETB_INFO_T](#) * [pEtbInfoList](#)
ETB information list for the consist Ordered list starting with lowest etbld.
- [UINT16](#) [reserved04](#)
reserved for future use (= 0)
- [UINT16](#) [vehCnt](#)
number of vehicles in consist 1..32

- [TRDP_VEHICLE_INFO_T](#) * [pVehInfoList](#)
vehicle info list for the vehicles in the consist Ordered list starting with cstVehNo==1
- UINT16 [reserved05](#)
reserved for future use (= 0)
- UINT16 [fctCnt](#)
number of consist functions value range 0..1024
- [TRDP_FUNCTION_INFO_T](#) * [pFctInfoList](#)
function info list for the functions in consist lexicographical ordered by fctName
- UINT16 [reserved06](#)
reserved for future use (= 0)
- UINT16 [cltrCstCnt](#)
number of original consists in closed train value range: 0..32, 0 = consist is no closed train
- [TRDP_CLTR_CST_INFO_T](#) * [pCltrCstInfoList](#)
info on closed train composition Ordered list starting with cltrCstNo == 1
- UINT32 [cstTopoCnt](#)
consist topology counter computed as defined in 5.3.3.2.16, seed value: 'FFFFFFFF'H

4.13.1 Detailed Description

consist information structure

4.13.2 Field Documentation

4.13.2.1 cstId

[TRDP_NET_LABEL_T](#) [TRDP_CONSIST_INFO_T::cstId](#)

application defined consist identifier, e.g.

UIC identifier

4.13.2.2 cstOwner

[TRDP_NET_LABEL_T](#) [TRDP_CONSIST_INFO_T::cstOwner](#)

consist owner, e.g.

"trenitalia.it", "snCF.fr", "db.de"

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

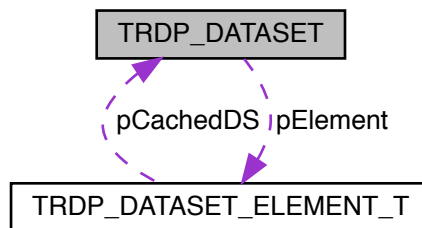
- [tau_tti_types.h](#)

4.14 TRDP_DATASET Struct Reference

Dataset definition.

```
#include <trdp_types.h>
```

Collaboration diagram for TRDP_DATASET:



Data Fields

- `UINT32 id`
dataset identifier > 1000
- `UINT16 reserved1`
Reserved for future use, must be zero.
- `UINT16 numElement`
Number of elements.
- `TRDP_DATASET_ELEMENT_T pElement []`
Pointer to a dataset element, used as array.

4.14.1 Detailed Description

Dataset definition.

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

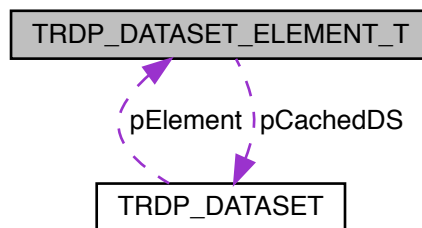
- `trdp_types.h`

4.15 TRDP_DATASET_ELEMENT_T Struct Reference

Dataset element definition.

```
#include <trdp_types.h>
```

Collaboration diagram for TRDP_DATASET_ELEMENT_T:



Data Fields

- **UINT32** [type](#)
Data type (TRDP_DATA_TYPE_T 1...99) or dataset id > 1000.
- **UINT32** [size](#)
Number of items or TRDP_VAR_SIZE (0)
- **CHAR8** * [name](#)
Name param, on special request (Ticket #211)
- **CHAR8** * [unit](#)
Unit text for visualisation.
- **REAL32** [scale](#)
Factor for visualisation.
- **INT32** [offset](#)
*Offset for visualisation (val = scale * x + offset)*
- **struct** [TRDP_DATASET](#) * [pCachedDS](#)
Used internally for marshallng speed-up.

4.15.1 Detailed Description

Dataset element definition.

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

- [trdp_types.h](#)

4.16 TRDP_DBG_CONFIG_T Struct Reference

Control for debug output device/file on application level.

```
#include <tau_xml.h>
```

Data Fields

- TRDP_DBG_OPTION_T [option](#)
Debug printout options for application use.
- UINT32 [maxFileSize](#)
Maximal file size.
- TRDP_FILE_NAME_T [fileName](#)
Debug file name and path.

4.16.1 Detailed Description

Control for debug output device/file on application level.

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

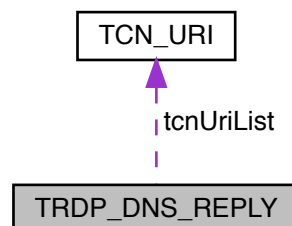
- [tau_xml.h](#)

4.17 TRDP_DNS_REPLY Struct Reference

TCN-DNS Reply telegram TCN_DNS_REP_DS.

```
#include <tau_dnr_types.h>
```

Collaboration diagram for TRDP_DNS_REPLY:



Data Fields

- TRDP_SHORT_VERSION_T [version](#)
1.0
- TRDP_NET_LABEL_T [deviceName](#)
function device of ED which sends the telegram
- UINT32 [etbTopoCnt](#)
ETB topography counter.
- UINT32 [opTrnTopoCnt](#)
operational train topography counter needed for TCN-URLs related to the operational train view = 0 if not used
- UINT8 [etbId](#)
identification of the related ETB 0 = ETB0 (operational network) 1 = ETB1 (multimedia network) 2 = ETB2 (other network) 3 = ETB3 (other network) 255 = don't care (for access to local DNS server)
- INT8 [dnsStatus](#)
0 = OK -1 = DNS Server not ready -2 = Inauguration in progress
- UINT8 [tcnUriCnt](#)
number of TCN-URLs to be resolved value range: 0 .
- TCN_URI_T [tcnUriList](#) [255]
defined for max size
- TRDP_ETB_CTRL_VDP_T [safetyTrail](#)
SDT trailer.

4.17.1 Detailed Description

TCN-DNS Reply telegram TCN_DNS_REP_DS.

4.17.2 Field Documentation

4.17.2.1 [tcnUriCnt](#)

UINT8 TRDP_DNS_REPLY::tcnUriCnt

number of TCN-URLs to be resolved value range: 0 .

. 255

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

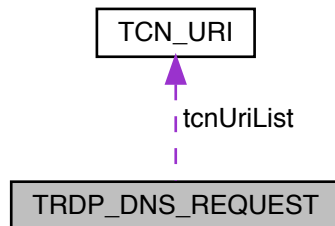
- [tau_dnr_types.h](#)

4.18 TRDP_DNS_REQUEST Struct Reference

TCN-DNS Request telegram TCN_DNS_REQ_DS.

```
#include <tau_dnr_types.h>
```

Collaboration diagram for TRDP_DNS_REQUEST:



Data Fields

- TRDP_SHORT_VERSION_T [version](#)
1.0
- TRDP_NET_LABEL_T [deviceName](#)
function device of ED which sends the telegram
- UINT32 [etbTopoCnt](#)
ETB topography counter.
- UINT32 [opTrnTopoCnt](#)
operational train topography counter needed for TCN-URIs related to the operational train view = 0 if not used
- UINT8 [etbld](#)
identification of the related ETB 0 = ETB0 (operational network) 1 = ETB1 (multimedia network) 2 = ETB2 (other network) 3 = ETB3 (other network) 255 = don't care (for access to local DNS server)
- UINT8 [tcnUriCnt](#)
number of TCN-URIs to be resolved value range: 0 .
- TCN_URI_T [tcnUriList](#) [255]
defined for max size
- TRDP_ETB_CTRL_VDP_T [safetyTrail](#)
SDT trailer.

4.18.1 Detailed Description

TCN-DNS Request telegram TCN_DNS_REQ_DS.

4.18.2 Field Documentation

4.18.2.1 tcnUriCnt

```
UINT8 TRDP_DNS_REQUEST::tcnUriCnt
```

number of TCN-URIs to be resolved value range: 0 .

. 255

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

- [tau_dnr_types.h](#)

4.19 TRDP_ETB_INFO_T Struct Reference

Types for train configuration information.

```
#include <tau_tti_types.h>
```

Data Fields

- UINT8 [etbId](#)
identification of train backbone; value range: 0..3
- UINT8 [cnCnt](#)
number of CNs within consist connected to this ETB value range 1..16 referring to cnId 0..15 acc.
- UINT16 [reserved01](#)
reserved for future use (= 0)

4.19.1 Detailed Description

Types for train configuration information.

ETB information

4.19.2 Field Documentation

4.19.2.1 cnCnt

```
UINT8 TRDP_ETB_INFO_T::cnCnt
```

number of CNs within consist connected to this ETB value range 1..16 referring to cnId 0..15 acc.

IEC61375-2-5

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

- [tau_tti_types.h](#)

4.20 TRDP_FUNCTION_INFO_T Struct Reference

function/device information structure

```
#include <tau_tti_types.h>
```

Data Fields

- [TRDP_NET_LABEL_T fctName](#)
function device or group label
- [UINT16 fctId](#)
host identification of the function device or group as defined in IEC 61375-2-5, application defined.
- [BOOL8 grp](#)
is a function group and will be resolved as IP multicast address
- [UINT8 reserved01](#)
reserved for future use (= 0)
- [UINT8 cstVehNo](#)
Sequence number of the vehicle in the consist the function belongs to.
- [UINT8 etbId](#)
number of connected train backbone.
- [UINT8 cnId](#)
identifier of connected consist network in the consist, related to the etbId.
- [UINT8 reserved02](#)
reserved for future use (= 0)

4.20.1 Detailed Description

function/device information structure

4.20.2 Field Documentation

4.20.2.1 cnId

```
UINT8 TRDP_FUNCTION_INFO_T::cnId
```

identifier of connected consist network in the consist, related to the etbId.

Value range: 0..31

4.20.2.2 cstVehNo

```
UINT8 TRDP_FUNCTION_INFO_T::cstVehNo
```

Sequence number of the vehicle in the consist the function belongs to.

Value range: 1..16, 0 = not defined

4.20.2.3 etbId

UINT8 TRDP_FUNCTION_INFO_T::etbId

number of connected train backbone.

Value range: 0..3

4.20.2.4 fctId

UINT16 TRDP_FUNCTION_INFO_T::fctId

host identification of the function device or group as defined in IEC 61375-2-5, application defined.

Value range: 1..16383 (device), 256..16383 (group)

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

- [tau_tti_types.h](#)

4.21 TRDP_HANDLE Struct Reference

Hidden handle definition, used as unique addressing item.

```
#include <trdp_private.h>
```

Data Fields

- [UINT32 comId](#)
comId for packets to send/receive
- [TRDP_IP_ADDR_T srcIpAddr](#)
source IP for PD/MD
- [TRDP_IP_ADDR_T srcIpAddr2](#)
second source IP for PD/MD
- [TRDP_IP_ADDR_T destIpAddr](#)
destination IP for PD
- [TRDP_IP_ADDR_T mcGroup](#)
multicast group to join for PD
- [UINT32 etbTopoCnt](#)
etb topocount belongs to addressing item
- [UINT32 opTrnTopoCnt](#)
opTrn topocount belongs to addressing item
- [UINT32 serviceId](#)
group of services this packet belongs to

4.21.1 Detailed Description

Hidden handle definition, used as unique addressing item.

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

- [trdp_private.h](#)

4.22 TRDP_IDX_TABLE_T Struct Reference

Settings for pre-allocation of index tables for application session initialization.

```
#include <trdp_types.h>
```

Data Fields

- UINT32 [maxNoOfLowCatSubscriptions](#)
Max.
- UINT32 [maxNoOfMidCatSubscriptions](#)
Max.
- UINT32 [maxNoOfHighCatSubscriptions](#)
Max.
- UINT32 [maxNoOfLowCatPublishers](#)
Max.
- UINT32 [maxDepthOfLowCatPublishers](#)
depth / overlapped publishers with intervals <= 100ms
- UINT32 [maxNoOfMidCatPublishers](#)
Max.
- UINT32 [maxDepthOfMidCatPublishers](#)
depth / overlapped publishers with intervals <= 1000ms
- UINT32 [maxNoOfHighCatPublishers](#)
Max.
- UINT32 [maxDepthOfHighCatPublishers](#)
depth / overlapped publishers with intervals <= 10000ms
- UINT32 [maxNoOfExtPublishers](#)
Max.

4.22.1 Detailed Description

Settings for pre-allocation of index tables for application session initialization.

4.22.2 Field Documentation

4.22.2.1 maxNoOfExtPublishers

UINT32 TRDP_IDX_TABLE_T::maxNoOfExtPublishers

Max.

number of expected publishers with intervals > 10000ms

4.22.2.2 maxNoOfHighCatPublishers

UINT32 TRDP_IDX_TABLE_T::maxNoOfHighCatPublishers

Max.

number of expected publishers with intervals <= 10000ms

4.22.2.3 maxNoOfHighCatSubscriptions

UINT32 TRDP_IDX_TABLE_T::maxNoOfHighCatSubscriptions

Max.

number of expected subscriptions with intervals > 1000ms

4.22.2.4 maxNoOfLowCatPublishers

UINT32 TRDP_IDX_TABLE_T::maxNoOfLowCatPublishers

Max.

number of expected publishers with intervals <= 100ms

4.22.2.5 maxNoOfLowCatSubscriptions

UINT32 TRDP_IDX_TABLE_T::maxNoOfLowCatSubscriptions

Max.

number of expected subscriptions with intervals <= 100ms

4.22.2.6 maxNoOfMidCatPublishers

UINT32 TRDP_IDX_TABLE_T::maxNoOfMidCatPublishers

Max.

number of expected publishers with intervals <= 1000ms

4.22.2.7 maxNoOfMidCatSubscriptions

UINT32 TRDP_IDX_TABLE_T::maxNoOfMidCatSubscriptions

Max.

number of expected subscriptions with intervals $\leq 1000\text{ms}$

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

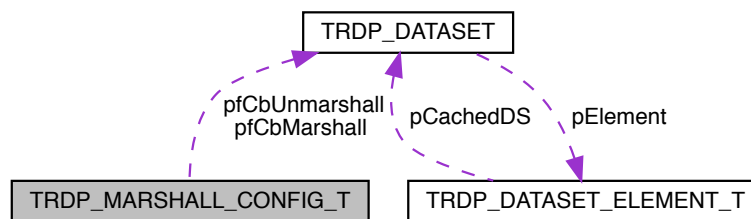
- [trdp_types.h](#)

4.23 TRDP_MARSHALL_CONFIG_T Struct Reference

Marshaling/unmarshalling configuration.

```
#include <trdp_types.h>
```

Collaboration diagram for TRDP_MARSHALL_CONFIG_T:



Data Fields

- [TRDP_MARSHALL_T pfCbMarshall](#)
Pointer to marshall callback function.
- [TRDP_UNMARSHALL_T pfCbUnmarshall](#)
Pointer to unmarshall callback function.
- void * [pRefCon](#)
Pointer to user context for call back.

4.23.1 Detailed Description

Marshaling/unmarshalling configuration.

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

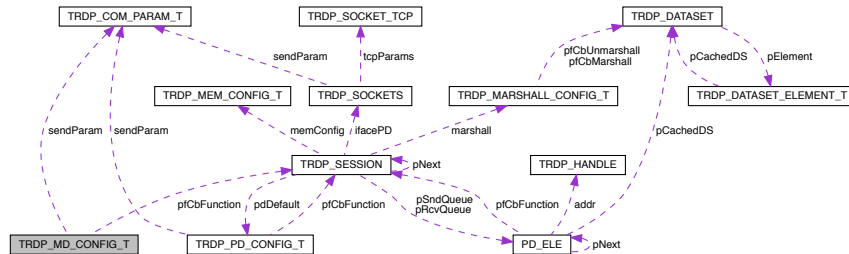
- [trdp_types.h](#)

4.24 TRDP_MD_CONFIG_T Struct Reference

Default MD configuration.

```
#include <trdp_types.h>
```

Collaboration diagram for TRDP_MD_CONFIG_T:



Data Fields

- [TRDP_MD_CALLBACK_T pCbFunction](#)
Pointer to MD callback function.
- void * [pRefCon](#)
Pointer to user context for call back.
- [TRDP_SEND_PARAM_T sendParam](#)
Default send parameters.
- [TRDP_FLAGS_T flags](#)
Default flags for MD packets.
- [UINT32 replyTimeout](#)
Default reply timeout in us.
- [UINT32 confirmTimeout](#)
Default confirmation timeout in us.
- [UINT32 connectTimeout](#)
Default connection timeout in us.
- [UINT32 sendingTimeout](#)
Default sending timeout in us.
- [UINT16 udpPort](#)
Port to be used for UDP MD communication (default: 17225)
- [UINT16 tcpPort](#)
Port to be used for TCP MD communication (default: 17225)
- [UINT32 maxNumSessions](#)
Maximal number of replier sessions.

4.24.1 Detailed Description

Default MD configuration.

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

- [trdp_types.h](#)

4.25 TRDP_MD_INFO_T Struct Reference

Message data info from received telegram; allows the application to generate responses.

```
#include <trdp_types.h>
```

Data Fields

- [TRDP_IP_ADDR_T srcIpAddr](#)
source IP address for filtering
- [TRDP_IP_ADDR_T destIpAddr](#)
destination IP address for filtering
- [UINT32 seqCount](#)
sequence counter
- [UINT16 protVersion](#)
Protocol version.
- [TRDP_MSG_T msgType](#)
Protocol ('PD', 'MD', ...)
- [UINT32 comId](#)
ComID.
- [UINT32 etbTopoCnt](#)
received topocount
- [UINT32 opTrnTopoCnt](#)
received topocount
- [BOOL8 aboutToDie](#)
session is about to die
- [UINT32 numRepliesQuery](#)
number of ReplyQuery received
- [UINT32 numConfirmSent](#)
number of Confirm sent
- [UINT32 numConfirmTimeout](#)
number of Confirm Timeouts (incremented by listeners)
- [UINT16 userStatus](#)
error code, user stat
- [TRDP_REPLY_STATUS_T replyStatus](#)
reply status
- [TRDP_UUID_T sessionId](#)
for response
- [UINT32 replyTimeout](#)
reply timeout in us given with the request
- [TRDP_URI_USER_T srcUserURI](#)
source URI user part from MD header
- [TRDP_URI_HOST_T srcHostURI](#)
source URI host part (unused)
- [TRDP_URI_USER_T destUserURI](#)
destination URI user part from MD header
- [TRDP_URI_HOST_T destHostURI](#)
destination URI host part (unused)
- [UINT32 numExpReplies](#)
number of expected replies, 0 if unknown

- `UINT32 numReplies`
actual number of replies for the request
- `const void * pUserRef`
User reference given with the local call.
- `TRDP_ERR_T resultCode`
error code

4.25.1 Detailed Description

Message data info from received telegram; allows the application to generate responses.

Note: Not all fields are relevant for each message type!

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

- [trdp_types.h](#)

4.26 TRDP_MEM_CONFIG_T Struct Reference

Enumeration type for memory pre-fragmentation, reuse of VOS definition.

```
#include <trdp_types.h>
```

Data Fields

- `UINT8 * p`
pointer to static or allocated memory
- `UINT32 size`
size of static or allocated memory
- `UINT32 prealloc [VOS_MEM_NBLOCKSIZES]`
memory block structure

4.26.1 Detailed Description

Enumeration type for memory pre-fragmentation, reuse of VOS definition.

Structure describing memory (and its pre-fragmentation)

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

- [trdp_types.h](#)

Data Fields

- [TRDP_IP_ADDR_T srcIpAddr](#)
source IP address for filtering
- [TRDP_IP_ADDR_T destIpAddr](#)
destination IP address for filtering
- [UINT32 seqCount](#)
sequence counter
- [UINT16 protVersion](#)
Protocol version.
- [TRDP_MSG_T msgType](#)
Protocol ('PD', 'MD', ...)
- [UINT32 comId](#)
ComID.
- [UINT32 etbTopoCnt](#)
received ETB topocount
- [UINT32 opTrnTopoCnt](#)
received operational train directory topocount
- [UINT32 replyComId](#)
ComID for reply (request only)
- [TRDP_IP_ADDR_T replyIpAddr](#)
IP address for reply (request only)
- `const void * pUserRef`
User reference given with the local subscribe.
- [TRDP_ERR_T resultCode](#)
error code
- [TRDP_URI_HOST_T srcHostURI](#)
source URI host part (unused)
- [TRDP_URI_HOST_T destHostURI](#)
destination URI host part (unused)
- [TRDP_TO_BEHAVIOR_T toBehavior](#)
callback can decide about handling of data on timeout
- [UINT32 servId](#)
the reserved field of the PD header

4.28.1 Detailed Description

Process data info from received telegram; allows the application to generate responses.

Note: Not all fields are relevant for each message type!

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

- [trdp_types.h](#)

4.29 TRDP_PROCESS_CONFIG_T Struct Reference

Various flags/general TRDP options for library initialization.

```
#include <trdp_types.h>
```


Data Fields

- TRDP_LABEL_T [hostName](#)
Host name.
- TRDP_LABEL_T [leaderName](#)
Leader name dependant on redundancy concept.
- UINT32 [cycleTime](#)
TRDP main process cycle time in us.
- UINT32 [priority](#)
TRDP main process priority (0-255, 0=default, 255=highest)
- TRDP_OPTION_T [options](#)
TRDP options.

4.29.1 Detailed Description

Various flags/general TRDP options for library initialization.

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

- [trdp_types.h](#)

4.30 TRDP_PROP_T Struct Reference

Application defined properties.

```
#include <tau_tti_types.h>
```

Data Fields

- TRDP_SHORT_VERSION_T [ver](#)
properties version information, application defined
- UINT16 [len](#)
properties length in number of octets, application defined, must be a multiple of 4 octets for alignment reasons value range: 0..32768
- UINT8 [prop](#) [1]
properties, application defined

4.30.1 Detailed Description

Application defined properties.

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

- [tau_tti_types.h](#)

4.31 TRDP_SDT_PAR_T Struct Reference

Types to read out the XML configuration.

```
#include <tau_xml.h>
```

Data Fields

- UINT32 [smi1](#)
Safe message identifier - unique for this message at consist level.
- UINT32 [smi2](#)
Safe message identifier - unique for this message at consist level.
- UINT32 [cmThr](#)
Channel monitoring threshold.
- UINT16 [udv](#)
User data version.
- UINT16 [rxPeriod](#)
Sink cycle time.
- UINT16 [txPeriod](#)
Source cycle time.
- UINT16 [nGuard](#)
Initial timeout cycles.
- UINT8 [nrxFsafe](#)
Timeout cycles.
- UINT8 [reserved1](#)
Reserved for future use.
- UINT16 [lmiMax](#)
Latency monitoring cycles.

4.31.1 Detailed Description

Types to read out the XML configuration.

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

- [tau_xml.h](#)

4.32 TRDP_SEQ_CNT_ENTRY_T Struct Reference

Tuples of last received sequence counter per comId.

```
#include <trdp_private.h>
```

Data Fields

- [UINT32 lastSeqCnt](#)
Sequence counter value for comId.
- [TRDP_IP_ADDR_T srcIpAddr](#)
Source IP address.
- [TRDP_MSG_T msgType](#)
message type

4.32.1 Detailed Description

Tuples of last received sequence counter per comId.

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

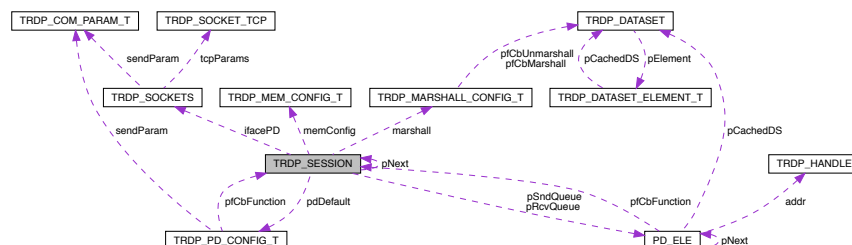
- [trdp_private.h](#)

4.33 TRDP_SESSION Struct Reference

Session/application variables store.

```
#include <trdp_private.h>
```

Collaboration diagram for TRDP_SESSION:



Data Fields

- [struct TRDP_SESSION * pNext](#)
Pointer to next session.
- [VOS_MUTEX_T mutex](#)
protect this session
- [VOS_MUTEX_T mutexTxPD](#)
protect the sending queue
- [VOS_MUTEX_T mutexRxPD](#)
protect the receiving queue
- [TRDP_IP_ADDR_T realIP](#)
Real IP address.

- [TRDP_IP_ADDR_T](#) `virtualIP`
Virtual IP address.
- `UINT32` [etbTopoCnt](#)
current valid topocount or zero
- `UINT32` [opTrnTopoCnt](#)
current valid topocount or zero
- [TRDP_TIME_T](#) `nextJob`
Store for next select interval.
- [TRDP_PRINT_DBG_T](#) `pPrintDebugString`
Pointer to function to print debug information.
- [TRDP_MARSHALL_CONFIG_T](#) `marshall`
Marshalling(unMarshalling) configuration.
- [TRDP_PD_CONFIG_T](#) `pdDefault`
Default configuration for process data.
- [TRDP_MEM_CONFIG_T](#) `memConfig`
Internal memory handling configuration.
- [TRDP_OPTION_T](#) `option`
Stack behavior options.
- [TRDP_SOCKETS_T](#) `ifacePD` [[TRDP_MAX_PD_SOCKET_CNT](#)]
Collection of sockets to use.
- [PD_ELE_T](#) * `pSndQueue`
pointer to first element of send queue
- [PD_ELE_T](#) * `pRcvQueue`
pointer to first element of rcv queue
- [PD_PACKET_T](#) * `pNewFrame`
pointer to received PD frame
- [TRDP_TIME_T](#) `initTime`
initialization time of session
- [TRDP_STATISTICS_T](#) `stats`
statistics of this session

4.33.1 Detailed Description

Session/application variables store.

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

- [trdp_private.h](#)

4.34 TRDP_SOCKET_TCP Struct Reference

TCP parameters.

```
#include <trdp_private.h>
```

Data Fields

- [TRDP_IP_ADDR_T cornerIp](#)
The other TCP corner Ip.
- BOOL8 [notSend](#)
If the message has been sent uncompleted.
- [TRDP_TIME_T connectionTimeout](#)
TCP socket connection Timeout.
- BOOL8 [sendNotOk](#)
The sending timeout will be start.
- [TRDP_TIME_T sendingTimeout](#)
The timeout sending the message.
- BOOL8 [addFileDesc](#)
Ready to add the socket in the fd.
- BOOL8 [morituri](#)
about to die

4.34.1 Detailed Description

TCP parameters.

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

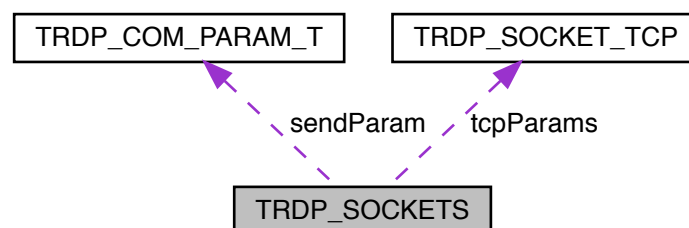
- [trdp_private.h](#)

4.35 TRDP_SOCKETS Struct Reference

Socket item.

```
#include <trdp_private.h>
```

Collaboration diagram for TRDP_SOCKETS:



Data Fields

- SOCKET [sock](#)
vos socket descriptor to use
- TRDP_IP_ADDR_T [bindAddr](#)
Defines the interface to use.
- TRDP_IP_ADDR_T [srcAddr](#)
Defines the source interface to use.
- TRDP_SEND_PARAM_T [sendParam](#)
Send parameters.
- TRDP_SOCKET_TYPE_T [type](#)
Usage of this socket.
- BOOL8 [rcvMostly](#)
Used for receiving.
- INT16 [usage](#)
No.
- TRDP_SOCKET_TCP_T [tcpParams](#)
Params used for TCP.
- TRDP_IP_ADDR_T [mcGroups](#) [VOS_MAX_MULTICAST_CNT]
List of multicast addresses for this socket.

4.35.1 Detailed Description

Socket item.

4.35.2 Field Documentation

4.35.2.1 usage

```
INT16 TRDP_SOCKETS::usage
```

No.

of current users of this socket

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

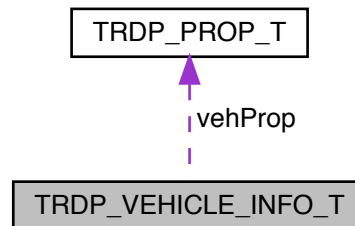
- [trdp_private.h](#)

4.36 TRDP_VEHICLE_INFO_T Struct Reference

vehicle information structure

```
#include <tau_tti_types.h>
```

Collaboration diagram for TRDP_VEHICLE_INFO_T:



Data Fields

- [TRDP_NET_LABEL_T](#) `vehId`
vehicle identifier label,application defined (e.g.
- [TRDP_NET_LABEL_T](#) `vehType`
vehicle type,application defined
- `UINT8` [vehOrient](#)
vehicle orientation '01'B = same as consist direction '10'B = inverse to consist direction
- `UINT8` [cstVehNo](#)
Sequence number of vehicle in consist(1..16)
- `ANTIVALENT8` [tractVeh](#)
vehicle is a traction vehicle '01'B = vehicle is not a traction vehicle '10'B = vehicle is a traction vehicle
- `UINT8` [reserved01](#)
for future use (= 0)
- [TRDP_PROP_T](#) `vehProp`
static vehicle properties

4.36.1 Detailed Description

vehicle information structure

4.36.2 Field Documentation

4.36.2.1 vehId

`TRDP_NET_LABEL_T TRDP_VEHICLE_INFO_T::vehId`

vehicle identifier label,application defined (e.g.

UIC vehicle identification number) vehId of vehicle with vehNo==1 is used also as cstId

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

- [tau_tti_types.h](#)

4.37 TRDP_XML_DOC_HANDLE_T Struct Reference

Parsed XML document handle.

```
#include <tau_xml.h>
```

Data Fields

- struct XML_HANDLE * [pXmlDocument](#)
XML document context.

4.37.1 Detailed Description

Parsed XML document handle.

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

- [tau_xml.h](#)

4.38 VOS_SOCK_OPT_T Struct Reference

Common socket options.

```
#include <vos_sock.h>
```


Data Fields

- [UINT8 qos](#)
quality/type of service 0...7
- [UINT8 ttl](#)
time to live for unicast (default 64)
- [UINT8 ttl_multicast](#)
time to live for multicast
- [BOOL8 reuseAddrPort](#)
allow reuse of address and port
- [BOOL8 nonBlocking](#)
use non blocking calls
- [BOOL8 no_mc_loop](#)
no multicast loop back
- [BOOL8 no_udp_crc](#)
supress udp crc computation
- [BOOL8 txTime](#)
use transmit time on send, if available
- [BOOL8 raw](#)
use raw socket, not for receiver!
- [CHAR8 ifName \[VOS_MAX_IF_NAME_SIZE\]](#)
interface name if available

4.38.1 Detailed Description

Common socket options.

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

- [vos_sock.h](#)

4.39 VOS_VERSION_T Struct Reference

Version information.

```
#include <vos_types.h>
```

Data Fields

- [UINT8 ver](#)
Version - incremented for incompatible changes.
- [UINT8 rel](#)
Release - incremented for compatible changes.
- [UINT8 upd](#)
Update - incremented for bug fixes.
- [UINT8 evo](#)
Evolution - incremented for build.

4.39.1 Detailed Description

Version information.

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

- [vos_types.h](#)

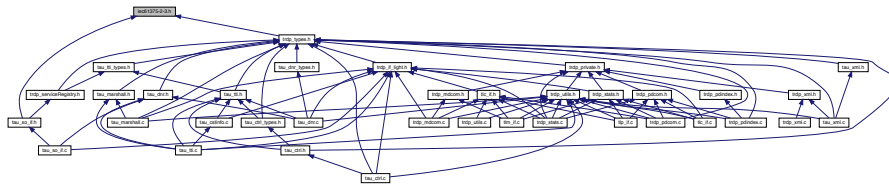
Chapter 5

File Documentation

5.1 iec61375-2-3.h File Reference

All definitions from IEC 61375-2-3.

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



Macros

- `#define ETB_WAIT_TIMER_VALUE 5u` /* Compute train dir. IEC61375-2-3 Ch. 5.3.2.3 */
Time out values (in seconds)
- `#define TRDP_PD_UDP_PORT 17224u`
TRDP defines (from former trpd_proto.h)
- `#define TRDP_MD_UDP_PORT 17225u`
IANA assigned message data UDP port.
- `#define TRDP_MD_TCP_PORT 17225u`
IANA assigned message data TCP port.
- `#define TRDP_PROTO_VER 0x0100u`
Protocol version.
- `#define TRDP_PROTOCOL_VERSION_CHECK_MASK 0xFF00u`
Version check, two digits are relevant.
- `#define TRDP_SESS_ID_SIZE 16u`
Session ID (UUID) size in MD header.
- `#define TRDP_USR_URI_SIZE 32u`
max.
- `#define TRDP_MD_INFINITE_TIME (0)`
Definitions for time out behaviour accd.

- `#define TRDP_MD_DEFAULT_REPLY_TIMEOUT 5000000u`
Default MD communication parameters.
- `#define TRDP_MD_DEFAULT_CONFIRM_TIMEOUT 1000000u`
[us] default confirm time out 1s
- `#define TRDP_MD_DEFAULT_CONNECTION_TIMEOUT 60000000u`
[us] Socket connection time out 1min
- `#define TRDP_MD_DEFAULT_SENDING_TIMEOUT 5000000u`
[us] Socket sending time out 5s
- `#define TRDP_PD_DEFAULT_QOS 5u`
Default PD communication parameters.
- `#define TRDP_PD_DEFAULT_TIMEOUT 100000u`
[us] 100ms default PD timeout
- `#define TRDP_PROCESS_DEFAULT_CYCLE_TIME 10000u`
Default TRDP process options.
- `#define TRDP_PROCESS_DEFAULT_PRIORITY 64u`
Default priority of TRDP process.
- `#define TRDP_PROCESS_DEFAULT_OPTIONS TRDP_OPTION_TRAFFIC_SHAPING`
Default options for TRDP process.
- `#define TRDP_MIN_PD_HEADER_SIZE sizeof(PD_HEADER_T)`
PD packet properties.
- `#define TRDP_MAX_PD_DATA_SIZE 1432u`
PD data.
- `#define TRDP_MAX_MD_DATA_SIZE 65388u`
MD packet properties.
- `#define TRDP_MAX_MD_RETRIES 2u`
Maximum values.
- `#define TRDP_MAX_LABEL_LEN 16u`
label length incl.
- `#define TRDP_MAX_URI_USER_LEN (2u * TRDP_MAX_LABEL_LEN)`
URI user part incl.
- `#define TRDP_MAX_URI_HOST_LEN (5u * TRDP_MAX_LABEL_LEN)`
URI host part incl.
- `#define TRDP_MAX_URI_LEN (7u * TRDP_MAX_LABEL_LEN)`
URI length incl.
- `#define TRDP_MAX_FILE_NAME_LEN 128u`
path and file name length incl.
- `#define TRDP_VAR_SIZE 0u`
Variable size dataset.
- `#define TRDP_MSG_PD 0x5064u`
Message Types.
- `#define TRDP_MSG_PP 0x5070u`
'Pp' PD Data (Pull Reply)
- `#define TRDP_MSG_PR 0x5072u`
'Pr' PD Request
- `#define TRDP_MSG_PE 0x5065u`
'Pe' PD Error
- `#define TRDP_MSG_MN 0x4D6Eu`
'Mn' MD Notification (Request w/o reply)
- `#define TRDP_MSG_MR 0x4D72u`
'Mr' MD Request with reply
- `#define TRDP_MSG_MP 0x4D70u`

- *'Mp' MD Reply without confirmation*
- #define [TRDP_MSG_MQ](#) 0x4D71u
- *'Mq' MD Reply with confirmation*
- #define [TRDP_MSG_MC](#) 0x4D63u
- *'Mc' MD Confirm*
- #define [TRDP_MSG_ME](#) 0x4D65u
- *'Me' MD Error*
- #define [ETB0_ALL_END_DEVICES_IP](#) "239.193.0.0"
- *from Table 22*
- #define [ETB_CTRL_COMID](#) 1u
- *Reserved COMIDs in the range 1 ...*
- #define [ETB_CTRL_CYCLE](#) 500000u
- *[us] 0.5s*
- #define [ETB_CTRL_TO_US](#) 300000u
- *[us] 3s*
- #define [TRDP_ETBCTRL_COMID](#) [ETB_CTRL_COMID](#)
- *alternative name*
- #define [CSTINFO_COMID](#) 2u
- *Consist Info telegram (Message data notification 'Mn')*
- #define [TRDP_CSTINFO_COMID](#) [CSTINFO_COMID](#)
- *alternative name*
- #define [CSTINFOCTRL_COMID](#) 3u
- *Consist Info control/request telegram (Message data notification 'Mn')*
- #define [TRDP_CSTINFOCTRL_COMID](#) [CSTINFOCTRL_COMID](#)
- *alternative name*
- #define [TRDP_COMID_ECHO](#) 10u
- *Reserved in Annex D & E.*
- #define [TRDP_STATISTICS_PULL_COMID](#) 31u
- *reserved in Table A.2*
- #define [TRDP_GLOBAL_STATS_REPLY_COMID](#) 31u
- *reserved in D.3*
- #define [TTDB_STATUS_COMID](#) 100u
- *TTDB manager telegram PD.*
- #define [TTDB_STATUS_CYCLE](#) 1000000u
- *[us] 1s Push*
- #define [TTDB_STATUS_TO_US](#) 5000000u
- *[us] 5s*
- #define [TTDB_OP_DIR_INFO_COMID](#) 101u
- *TTDB manager telegram MD: Push the OP_TRAIN_DIRECTORY.*
- #define [TTDB_OP_DIR_INFO_DS](#) "TTDB_OP_TRAIN_DIRECTORY_INFO"
- *OP_TRAIN_DIRECTORY.*
- #define [TTDB_TRN_DIR_REQ_COMID](#) 102u
- *TTDB manager telegram MD: Get the TRAIN_DIRECTORY.*
- #define [TTDB_TRN_DIR_REQ_TO_US](#) 3000000u
- *3s timeout*
- #define [TTDB_TRN_DIR_REP_COMID](#) 103u
- *MD reply.*
- #define [TTDB_TRN_DIR_REP_DS](#) "TTDB_TRAIN_DIRECTORY_INFO_REPLY"
- *TRAIN_DIRECTORY.*
- #define [TTDB_STAT_CST_REQ_COMID](#) 104u
- *TTDB manager telegram MD: Get the static consist information.*

- #define `TTDB_STAT_CST_REQ_TO_US` 3000000u
[us] 3s timeout
- #define `TTDB_STAT_CST_REP_DS` "TTDB_STATIC_CONSIST_INFO_REPLY"
CONSIST_INFO.
- #define `TTDB_NET_DIR_REQ_COMID` 106u
TTDB manager telegram MD: Get the NETWORK_TRAIN_DIRECTORY.
- #define `TTDB_NET_DIR_REQ_TO_US` 3000000u
[us] 3s timeout
- #define `TTDB_NET_DIR_REP_COMID` 107u
MD reply.
- #define `TTDB_NET_DIR_REP_DS` "TTDB_TRAIN_NETWORK_DIRECTORY_INFO_REPLY"
TRAIN_NETWORK_DIRECTORY.
- #define `TTDB_OP_DIR_INFO_REQ_COMID` 108u
TTDB manager telegram MD: Get the OP_TRAIN_DIRECTORY.
- #define `TTDB_OP_DIR_INFO_REQ_TO_US` 3000000u
[us] 3s timeout
- #define `TTDB_OP_DIR_INFO_REP_DS` "TTDB_OP_TRAIN_DIR_INFO"
OP_TRAIN_DIRECTORY.
- #define `TTDB_READ_CMPLT_REQ_COMID` 110u
TTDB manager telegram MD: Get the TTDB.
- #define `TTDB_READ_CMPLT_REQ_DS` "TTDB_READ_COMPLETE_REQUEST"
ETBx.
- #define `TTDB_READ_CMPLT_REQ_TO_US` 3000000u
[us] 3s timeout
- #define `TTDB_READ_CMPLT_REP_COMID` 111u
MD reply.
- #define `TTDB_READ_CMPLT_REP_DS` "TTDB_READ_COMPLETE_REPLY"
TRDP_READ_COMPLETE_REPLY.T.
- #define `ECSP_CTRL_COMID` 120u
ECSP Control telegram.
- #define `ECSP_CTRL_CYCLE` 1000000u
[us] 1s
- #define `ECSP_CTRL_TO_US` 5000000u
[us] 5s
- #define `ECSP_CTRL_DEST_URI` "devECSP.anyVeh.ICst.ICITrn.ITrn"
10.0.0.1
- #define `TRDP_ECSP_CTRL_COMID` `ECSP_CTRL_COMID`
Etb control message.
- #define `ECSP_STATUS_COMID` 121u
ECSP status telegram.
- #define `ECSP_STATUS_CYCLE` 1000000u
[us] 1s
- #define `ECSP_STATUS_TO_US` 5000000u
[us] 5s
- #define `ECSP_STATUS_DEST_URI` "devECSC.anyVeh.ICst.ICITrn.ITrn"
10.0.0.100
- #define `ECSP_CONF_REQ_COMID` 122u
ECSP Confirmation Request telegram MD:
- #define `ECSP_CONF_REQ_TO_US` 3000000u
[us]
- #define `ECSP_CONF_REQ_URI` "devECSP.anyVeh.ICst.ICITrn.ITrn"

- 10.0.0.1
- #define ECSP_CONF_REP_TO_US 3000000u
[us]
 - #define ETBN_CTRL_REQ_COMID 130u
ETBN Control & Status Telegram MD.
 - #define ETBN_CTRL_REQ_DS "ETBN_CTRL"
ETBx.
 - #define ETBN_CTRL_REQ_TO_US 3000000u
[us] 3s timeout
 - #define ETBN_CTRL_REP_DS "ETBN_STATUS"
ETBN status reply.
 - #define ETBN_TRN_NET_DIR_REQ_COMID 132u
ETBN Control Telegram MD.
 - #define ETBN_TRN_NET_DIR_REQ_TO_US 3000000u
[us] 3s timeout
 - #define TCN_DNS_REQ_COMID 140u
TCN-DNS Request Telegram MD.
 - #define TCN_DNS_REQ_TO_US 3000000u
[us] 3s timeout
 - #define TRDP_ETBCTRL_DSID 1u
TRDP reserved data set ids in the range 1 ...

5.1.1 Detailed Description

All definitions from IEC 61375-2-3.

Note

Project: TCNOpen TRDP

Author

Bernd Loehr, NewTec GmbH, 2015-09-11

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>.

5.1.2 Macro Definition Documentation

5.1.2.1 ETB_CTRL_COMID

```
#define ETB_CTRL_COMID 1u
```

Reserved COMIDs in the range 1 ...

1000 ETB Control telegram

5.1.2.2 TRDP_ETBCTRL_DSID

```
#define TRDP_ETBCTRL_DSID 1u
```

TRDP reserved data set ids in the range 1 ...

1000

5.1.2.3 TRDP_MAX_FILE_NAME_LEN

```
#define TRDP_MAX_FILE_NAME_LEN 128u
```

path and file name length incl.

terminating '0'

5.1.2.4 TRDP_MAX_LABEL_LEN

```
#define TRDP_MAX_LABEL_LEN 16u
```

label length incl.

terminating '0'

5.1.2.5 TRDP_MAX_MD_DATA_SIZE

```
#define TRDP_MAX_MD_DATA_SIZE 65388u
```

MD packet properties.

MD payload size

5.1.2.6 TRDP_MAX_URI_HOST_LEN

```
#define TRDP_MAX_URI_HOST_LEN (5u * TRDP_MAX_LABEL_LEN)
```

URI host part incl.

terminating '0'

5.1.2.7 TRDP_MAX_URI_LEN

```
#define TRDP_MAX_URI_LEN (7u * TRDP_MAX_LABEL_LEN)
```

URI length incl.

':', '@' and terminating '0'

5.1.2.8 TRDP_MAX_URI_USER_LEN

```
#define TRDP_MAX_URI_USER_LEN (2u * TRDP_MAX_LABEL_LEN)
```

URI user part incl.

'.' and terminating '0'

5.1.2.9 TRDP_MD_DEFAULT_REPLY_TIMEOUT

```
#define TRDP_MD_DEFAULT_REPLY_TIMEOUT 5000000u
```

Default MD communication parameters.

[us] default reply timeout 5s

5.1.2.10 TRDP_MD_INFINITE_TIME

```
#define TRDP_MD_INFINITE_TIME (0)
```

Definitions for time out behaviour accd.

table A.18

5.1.2.11 TRDP_MIN_PD_HEADER_SIZE

```
#define TRDP_MIN_PD_HEADER_SIZE sizeof(PD_HEADER_T)
```

PD packet properties.

PD header size with FCS

5.1.2.12 TRDP_MSG_PD

```
#define TRDP_MSG_PD 0x5064u
```

Message Types.

'Pd' PD Data

5.1.2.13 TRDP_PD_UDP_PORT

```
#define TRDP_PD_UDP_PORT 17224u
```

TRDP defines (from former trpd_proto.h)

IANA assigned process data UDP port

5.1.2.14 TRDP_PROCESS_DEFAULT_CYCLE_TIME

```
#define TRDP_PROCESS_DEFAULT_CYCLE_TIME 10000u
```

Default TRDP process options.

[us] 10ms cycle time for TRDP process

5.1.2.15 TRDP_USR_URI_SIZE

```
#define TRDP_USR_URI_SIZE 32u
```

max.

User URI size in MD header

5.1.2.16 TTDB_NET_DIR_REQ_COMID

```
#define TTDB_NET_DIR_REQ_COMID 106u
```

TTDB manager telegram MD: Get the NETWORK_TRAIN_DIRECTORY.

MD request

5.1.2.17 TTDB_OP_DIR_INFO_COMID

```
#define TTDB_OP_DIR_INFO_COMID 101u
```

TTDB manager telegram MD: Push the OP_TRAIN_DIRECTORY.

MD notification

5.1.2.18 TTDB_STAT_CST_REQ_COMID

```
#define TTDB_STAT_CST_REQ_COMID 104u
```

TTDB manager telegram MD: Get the static consist information.

MD request

5.1.2.19 TTDB_TRN_DIR_REQ_COMID

```
#define TTDB_TRN_DIR_REQ_COMID 102u
```

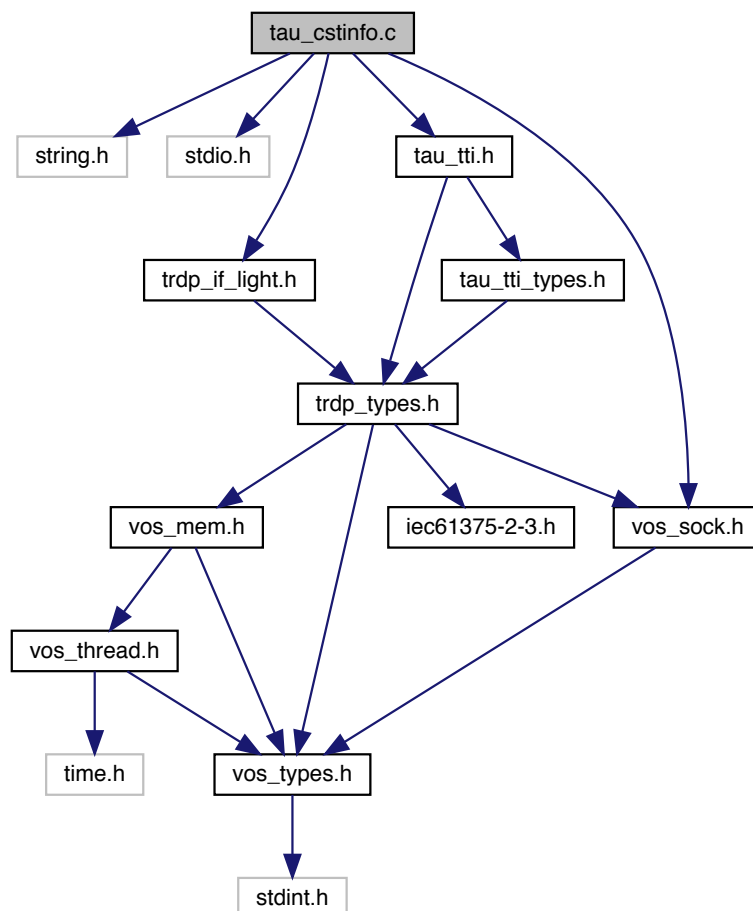
TTDB manager telegram MD: Get the TRAIN_DIRECTORY.

MD request

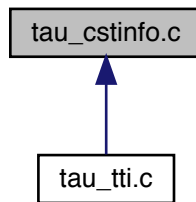
5.2 tau_cstinfo.c File Reference

Functions for consist information access.

```
#include <string.h>
#include <stdio.h>
#include "trdp_if_light.h"
#include "tau_tti.h"
#include "vos_sock.h"
Include dependency graph for tau_cstinfo.c:
```



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



Functions

- UINT16 `cstInfoGetPropSize` (`TRDP_CONSIST_INFO_T *pCstInfo`)
Getter function to retrieve a value from the consist info telegram value.

5.2.1 Detailed Description

Functions for consist information access.

Note

Project: TCNOpen TRDP prototype stack

Author

B. Loehr (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2015. All rights reserved.

5.2.2 Function Documentation

5.2.2.1 `cstInfoGetPropSize()`

```
UINT16 cstInfoGetPropSize (  
    TRDP_CONSIST_INFO_T * pCstInfo )
```

Getter function to retrieve a value from the consist info telegram value.

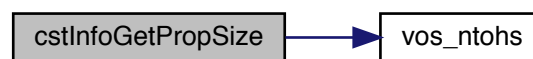
Parameters

in	<i>pCstInfo</i>	pointer to packed consist info in network byte order
----	-----------------	--

Return values

<i>len</i>	
------------	--

Here is the call graph for this function:

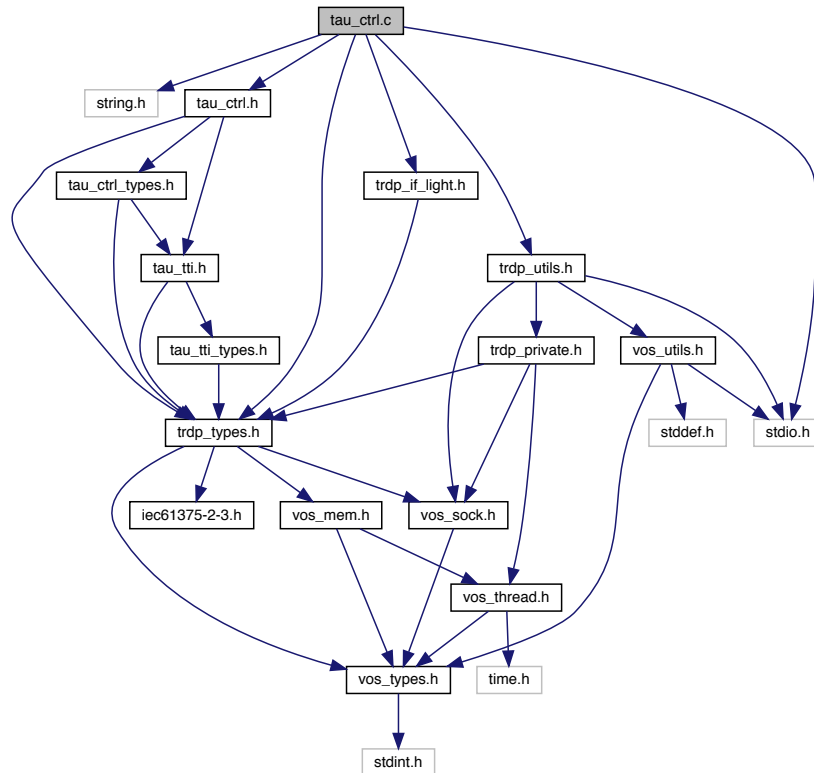


5.3 tau_ctrl.c File Reference

Functions for train switch control.

```
#include <string.h>
#include <stdio.h>
#include "trdp_types.h"
#include "trdp_utils.h"
#include "trdp_if_light.h"
#include "tau_ctrl.h"
```

Include dependency graph for tau_ctrl.c:



Functions

- EXT_DECL [TRDP_ERR_T](#) tau_initEcspCtrl ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_IP_ADDR_T](#) ecsplpAddr)
Function to init ECSP control interface.
- EXT_DECL [TRDP_ERR_T](#) tau_terminateEcspCtrl ([TRDP_APP_SESSION_T](#) appHandle)
Function to close ECSP control interface.
- EXT_DECL [TRDP_ERR_T](#) tau_setEcspCtrl ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_ECSP_CTRL_T](#) *pEcspCtrl)
Function to set ECSP control information.
- EXT_DECL [TRDP_ERR_T](#) tau_getEcspStat ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_ECSP_STAT_T](#) *pEcspStat, [TRDP_PD_INFO_T](#) *pPdInfo)
Function to get ECSP status information.
- EXT_DECL [TRDP_ERR_T](#) tau_requestEcspConfirm ([TRDP_APP_SESSION_T](#) appHandle, const void *pUserRef, [TRDP_MD_CALLBACK_T](#) pfCbFunction, [TRDP_ECSP_CONF_REQUEST_T](#) *pEcspConfRequest)
Function for ECSP confirmation/correction request, reply will be received via call back.

5.3.1 Detailed Description

Functions for train switch control.

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013-2019. All rights reserved.

5.3.2 Function Documentation**5.3.2.1 tau_getEcspStat()**

```
EXT_DECL TRDP_ERR_T tau_getEcspStat (
    TRDP_APP_SESSION_T appHandle,
    TRDP_ECSP_STAT_T * pEcspStat,
    TRDP_PD_INFO_T * pPdInfo )
```

Function to get ECSP status information.

Parameters

in	<i>appHandle</i>	Application handle
in, out	<i>pEcspStat</i>	Pointer to the ECSP status structure
in, out	<i>pPdInfo</i>	Pointer to PD status information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

5.3.2.2 tau_initEcspCtrl()

```
EXT_DECL TRDP_ERR_T tau_initEcspCtrl (
    TRDP_APP_SESSION_T appHandle,
    TRDP_IP_ADDR_T ecspIpAddress )
```

Function to init ECSP control interface.

Parameters

in	<i>appHandle</i>	Application handle
in	<i>ecsplpAddr</i>	ECSP address

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

5.3.2.3 tau_requestEcspConfirm()

```
EXT_DECL TRDP_ERR_T tau_requestEcspConfirm (
    TRDP_APP_SESSION_T appHandle,
    const void * pUserRef,
    TRDP_MD_CALLBACK_T pCbFunction,
    TRDP_ECSP_CONF_REQUEST_T * pEcspConfRequest )
```

Function for ECSP confirmation/correction request, reply will be received via call back.

Parameters

in	<i>appHandle</i>	Application Handle
in	<i>pUserRef</i>	user reference returned with reply
in	<i>pCbFunction</i>	Pointer to callback function, NULL for default
in	<i>pEcspConfRequest</i>	Pointer to confirmation data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

5.3.2.4 tau_setEcspCtrl()

```
EXT_DECL TRDP_ERR_T tau_setEcspCtrl (
    TRDP_APP_SESSION_T appHandle,
    TRDP_ECSP_CTRL_T * pEcspCtrl )
```

Function to set ECSP control information.

Parameters

in	<i>appHandle</i>	Application handle
in	<i>pEcspCtrl</i>	Pointer to the ECSP control structure

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

5.3.2.5 tau_terminateEcspCtrl()

```
EXT_DECL TRDP_ERR_T tau_terminateEcspCtrl (  
    TRDP_APP_SESSION_T appHandle )
```

Function to close ECSP control interface.

Parameters

in	<i>appHandle</i>	Application handle
----	------------------	--------------------

Return values

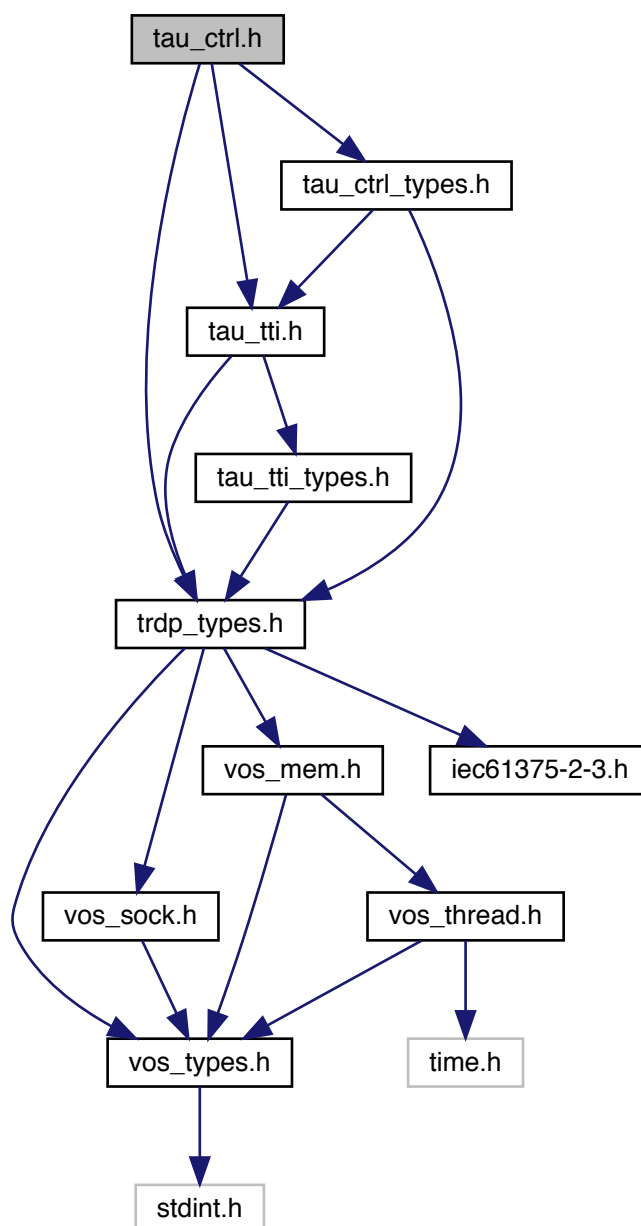
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_UNKNOWN_ERR</i>	undefined error

5.4 tau_ctrl.h File Reference

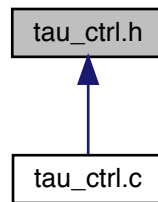
TRDP utility interface definitions.

```
#include "trdp_types.h"  
#include "tau_tti.h"  
#include "tau_ctrl_types.h"
```

Include dependency graph for tau_ctrl.h:



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



Functions

- EXT_DECL [TRDP_ERR_T](#) [tau_initEcspCtrl](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_IP_ADDR_T](#) [ecsplpAddr](#))
Function to init ECSP control interface.
- EXT_DECL [TRDP_ERR_T](#) [tau_terminateEcspCtrl](#) ([TRDP_APP_SESSION_T](#) appHandle)
Function to close ECSP control interface.
- EXT_DECL [TRDP_ERR_T](#) [tau_setEcspCtrl](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_ECSP_CTRL_T](#) *pEcspCtrl)
Function to set ECSP control information.
- EXT_DECL [TRDP_ERR_T](#) [tau_getEcspStat](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_ECSP_STAT_T](#) *pEcspStat, [TRDP_PD_INFO_T](#) *pPdInfo)
Function to get ECSP status information.
- EXT_DECL [TRDP_ERR_T](#) [tau_requestEcspConfirm](#) ([TRDP_APP_SESSION_T](#) appHandle, const void *pUserRef, [TRDP_MD_CALLBACK_T](#) pfCbFunction, [TRDP_ECSP_CONF_REQUEST_T](#) *pEcspConfRequest)
Function for ECSP confirmation/correction request, reply will be received via call back.

5.4.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- ETB control

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.4.2 Function Documentation

5.4.2.1 tau_getEcspStat()

```
EXT_DECL TRDP_ERR_T tau_getEcspStat (
    TRDP_APP_SESSION_T appHandle,
    TRDP_ECSP_STAT_T * pEcspStat,
    TRDP_PD_INFO_T * pPdInfo )
```

Function to get ECSP status information.

Parameters

in	<i>appHandle</i>	Application Handle
in, out	<i>pEcspStat</i>	Pointer to the ECSP status structure
in, out	<i>pPdInfo</i>	Pointer to PD status information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

Parameters

in	<i>appHandle</i>	Application handle
in, out	<i>pEcspStat</i>	Pointer to the ECSP status structure
in, out	<i>pPdInfo</i>	Pointer to PD status information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

5.4.2.2 tau_initEcspCtrl()

```
EXT_DECL TRDP_ERR_T tau_initEcspCtrl (
    TRDP_APP_SESSION_T appHandle,
    TRDP_IP_ADDR_T ecspIpAddress )
```

Function to init ECSP control interface.

Parameters

in	<i>appHandle</i>	Application handle
in	<i>ecspIpAddr</i>	ECSP address

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

5.4.2.3 tau_requestEcspConfirm()

```
EXT_DECL TRDP_ERR_T tau_requestEcspConfirm (
    TRDP_APP_SESSION_T appHandle,
    const void * pUserRef,
    TRDP_MD_CALLBACK_T pCbFunction,
    TRDP_ECSP_CONF_REQUEST_T * pEcspConfRequest )
```

Function for ECSP confirmation/correction request, reply will be received via call back.

Parameters

in	<i>appHandle</i>	Application Handle
in	<i>pUserRef</i>	user reference returned with reply
in	<i>pCbFunction</i>	Pointer to callback function, NULL for default
in	<i>pEcspConfRequest</i>	Pointer to confirmation data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

5.4.2.4 tau_setEcspCtrl()

```
EXT_DECL TRDP_ERR_T tau_setEcspCtrl (
    TRDP_APP_SESSION_T appHandle,
    TRDP_ECSP_CTRL_T * pEcspCtrl )
```

Function to set ECSP control information.

Parameters

in	<i>appHandle</i>	Application handle
in	<i>pEcspCtrl</i>	Pointer to the ECSP control structure

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

5.4.2.5 tau_terminateEcspCtrl()

```
EXT_DECL TRDP_ERR_T tau_terminateEcspCtrl (
    TRDP_APP_SESSION_T appHandle )
```

Function to close ECSP control interface.

Parameters

in	<i>appHandle</i>	Application handle
----	------------------	--------------------

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_UNKNOWN_ERR</i>	undefined error

Parameters

in	<i>appHandle</i>	Application handle
----	------------------	--------------------

Return values

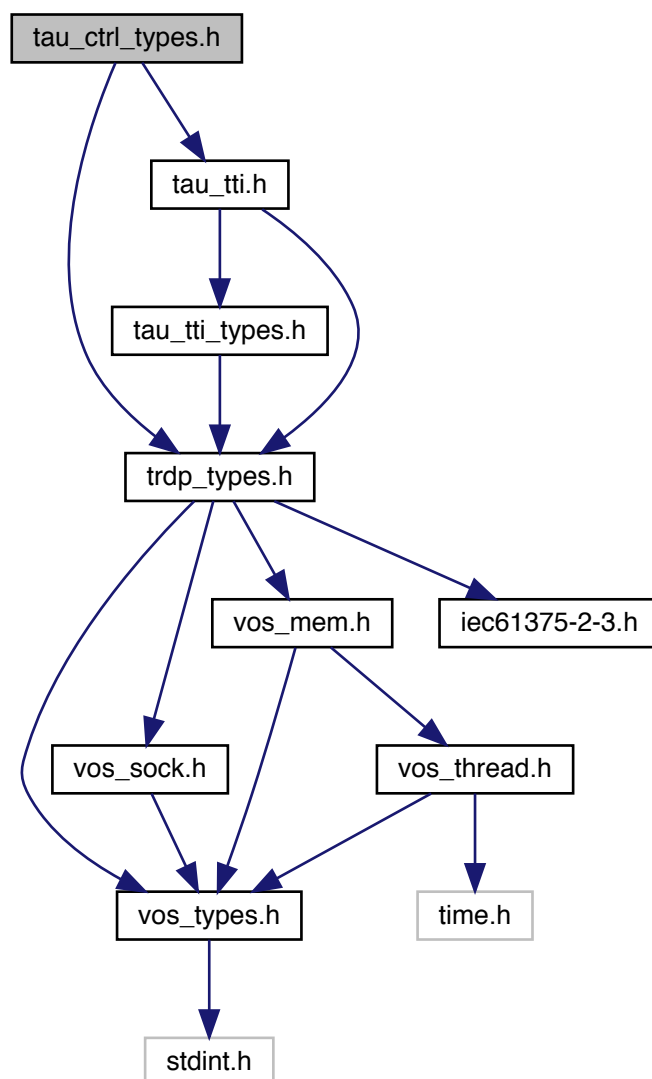
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_UNKNOWN_ERR</i>	undefined error

5.5 tau_ctrl_types.h File Reference

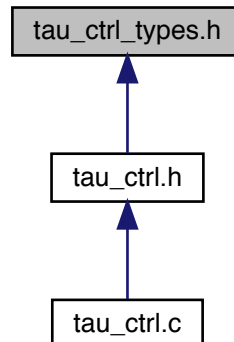
TRDP utility interface definitions.

```
#include "trdp_types.h"
#include "tau_tti.h"
```

Include dependency graph for tau_ctrl_types.h:



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



Data Structures

- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.

5.5.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following

- ETB control type definitions acc. to IEC61375-2-3

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

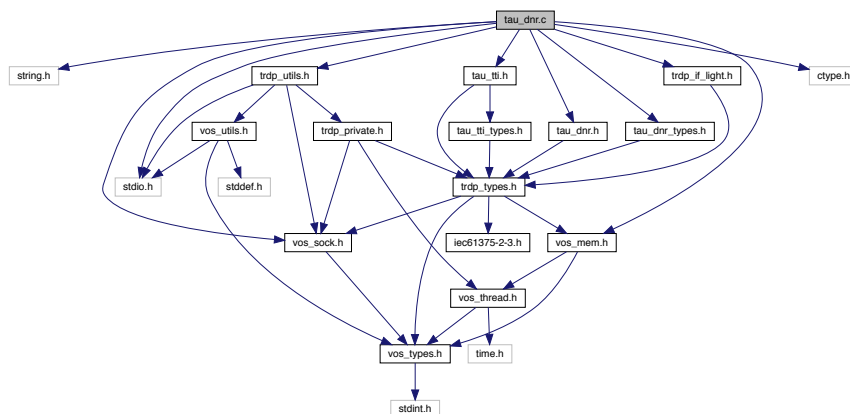
This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.6 tau_dnr.c File Reference

Functions for domain name resolution.

```
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include "tau_tti.h"
#include "tau_dnr.h"
#include "tau_dnr_types.h"
#include "trdp_utils.h"
#include "trdp_if_light.h"
#include "vos_mem.h"
#include "vos_sock.h"
```

Include dependency graph for tau_dnr.c:

**Data Structures**

- struct [DNS_HEADER](#)
DNS header structure.

Macros

- `#define TAU_MAX_NO_IF 4u`
Default interface should be in the first 4.
- `#define TAU_DNS_TIME_OUT_LONG 10u`
Timeout in seconds for DNS server reply, if no hosts file provided.
- `#define TAU_DNS_TIME_OUT_SHORT 1u`
Timeout in seconds for DNS server reply, if hosts file was provided.

Typedefs

- `typedef struct DNS_HEADER TAU_DNS_HEADER_T`
DNS header structure.

Functions

- `EXT_DECL TRDP_ERR_T tau_initDnr (TRDP_APP_SESSION_T appHandle, TRDP_IP_ADDR_T dnsIp, Addr, UINT16 dnsPort, const CHAR8 *pHostsFileName, TRDP_DNR_OPTS_T dnsOptions, BOOL8 waitForDnr)`
Function to init the DNR subsystem Initialize the DNR resolver.
- `EXT_DECL void tau_deinitDnr (TRDP_APP_SESSION_T appHandle)`
Function to deinit DNR.
- `EXT_DECL TRDP_DNR_STATE_T tau_DNRstatus (TRDP_APP_SESSION_T appHandle)`
Function to get the status of DNR.
- `EXT_DECL TRDP_IP_ADDR_T tau_getOwnAddr (TRDP_APP_SESSION_T appHandle)`
Function to get the own IP address.
- `EXT_DECL TRDP_ERR_T tau_uri2Addr (TRDP_APP_SESSION_T appHandle, TRDP_IP_ADDR_T *pAddr, const TRDP_URI_T pUri)`
Function to convert a URI to an IP address.
- `EXT_DECL TRDP_ERR_T tau_addr2Uri (TRDP_APP_SESSION_T appHandle, TRDP_URI_HOST_T pUri, TRDP_IP_ADDR_T addr)`
Function to convert an IP address to a URI.

5.6.1 Detailed Description

Functions for domain name resolution.

Note

Project: TCNOpen TRDP prototype stack

Author

B. Loehr (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.6.2 Function Documentation

5.6.2.1 tau_addr2Uri()

```
EXT_DECL TRDP_ERR_T tau_addr2Uri (
    TRDP_APP_SESSION_T appHandle,
    TRDP_URI_HOST_T pUri,
    TRDP_IP_ADDR_T addr )
```

Function to convert an IP address to a URI.

Receives an IP-Address and translates it into the host part of the corresponding URI. Both unicast and multicast addresses are accepted.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession()
out	<i>pUri</i>	Pointer to a string to return the URI host part
in	<i>addr</i>	IP address, 0==own address

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.6.2.2 tau_deinitDnr()

```
EXT_DECL void tau_deinitDnr (
    TRDP_APP_SESSION_T appHandle )
```

Function to deinit DNR.

Release any resources allocated by DNR.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession()
----	------------------	--

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.6.2.3 tau_DNRstatus()

```
EXT_DECL TRDP_DNR_STATE_T tau_DNRstatus (
    TRDP_APP_SESSION_T appHandle )
```

Function to get the status of DNR.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession()
----	------------------	--

Return values

<i>TRDP_DNR_NOT_AVAILABLE</i>	no error
<i>TRDP_DNR_UNKNOWN</i>	enabled, but cache is empty
<i>TRDP_DNR_ACTIVE</i>	enabled, cache has values
<i>TRDP_DNR_HOSTSFILE</i>	enabled, hostsfile used (static mode)

5.6.2.4 tau_getOwnAddr()

```
EXT_DECL TRDP_IP_ADDR_T tau_getOwnAddr (
    TRDP_APP_SESSION_T appHandle )
```

Function to get the own IP address.

Returns the IP address set by openSession. If it was 0 (INADDR_ANY), the address of the default adapter will be returned.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession()
----	------------------	--

Return values

<i>own</i>	IP address
------------	------------

5.6.2.5 tau_initDnr()

```
EXT_DECL TRDP_ERR_T tau_initDnr (
    TRDP_APP_SESSION_T appHandle,
    TRDP_IP_ADDR_T dnsIpAddr,
    UINT16 dnsPort,
    const CHAR8 * pHostsFileName,
    TRDP_DNR_OPTS_T dnsOptions,
    BOOL8 waitForDnr )
```

Function to init the DNR subsystem Initialize the DNR resolver.

Function to init DNR.

Depending on the supplied options, three operational modes are supported:

1. TRDP_DNR_COMMON_THREAD (default) Expect `tlc_process` running in a different, separate thread
2. TRDP_DNR_OWN_THREAD For single threaded systems only! Internally call [tlc_process\(\)](#)
3. TRDP_DNR_STANDARD_DNS Use standard DNS instead of TCN-DNS. Default `dnsPort` (= 0) for TCN-DNS is 17225, for standard DNS it is 53.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
in	<i>dnsIpAddr</i>	DNS/ECSP IP address.
in	<i>dnsPort</i>	DNS port number.
in	<i>pHostsFileName</i>	Optional host file name as ECSP replacement/addition.
in	<i>dnsOptions</i>	Use existing thread (recommended), use own <code>tlc_process</code> loop or use standard DNS
in	<i>waitForDnr</i>	Waits for DNR if true(recommended), doesn't wait for DNR if false(for testing).

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

< default DNR/ECSP settings

5.6.2.6 tau_uri2Addr()

```
EXT_DECL TRDP_ERR_T tau_uri2Addr (
    TRDP_APP_SESSION_T appHandle,
    TRDP_IP_ADDR_T * pAddr,
    const TRDP_URI_T pUri )
```

Function to convert a URI to an IP address.

Receives an URI as input variable and translates this URI to an IP-Address. The URI may specify either a unicast or a multicast IP-Address.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession()
out	<i>pAddr</i>	Pointer to return the IP address
in	<i>pUri</i>	Pointer to an URI or an IP Address string, NULL==own URI

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_UNRESOLVED_ERR</i>	Could not resolve error

Return values

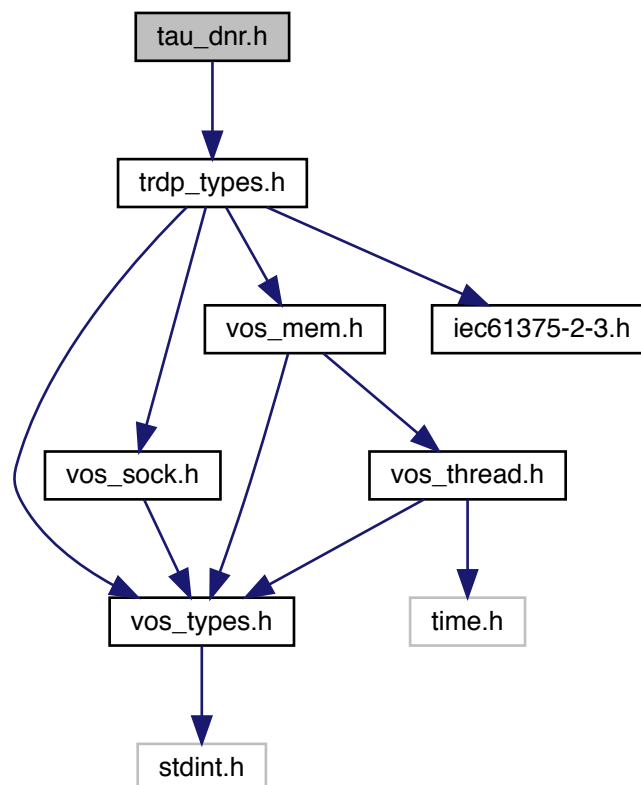
<i>TRDP_TOPO_ERR</i>	Cache/DB entry is invalid
----------------------	---------------------------

5.7 tau_dnr.h File Reference

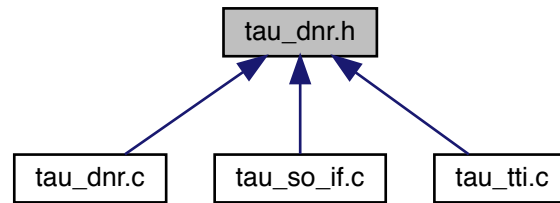
TRDP utility interface definitions.

```
#include "trdp_types.h"
```

Include dependency graph for tau_dnr.h:



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



Typedefs

- typedef enum [TRDP_DNR_STATE](#) [TRDP_DNR_STATE_T](#)
DNR state.
- typedef enum [TRDP_DNR_OPTS](#) [TRDP_DNR_OPTS_T](#)
DNR options.

Enumerations

- enum [TRDP_DNR_STATE](#)
DNR state.
- enum [TRDP_DNR_OPTS](#) { , [TRDP_DNR_OWN_THREAD](#) = 1 }
DNR options.

Functions

- EXT_DECL [TRDP_ERR_T](#) [tau_initDnr](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_IP_ADDR_T](#) dnsIp↔
Addr, UINT16 dnsPort, const CHAR8 *pHostsFileName, [TRDP_DNR_OPTS_T](#) dnsOptions, BOOL8 wait↔
ForDnr)
Function to init DNR.
- EXT_DECL void [tau_deinitDnr](#) ([TRDP_APP_SESSION_T](#) appHandle)
Release any resources allocated by DNR.
- EXT_DECL [TRDP_DNR_STATE_T](#) [tau_DNRstatus](#) ([TRDP_APP_SESSION_T](#) appHandle)
Function to get the status of DNR.
- EXT_DECL [TRDP_IP_ADDR_T](#) [tau_getOwnAddr](#) ([TRDP_APP_SESSION_T](#) appHandle)
Function to get the own IP address.
- EXT_DECL [TRDP_ERR_T](#) [tau_uri2Addr](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_IP_ADDR_T](#) *p↔
Addr, const [TRDP_URI_T](#) pUri)
Function to convert a URI to an IP address.
- EXT_DECL [TRDP_ERR_T](#) [tau_addr2Uri](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_URI_HOST_T](#) pUri,
[TRDP_IP_ADDR_T](#) addr)
Function to convert an IP address to a URI.

5.7.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- IP - URI address translation

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.7.2 Enumeration Type Documentation

5.7.2.1 TRDP_DNR_OPTS

enum `TRDP_DNR_OPTS`

DNR options.

Enumerator

TRDP_DNR_OWN_THREAD	For single threaded systems only! Internally call <code>tlc_process()</code>
---------------------	--

5.7.3 Function Documentation

5.7.3.1 tau_addr2Uri()

```
EXT_DECL TRDP_ERR_T tau_addr2Uri (
    TRDP_APP_SESSION_T appHandle,
```



```
TRDP_URI_HOST_T pUri,
TRDP_IP_ADDR_T addr )
```

Function to convert an IP address to a URI.

Receives an IP-Address and translates it into the host part of the corresponding URI. Both unicast and multicast addresses are accepted.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pUri</i>	Pointer to a string to return the URI host part
in	<i>addr</i>	IP address, 0==own address

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

Receives an IP-Address and translates it into the host part of the corresponding URI. Both unicast and multicast addresses are accepted.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession()
out	<i>pUri</i>	Pointer to a string to return the URI host part
in	<i>addr</i>	IP address, 0==own address

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.7.3.2 tau_deinitDnr()

```
EXT_DECL void tau_deinitDnr (
    TRDP_APP_SESSION_T appHandle )
```

Release any resources allocated by DNR.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
----	------------------	--

Return values

<i>none</i>	Release any resources allocated by DNR.
-------------	---

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession()
----	------------------	--

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.7.3.3 tau_DNRstatus()

```
EXT_DECL TRDP_DNR_STATE_T tau_DNRstatus (  
    TRDP_APP_SESSION_T appHandle )
```

Function to get the status of DNR.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession()
----	------------------	--

Return values

<i>TRDP_DNR_NOT_AVAILABLE</i>	no error
<i>TRDP_DNR_UNKNOWN</i>	enabled, but cache is empty
<i>TRDP_DNR_ACTIVE</i>	enabled, cache has values
<i>TRDP_DNR_HOSTSFILE</i>	enabled, hostsfile used (static mode)

5.7.3.4 tau_getOwnAddr()

```
EXT_DECL TRDP_IP_ADDR_T tau_getOwnAddr (  
    TRDP_APP_SESSION_T appHandle )
```

Function to get the own IP address.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
----	------------------	--

Return values

<i>own</i>	IP address
------------	------------

Returns the IP address set by openSession. If it was 0 (INADDR_ANY), the address of the default adapter will be returned.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession()
----	------------------	--

Return values

<i>own</i>	IP address
------------	------------

5.7.3.5 tau_initDnr()

```
EXT_DECL TRDP_ERR_T tau_initDnr (
    TRDP_APP_SESSION_T appHandle,
    TRDP_IP_ADDR_T dnsIpAddr,
    UINT16 dnsPort,
    const CHAR8 * pHostsFileName,
    TRDP_DNR_OPTS_T dnsOptions,
    BOOL8 waitForDnr )
```

Function to init DNR.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
in	<i>dnsIpAddr</i>	DNS/ECSP IP address.
in	<i>dnsPort</i>	DNS port number.
in	<i>pHostsFileName</i>	Optional host file name as ECSP replacement/addition.
in	<i>dnsOptions</i>	Use existing thread (recommended), use own tlc_process loop or use standard DNS
in	<i>waitForDnr</i>	Waits for DNR if true(recommended), doesn't wait for DNR if false(for testing).

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

Function to init DNR.

Depending on the supplied options, three operational modes are supported:

1. TRDP_DNR_COMMON_THREAD (default) Expect tlc_process running in a different, separate thread
2. TRDP_DNR_OWN_THREAD For single threaded systems only! Internally call [tlc_process\(\)](#)
3. TRDP_DNR_STANDARD_DNS Use standard DNS instead of TCN-DNS. Default dnsPort (= 0) for TCN-DNS is 17225, for standard DNS it is 53.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
in	<i>dnsIpAddr</i>	DNS/ECSP IP address.

Parameters

in	<i>dnsPort</i>	DNS port number.
in	<i>pHostsFileName</i>	Optional host file name as ECSP replacement/addition.
in	<i>dnsOptions</i>	Use existing thread (recommended), use own tlc_process loop or use standard DNS
in	<i>waitForDnr</i>	Waits for DNR if true(recommended), doesn't wait for DNR if false(for testing).

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

< default DNR/ECSP settings

5.7.3.6 tau_uri2Addr()

```
EXT_DECL TRDP_ERR_T tau_uri2Addr (
    TRDP_APP_SESSION_T appHandle,
    TRDP_IP_ADDR_T * pAddr,
    const TRDP_URI_T pUri )
```

Function to convert a URI to an IP address.

Receives a URI as input variable and translates this URI to an IP-Address. The URI may specify either a unicast or a multicast IP-Address. The caller may specify a topographic counter, which will be checked.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pAddr</i>	Pointer to return the IP address
in	<i>pUri</i>	Pointer to a URI or an IP Address string, NULL==own URI

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

Receives an URI as input variable and translates this URI to an IP-Address. The URI may specify either a unicast or a multicast IP-Address.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession()
out	<i>pAddr</i>	Pointer to return the IP address
in	<i>pUri</i>	Pointer to an URI or an IP Address string, NULL==own URI

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_UNRESOLVED_ERR</i>	Could not resolve error

Return values

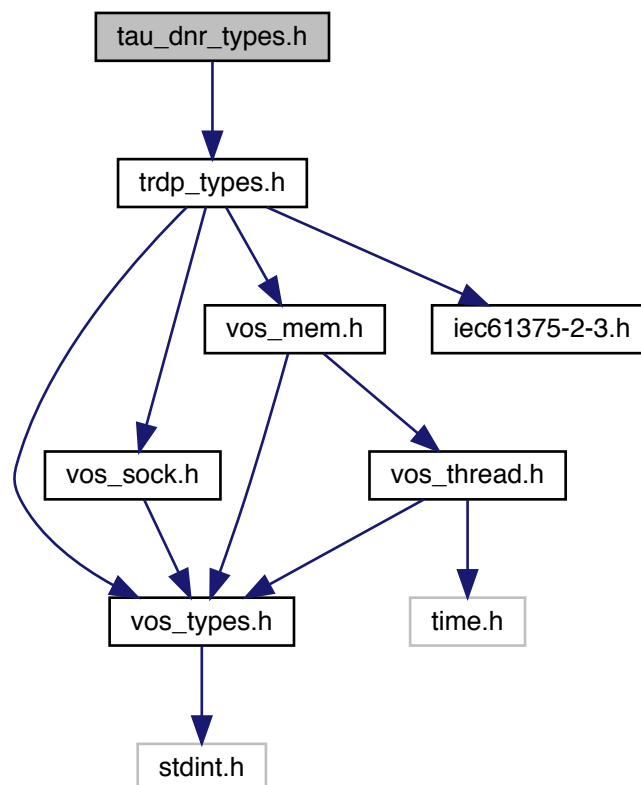
<i>TRDP_TOPO_ERR</i>	Cache/DB entry is invalid
----------------------	---------------------------

5.8 tau_dnr_types.h File Reference

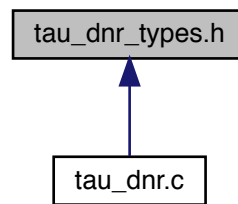
TRDP utility interface definitions.

```
#include "trdp_types.h"
```

Include dependency graph for tau_dnr_types.h:



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



Data Structures

- struct [TCN_URI](#)
TCN-DNS simplified header structures.
- struct [TRDP_DNS_REQUEST](#)
TCN-DNS Request telegram TCN_DNS_REQ_DS.
- struct [TRDP_DNS_REPLY](#)
TCN-DNS Reply telegram TCN_DNS_REP_DS.

Typedefs

- typedef struct [TCN_URI](#) [TCN_URI_T](#)
TCN-DNS simplified header structures.
- typedef struct [TRDP_DNS_REQUEST](#) [TRDP_DNS_REQUEST_T](#)
TCN-DNS Request telegram TCN_DNS_REQ_DS.
- typedef struct [TRDP_DNS_REPLY](#) [TRDP_DNS_REPLY_T](#)
TCN-DNS Reply telegram TCN_DNS_REP_DS.

5.8.1 Detailed Description

TRDP utility interface definitions.

This module provides typedefs to the following utilities

- IP - URI address translation

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr (initial version)

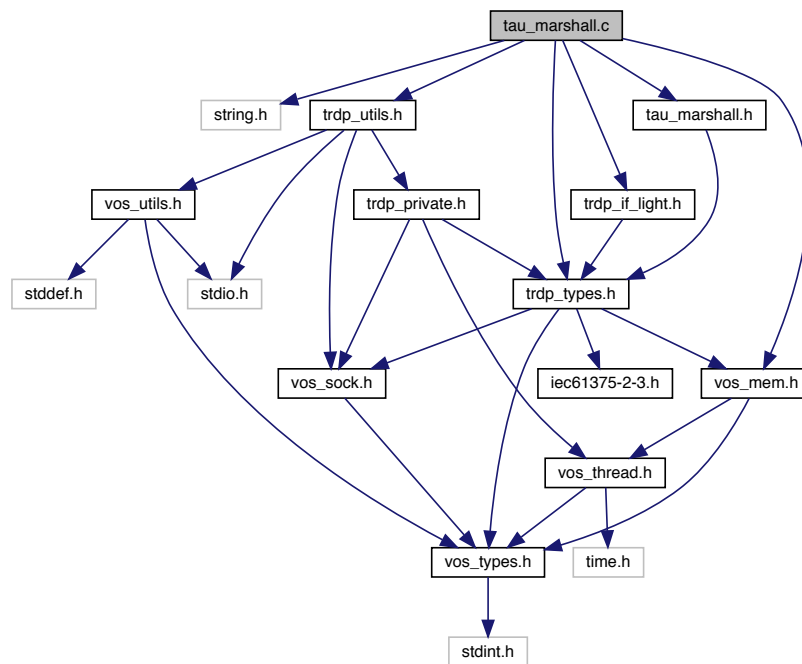
Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright NewTec GmbH, 2017. All rights reserved.

5.9 tau_marshall.c File Reference

Marshalling functions for TRDP.

```
#include <string.h>
#include "trdp_types.h"
#include "trdp_if_light.h"
#include "trdp_utils.h"
#include "vos_mem.h"
#include "tau_marshall.h"
Include dependency graph for tau_marshall.c:
```



Data Structures

- struct [TAU_MARSHALL_INFO_T](#)
Marshalling info, used to and from wire.

Functions

- EXT_DECL [TRDP_ERR_T](#) [tau_initMarshall](#) (void **ppRefCon, UINT32 numComId, [TRDP_COMID_DSID_MAP_T](#) *pComIdDsIdMap, UINT32 numDataSet, [TRDP_DATASET_T](#) *pDataset[])
Function to initialise the marshalling/unmarshalling.
- EXT_DECL [TRDP_ERR_T](#) [tau_marshall](#) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDest, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)
marshall function.
- EXT_DECL [TRDP_ERR_T](#) [tau_unmarshall](#) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDest, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)

unmarshall function.

- EXT_DECL [TRDP_ERR_T tau_marshallDs](#) (void *pRefCon, UINT32 dsId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDest, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)

marshall data set function.

- EXT_DECL [TRDP_ERR_T tau_unmarshallDs](#) (void *pRefCon, UINT32 dsId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDest, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)

unmarshall data set function.

- EXT_DECL [TRDP_ERR_T tau_calcDatasetSize](#) (void *pRefCon, UINT32 dsId, UINT8 *pSrc, UINT32 srcSize, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)

Calculate data set size by given data set id.

- EXT_DECL [TRDP_ERR_T tau_calcDatasetSizeByComId](#) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)

Calculate data set size by given ComId.

5.9.1 Detailed Description

Marshalling functions for TRDP.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.9.2 Function Documentation

5.9.2.1 tau_calcDatasetSize()

```
EXT_DECL TRDP\_ERR\_T tau_calcDatasetSize (
    void * pRefCon,
    UINT32 dsId,
    UINT8 * pSrc,
    UINT32 srcSize,
    UINT32 * pDestSize,
    TRDP\_DATASET\_T ** ppDSPointer )
```

Calculate data set size by given data set id.

Parameters

in	<i>pRefCon</i>	Pointer to user context
in	<i>dsId</i>	Dataset id to identify the structure out of a configuration
in	<i>pSrc</i>	Pointer to received original message
in	<i>srcSize</i>	size of the source buffer
out	<i>pDestSize</i>	Pointer to the size of the data set
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset, set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_STATE_ERR</i>	Too deep recursion
<i>TRDP_COMID_ERR</i>	comid not existing
<i>TRDP_MARSHALLING_ERR</i>	dataset/source size mismatch

5.9.2.2 tau_calcDatasetSizeByComId()

```
EXT_DECL TRDP_ERR_T tau_calcDatasetSizeByComId (
    void * pRefCon,
    UINT32 comId,
    UINT8 * pSrc,
    UINT32 srcSize,
    UINT32 * pDestSize,
    TRDP_DATASET_T ** ppDSPointer )
```

Calculate data set size by given ComId.

Parameters

in	<i>pRefCon</i>	Pointer to user context
in	<i>comId</i>	ComId id to identify the structure out of a configuration
in	<i>pSrc</i>	Pointer to received original message
in	<i>srcSize</i>	size of the source buffer
out	<i>pDestSize</i>	Pointer to the size of the data set
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset, set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_STATE_ERR</i>	Too deep recursion

Return values

<i>TRDP_COMID_ERR</i>	comid not existing
<i>TRDP_MARSHALLING_ERR</i>	dataset/source size mismatch

5.9.2.3 tau_initMarshall()

```
EXT_DECL TRDP_ERR_T tau_initMarshall (
    void ** ppRefCon,
    UINT32 numComId,
    TRDP_COMID_DSID_MAP_T * pComIdDsIdMap,
    UINT32 numDataSet,
    TRDP_DATASET_T * pDataset[ ] )
```

Function to initialise the marshalling/unmarshalling.

Types for marshalling / unmarshalling.

The supplied array must be sorted by ComIds. The array must exist during the use of the marshalling functions (until [tlc_terminate\(\)](#)).

Parameters

in, out	<i>ppRefCon</i>	Returns a pointer to be used for the reference context of marshalling/unmarshalling
in	<i>numComId</i>	Number of datasets found in the configuration
in	<i>pComIdDsIdMap</i>	Pointer to an array of structures of type TRDP_DATASET_T
in	<i>numDataSet</i>	Number of datasets found in the configuration
in	<i>pDataset</i>	Pointer to an array of pointers to structures of type TRDP_DATASET_T

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer too small
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.4 tau_marshall()

```
EXT_DECL TRDP_ERR_T tau_marshall (
    void * pRefCon,
    UINT32 comId,
    UINT8 * pSrc,
    UINT32 srcSize,
    UINT8 * pDest,
    UINT32 * pDestSize,
    TRDP_DATASET_T ** ppDSPointer )
```

marshall function.

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>comId</i>	ComId to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDest</i>	pointer to a buffer for the treated message
in, out	<i>pDestSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_STATE_ERR</i>	Too deep recursion
<i>TRDP_MARSHALLING_ERR</i>	dataset/source size mismatch

5.9.2.5 tau_marshallDs()

```
EXT_DECL TRDP_ERR_T tau_marshallDs (
    void * pRefCon,
    UINT32 dsId,
    UINT8 * pSrc,
    UINT32 srcSize,
    UINT8 * pDest,
    UINT32 * pDestSize,
    TRDP_DATASET_T ** ppDSPointer )
```

marshall data set function.

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>dsId</i>	Data set id to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDest</i>	pointer to a buffer for the treated message
in, out	<i>pDestSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_STATE_ERR</i>	Too deep recursion
<i>TRDP_COMID_ERR</i>	comid not existing

Return values

<code>TRDP_MARSHALLING_ERR</code>	dataset/source size mismatch
-----------------------------------	------------------------------

5.9.2.6 tau_unmarshall()

```
EXT_DECL TRDP_ERR_T tau_unmarshall (
    void * pRefCon,
    UINT32 comId,
    UINT8 * pSrc,
    UINT32 srcSize,
    UINT8 * pDest,
    UINT32 * pDestSize,
    TRDP_DATASET_T ** ppDSPointer )
```

unmarshall function.

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>comId</i>	ComId to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDest</i>	pointer to a buffer for the treated message
in, out	<i>pDestSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

<code>TRDP_INIT_ERR</code>	marshalling not initialised
<code>TRDP_NO_ERR</code>	no error
<code>TRDP_MEM_ERR</code>	provided buffer too small
<code>TRDP_PARAM_ERR</code>	Parameter error
<code>TRDP_STATE_ERR</code>	Too deep recursion
<code>TRDP_COMID_ERR</code>	comid not existing
<code>TRDP_MARSHALLING_ERR</code>	dataset/source size mismatch

5.9.2.7 tau_unmarshallDs()

```
EXT_DECL TRDP_ERR_T tau_unmarshallDs (
    void * pRefCon,
    UINT32 dsId,
    UINT8 * pSrc,
    UINT32 srcSize,
    UINT8 * pDest,
```

```
UINT32 * pDestSize,
TRDP_DATASET_T ** ppDSPointer )
```

unmarshall data set function.

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>dsId</i>	Data set id to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDest</i>	pointer to a buffer for the treated message
in, out	<i>pDestSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

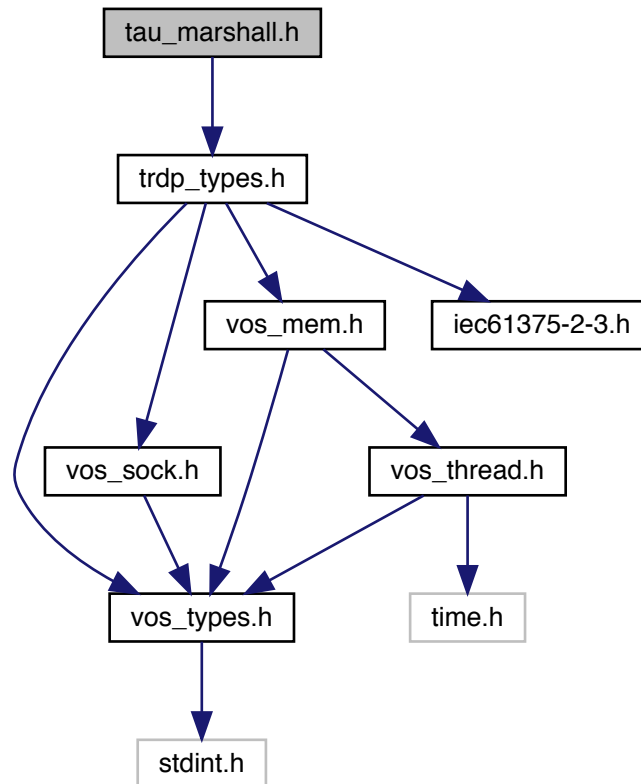
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_STATE_ERR</i>	Too deep recursion
<i>TRDP_COMID_ERR</i>	comid not existing
<i>TRDP_MARSHALLING_ERR</i>	dataset/source size mismatch

5.10 tau_marshall.h File Reference

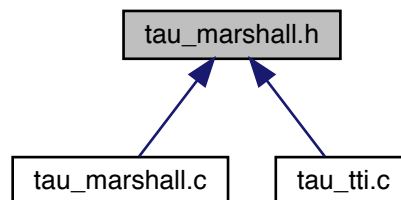
TRDP utility interface definitions.

```
#include "trdp_types.h"
```

Include dependency graph for tau_marshall.h:



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



Functions

- EXT_DECL [TRDP_ERR_T](#) [tau_initMarshall](#) (void **ppRefCon, UINT32 numComId, [TRDP_COMID_DSID_T](#) *pComIdDsIdMap, UINT32 numDataSet, [TRDP_DATASET_T](#) *pDataset[])

Types for marshalling / unmarshall.

- EXT_DECL [TRDP_ERR_T tau_marshall](#) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDest, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)

marshall function.

- EXT_DECL [TRDP_ERR_T tau_marshallDs](#) (void *pRefCon, UINT32 dsId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDest, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)

marshall data set function.

- EXT_DECL [TRDP_ERR_T tau_unmarshall](#) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDest, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)

unmarshall function.

- EXT_DECL [TRDP_ERR_T tau_unmarshallDs](#) (void *pRefCon, UINT32 dsId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDest, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)

unmarshall data set function.

- EXT_DECL [TRDP_ERR_T tau_calcDatasetSize](#) (void *pRefCon, UINT32 dsId, UINT8 *pSrc, UINT32 srcSize, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)

Calculate data set size by given data set id.

- EXT_DECL [TRDP_ERR_T tau_calcDatasetSizeByComId](#) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)

Calculate data set size by given ComId.

5.10.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- marshalling/unmarshall

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.10.2 Function Documentation

5.10.2.1 tau_calcDatasetSize()

```
EXT_DECL TRDP\_ERR\_T tau_calcDatasetSize (
    void * pRefCon,
    UINT32 dsId,
    UINT8 * pSrc,
    UINT32 srcSize,
    UINT32 * pDestSize,
    TRDP\_DATASET\_T ** ppDSPointer )
```

Calculate data set size by given data set id.

Parameters

in	<i>pRefCon</i>	Pointer to user context
in	<i>dsId</i>	Dataset id to identify the structure out of a configuration
in	<i>pSrc</i>	Pointer to received original message
in	<i>srcSize</i>	size of the source buffer
out	<i>pDestSize</i>	Pointer to the size of the data set
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset, set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_PARAM_ERR</i>	data set id not existing

Parameters

in	<i>pRefCon</i>	Pointer to user context
in	<i>dsId</i>	Dataset id to identify the structure out of a configuration
in	<i>pSrc</i>	Pointer to received original message
in	<i>srcSize</i>	size of the source buffer
out	<i>pDestSize</i>	Pointer to the size of the data set
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset, set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer too small
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_STATE_ERR</i>	Too deep recursion
<i>TRDP_COMID_ERR</i>	comid not existing
<i>TRDP_MARSHALLING_ERR</i>	dataset/source size mismatch

5.10.2.2 tau_calcDatasetSizeByComId()

```
EXT_DECL TRDP_ERR_T tau_calcDatasetSizeByComId (
    void * pRefCon,
    UINT32 comId,
    UINT8 * pSrc,
    UINT32 srcSize,
    UINT32 * pDestSize,
    TRDP_DATASET_T ** ppDSPointer )
```

Calculate data set size by given ComId.

Parameters

in	<i>pRefCon</i>	Pointer to user context
in	<i>comId</i>	ComId id to identify the structure out of a configuration
in	<i>pSrc</i>	Pointer to received original message
in	<i>srcSize</i>	size of the source buffer
out	<i>pDestSize</i>	Pointer to the size of the data set
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset, set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_PARAM_ERR</i>	data set id not existing

Parameters

in	<i>pRefCon</i>	Pointer to user context
in	<i>comId</i>	ComId id to identify the structure out of a configuration
in	<i>pSrc</i>	Pointer to received original message
in	<i>srcSize</i>	size of the source buffer
out	<i>pDestSize</i>	Pointer to the size of the data set
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset, set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer too small
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_STATE_ERR</i>	Too deep recursion
<i>TRDP_COMID_ERR</i>	comid not existing
<i>TRDP_MARSHALLING_ERR</i>	dataset/source size mismatch

5.10.2.3 tau_initMarshall()

```
EXT_DECL TRDP_ERR_T tau_initMarshall (
    void ** ppRefCon,
    UINT32 numComId,
    TRDP_COMID_DSID_MAP_T * pComIdDsIdMap,
    UINT32 numDataSet,
    TRDP_DATASET_T * pDataset[] )
```

Types for marshalling / unmarshalling.

Function to initialise the marshalling/unmarshalling.

Parameters

in, out	<i>ppRefCon</i>	Returns a pointer to be used for the reference context of marshalling/unmarshalling
in	<i>numComId</i>	Number of datasets found in the configuration
in	<i>pComIdDsIdMap</i>	Pointer to an array of structures of type TRDP_DATASET_T
in	<i>numDataSet</i>	Number of datasets found in the configuration
in	<i>pDataset</i>	Pointer to an array of pointers to structures of type TRDP_DATASET_T

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error

Types for marshalling / unmarshalling.

The supplied array must be sorted by ComIds. The array must exist during the use of the marshalling functions (until [tlc_terminate\(\)](#)).

Parameters

in, out	<i>ppRefCon</i>	Returns a pointer to be used for the reference context of marshalling/unmarshalling
in	<i>numComId</i>	Number of datasets found in the configuration
in	<i>pComIdDsIdMap</i>	Pointer to an array of structures of type TRDP_DATASET_T
in	<i>numDataSet</i>	Number of datasets found in the configuration
in	<i>pDataset</i>	Pointer to an array of pointers to structures of type TRDP_DATASET_T

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error

5.10.2.4 tau_marshall()

```
EXT_DECL TRDP_ERR_T tau_marshall (
    void * pRefCon,
    UINT32 comId,
    UINT8 * pSrc,
    UINT32 srcSize,
    UINT8 * pDest,
    UINT32 * pDestSize,
    TRDP_DATASET_T ** ppDSPointer )
```

marshall function.

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>comId</i>	ComId to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDest</i>	pointer to a buffer for the treated message
in, out	<i>pDestSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_COMID_ERR</i>	comid not existing
<i>TRDP_PARAM_ERR</i>	Parameter error

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>comId</i>	ComId to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDest</i>	pointer to a buffer for the treated message
in, out	<i>pDestSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_STATE_ERR</i>	Too deep recursion
<i>TRDP_MARSHALLING_ERR</i>	dataset/source size mismatch

5.10.2.5 tau_marshallDs()

```
EXT_DECL TRDP_ERR_T tau_marshallDs (
    void * pRefCon,
    UINT32 dsId,
    UINT8 * pSrc,
    UINT32 srcSize,
    UINT8 * pDest,
    UINT32 * pDestSize,
    TRDP_DATASET_T ** ppDSPointer )
```

marshall data set function.

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>dsId</i>	Data set id to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDest</i>	pointer to a buffer for the treated message
in, out	<i>pDestSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_COMID_ERR</i>	comid not existing
<i>TRDP_PARAM_ERR</i>	Parameter error

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>dsId</i>	Data set id to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDest</i>	pointer to a buffer for the treated message
in, out	<i>pDestSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_STATE_ERR</i>	Too deep recursion
<i>TRDP_COMID_ERR</i>	comid not existing
<i>TRDP_MARSHALLING_ERR</i>	dataset/source size mismatch

5.10.2.6 tau_unmarshall()

```
EXT_DECL TRDP_ERR_T tau_unmarshall (
    void * pRefCon,
    UINT32 comId,
    UINT8 * pSrc,
    UINT32 srcSize,
    UINT8 * pDest,
```

```
UINT32 * pDestSize,  
TRDP_DATASET_T ** ppDSPointer )
```

unmarshall function.

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>comId</i>	ComId to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDest</i>	pointer to a buffer for the treated message
in, out	<i>pDestSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_COMID_ERR</i>	comid not existing

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>comId</i>	ComId to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDest</i>	pointer to a buffer for the treated message
in, out	<i>pDestSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_STATE_ERR</i>	Too deep recursion
<i>TRDP_COMID_ERR</i>	comid not existing
<i>TRDP_MARSHALLING_ERR</i>	dataset/source size mismatch

5.10.2.7 tau_unmarshallDs()

```
EXT_DECL TRDP_ERR_T tau_unmarshallDs (
    void * pRefCon,
    UINT32 dsId,
    UINT8 * pSrc,
    UINT32 srcSize,
    UINT8 * pDest,
    UINT32 * pDestSize,
    TRDP_DATASET_T ** ppDSPointer )
```

unmarshall data set function.

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>dsId</i>	Data set id to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDest</i>	pointer to a buffer for the treated message
in, out	<i>pDestSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_COMID_ERR</i>	comid not existing

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>dsId</i>	Data set id to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDest</i>	pointer to a buffer for the treated message
in, out	<i>pDestSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppDSPointer</i>	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_STATE_ERR</i>	Too deep recursion
<i>TRDP_COMID_ERR</i>	comid not existing
<i>TRDP_MARSHALLING_ERR</i>	dataset/source size mismatch

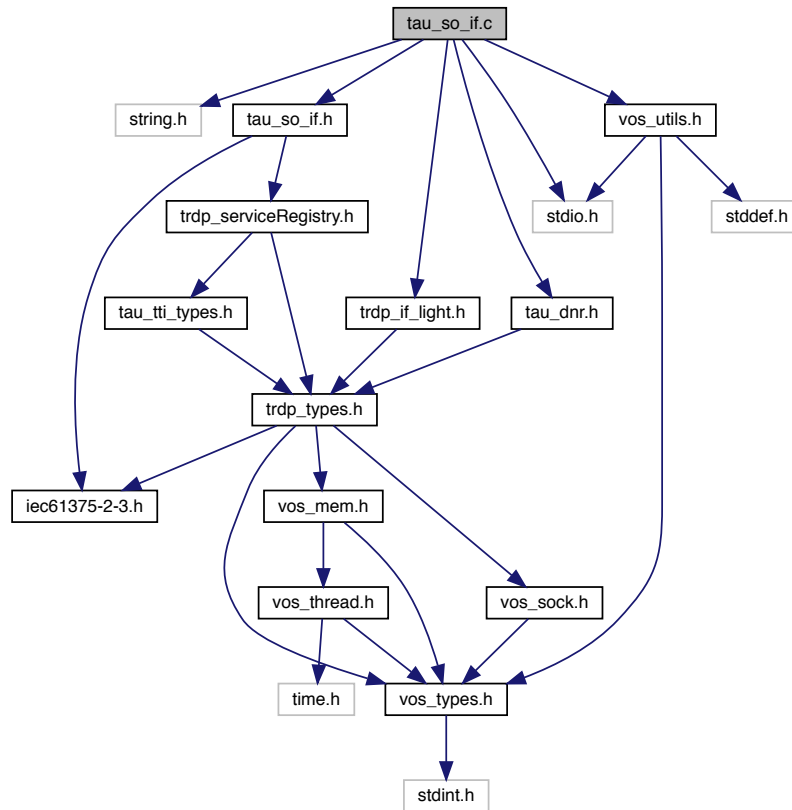
5.11 tau_so_if.c File Reference

Access to service oriented functions of the SRM.

```
#include <string.h>
#include <stdio.h>
#include "trdp_if_light.h"
#include "tau_dnr.h"
#include "tau_so_if.h"
```

```
#include "vos_utils.h"
```

Include dependency graph for tau_so_if.c:



Functions

- EXT_DECL [TRDP_ERR_T tau_addService](#) (TRDP_APP_SESSION_T appHandle, [SRM_SERVICE_INFO↔O_T](#) *pServiceToAdd, BOOL8 waitForCompletion)
Function to add to the service registry of the consist-local SRM.
- EXT_DECL [TRDP_ERR_T tau_delService](#) (TRDP_APP_SESSION_T appHandle, [SRM_SERVICE_INFO↔_T](#) *pServiceToRemove, BOOL8 waitForCompletion)
Remove the defined service from the service registry of the consist-local SRM.
- EXT_DECL [TRDP_ERR_T tau_updService](#) (TRDP_APP_SESSION_T appHandle, [SRM_SERVICE_INFO↔O_T](#) *pServiceToUpdate, BOOL8 waitForCompletion)
Register an update a service.
- EXT_DECL [TRDP_ERR_T tau_getServicesList](#) (TRDP_APP_SESSION_T appHandle, SRM_SERVICE_↔ENTRIES_T **ppServicesListBuffer, UINT32 *pNoOfServices, SRM_SERVICE_ENTRIES_T *pFilterEntry)
Get a list of the services known by the service registry of the local TTDB / SRM.
- EXT_DECL void [tau_freeServicesList](#) (SRM_SERVICE_ENTRIES_T *pServicesListBuffer)
Release the memory of a list received by tau_getServiceList.

5.11.1 Detailed Description

Access to service oriented functions of the SRM.

Because of the asynchronous behavior of the TTI subsystem, the source functions (add/upd/del) will return TRDP_↵P_NODATA_ERR if called with the the no-wait option.

Note

Project: TCNOpen TRDP prototype stack

Author

B. Loehr (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright NewTec GmbH 2019. All rights reserved.

5.11.2 Function Documentation

5.11.2.1 tau_addService()

```
EXT_DECL TRDP_ERR_T tau_addService (
    TRDP_APP_SESSION_T appHandle,
    SRM_SERVICE_INFO_T * pServiceToAdd,
    BOOL8 waitForCompletion )
```

Function to add to the service registry of the consist-local SRM.

Note: If waitForCompletion == TRUE, this function will block until completion (or timeout).

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
in, out	<i>pServiceToAdd</i>	Pointer to a service registry structure to be set and/or updated (returned)
in	<i>waitForCompletion</i>	if true, block for reply

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Data currently not available, try again later
<i>TRDP_TIMEOUT_ERR</i>	Reply timed out
<i>TRDP_SEMA_ERR</i>	Semaphore could not be aquired

5.11.2.2 tau_delService()

```
EXT_DECL TRDP_ERR_T tau_delService (
    TRDP_APP_SESSION_T appHandle,
    SRM_SERVICE_INFO_T * pServiceToRemove,
    BOOL8 waitForCompletion )
```

Remove the defined service from the service registry of the consist-local SRM.

Note: waitForCompletion is currently ignored, this function does not block.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
in, out	<i>pServiceToRemove</i>	Pointer to a service registry structure to be set and/or updated (returned)
in	<i>waitForCompletion</i>	if true, block for reply

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Data currently not available, try again later
<i>TRDP_TIMEOUT_ERR</i>	Reply timed out
<i>TRDP_SEMA_ERR</i>	Semaphore could not be aquired

5.11.2.3 tau_freeServicesList()

```
EXT_DECL void tau_freeServicesList (
    SRM_SERVICE_ENTRIES_T * pServicesListBuffer )
```

Release the memory of a list received by tau_getServiceList.

Parameters

in	<i>pServicesListBuffer</i>	Pointer to list aquired by getServiceList.
----	----------------------------	--

Return values

<i>none</i>	
-------------	--

5.11.2.4 tau_getServicesList()

```
EXT_DECL TRDP_ERR_T tau_getServicesList (
    TRDP_APP_SESSION_T appHandle,
```

```
SRM_SERVICE_ENTRIES_T ** ppServicesListBuffer,
UINT32 * pNoOfServices,
SRM_SERVICE_ENTRIES_T * pFilterEntry )
```

Get a list of the services known by the service registry of the local TTDB / SRM.

Note: This function will block until completion (or timeout). The buffer must be provided by the caller.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>ppServicesListBuffer</i>	Pointer to pointer containing the list. Has to be vos_memfree'd by user
out	<i>pNoOfServices</i>	Pointer to no. of services in returned list
in	<i>pFilterEntry</i>	Pointer to entry for filtering

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Data currently not available, try again later
<i>TRDP_TIMEOUT_ERR</i>	Reply timed out

5.11.2.5 tau_updService()

```
EXT_DECL TRDP_ERR_T tau_updService (
    TRDP_APP_SESSION_T appHandle,
    SRM_SERVICE_INFO_T * pServiceToUpdate,
    BOOL8 waitForCompletion )
```

Register an update a service.

Same as addService. Note: If waitForCompletion == TRUE, this function will block until completion (or timeout).

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
in, out	<i>pServiceToUpdate</i>	Pointer to a service registry structure to be updated
in	<i>waitForCompletion</i>	if true, block for reply

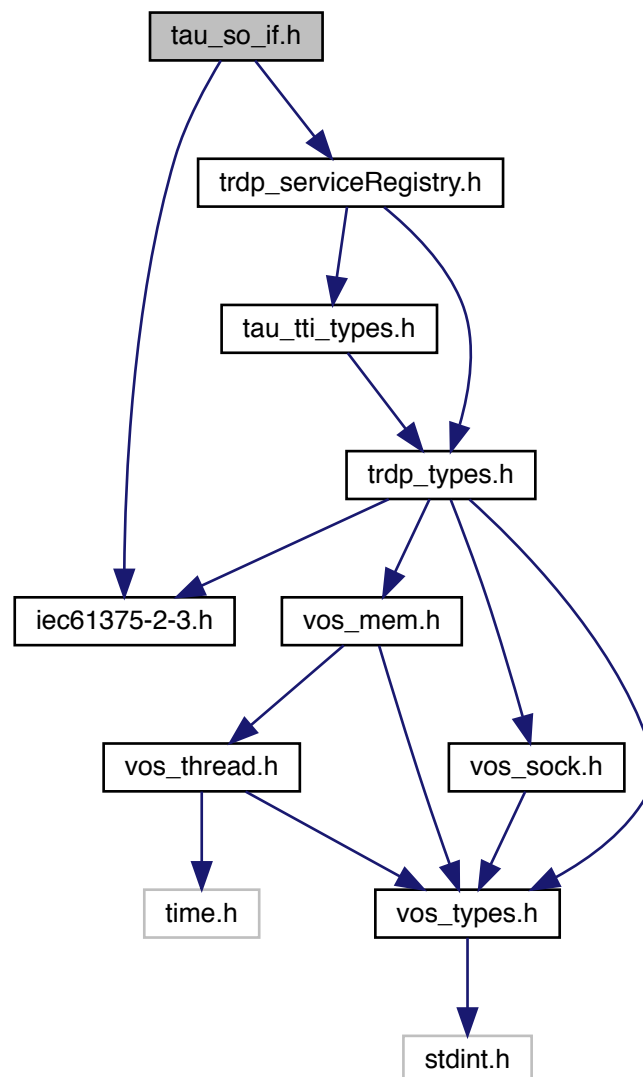
Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Data currently not available, try again later
<i>TRDP_TIMEOUT_ERR</i>	Reply timed out
<i>TRDP_SEMA_ERR</i>	Semaphore could not be aquired

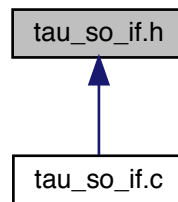
5.12 tau_so_if.h File Reference

Access to the Service Registry.

```
#include "iec61375-2-3.h"
#include "trdp_serviceRegistry.h"
Include dependency graph for tau_so_if.h:
```



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



Functions

- EXT_DECL [TRDP_ERR_T tau_addService](#) ([TRDP_APP_SESSION_T](#) appHandle, [SRM_SERVICE_INFO_T](#) *pServiceToAdd, [BOOL8](#) waitForCompletion)
Function to add to the service registry of the consist-local SRM.
- EXT_DECL [TRDP_ERR_T tau_delService](#) ([TRDP_APP_SESSION_T](#) appHandle, [SRM_SERVICE_INFO_T](#) *pServiceToAdd, [BOOL8](#) waitForCompletion)
Remove the defined service from the service registry of the consist-local SRM.
- EXT_DECL [TRDP_ERR_T tau_updService](#) ([TRDP_APP_SESSION_T](#) appHandle, [SRM_SERVICE_INFO_T](#) *pServiceToAdd, [BOOL8](#) waitForCompletion)
Register an update a service.
- EXT_DECL [TRDP_ERR_T tau_getServicesList](#) ([TRDP_APP_SESSION_T](#) appHandle, [SRM_SERVICE_ENTRIES_T](#) **ppServicesToAdd, [UINT32](#) *pNoOfServices, [SRM_SERVICE_ENTRIES_T](#) *pFilterEntry)
Get a list of the services known by the service registry of the local TTDB / SRM.
- EXT_DECL void [tau_freeServicesList](#) ([SRM_SERVICE_ENTRIES_T](#) *pServicesListBuffer)
Release the memory of a list received by tau_getServiceList.

5.12.1 Detailed Description

Access to the Service Registry.

This header file defines the proposed extensions and additions to access the service interface (proposed as extension to the TTDB defined in IEC61375-2-3:2017

Note

Project: TCNOpen TRDP prototype stack & FDF/DbD

Author

Bernd Loehr, NewTec GmbH, 2019-06-17

Remarks

Copyright 2019, NewTec GmbH

Id

[tau_so_if.h](#) 2091 2019-10-15 08:48:18Z s-bender

5.12.2 Function Documentation

5.12.2.1 tau_addService()

```
EXT_DECL TRDP_ERR_T tau_addService (
    TRDP_APP_SESSION_T appHandle,
    SRM_SERVICE_INFO_T * pServiceToAdd,
    BOOL8 waitForCompletion )
```

Function to add to the service registry of the consist-local SRM.

Note: If `waitForCompletion == TRUE`, this function will block until completion (or timeout).

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
in, out	<i>pServiceToAdd</i>	Pointer to a service registry structure to be set and/or updated (returned)
in	<i>waitForCompletion</i>	if true, block for reply

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Data currently not available, try again later
<i>TRDP_TIMEOUT_ERR</i>	Reply timed out
<i>TRDP_SEMA_ERR</i>	Semaphore could not be acquired

5.12.2.2 tau_delService()

```
EXT_DECL TRDP_ERR_T tau_delService (
    TRDP_APP_SESSION_T appHandle,
    SRM_SERVICE_INFO_T * pServiceToRemove,
    BOOL8 waitForCompletion )
```

Remove the defined service from the service registry of the consist-local SRM.

Note: `waitForCompletion` is currently ignored, this function does not block.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
in, out	<i>pServiceToRemove</i>	Pointer to a service registry structure to be set and/or updated (returned)
in	<i>waitForCompletion</i>	if true, block for reply

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Data currently not available, try again later
<i>TRDP_TIMEOUT_ERR</i>	Reply timed out
<i>TRDP_SEMA_ERR</i>	Semaphore could not be aquired

5.12.2.3 tau_freeServicesList()

```
EXT_DECL void tau_freeServicesList (
    SRM_SERVICE_ENTRIES_T * pServicesListBuffer )
```

Release the memory of a list received by tau_getServiceList.

Parameters

in	<i>pServicesListBuffer</i>	Pointer to list aquired by getServiceList.
----	----------------------------	--

Return values

<i>none</i>	
-------------	--

5.12.2.4 tau_getServicesList()

```
EXT_DECL TRDP_ERR_T tau_getServicesList (
    TRDP_APP_SESSION_T appHandle,
    SRM_SERVICE_ENTRIES_T ** ppServicesListBuffer,
    UINT32 * pNoOfServices,
    SRM_SERVICE_ENTRIES_T * pFilterEntry )
```

Get a list of the services known by the service registry of the local TTDB / SRM.

Note: This function will block until completion (or timeout). The buffer must be provided by the caller.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>ppServicesListBuffer</i>	Pointer to pointer containing the list. Has to be vos_memfree'd by user
out	<i>pNoOfServices</i>	Pointer to no. of services in returned list
in	<i>pFilterEntry</i>	Pointer to entry for filtering

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

Return values

<i>TRDP_NODATA_ERR</i>	Data currently not available, try again later
<i>TRDP_TIMEOUT_ERR</i>	Reply timed out

5.12.2.5 tau_updService()

```
EXT_DECL TRDP_ERR_T tau_updService (
    TRDP_APP_SESSION_T appHandle,
    SRM_SERVICE_INFO_T * pServiceToUpdate,
    BOOL8 waitForCompletion )
```

Register an update a service.

Same as addService. Note: If waitForCompletion == TRUE, this function will block until completion (or timeout).

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
in, out	<i>pServiceToUpdate</i>	Pointer to a service registry structure to be updated
in	<i>waitForCompletion</i>	if true, block for reply

Return values

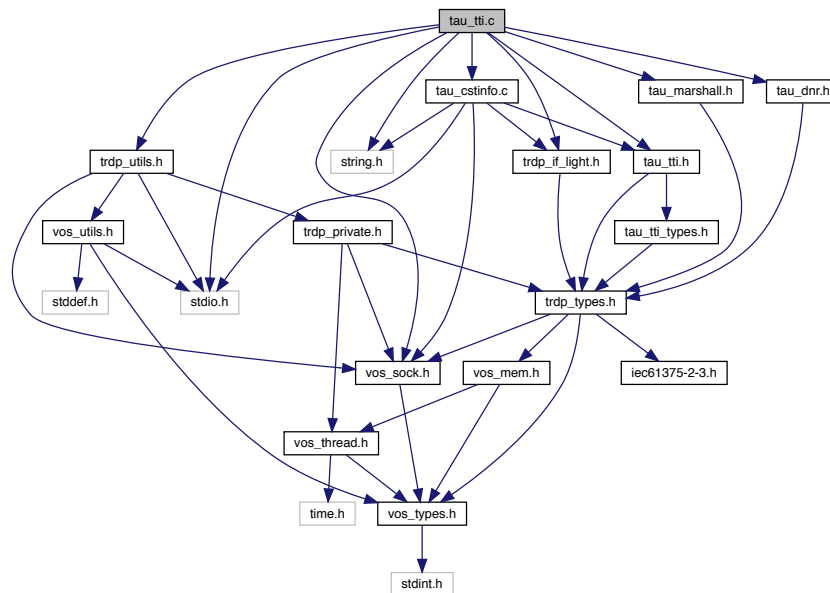
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Data currently not available, try again later
<i>TRDP_TIMEOUT_ERR</i>	Reply timed out
<i>TRDP_SEMA_ERR</i>	Semaphore could not be aquired

5.13 tau_tti.c File Reference

Functions for train topology information access.

```
#include <string.h>
#include <stdio.h>
#include "trdp_if_light.h"
#include "trdp_utils.h"
#include "tau_marshall.h"
#include "tau_tti.h"
#include "vos_sock.h"
#include "tau_dnr.h"
#include "tau_cstinfo.c"
```


Include dependency graph for tau_tti.c:



Macros

- `#define TTI_CACHED_CONSISTS 8u`
We hold this number of consist infos (ca.

Functions

- EXT_DECL `TRDP_ERR_T tau_initTTIaccess (TRDP_APP_SESSION_T appHandle, VOS_SEMA_T user↔ Action, TRDP_IP_ADDR_T ecsplpAddr, CHAR8 *hostsFileName)`
Function to init TTI access.
- EXT_DECL `void tau_delInitTTI (TRDP_APP_SESSION_T appHandle)`
Release any resources allocated by TTI Must be called before closing the session.
- EXT_DECL `TRDP_ERR_T tau_getOpTrDirectory (TRDP_APP_SESSION_T appHandle, TRDP_OP_TRA↔ IN_DIR_STATE_T *pOpTrnDirState, TRDP_OP_TRAIN_DIR_T *pOpTrnDir)`
Function to retrieve the operational train directory state.
- EXT_DECL `TRDP_ERR_T tau_getOpTrnDirectoryStatusInfo (TRDP_APP_SESSION_T appHandle, TRD↔ P_OP_TRAIN_DIR_STATUS_INFO_T *pOpTrnDirStatusInfo)`
Function to retrieve the operational train directory state info.
- EXT_DECL `TRDP_ERR_T tau_getTrDirectory (TRDP_APP_SESSION_T appHandle, TRDP_TRAIN_DIR↔ _T *pTrnDir)`
Function to retrieve the train directory.
- EXT_DECL `TRDP_ERR_T tau_getStaticCstInfo (TRDP_APP_SESSION_T appHandle, TRDP_CONSIST↔ _INFO_T *pCstInfo, TRDP_UUID_T const cstUUID)`
Function to retrieve the consist info.
- EXT_DECL `TRDP_ERR_T tau_getTTI (TRDP_APP_SESSION_T appHandle, TRDP_OP_TRAIN_DIR_S↔ TATE_T *pOpTrnDirState, TRDP_OP_TRAIN_DIR_T *pOpTrnDir, TRDP_TRAIN_DIR_T *pTrnDir, TRDP↔ _TRAIN_NET_DIR_T *pTrnNetDir)`
Function to retrieve the operational train directory.

- EXT_DECL [TRDP_ERR_T tau_getTrnCstCnt](#) ([TRDP_APP_SESSION_T](#) appHandle, [UINT16](#) *pTrnCstCnt)
Function to retrieve the total number of consists in the train.
- EXT_DECL [TRDP_ERR_T tau_getTrnVehCnt](#) ([TRDP_APP_SESSION_T](#) appHandle, [UINT16](#) *pTrnVehCnt)
Function to retrieve the total number of vehicles in the train.
- EXT_DECL [TRDP_ERR_T tau_getCstVehCnt](#) ([TRDP_APP_SESSION_T](#) appHandle, [UINT16](#) *pCstVehCnt, [const](#) [TRDP_LABEL_T](#) pCstLabel)
Function to retrieve the total number of vehicles in a consist.
- EXT_DECL [TRDP_ERR_T tau_getCstFctCnt](#) ([TRDP_APP_SESSION_T](#) appHandle, [UINT16](#) *pCstFctCnt, [const](#) [TRDP_LABEL_T](#) pCstLabel)
Function to retrieve the total number of functions in a consist.
- EXT_DECL [TRDP_ERR_T tau_getCstFctInfo](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_FUNCTION_↵](#)
[INFO_T](#) *pFctInfo, [const](#) [TRDP_LABEL_T](#) pCstLabel, [UINT16](#) maxFctCnt)
Function to retrieve the function information of the consist.
- EXT_DECL [TRDP_ERR_T tau_getVehInfo](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_VEHICLE_INF_↵](#)
[O_T](#) *pVehInfo, [const](#) [TRDP_LABEL_T](#) pVehLabel, [const](#) [TRDP_LABEL_T](#) pCstLabel)
Function to retrieve the vehicle information of a consist's vehicle.
- EXT_DECL [TRDP_ERR_T tau_getCstInfo](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_CONSIST_INF_↵](#)
[O_T](#) *pCstInfo, [const](#) [TRDP_LABEL_T](#) pCstLabel)
Function to retrieve the consist information of a train's consist.
- EXT_DECL [TRDP_ERR_T tau_getVehOrient](#) ([TRDP_APP_SESSION_T](#) appHandle, [UINT8](#) *pVehOrient, [UINT8](#) *pCstOrient, [TRDP_LABEL_T](#) pVehLabel, [TRDP_LABEL_T](#) pCstLabel)
Function to retrieve the orientation of the given vehicle.
- EXT_DECL [TRDP_ERR_T tau_getOwnIds](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_LABEL_T](#) *p↵
DevId, [TRDP_LABEL_T](#) *pVehId, [TRDP_LABEL_T](#) *pCstId)
Who am I ?.
- EXT_DECL [UINT8 tau_getOwnOpCstNo](#) ([TRDP_APP_SESSION_T](#) appHandle)
Get own operational consist number.
- EXT_DECL [UINT8 tau_getOwnTrnCstNo](#) ([TRDP_APP_SESSION_T](#) appHandle)
Get own train consist number.

5.13.1 Detailed Description

Functions for train topology information access.

The TTI subsystem maintains a pointer to the TAU_TTDB struct in the TRDP session struct. That TAU_TTDB struct keeps the subscription and listener handles, the current TTDB directories and a pointer list to consist infos (in network format). On init, most TTDB data is requested from the ECSP plus the own consist info. This data is automatically updated if an inauguration is detected. Additional consist infos are requested on demand, only. Because of the asynchronous behavior of the TTI subsystem, most functions in [tau_tti.c](#) may return [TRDP_N↵](#)
[ODATA_ERR](#) on first invocation. They should be called again after 1...3 seconds (3s is the timeout for most MD replies).

Note

Project: TCNOpen TRDP prototype stack

Author

B. Loehr (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2016-2019. All rights reserved.

5.13.2 Macro Definition Documentation

5.13.2.1 TTI_CACHED_CONSISTS

```
#define TTI_CACHED_CONSISTS 8u
```

We hold this number of consist infos (ca.

105kB)

5.13.3 Function Documentation

5.13.3.1 tau_deinitTTI()

```
EXT_DECL void tau_deinitTTI (  
    TRDP_APP_SESSION_T appHandle )
```

Release any resources allocated by TTI Must be called before closing the session.

Function to terminate TTI access.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
----	------------------	--

Return values

<i>none</i>	
-------------	--

5.13.3.2 tau_getCstFctCnt()

```
EXT_DECL TRDP_ERR_T tau_getCstFctCnt (  
    TRDP_APP_SESSION_T appHandle,  
    UINT16 * pCstFctCnt,  
    const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the total number of functions in a consist.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pCstFctCnt</i>	Pointer to the number of functions to be returned
in	<i>pCstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.13.3.3 tau_getCstFctInfo()

```
EXT_DECL TRDP_ERR_T tau_getCstFctInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FUNCTION_INFO_T * pFctInfo,
    const TRDP_LABEL_T pCstLabel,
    UINT16 maxFctCnt )
```

Function to retrieve the function information of the consist.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pFctInfo</i>	Pointer to function info list to be returned. Memory needs to be provided by application. Set NULL if not used.
in	<i>pCstLabel</i>	Pointer to a consist label. NULL means own consist.
in	<i>maxFctCnt</i>	Maximal number of functions to be returned in provided buffer.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.13.3.4 tau_getCstInfo()

```
EXT_DECL TRDP_ERR_T tau_getCstInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_CONSIST_INFO_T * pCstInfo,
    const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the consist information of a train's consist.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pCstInfo</i>	Pointer to the consist info to be returned.
in	<i>pCstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.13.3.5 tau_getCstVehCnt()

```
EXT_DECL TRDP_ERR_T tau_getCstVehCnt (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pCstVehCnt,
    const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the total number of vehicles in a consist.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pCstVehCnt</i>	Pointer to the number of vehicles to be returned
in	<i>pCstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Try again

5.13.3.6 tau_getOpTrDirectory()

```
EXT_DECL TRDP_ERR_T tau_getOpTrDirectory (
    TRDP_APP_SESSION_T appHandle,
    TRDP_OP_TRAIN_DIR_STATE_T * pOpTrnDirState,
    TRDP_OP_TRAIN_DIR_T * pOpTrnDir )
```

Function to retrieve the operational train directory state.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pOpTrnDirState</i>	Pointer to an operational train directory state structure to be returned.
out	<i>pOpTrnDir</i>	Pointer to an operational train directory structure to be returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Data currently not available, try again later

5.13.3.7 tau_getOpTrnDirectoryStatusInfo()

```
EXT_DECL TRDP_ERR_T tau_getOpTrnDirectoryStatusInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_OP_TRAIN_DIR_STATUS_INFO_T * pOpTrnDirStatusInfo )
```

Function to retrieve the operational train directory state info.

Return a copy of the last received PD 100 telegram. Note: The values are in host endianness! When validating (v2), network endianness must be ensured.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pOpTrnDirStatusInfo</i>	Pointer to an operational train directory state structure to be returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.13.3.8 tau_getOwnIds()

```
EXT_DECL TRDP_ERR_T tau_getOwnIds (
    TRDP_APP_SESSION_T appHandle,
    TRDP_LABEL_T * pDevId,
    TRDP_LABEL_T * pVehId,
    TRDP_LABEL_T * pCstId )
```

Who am I ?.

Realizes a kind of 'Who am I' function. It is used to determine the own identifiers (i.e. the own labels), which may be used as host part of the own fully qualified domain name.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession()
out	<i>pDevId</i>	Returns the device label (host name)
out	<i>pVehId</i>	Returns the vehicle label
out	<i>pCstId</i>	Returns the consist label

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Data currently not available, call again

5.13.3.9 tau_getOwnOpCstNo()

```
EXT_DECL UINT8 tau_getOwnOpCstNo (
    TRDP_APP_SESSION_T appHandle )
```

Get own operational consist number.

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_init</code>
----	------------------	--

Return values

<i>ownOpCstNo</i>	own operational consist number value 0 on error
-------------------	---

5.13.3.10 tau_getOwnTrnCstNo()

```
EXT_DECL UINT8 tau_getOwnTrnCstNo (
    TRDP_APP_SESSION_T appHandle )
```

Get own train consist number.

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_init</code>
----	------------------	--

Return values

<i>ownTrnCstNo</i>	own train consist number value 0 on error
--------------------	---

5.13.3.11 tau_getStaticCstInfo()

```
EXT_DECL TRDP_ERR_T tau_getStaticCstInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_CONSIST_INFO_T * pCstInfo,
    TRDP_UUID_T const cstUUID )
```

Function to retrieve the consist info.

Function to retrieve the operational train directory.

Parameters

in	<i>appHandle</i>	Handle returned by <code>tlc_openSession()</code> .
out	<i>pCstInfo</i>	Pointer to a consist info structure to be returned.
in	<i>cstUUID</i>	UUID of the consist the consist info is requested for.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.13.3.12 tau_getTrDirectory()

```
EXT_DECL TRDP_ERR_T tau_getTrDirectory (
    TRDP_APP_SESSION_T appHandle,
    TRDP_TRAIN_DIR_T * pTrnDir )
```

Function to retrieve the train directory.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pTrnDir</i>	Pointer to a train directory structure to be returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Try later

5.13.3.13 tau_getTrnCstCnt()

```
EXT_DECL TRDP_ERR_T tau_getTrnCstCnt (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pTrnCstCnt )
```

Function to retrieve the total number of consists in the train.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pTrnCstCnt</i>	Pointer to the number of consists to be returned

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Try again

5.13.3.14 tau_getTrnVehCnt()

```
EXT_DECL TRDP_ERR_T tau_getTrnVehCnt (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pTrnVehCnt )
```

Function to retrieve the total number of vehicles in the train.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pTrnVehCnt</i>	Pointer to the number of vehicles to be returned

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Try again

5.13.3.15 tau_getTTI()

```
EXT_DECL TRDP_ERR_T tau_getTTI (
    TRDP_APP_SESSION_T appHandle,
    TRDP_OP_TRAIN_DIR_STATE_T * pOpTrnDirState,
    TRDP_OP_TRAIN_DIR_T * pOpTrnDir,
    TRDP_TRAIN_DIR_T * pTrnDir,
    TRDP_TRAIN_NET_DIR_T * pTrnNetDir )
```

Function to retrieve the operational train directory.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pOpTrnDirState</i>	Pointer to an operational train directory state structure to be returned.
out	<i>pOpTrnDir</i>	Pointer to an operational train directory structure to be returned.
out	<i>pTrnDir</i>	Pointer to a train directory structure to be returned.
out	<i>pTrnNetDir</i>	Pointer to a train network directory structure to be returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.13.3.16 tau_getVehInfo()

```
EXT_DECL TRDP_ERR_T tau_getVehInfo (
    TRDP_APP_SESSION_T appHandle,
```

```

TRDP_VEHICLE_INFO_T * pVehInfo,
const TRDP_LABEL_T pVehLabel,
const TRDP_LABEL_T pCstLabel )

```

Function to retrieve the vehicle information of a consist's vehicle.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pVehInfo</i>	Pointer to the vehicle info to be returned.
in	<i>pVehLabel</i>	Pointer to a vehicle label. NULL means own vehicle if cstLabel refers to own consist.
in	<i>pCstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.13.3.17 tau_getVehOrient()

```

EXT_DECL TRDP_ERR_T tau_getVehOrient (
    TRDP_APP_SESSION_T appHandle,
    UINT8 * pVehOrient,
    UINT8 * pCstOrient,
    TRDP_LABEL_T pVehLabel,
    TRDP_LABEL_T pCstLabel )

```

Function to retrieve the orientation of the given vehicle.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pVehOrient</i>	Pointer to the vehicle orientation to be returned '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction
out	<i>pCstOrient</i>	Pointer to the consist orientation to be returned '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction
in	<i>pVehLabel</i>	vehLabel = NULL means own vehicle if cstLabel == NULL, currently ignored.
in	<i>pCstLabel</i>	cstLabel = NULL means own consist

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.13.3.18 tau_initTTIaccess()

```

EXT_DECL TRDP_ERR_T tau_initTTIaccess (

```

```
TRDP_APP_SESSION_T appHandle,  
VOS_SEMA_T userAction,  
TRDP_IP_ADDR_T ecspIpAddr,  
CHAR8 * hostsFileName )
```

Function to init TTI access.

Subscribe to necessary process data for correct ECSP handling, further calls need DNS!

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
in	<i>userAction</i>	Semaphore to fire if inauguration took place.
in	<i>ecspIpAddr</i>	ECSP IP address. Currently not used.
in	<i>hostsFileName</i>	Optional host file name as ECSP replacement. Currently not implemented.

Return values

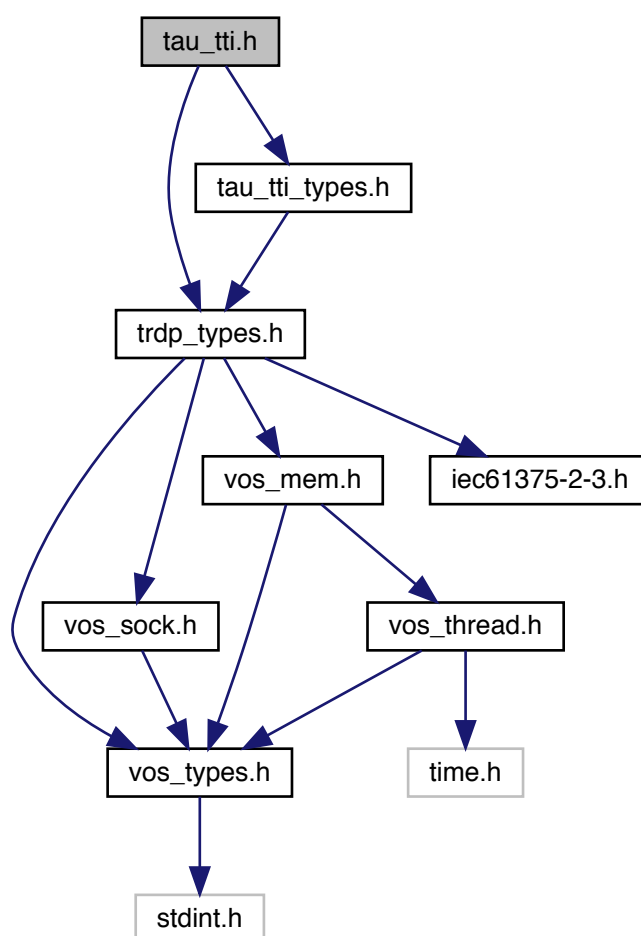
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

5.14 tau_tti.h File Reference

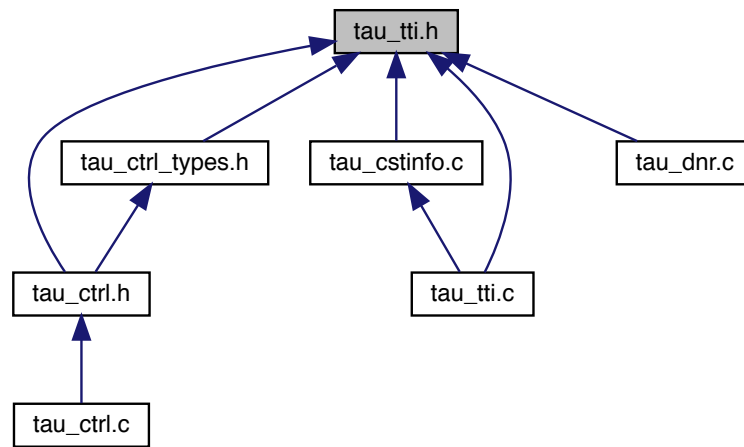
TRDP utility interface definitions.

```
#include "trdp_types.h"  
#include "tau_tti_types.h"
```

Include dependency graph for tau_tti.h:



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



Functions

- EXT_DECL [TRDP_ERR_T tau_initTTIaccess](#) ([TRDP_APP_SESSION_T](#) appHandle, [VOS_SEMA_T](#) user↔ Action, [TRDP_IP_ADDR_T](#) ecsplpAddr, [CHAR8](#) *hostsFileName)
Function to init TTI access.
- EXT_DECL void [tau_deinitTTI](#) ([TRDP_APP_SESSION_T](#) appHandle)
Function to terminate TTI access.
- EXT_DECL [TRDP_ERR_T tau_getOpTrDirectory](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_OP_TRA↔ IN_DIR_STATE_T](#) *pOpTrnDirState, [TRDP_OP_TRAIN_DIR_T](#) *pOpTrnDir)
Function to retrieve the operational train directory state.
- EXT_DECL [TRDP_ERR_T tau_getOpTrnDirectoryStatusInfo](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRD↔ P_OP_TRAIN_DIR_STATUS_INFO_T](#) *pOpTrnDirStatusInfo)
Function to retrieve the operational train directory state info.
- EXT_DECL [TRDP_ERR_T tau_getTrDirectory](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_TRAIN_DIR↔ _T](#) *pTrnDir)
Function to retrieve the train directory.
- EXT_DECL [TRDP_ERR_T tau_getStaticCstInfo](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_CONSIST↔ _INFO_T](#) *pCstInfo, [TRDP_UUID_T](#) const cstUUID)
Function to retrieve the operational train directory.
- EXT_DECL [TRDP_ERR_T tau_getTTI](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_OP_TRAIN_DIR_S↔ TATE_T](#) *pOpTrnDirState, [TRDP_OP_TRAIN_DIR_T](#) *pOpTrnDir, [TRDP_TRAIN_DIR_T](#) *pTrnDir, [TRDP↔ _TRAIN_NET_DIR_T](#) *pTrnNetDir)
Function to retrieve the operational train directory.
- EXT_DECL [TRDP_ERR_T tau_getTrnCstCnt](#) ([TRDP_APP_SESSION_T](#) appHandle, [UINT16](#) *pTrnCstCnt)
Function to retrieve the total number of consists in the train.
- EXT_DECL [TRDP_ERR_T tau_getTrnVehCnt](#) ([TRDP_APP_SESSION_T](#) appHandle, [UINT16](#) *pTrnVehCnt)
Function to retrieve the total number of vehicles in the train.
- EXT_DECL [TRDP_ERR_T tau_getCstVehCnt](#) ([TRDP_APP_SESSION_T](#) appHandle, [UINT16](#) *pCstVehCnt, [const TRDP_LABEL_T](#) pCstLabel)
Function to retrieve the total number of vehicles in a consist.

- EXT_DECL [TRDP_ERR_T tau_getCstFctCnt](#) ([TRDP_APP_SESSION_T](#) appHandle, [UINT16](#) *pCstFctCnt, [const TRDP_LABEL_T](#) pCstLabel)
Function to retrieve the total number of functions in a consist.
- EXT_DECL [TRDP_ERR_T tau_getCstFctInfo](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_FUNCTION_INFO_T](#) *pFctInfo, [const TRDP_LABEL_T](#) pCstLabel, [UINT16](#) maxFctCnt)
Function to retrieve the function information of the consist.
- EXT_DECL [TRDP_ERR_T tau_getVehInfo](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_VEHICLE_INFO_T](#) *pVehInfo, [const TRDP_LABEL_T](#) pVehLabel, [const TRDP_LABEL_T](#) pCstLabel)
Function to retrieve the vehicle information of a consist's vehicle.
- EXT_DECL [TRDP_ERR_T tau_getCstInfo](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_CONSIST_INFO_T](#) *pCstInfo, [const TRDP_LABEL_T](#) pCstLabel)
Function to retrieve the consist information of a train's consist.
- EXT_DECL [TRDP_ERR_T tau_getVehOrient](#) ([TRDP_APP_SESSION_T](#) appHandle, [UINT8](#) *pVehOrient, [UINT8](#) *pCstOrient, [TRDP_LABEL_T](#) pVehLabel, [TRDP_LABEL_T](#) pCstLabel)
Function to retrieve the orientation of the given vehicle.
- EXT_DECL [TRDP_ERR_T tau_getOwnIds](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_LABEL_T](#) *pDevId, [TRDP_LABEL_T](#) *pVehId, [TRDP_LABEL_T](#) *pCstId)
Who am I ?.
- EXT_DECL [UINT8 tau_getOwnOpCstNo](#) ([TRDP_APP_SESSION_T](#) appHandle)
Get own operational consist number.
- EXT_DECL [UINT8 tau_getOwnTrnCstNo](#) ([TRDP_APP_SESSION_T](#) appHandle)
Get own train consist number.

5.14.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- train topology information access

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2014. All rights reserved.

5.14.2 Function Documentation

5.14.2.1 tau_deinitTTI()

```
EXT_DECL void tau_deinitTTI (
    TRDP\_APP\_SESSION\_T appHandle )
```

Function to terminate TTI access.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
----	------------------	--

Return values

<i>none</i>	Function to terminate TTI access.
-------------	-----------------------------------

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
----	------------------	--

Return values

<i>none</i>	
-------------	--

5.14.2.2 tau_getCstFctCnt()

```
EXT_DECL TRDP_ERR_T tau_getCstFctCnt (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pCstFctCnt,
    const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the total number of functions in a consist.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pCstFctCnt</i>	Pointer to the number of functions to be returned
in	<i>pCstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.14.2.3 tau_getCstFctInfo()

```
EXT_DECL TRDP_ERR_T tau_getCstFctInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FUNCTION_INFO_T * pFctInfo,
    const TRDP_LABEL_T pCstLabel,
    UINT16 maxFctCnt )
```

Function to retrieve the function information of the consist.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pFctInfo</i>	Pointer to function info list to be returned. Memory needs to be provided by application. Set NULL if not used.
in	<i>pCstLabel</i>	Pointer to a consist label. NULL means own consist.
in	<i>maxFctCnt</i>	Maximal number of functions to be returned in provided buffer.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.14.2.4 tau_getCstInfo()

```
EXT_DECL TRDP_ERR_T tau_getCstInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_CONSIST_INFO_T * pCstInfo,
    const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the consist information of a train's consist.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pCstInfo</i>	Pointer to the consist info to be returned.
in	<i>pCstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.14.2.5 tau_getCstVehCnt()

```
EXT_DECL TRDP_ERR_T tau_getCstVehCnt (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pCstVehCnt,
    const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the total number of vehicles in a consist.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pCstVehCnt</i>	Pointer to the number of vehicles to be returned
in	<i>pCstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pCstVehCnt</i>	Pointer to the number of vehicles to be returned
in	<i>pCstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Try again

5.14.2.6 tau_getOpTrDirectory()

```
EXT_DECL TRDP\_ERR\_T tau_getOpTrDirectory (
    TRDP\_APP\_SESSION\_T appHandle,
    TRDP\_OP\_TRAIN\_DIR\_STATE\_T * pOpTrnDirState,
    TRDP\_OP\_TRAIN\_DIR\_T * pOpTrnDir )
```

Function to retrieve the operational train directory state.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pOpTrnDirState</i>	Pointer to an operational train directory state structure to be returned.
out	<i>pOpTrnDir</i>	Pointer to an operational train directory structure to be returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pOpTrnDirState</i>	Pointer to an operational train directory state structure to be returned.
out	<i>pOpTrnDir</i>	Pointer to an operational train directory structure to be returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Data currently not available, try again later

5.14.2.7 tau_getOpTrnDirectoryStatusInfo()

```
EXT_DECL TRDP_ERR_T tau_getOpTrnDirectoryStatusInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_OP_TRAIN_DIR_STATUS_INFO_T * pOpTrnDirStatusInfo )
```

Function to retrieve the operational train directory state info.

Return a copy of the last received PD 100 telegram. Note: The values are in host endianness! When validating (SDTv2), network endianness must be ensured.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pOpTrnDirStatusInfo</i>	Pointer to an operational train directory state structure to be returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

Return a copy of the last received PD 100 telegram. Note: The values are in host endianness! When validating (v2), network endianness must be ensured.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pOpTrnDirStatusInfo</i>	Pointer to an operational train directory state structure to be returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.14.2.8 tau_getOwnIds()

```
EXT_DECL TRDP_ERR_T tau_getOwnIds (
    TRDP_APP_SESSION_T appHandle,
    TRDP_LABEL_T * pDevId,
    TRDP_LABEL_T * pVehId,
    TRDP_LABEL_T * pCstId )
```

Who am I ?.

Realizes a kind of 'Who am I' function. It is used to determine the own identifiers (i.e. the own labels), which may be used as host part of the own fully qualified domain name.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession()
out	<i>pDevId</i>	Returns the device label (host name)
out	<i>pVehId</i>	Returns the vehicle label
out	<i>pCstId</i>	Returns the consist label

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Data currently not available, call again

5.14.2.9 tau_getOwnOpCstNo()

```
EXT_DECL_UINT8 tau_getOwnOpCstNo (  
    TRDP_APP_SESSION_T appHandle )
```

Get own operational consist number.

Parameters

in	<i>appHandle</i>	The handle returned by tlc_init
----	------------------	---

Return values

<i>ownOpCstNo</i>	own operational consist number value 0 on error
-------------------	---

5.14.2.10 tau_getOwnTrnCstNo()

```
EXT_DECL_UINT8 tau_getOwnTrnCstNo (  
    TRDP_APP_SESSION_T appHandle )
```

Get own train consist number.

Parameters

in	<i>appHandle</i>	The handle returned by tlc_init
----	------------------	---

Return values

<i>ownTrnCstNo</i>	own train consist number value 0 on error
--------------------	---

5.14.2.11 tau_getStaticCstInfo()

```
EXT_DECL TRDP_ERR_T tau_getStaticCstInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_CONSIST_INFO_T * pCstInfo,
    TRDP_UUID_T const cstUUID )
```

Function to retrieve the operational train directory.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pCstInfo</i>	Pointer to a consist info structure to be returned.
in	<i>cstUUID</i>	UUID of the consist the consist info is requested for.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

Function to retrieve the operational train directory.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pCstInfo</i>	Pointer to a consist info structure to be returned.
in	<i>cstUUID</i>	UUID of the consist the consist info is requested for.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.14.2.12 tau_getTrDirectory()

```
EXT_DECL TRDP_ERR_T tau_getTrDirectory (
    TRDP_APP_SESSION_T appHandle,
    TRDP_TRAIN_DIR_T * pTrnDir )
```

Function to retrieve the train directory.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pTrnDir</i>	Pointer to a train directory structure to be returned.

Return values

<i>TRDP_NO_ERR</i>	no error
--------------------	----------

Return values

<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Try later

5.14.2.13 tau_getTrnCstCnt()

```
EXT_DECL TRDP_ERR_T tau_getTrnCstCnt (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pTrnCstCnt )
```

Function to retrieve the total number of consists in the train.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pTrnCstCnt</i>	Pointer to the number of consists to be returned

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pTrnCstCnt</i>	Pointer to the number of consists to be returned

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Try again

5.14.2.14 tau_getTrnVehCnt()

```
EXT_DECL TRDP_ERR_T tau_getTrnVehCnt (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pTrnVehCnt )
```

Function to retrieve the total number of vehicles in the train.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pTrnVehCnt</i>	Pointer to the number of vehicles to be returned

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pTrnVehCnt</i>	Pointer to the number of vehicles to be returned

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Try again

5.14.2.15 tau_getTTI()

```
EXT_DECL TRDP_ERR_T tau_getTTI (
    TRDP_APP_SESSION_T appHandle,
    TRDP_OP_TRAIN_DIR_STATE_T * pOpTrnDirState,
    TRDP_OP_TRAIN_DIR_T * pOpTrnDir,
    TRDP_TRAIN_DIR_T * pTrnDir,
    TRDP_TRAIN_NET_DIR_T * pTrnNetDir )
```

Function to retrieve the operational train directory.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pOpTrnDirState</i>	Pointer to an operational train directory state structure to be returned.
out	<i>pOpTrnDir</i>	Pointer to an operational train directory structure to be returned.
out	<i>pTrnDir</i>	Pointer to a train directory structure to be returned.
out	<i>pTrnNetDir</i>	Pointer to a train network directory structure to be returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.14.2.16 tau_getVehInfo()

```
EXT_DECL TRDP_ERR_T tau_getVehInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_VEHICLE_INFO_T * pVehInfo,
```

```
const TRDP_LABEL_T pVehLabel,
const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the vehicle information of a consist's vehicle.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pVehInfo</i>	Pointer to the vehicle info to be returned.
in	<i>pVehLabel</i>	Pointer to a vehicle label. NULL means own vehicle if cstLabel refers to own consist.
in	<i>pCstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.14.2.17 tau_getVehOrient()

```
EXT_DECL TRDP_ERR_T tau_getVehOrient (
    TRDP_APP_SESSION_T appHandle,
    UINT8 * pVehOrient,
    UINT8 * pCstOrient,
    TRDP_LABEL_T pVehLabel,
    TRDP_LABEL_T pCstLabel )
```

Function to retrieve the orientation of the given vehicle.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pVehOrient</i>	Pointer to the vehicle orientation to be returned '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction
out	<i>pCstOrient</i>	Pointer to the consist orientation to be returned '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction
in	<i>pVehLabel</i>	vehLabel = NULL means own vehicle if cstLabel == NULL
in	<i>pCstLabel</i>	cstLabel = NULL means own consist

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pVehOrient</i>	Pointer to the vehicle orientation to be returned '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction
out	<i>pCstOrient</i>	Pointer to the consist orientation to be returned '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction

Parameters

in	<i>pVehLabel</i>	vehLabel = NULL means own vehicle if cstLabel == NULL, currently ignored.
in	<i>pCstLabel</i>	cstLabel = NULL means own consist

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.14.2.18 tau_initTTIaccess()

```
EXT_DECL TRDP_ERR_T tau_initTTIaccess (
    TRDP_APP_SESSION_T appHandle,
    VOS_SEMA_T userAction,
    TRDP_IP_ADDR_T ecspIpAddr,
    CHAR8 * hostsFileName )
```

Function to init TTI access.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
in	<i>userAction</i>	Semaphore to fire if inauguration took place.
in	<i>ecspIpAddr</i>	ECSP IP address.
in	<i>hostsFileName</i>	Optional host file name as ECSP replacement.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

Subscribe to necessary process data for correct ECSP handling, further calls need DNS!

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
in	<i>userAction</i>	Semaphore to fire if inauguration took place.
in	<i>ecspIpAddr</i>	ECSP IP address. Currently not used.
in	<i>hostsFileName</i>	Optional host file name as ECSP replacement. Currently not implemented.

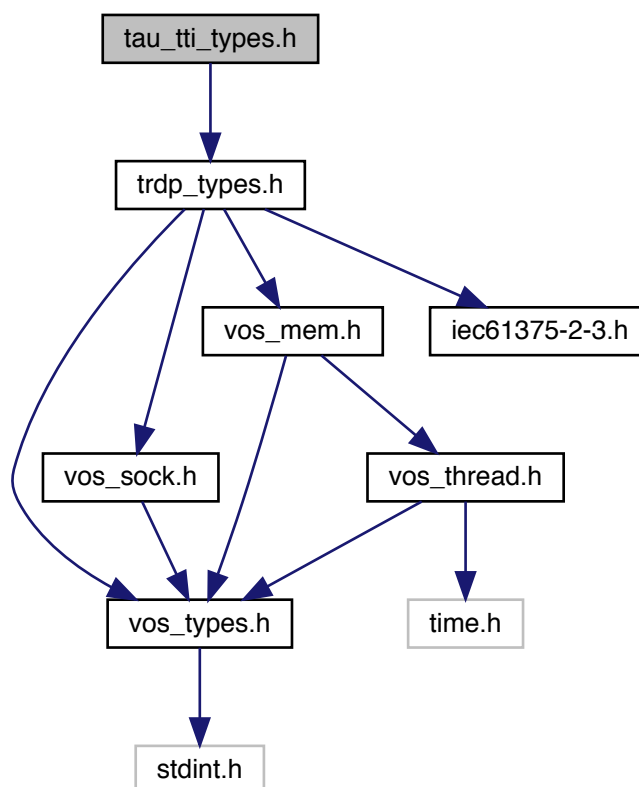
Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

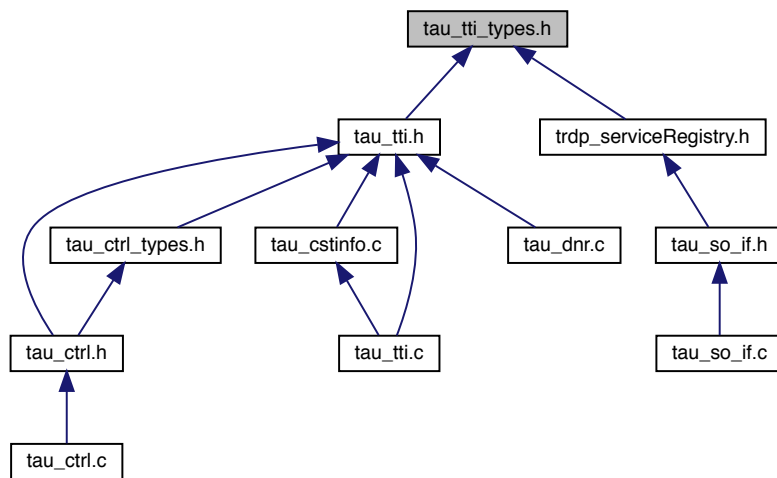
5.15 tau_tti_types.h File Reference

TRDP utility interface definitions.

```
#include "trdp_types.h"
Include dependency graph for tau_tti_types.h:
```



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



Data Structures

- struct [GNU_PACKED](#)
Types for ETB control.
- struct [TRDP_ETB_INFO_T](#)
Types for train configuration information.
- struct [TRDP_CLTR_CST_INFO_T](#)
Closed train consists information.
- struct [TRDP_PROP_T](#)
Application defined properties.
- struct [TRDP_FUNCTION_INFO_T](#)
function/device information structure
- struct [TRDP_VEHICLE_INFO_T](#)
vehicle information structure
- struct [TRDP_CONSIST_INFO_T](#)
consist information structure
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.

Types for ETB control.

- struct [GNU_PACKED](#)

Types for ETB control.

- struct [GNU_PACKED](#)

Types for ETB control.

- struct [GNU_PACKED](#)

Types for ETB control.

- struct [GNU_PACKED](#)

Types for ETB control.

- struct [GNU_PACKED](#)

Types for ETB control.

Macros

- #define [TRDP_MAX_CST_CNT](#) 63u
max number of consists per train
- #define [TRDP_MAX_VEH_CNT](#) 63u
max number of vehicles per train

5.15.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- train topology information access type definitions acc. to IEC61375-2-3

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

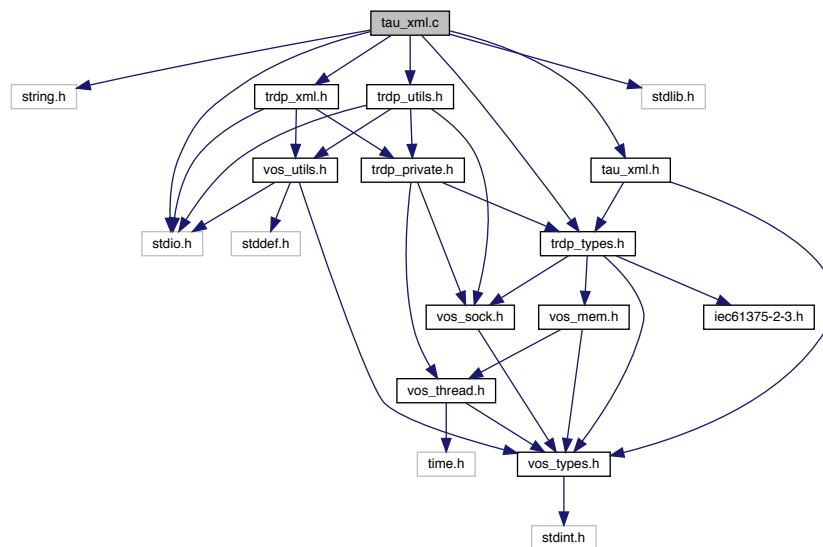
This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2014. All rights reserved.

5.16 tau_xml.c File Reference

Functions for XML file parsing.

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "trdp_types.h"
#include "trdp_utils.h"
#include "tau_xml.h"
#include "trdp_xml.h"
```

Include dependency graph for tau_xml.c:



Macros

- `#define TRDP_SDT_DEFAULT_SMI2 0u`
Default SDT safe message identifier.
- `#define TRDP_SDT_DEFAULT_NRXSAFE 3u`
Default SDT timeout cycles.
- `#define TRDP_SDT_DEFAULT_NGUARD 100u`
Default SDT initial timeout cycles.
- `#define TRDP_SDT_DEFAULT_CMTHR 10u`
Default SDT chan.
- `#define TRDP_SDT_DEFAULT_LMIMAX (11u*TRDP_SDT_DEFAULT_NRXSAFE)`
Default SDT chan.

Functions

- `EXT_DECL TRDP_ERR_T tau_prepareXmlDoc (const CHAR8 *pFileName, TRDP_XML_DOC_HANDLE_T *pDocHnd)`
Open XML file, prepare XPath context.

- EXT_DECL [TRDP_ERR_T tau_prepareXmlMem](#) (char *pBuffer, size_t bufSize, [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd)
Open XML stream, prepare XPath context.
- EXT_DECL void [tau_freeXmlDoc](#) ([TRDP_XML_DOC_HANDLE_T](#) *pDocHnd)
Free all the memory allocated by tau_prepareXmlDoc.
- EXT_DECL [TRDP_ERR_T tau_readXmlInterfaceConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, const CHAR8 *plfName, [TRDP_PROCESS_CONFIG_T](#) *pProcessConfig, [TRDP_PD_CONFIG_T](#) *pPdConfig, [TRDP_MD_CONFIG_T](#) *pMdConfig, UINT32 *pNumExchgPar, [TRDP_EXCHG_PAR_T](#) **ppExchgPar)
Read the interface relevant telegram parameters (except data set configuration) out of the configuration file .
- EXT_DECL void [tau_freeTelegrams](#) (UINT32 numExchgPar, [TRDP_EXCHG_PAR_T](#) *pExchgPar)
Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.
- EXT_DECL [TRDP_ERR_T tau_readXmlDeviceConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, [TRDP_MEM_CONFIG_T](#) *pMemConfig, [TRDP_DBG_CONFIG_T](#) *pDbgConfig, UINT32 *pNumComPar, [TRDP_COM_PAR_T](#) **ppComPar, UINT32 *pNumIfConfig, [TRDP_IF_CONFIG_T](#) **pplfConfig)
Function to read the TRDP device configuration parameters out of the XML configuration file.
- EXT_DECL [TRDP_ERR_T tau_readXmlMappedDevices](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, UINT32 *pNumProcConfig, [TRDP_PROCESS_CONFIG_T](#) **ppProcessConfig)
Function to read the TRDP mapped devices out of the XML configuration file.
- EXT_DECL [TRDP_ERR_T tau_readXmlMappedDeviceConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, const CHAR8 *pHostname, UINT32 *pNumIfConfig, [TRDP_IF_CONFIG_T](#) **pplfConfig)
Function to read the TRDP mapped device configuration parameters for a particular host out of the XML configuration file.
- EXT_DECL [TRDP_ERR_T tau_readXmlMappedInterfaceConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, const CHAR8 *pHostname, const CHAR8 *plfName, UINT32 *pNumExchgPar, [TRDP_EXCHG_PAR_T](#) **ppExchgPar)
Read the interface relevant mapped telegram parameters for a particular host and it's interface out of the configuration file .
- EXT_DECL [TRDP_ERR_T tau_readXmlDatasetConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, UINT32 *pNumComId, [TRDP_COMID_DSID_MAP_T](#) **ppComIdDsidMap, UINT32 *pNumDataset, [TRDP_DATASET_T](#) *pDataset)
Function to read the DataSet configuration out of the XML configuration file.
- EXT_DECL void [tau_freeXmlDatasetConfig](#) (UINT32 numComId, [TRDP_COMID_DSID_MAP_T](#) *pComIdDsidMap, UINT32 numDataset, [TRDP_DATASET_T](#) **ppDataset)
Function to free the memory for the DataSet configuration.
- EXT_DECL [TRDP_ERR_T tau_readXmlServiceConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, UINT32 *pNumServiceDefs, [TRDP_SERVICE_DEF_T](#) **ppServiceDefs)
Function to read the TRDP device service definitions out of the XML configuration file.

5.16.1 Detailed Description

Functions for XML file parsing.

SOX parsing of XML configuration file

Note

Project: TCNOpen TRDP prototype stack

Author

B. Loehr, NewTec GmbH, Tomas Svoboda, UniControls a.s.

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright NewTec GmbH, 2016-2019. All rights reserved.

5.16.2 Macro Definition Documentation

5.16.2.1 TRDP_SDT_DEFAULT_CMTHR

```
#define TRDP_SDT_DEFAULT_CMTHR 10u
```

Default SDT chan.

monitoring threshold

5.16.2.2 TRDP_SDT_DEFAULT_LMIMAX

```
#define TRDP_SDT_DEFAULT_LMIMAX (11u*TRDP_SDT_DEFAULT_NRXSAFE)
```

Default SDT chan.

latency monitoring cycles

5.16.3 Function Documentation

5.16.3.1 tau_freeTelegrams()

```
EXT_DECL void tau_freeTelegrams (
    UINT32 numExchgPar,
    TRDP_EXCHG_PAR_T * pExchgPar )
```

Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.

Parameters

in	<i>numExchgPar</i>	Number of telegram configurations in the array
in	<i>pExchgPar</i>	Pointer to array of telegram configurations

5.16.3.2 tau_freeXmlDatasetConfig()

```
EXT_DECL void tau_freeXmlDatasetConfig (
    UINT32 numComId,
    TRDP_COMID_DSID_MAP_T * pComIdDsIdMap,
    UINT32 numDataset,
    TRDP_DATASET_T ** ppDataset )
```

Function to free the memory for the DataSet configuration.

Free the memory for the DataSet configuration which was allocated when parsing the XML configuration file.

Parameters

in	<i>numComId</i>	The number of entries in the ComId DataSetId mapping list
in	<i>pComIdDsIdMap</i>	Pointer to an array of structures of type TRDP_COMID_DSID_MAP_T
in	<i>numDataset</i>	The number of datasets found in the configuration
in	<i>ppDataset</i>	Pointer to an array of pointers to a structures of type TRDP_DATASET_T

Return values

<i>none</i>	
-------------	--

5.16.3.3 tau_freeXmlDoc()

```
EXT_DECL void tau_freeXmlDoc (
    TRDP\_XML\_DOC\_HANDLE\_T * pDocHnd )
```

Free all the memory allocated by tau_prepareXmlDoc.

Parameters

in	<i>pDocHnd</i>	Handle of the parsed XML file
----	----------------	-------------------------------

5.16.3.4 tau_prepareXmlDoc()

```
EXT_DECL TRDP\_ERR\_T tau_prepareXmlDoc (
    const CHAR8 * pFileName,
    TRDP\_XML\_DOC\_HANDLE\_T * pDocHnd )
```

Open XML file, prepare XPath context.

Load XML file into DOM tree, prepare XPath context.

Parameters

in	<i>pFileName</i>	Path and filename of the xml configuration file
out	<i>pDocHnd</i>	Handle of the parsed XML file

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	File does not exist

5.16.3.5 tau_prepareXmlMem()

```
EXT_DECL TRDP_ERR_T tau_prepareXmlMem (
    char * pBuffer,
    size_t bufSize,
    TRDP_XML_DOC_HANDLE_T * pDocHnd )
```

Open XML stream, prepare XPath context.

Parameters

in	<i>pBuffer</i>	Pointer to the xml configuration stream buffer
in	<i>bufSize</i>	Size of the xml configuration stream buffer
out	<i>pDocHnd</i>	Pointer to the handle of the parsed XML file

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	File does not exist

5.16.3.6 tau_readXmlDatasetConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlDatasetConfig (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    UINT32 * pNumComId,
    TRDP_COMID_DSID_MAP_T ** ppComIdDsIdMap,
    UINT32 * pNumDataset,
    apTRDP_DATASET_T * apDataset )
```

Function to read the DataSet configuration out of the XML configuration file.

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
out	<i>pNumComId</i>	Pointer to the number of entries in the ComId DatasetId mapping list
out	<i>ppComIdDsIdMap</i>	Pointer to an array of a structures of type TRDP_COMID_DSID_MAP_T
out	<i>pNumDataset</i>	Pointer to the number of datasets found in the configuration
out	<i>apDataset</i>	Pointer to an array of pointers to a structure of type TRDP_DATASET_T

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.16.3.7 tau_readXmlDeviceConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlDeviceConfig (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    TRDP_MEM_CONFIG_T * pMemConfig,
    TRDP_DBG_CONFIG_T * pDbgConfig,
    UINT32 * pNumComPar,
    TRDP_COM_PAR_T ** ppComPar,
    UINT32 * pNumIfConfig,
    TRDP_IF_CONFIG_T ** ppIfConfig )
```

Function to read the TRDP device configuration parameters out of the XML configuration file.

The user must release the memory for ppComPar and pplfConfig (using vos_memFree)

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
out	<i>pMemConfig</i>	Memory configuration
out	<i>pDbgConfig</i>	Debug printout configuration for application use
out	<i>pNumComPar</i>	Number of configured com parameters
out	<i>ppComPar</i>	Pointer to array of com parameters
out	<i>pNumIfConfig</i>	Number of configured interfaces
out	<i>ppIfConfig</i>	Pointer to an array of interface parameter sets

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.16.3.8 tau_readXmlInterfaceConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlInterfaceConfig (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    const CHAR8 * pIfName,
    TRDP_PROCESS_CONFIG_T * pProcessConfig,
    TRDP_PD_CONFIG_T * pPdConfig,
    TRDP_MD_CONFIG_T * pMdConfig,
    UINT32 * pNumExchgPar,
    TRDP_EXCHG_PAR_T ** ppExchgPar )
```

Read the interface relevant telegram parameters (except data set configuration) out of the configuration file .

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
in	<i>pIfName</i>	Interface name
out	<i>pProcessConfig</i>	TRDP process (session) configuration for the interface
out	<i>pPdConfig</i>	PD default configuration for the interface
out	<i>pMdConfig</i>	MD default configuration for the interface
out	<i>pNumExchgPar</i>	Number of configured telegrams
out	<i>ppExchgPar</i>	Pointer to array of telegram configurations

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.16.3.9 tau_readXmlMappedDeviceConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlMappedDeviceConfig (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    const CHAR8 * pHostname,
    UINT32 * pNumIfConfig,
    TRDP_IF_CONFIG_T ** ppIfConfig )
```

Function to read the TRDP mapped device configuration parameters for a particular host out of the XML configuration file.

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
in	<i>pHostname</i>	Host name for which interface config is to be read
out	<i>pNumIfConfig</i>	Number of configured interfaces for this host
out	<i>ppIfConfig</i>	Pointer to an array of interface parameter sets

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.16.3.10 tau_readXmlMappedDevices()

```
EXT_DECL TRDP_ERR_T tau_readXmlMappedDevices (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    UINT32 * pNumProcConfig,
    TRDP_PROCESS_CONFIG_T ** ppProcessConfig )
```

Function to read the TRDP mapped devices out of the XML configuration file.

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
out	<i>pNumProcConfig</i>	Number of configured mapped devices
out	<i>ppProcessConfig</i>	Pointer to an array of mapped devices configuration

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.16.3.11 tau_readXmlMappedInterfaceConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlMappedInterfaceConfig (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    const CHAR8 * pHostname,
    const CHAR8 * pIfName,
    UINT32 * pNumExchgPar,
    TRDP_EXCHG_PAR_T ** ppExchgPar )
```

Read the interface relevant mapped telegram parameters for a particular host and it's interface out of the configuration file .

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
in	<i>pHostname</i>	Host name
in	<i>pIfName</i>	Interface name
out	<i>pNumExchgPar</i>	Number of configured telegrams
out	<i>ppExchgPar</i>	Pointer to array of telegram configurations

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.16.3.12 tau_readXmlServiceConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlServiceConfig (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    UINT32 * pNumServiceDefs,
    TRDP_SERVICE_DEF_T ** ppServiceDefs )
```

Function to read the TRDP device service definitions out of the XML configuration file.

The user must release the memory for pServiceDefs (using vos_memFree)

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
out	<i>pNumServiceDefs</i>	Pointer to number of defined Services
out	<i>ppServiceDefs</i>	Pointer to pointer of the defined Services

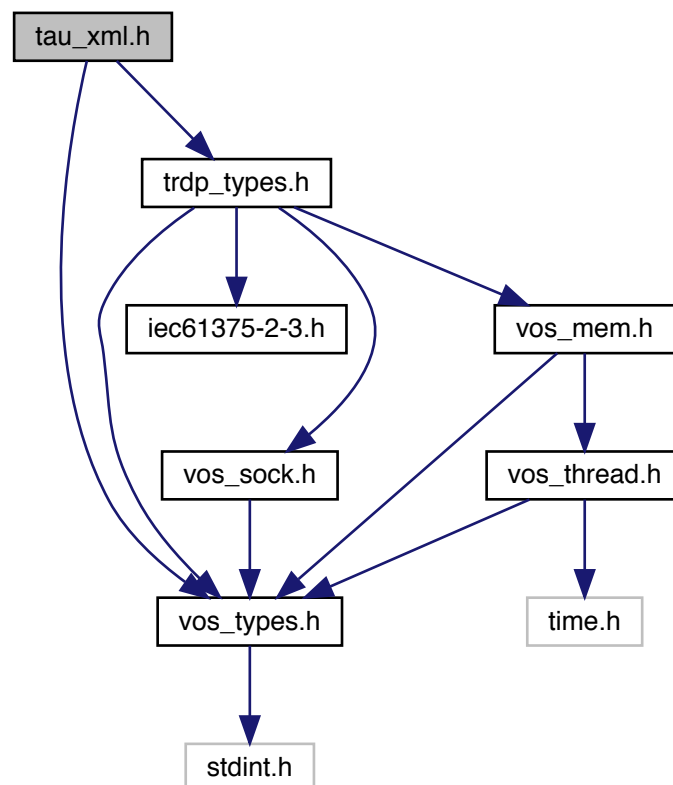
Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

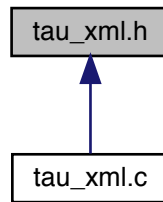
5.17 tau_xml.h File Reference

TRDP utility interface definitions.

```
#include "vos_types.h"
#include "trdp_types.h"
Include dependency graph for tau_xml.h:
```



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



Data Structures

- struct [TRDP_SDT_PAR_T](#)
Types to read out the XML configuration.
- struct [TRDP_DBG_CONFIG_T](#)
Control for debug output device/file on application level.
- struct [TRDP_XML_DOC_HANDLE_T](#)
Parsed XML document handle.

Macros

- #define [TRDP_DBG_DEFAULT](#) 0
Control for debug output format on application level.
- #define [TRDP_DBG_OFF](#) 0x01
Printout off.
- #define [TRDP_DBG_ERR](#) 0x02
Printout error.
- #define [TRDP_DBG_WARN](#) 0x04
Printout warning and error.
- #define [TRDP_DBG_INFO](#) 0x08
Printout info, warning and error.
- #define [TRDP_DBG_DBG](#) 0x10
Printout debug, info, warning and error.
- #define [TRDP_DBG_TIME](#) 0x20
Printout timestamp.
- #define [TRDP_DBG_LOC](#) 0x40
Printout file name and line.
- #define [TRDP_DBG_CAT](#) 0x80
Printout category (DBG, INFO, WARN, ERR)

Enumerations

- enum [TRDP_EXCHG_OPTION_T](#) {
[TRDP_EXCHG_UNSET](#) = 0,
[TRDP_EXCHG_SOURCE](#) = 1,
[TRDP_EXCHG_SINK](#) = 2,
[TRDP_EXCHG_SOURCESINK](#) = 3 }
Type attribute for telegrams.

Functions

- EXT_DECL [TRDP_ERR_T tau_prepareXmlDoc](#) (const CHAR8 *pFileName, [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd)
Load XML file into DOM tree, prepare XPath context.
- EXT_DECL [TRDP_ERR_T tau_prepareXmlMem](#) (char *pBuffer, size_t bufSize, [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd)
Open XML stream, prepare XPath context.
- EXT_DECL void [tau_freeXmlDoc](#) ([TRDP_XML_DOC_HANDLE_T](#) *pDocHnd)
Free all the memory allocated by tau_prepareXmlDoc.
- EXT_DECL [TRDP_ERR_T tau_readXmlDeviceConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, [TRDP_MEM_CONFIG_T](#) *pMemConfig, [TRDP_DBG_CONFIG_T](#) *pDbgConfig, UINT32 *pNumComPar, [TRDP_COM_PAR_T](#) **ppComPar, UINT32 *pNumIfConfig, [TRDP_IF_CONFIG_T](#) **ppIfConfig)
Function to read the TRDP device configuration parameters out of the XML configuration file.
- EXT_DECL [TRDP_ERR_T tau_readXmlInterfaceConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, const CHAR8 *plfName, [TRDP_PROCESS_CONFIG_T](#) *pProcessConfig, [TRDP_PD_CONFIG_T](#) *pPdConfig, [TRDP_MD_CONFIG_T](#) *pMdConfig, UINT32 *pNumExchgPar, [TRDP_EXCHG_PAR_T](#) **ppExchgPar)
Read the interface relevant telegram parameters (except data set configuration) out of the configuration file .
- EXT_DECL [TRDP_ERR_T tau_readXmlDatasetConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, UINT32 *pNumComId, [TRDP_COMID_DSID_MAP_T](#) **ppComIdDsidMap, UINT32 *pNumDataset, [TRDP_DATASET_T](#) *pDataset)
Function to read the DataSet configuration out of the XML configuration file.
- EXT_DECL void [tau_freeXmlDatasetConfig](#) (UINT32 numComId, [TRDP_COMID_DSID_MAP_T](#) *pComIdDsidMap, UINT32 numDataset, [TRDP_DATASET_T](#) **ppDataset)
Function to free the memory for the DataSet configuration.
- EXT_DECL void [tau_freeTelegrams](#) (UINT32 numExchgPar, [TRDP_EXCHG_PAR_T](#) *pExchgPar)
Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.
- EXT_DECL [TRDP_ERR_T tau_readXmlServiceConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, UINT32 *pNumServiceDefs, [TRDP_SERVICE_DEF_T](#) **ppServiceDefs)
Function to read the TRDP device service definitions out of the XML configuration file.
- EXT_DECL [TRDP_ERR_T tau_readXmlMappedDevices](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, UINT32 *pNumProcConfig, [TRDP_PROCESS_CONFIG_T](#) **ppProcessConfig)
Function to read the TRDP mapped devices out of the XML configuration file.
- EXT_DECL [TRDP_ERR_T tau_readXmlMappedDeviceConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, const CHAR8 *pHostname, UINT32 *pNumIfConfig, [TRDP_IF_CONFIG_T](#) **ppIfConfig)
Function to read the TRDP mapped device configuration parameters for a particular host out of the XML configuration file.
- EXT_DECL [TRDP_ERR_T tau_readXmlMappedInterfaceConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, const CHAR8 *pHostname, const CHAR8 *plfName, UINT32 *pNumExchgPar, [TRDP_EXCHG_PAR_T](#) **ppExchgPar)
Read the interface relevant mapped telegram parameters for a particular host and it's interface out of the configuration file .

5.17.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- read xml configuration interpreter

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.17.2 Macro Definition Documentation

5.17.2.1 TRDP_DBG_DEFAULT

```
#define TRDP_DBG_DEFAULT 0
```

Control for debug output format on application level.

Printout default

5.17.3 Enumeration Type Documentation

5.17.3.1 TRDP_EXCHG_OPTION_T

```
enum TRDP_EXCHG_OPTION_T
```

Type attribute for telegrams.

Enumerator

TRDP_EXCHG_UNSET	default, direction is not defined
TRDP_EXCHG_SOURCE	telegram shall be published
TRDP_EXCHG_SINK	telegram shall be subscribed
TRDP_EXCHG_SOURCESINK	telegram shall be published and subscribed

5.17.4 Function Documentation

5.17.4.1 tau_freeTelegrams()

```
EXT_DECL void tau_freeTelegrams (
    UINT32 numExchgPar,
    TRDP_EXCHG_PAR_T * pExchgPar )
```

Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.

Parameters

in	<i>numExchgPar</i>	Number of telegram configurations in the array
in	<i>pExchgPar</i>	Pointer to array of telegram configurations

5.17.4.2 tau_freeXmlDatasetConfig()

```
EXT_DECL void tau_freeXmlDatasetConfig (
    UINT32 numComId,
    TRDP_COMID_DSID_MAP_T * pComIdDsIdMap,
    UINT32 numDataset,
    TRDP_DATASET_T ** ppDataset )
```

Function to free the memory for the DataSet configuration.

Free the memory for the DataSet configuration which was allocated when parsing the XML configuration file.

Parameters

in	<i>numComId</i>	The number of entries in the ComId DatasetId mapping list
in	<i>pComIdDsIdMap</i>	Pointer to an array of structures of type TRDP_COMID_DSID_MAP_T
in	<i>numDataset</i>	The number of datasets found in the configuration
in	<i>ppDataset</i>	Pointer to an array of pointers to a structures of type TRDP_DATASET_T

Return values

<i>none</i>	
-------------	--

5.17.4.3 tau_freeXmlDoc()

```
EXT_DECL void tau_freeXmlDoc (
    TRDP_XML_DOC_HANDLE_T * pDocHnd )
```

Free all the memory allocated by tau_prepareXmlDoc.

Parameters

in	<i>pDocHnd</i>	Handle of the parsed XML file
----	----------------	-------------------------------

5.17.4.4 tau_prepareXmlDoc()

```
EXT_DECL TRDP_ERR_T tau_prepareXmlDoc (
    const CHAR8 * pFileName,
    TRDP_XML_DOC_HANDLE_T * pDocHnd )
```

Load XML file into DOM tree, prepare XPath context.

Parameters

in	<i>pFileName</i>	Path and filename of the xml configuration file
out	<i>pDocHnd</i>	Handle of the parsed XML file

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	File does not exist

Load XML file into DOM tree, prepare XPath context.

Parameters

in	<i>pFileName</i>	Path and filename of the xml configuration file
out	<i>pDocHnd</i>	Handle of the parsed XML file

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	File does not exist

5.17.4.5 tau_prepareXmlMem()

```
EXT_DECL TRDP_ERR_T tau_prepareXmlMem (
    char * pBuffer,
    size_t bufSize,
    TRDP_XML_DOC_HANDLE_T * pDocHnd )
```

Open XML stream, prepare XPath context.

Parameters

in	<i>pBuffer</i>	Pointer to the xml configuration stream buffer
in	<i>bufSize</i>	Size of the xml configuration stream buffer
out	<i>pDocHnd</i>	Pointer to the handle of the parsed XML file

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	File does not exist

5.17.4.6 tau_readXmlDatasetConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlDatasetConfig (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    UINT32 * pNumComId,
    TRDP_COMID_DSID_MAP_T ** ppComIdDsIdMap,
    UINT32 * pNumDataset,
    papTRDP_DATASET_T papDataset )
```

Function to read the DataSet configuration out of the XML configuration file.

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
out	<i>pNumComId</i>	Pointer to the number of entries in the ComId DatasetId mapping list
out	<i>ppComIdDsIdMap</i>	Pointer to an array of a structures of type TRDP_COMID_DSID_MAP_T
out	<i>pNumDataset</i>	Pointer to the number of datasets found in the configuration
out	<i>papDataset</i>	Pointer to an array of pointers to a structures of type TRDP_DATASET_T

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.17.4.7 tau_readXmlDeviceConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlDeviceConfig (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    TRDP_MEM_CONFIG_T * pMemConfig,
    TRDP_DBG_CONFIG_T * pDbgConfig,
    UINT32 * pNumComPar,
    TRDP_COM_PAR_T ** ppComPar,
    UINT32 * pNumIfConfig,
    TRDP_IF_CONFIG_T ** ppIfConfig )
```

Function to read the TRDP device configuration parameters out of the XML configuration file.

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
out	<i>pMemConfig</i>	Memory configuration

Parameters

out	<i>pDbgConfig</i>	Debug printout configuration for application use
out	<i>pNumComPar</i>	Number of configured com parameters
out	<i>ppComPar</i>	Pointer to array of com parameters
out	<i>pNumIfConfig</i>	Number of configured interfaces
out	<i>ppIfConfig</i>	Pointer to an array of interface parameter sets

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

The user must release the memory for ppComPar and pplfConfig (using vos_memFree)

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
out	<i>pMemConfig</i>	Memory configuration
out	<i>pDbgConfig</i>	Debug printout configuration for application use
out	<i>pNumComPar</i>	Number of configured com parameters
out	<i>ppComPar</i>	Pointer to array of com parameters
out	<i>pNumIfConfig</i>	Number of configured interfaces
out	<i>ppIfConfig</i>	Pointer to an array of interface parameter sets

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.17.4.8 tau_readXmlInterfaceConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlInterfaceConfig (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    const CHAR8 * pIfName,
    TRDP_PROCESS_CONFIG_T * pProcessConfig,
    TRDP_PD_CONFIG_T * pPdConfig,
    TRDP_MD_CONFIG_T * pMdConfig,
    UINT32 * pNumExchgPar,
    TRDP_EXCHG_PAR_T ** ppExchgPar )
```

Read the interface relevant telegram parameters (except data set configuration) out of the configuration file .

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
----	----------------	--

Parameters

in	<i>plfName</i>	Interface name
out	<i>pProcessConfig</i>	TRDP process (session) configuration for the interface
out	<i>pPdConfig</i>	PD default configuration for the interface
out	<i>pMdConfig</i>	MD default configuration for the interface
out	<i>pNumExchgPar</i>	Number of configured telegrams
out	<i>ppExchgPar</i>	Pointer to array of telegram configurations

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.17.4.9 tau_readXmlMappedDeviceConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlMappedDeviceConfig (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    const CHAR8 * pHostname,
    UINT32 * pNumIfConfig,
    TRDP_IF_CONFIG_T ** ppIfConfig )
```

Function to read the TRDP mapped device configuration parameters for a particular host out of the XML configuration file.

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
in	<i>pHostname</i>	Host name for which interface config is to be read
out	<i>pNumIfConfig</i>	Number of configured interfaces for this host
out	<i>ppIfConfig</i>	Pointer to an array of interface parameter sets

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.17.4.10 tau_readXmlMappedDevices()

```
EXT_DECL TRDP_ERR_T tau_readXmlMappedDevices (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    UINT32 * pNumProcConfig,
    TRDP_PROCESS_CONFIG_T ** ppProcessConfig )
```

Function to read the TRDP mapped devices out of the XML configuration file.

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
out	<i>pNumProcConfig</i>	Number of configured mapped devices
out	<i>ppProcessConfig</i>	Pointer to an array of mapped devices configuration

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.17.4.11 tau_readXmlMappedInterfaceConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlMappedInterfaceConfig (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    const CHAR8 * pHostname,
    const CHAR8 * pIfName,
    UINT32 * pNumExchgPar,
    TRDP_EXCHG_PAR_T ** ppExchgPar )
```

Read the interface relevant mapped telegram parameters for a particular host and it's interface out of the configuration file .

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
in	<i>pHostname</i>	Host name
in	<i>pIfName</i>	Interface name
out	<i>pNumExchgPar</i>	Number of configured telegrams
out	<i>ppExchgPar</i>	Pointer to array of telegram configurations

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.17.4.12 tau_readXmlServiceConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlServiceConfig (
    const TRDP_XML_DOC_HANDLE_T * pDocHnd,
    UINT32 * pNumServiceDefs,
    TRDP_SERVICE_DEF_T ** ppServiceDefs )
```

Function to read the TRDP device service definitions out of the XML configuration file.

The user must release the memory for pServiceDefs (using vos_memFree)

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
out	<i>pNumServiceDefs</i>	Number of defined Services
out	<i>ppServiceDefs</i>	Pointer to pointer of the defined Services

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

The user must release the memory for pServiceDefs (using vos_memFree)

Parameters

in	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
out	<i>pNumServiceDefs</i>	Pointer to number of defined Services
out	<i>ppServiceDefs</i>	Pointer to pointer of the defined Services

Return values

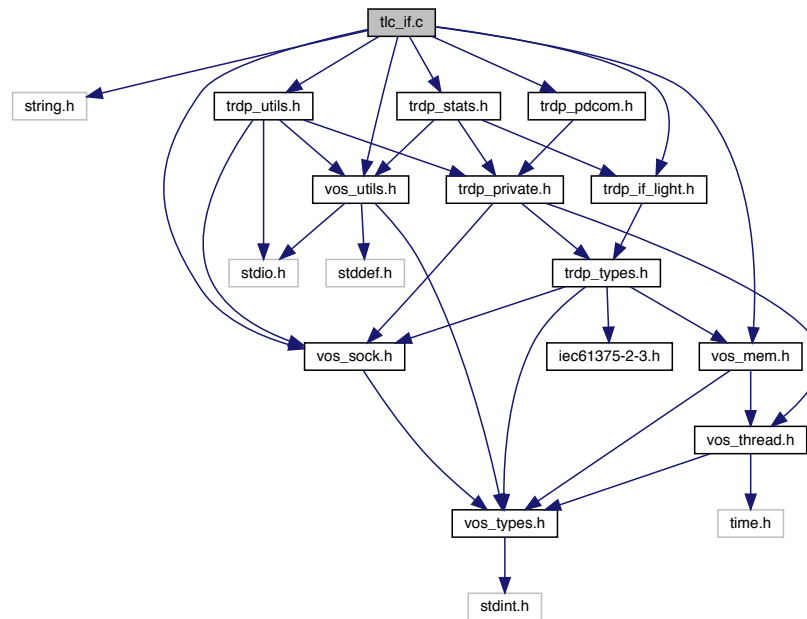
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.18 tlc_if.c File Reference

Functions for ECN communication.

```
#include <string.h>
#include "trdp_if_light.h"
#include "trdp_utils.h"
#include "trdp_pdcom.h"
#include "trdp_stats.h"
#include "vos_sock.h"
#include "vos_mem.h"
#include "vos_utils.h"
```


Include dependency graph for tlc_if.c:



Functions

- `BOOL8 trdp_isValidSession (TRDP_APP_SESSION_T pSessionHandle)`
Check if the session handle is valid.
- `TRDP_APP_SESSION_T * trdp_sessionQueue (void)`
Get the session queue head pointer.
- `TRDP_ERR_T trdp_getAccess (TRDP_APP_SESSION_T appHandle, int force)`
Get mutual access to the session Take all mutexes of that session.
- `void trdp_releaseAccess (TRDP_APP_SESSION_T appHandle)`
Release access to the session.
- `EXT_DECL TRDP_IP_ADDR_T tlc_getOwnIpAddress (TRDP_APP_SESSION_T appHandle)`
Get the interface address.
- `EXT_DECL TRDP_ERR_T tlc_init (const TRDP_PRINT_DBG_T pPrintDebugString, void *pRefCon, const TRDP_MEM_CONFIG_T *pMemConfig)`
Initialize the TRDP stack.
- `EXT_DECL TRDP_ERR_T tlc_openSession (TRDP_APP_SESSION_T *pAppHandle, TRDP_IP_ADDR_T ownIpAddr, TRDP_IP_ADDR_T leaderIpAddr, const TRDP_MARSHALL_CONFIG_T *pMarshall, const TRDP_PD_CONFIG_T *pPdDefault, const TRDP_MD_CONFIG_T *pMdDefault, const TRDP_PROCESS_CONFIG_T *pProcessConfig)`
Open a session with the TRDP stack.
- `EXT_DECL TRDP_ERR_T tlc_configSession (TRDP_APP_SESSION_T appHandle, const TRDP_MARSHALL_CONFIG_T *pMarshall, const TRDP_PD_CONFIG_T *pPdDefault, const TRDP_MD_CONFIG_T *pMdDefault, const TRDP_PROCESS_CONFIG_T *pProcessConfig)`
(Re-)configure a session.
- `EXT_DECL TRDP_ERR_T tlc_updateSession (TRDP_APP_SESSION_T appHandle)`
Update a session.

- EXT_DECL [TRDP_ERR_T tlc_presetIndexSession](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_IDX_T↔
ABLE_T](#) *pIndexTableSizes)
Preset the index table sizes of a session.
- EXT_DECL [TRDP_ERR_T tlc_closeSession](#) ([TRDP_APP_SESSION_T](#) appHandle)
Close a session.
- EXT_DECL [TRDP_ERR_T tlc_terminate](#) (void)
Un-Initialize.
- EXT_DECL [TRDP_ERR_T tlc_reinitSession](#) ([TRDP_APP_SESSION_T](#) appHandle)
Re-Initialize.
- EXT_DECL [TRDP_ERR_T tlc_getInterval](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_TIME_T](#) *pInterval,
[TRDP_FDS_T](#) *pFileDesc, INT32 *pNoDesc)
Get the lowest time interval for PDs.
- EXT_DECL [TRDP_ERR_T tlc_process](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_FDS_T](#) *pRfds, INT32
*pCount)
Work loop of the TRDP handler.
- const char * [tlc_getVersionString](#) (void)
Return a human readable version representation.
- EXT_DECL const [TRDP_VERSION_T](#) * [tlc_getVersion](#) (void)
Return version.
- EXT_DECL [TRDP_ERR_T tlc_setETBTopoCount](#) ([TRDP_APP_SESSION_T](#) appHandle, UINT32 etbTopo↔
Cnt)
Set new topocount for trainwide communication.
- EXT_DECL [TRDP_ERR_T tlc_setOpTrainTopoCount](#) ([TRDP_APP_SESSION_T](#) appHandle, UINT32 op↔
TrnTopoCnt)
Set new operational train topocount for direction/orientation sensitive communication.
- EXT_DECL UINT32 [tlc_getETBTopoCount](#) ([TRDP_APP_SESSION_T](#) appHandle)
Set new topocount for trainwide communication.
- EXT_DECL UINT32 [tlc_getOpTrainTopoCount](#) ([TRDP_APP_SESSION_T](#) appHandle)
Set new operational train topocount for direction/orientation sensitive communication.

5.18.1 Detailed Description

Functions for ECN communication.

API implementation of TRDP Light

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013-2019. All rights reserved.

5.18.2 Function Documentation

5.18.2.1 tlc_closeSession()

```
EXT_DECL TRDP_ERR_T tlc_closeSession (
    TRDP_APP_SESSION_T appHandle )
```

Close a session.

Clean up and release all resources of that session

Parameters

in	<i>appHandle</i>	The handle returned by tlc_openSession
----	------------------	--

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	handle NULL

5.18.2.2 tlc_configSession()

```
EXT_DECL TRDP_ERR_T tlc_configSession (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_MARSHALL_CONFIG_T * pMarshall,
    const TRDP_PD_CONFIG_T * pPdDefault,
    const TRDP_MD_CONFIG_T * pMdDefault,
    const TRDP_PROCESS_CONFIG_T * pProcessConfig )
```

(Re-)configure a session.

tlc_configSession is called by openSession, but may also be called later on to change the defaults. Only the supplied settings (pointer != NULL) will be evaluated.

Parameters

in	<i>appHandle</i>	A handle for further calls to the trdp stack
in	<i>pMarshall</i>	Pointer to marshalling configuration
in	<i>pPdDefault</i>	Pointer to default PD configuration
in	<i>pMdDefault</i>	Pointer to default MD configuration
in	<i>pProcessConfig</i>	Pointer to process configuration only option parameter is used here to define session behavior all other parameters are only used to feed statistics

Return values

<i>TRDP_NO_ERR</i>	no error
--------------------	----------

Return values

<i>TRDP_INIT_ERR</i>	not yet initied
<i>TRDP_PARAM_ERR</i>	parameter error

5.18.2.3 `tlc_getETBTopoCount()`

```
EXT_DECL UINT32 tlc_getETBTopoCount (
    TRDP_APP_SESSION_T appHandle )
```

Set new topocount for trainwide communication.

This value is used for validating outgoing and incoming packets only!

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
----	------------------	---

Return values

<i>etbTopoCnt</i>	
-------------------	--

5.18.2.4 `tlc_getInterval()`

```
EXT_DECL TRDP_ERR_T tlc_getInterval (
    TRDP_APP_SESSION_T appHandle,
    TRDP_TIME_T * pInterval,
    TRDP_FDS_T * pFileDesc,
    INT32 * pNoDesc )
```

Get the lowest time interval for PDs.

Return the maximum time interval suitable for 'select()' so that we can send due PD packets in time. If the PD send queue is empty, return zero time

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
out	<i>pInterval</i>	pointer to needed interval
in, out	<i>pFileDesc</i>	pointer to file descriptor set
out	<i>pNoDesc</i>	pointer to put no of highest used descriptors (for select())

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.18.2.5 tlc_getOpTrainTopoCount()

```
EXT_DECL UINT32 tlc_getOpTrainTopoCount (
    TRDP_APP_SESSION_T appHandle )
```

Set new operational train topocount for direction/orientation sensitive communication.

This value is used for validating outgoing and incoming packets only!

Parameters

in	<i>appHandle</i>	The handle returned by tlc_openSession
----	------------------	--

Return values

<i>opTrnTopoCnt</i>	New operational topocount value
---------------------	---------------------------------

5.18.2.6 tlc_getOwnIpAddress()

```
EXT_DECL TRDP_IP_ADDR_T tlc_getOwnIpAddress (
    TRDP_APP_SESSION_T appHandle )
```

Get the interface address.

Parameters

out	<i>appHandle</i>	A handle for further calls to the trdp stack
-----	------------------	--

Return values

<i>real</i> ↔ <i>IP</i>	
----------------------------	--

5.18.2.7 tlc_getVersion()

```
EXT_DECL const TRDP_VERSION_T* tlc_getVersion (
    void )
```

Return version.

Return pointer to version structure

Return values

<i>TRDP_VERSION</i>	↔	
<i>_T</i>		

5.18.2.8 tlc_getVersionString()

```
const char* tlc_getVersionString (
    void )
```

Return a human readable version representation.

Return string in the form 'v.r.u.b'

Return values

<i>const</i>	string
--------------	--------

5.18.2.9 tlc_init()

```
EXT_DECL TRDP_ERR_T tlc_init (
    const TRDP_PRINT_DBG_T pPrintDebugString,
    void * pRefCon,
    const TRDP_MEM_CONFIG_T * pMemConfig )
```

Initialize the TRDP stack.

Support for message data can only be excluded during compile time!

tlc_init initializes the memory subsystem and takes a function pointer to an output function for logging.

Parameters

in	<i>pPrintDebugString</i>	Pointer to debug print function
in	<i>pRefCon</i>	user context
in	<i>pMemConfig</i>	Pointer to memory configuration

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	memory allocation failed
<i>TRDP_PARAM_ERR</i>	initialization error

5.18.2.10 tlc_openSession()

```
EXT_DECL TRDP_ERR_T tlc_openSession (
    TRDP_APP_SESSION_T * pAppHandle,
    TRDP_IP_ADDR_T ownIpAddr,
    TRDP_IP_ADDR_T leaderIpAddr,
    const TRDP_MARSHALL_CONFIG_T * pMarshall,
    const TRDP_PD_CONFIG_T * pPdDefault,
    const TRDP_MD_CONFIG_T * pMdDefault,
    const TRDP_PROCESS_CONFIG_T * pProcessConfig )
```

Open a session with the TRDP stack.

tlc_openSession returns in pAppHandle a unique handle to be used in further calls to the stack.

Parameters

out	<i>pAppHandle</i>	A handle for further calls to the trdp stack
in	<i>ownIpAddr</i>	Own IP address, can be different for each process in multihoming systems, if zero, the default interface / IP will be used.
in	<i>leaderIpAddr</i>	IP address of redundancy leader
in	<i>pMarshall</i>	Pointer to marshalling configuration
in	<i>pPdDefault</i>	Pointer to default PD configuration
in	<i>pMdDefault</i>	Pointer to default MD configuration
in	<i>pProcessConfig</i>	Pointer to process configuration only option parameter is used here to define session behavior all other parameters are only used to feed statistics

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	not yet initied
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP SOCK_ERR</i>	socket error

5.18.2.11 tlc_presetIndexSession()

```
EXT_DECL TRDP_ERR_T tlc_presetIndexSession (
    TRDP_APP_SESSION_T appHandle,
    TRDP_IDX_TABLE_T * pIndexTableSizes )
```

Preset the index table sizes of a session.

tlc_presetIndexSession allows to preallocate the table sizes in HIGH_PERF_INDEXED mode. If no table sizes are provided, the default sizes are used. In normal mode, this is a no-op. This function should be called during initialisation stage, e.g. right after a session has been opened.

Parameters

in	<i>appHandle</i>	The handle returned by tlc_openSession
in	<i>pIndexTableSizes</i>	Pointer to a table of sizes to reserve the memory

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	not yet initied
<i>TRDP_PARAM_ERR</i>	parameter error

5.18.2.12 `tlc_process()`

```
EXT_DECL TRDP_ERR_T tlc_process (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FDS_T * pRfds,
    INT32 * pCount )
```

Work loop of the TRDP handler.

Search the queue for pending PDs and MDs to be sent Search the receive queue for pending PDs and MDs (time out)

Note: If using `tlc_process()`, do not use `tlp_process*()` and `tlm_process()` calls at the same time! Single thread usage -> use `tlc_getInterval()`, `vos_select()`, `tlc_process()` Multiple threads -> thread 1: use `tlp_getInterval()`, `vos_select()`, `tlp_processReceive()` -> thread 2: cyclically call `tlp_processSend()` -> thread 3: use `tlm_getInterval()`, `vos_select()`, `tlm_process()` for message data

Also see User Manual.

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.18.2.13 `tlc_reinitSession()`

```
EXT_DECL TRDP_ERR_T tlc_reinitSession (
    TRDP_APP_SESSION_T appHandle )
```

Re-Initialize.

Should be called by the application when a link-down/link-up event has occurred during normal operation. We need to re-join the multicast groups...

Parameters

in	<i>appHandle</i>	The handle returned by tlc_openSession
----	------------------	--

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	handle NULL

5.18.2.14 tlc_setETBTopoCount()

```
EXT_DECL TRDP_ERR_T tlc_setETBTopoCount (
    TRDP_APP_SESSION_T appHandle,
    UINT32 etbTopoCnt )
```

Set new topocount for trainwide communication.

This value is used for validating outgoing and incoming packets only!

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>etbTopoCnt</i>	New etbTopoCnt value

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.18.2.15 tlc_setOpTrainTopoCount()

```
EXT_DECL TRDP_ERR_T tlc_setOpTrainTopoCount (
    TRDP_APP_SESSION_T appHandle,
    UINT32 opTrnTopoCnt )
```

Set new operational train topocount for direction/orientation sensitive communication.

This value is used for validating outgoing and incoming packets only!

Parameters

in	<i>appHandle</i>	The handle returned by tlc_openSession
in	<i>opTrnTopoCnt</i>	New operational topocount value

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.18.2.16 `tlc_terminate()`

```
EXT_DECL TRDP_ERR_T tlc_terminate (
    void )
```

Un-Initialize.

Clean up and close all sessions. Mainly used for debugging/test runs. No further calls to library allowed

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	TrafficStore nothing
<i>TRDP_MUTEX_ERR</i>	TrafficStore mutex err

5.18.2.17 `tlc_updateSession()`

```
EXT_DECL TRDP_ERR_T tlc_updateSession (
    TRDP_APP_SESSION_T appHandle )
```

Update a session.

`tlc_updateSession` signals the end of the set-up phase to the stack. It shall be called after the last publisher and subscriber was added and will create and compute the index tables to be used by the high-performance targets. This function is currently a no-op on standard targets.

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
----	------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	not yet initied
<i>TRDP_PARAM_ERR</i>	parameter error

5.18.2.18 trdp_getAccess()

```
TRDP_ERR_T trdp_getAccess (
    TRDP_APP_SESSION_T appHandle,
    int force )
```

Get mutual access to the session Take all mutexes of that session.

Parameters

in	<i>appHandle</i>	A handle for further calls to the trdp stack
in	<i>force</i>	If TRUE, access the session even if we cannot get the mutex.

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_INIT_ERR</i>	
<i>TRDP_MUTEX_ERR</i>	

5.18.2.19 trdp_isValidSession()

```
BOOL8 trdp_isValidSession (
    TRDP_APP_SESSION_T pSessionHandle )
```

Check if the session handle is valid.

Parameters

in	<i>pSessionHandle</i>	pointer to packet data (dataset)
----	-----------------------	----------------------------------

Return values

<i>TRUE</i>	is valid
<i>FALSE</i>	is invalid

5.18.2.20 trdp_releaseAccess()

```
void trdp_releaseAccess (
    TRDP_APP_SESSION_T appHandle )
```

Release access to the session.

Parameters

in	<i>appHandle</i>	A handle for further calls to the trdp stack
----	------------------	--

Return values

<i>real</i> ↔	
<i>IP</i>	

Here is the call graph for this function:



5.18.2.21 trdp_sessionQueue()

```
TRDP_APP_SESSION_T * trdp_sessionQueue (  
    void )
```

Get the session queue head pointer.

Return values

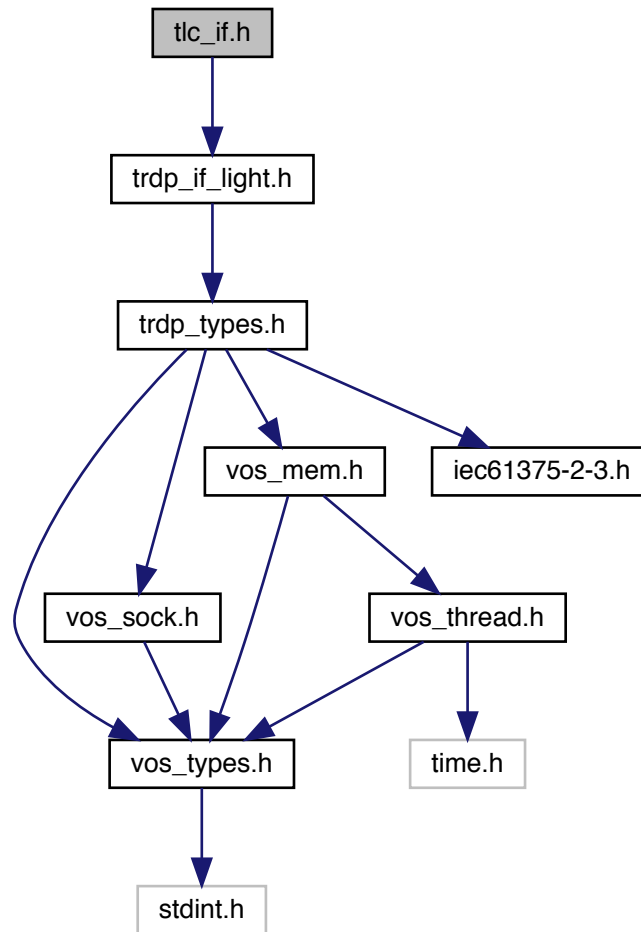
<i>&sSession</i>	
----------------------	--

5.19 tlc_if.h File Reference

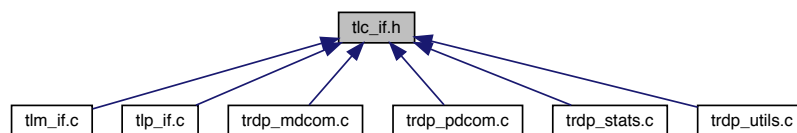
Typedefs for TRDP communication.

```
#include "trdp_if_light.h"
```

Include dependency graph for tlc_if.h:



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



Functions

- BOOL8 `trdp_isValidSession` (TRDP_APP_SESSION_T pSessionHandle)

Check if the session handle is valid.

- `TRDP_APP_SESSION_T * trdp_sessionQueue` (void)

Get the session queue head pointer.

5.19.1 Detailed Description

Typedefs for TRDP communication.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013-2019. All rights reserved.

5.19.2 Function Documentation

5.19.2.1 trdp_isValidSession()

```
BOOL8 trdp_isValidSession (
    TRDP_APP_SESSION_T pSessionHandle )
```

Check if the session handle is valid.

Parameters

in	<i>pSessionHandle</i>	pointer to packet data (dataset)
----	-----------------------	----------------------------------

Return values

<i>TRUE</i>	is valid
<i>FALSE</i>	is invalid

5.19.2.2 trdp_sessionQueue()

```
TRDP_APP_SESSION_T* trdp_sessionQueue (
    void )
```

Get the session queue head pointer.

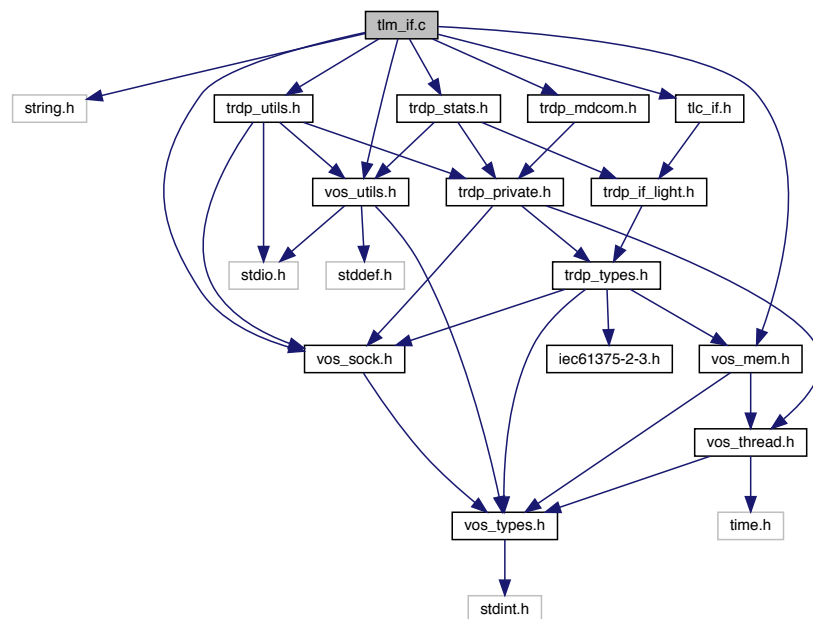
Return values

<code>&sSession</code>

5.20 tlm_if.c File Reference

Functions for Message Data Communication.

```
#include <string.h>
#include "tlc_if.h"
#include "trdp_utils.h"
#include "trdp_mdcom.h"
#include "trdp_stats.h"
#include "vos_sock.h"
#include "vos_mem.h"
#include "vos_utils.h"
Include dependency graph for tlm_if.c:
```



Functions

- EXT_DECL [TRDP_ERR_T tlm_getInterval](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_TIME_T](#) *pInterval, [TRDP_FDS_T](#) *pFileDesc, INT32 *pNoDesc)
Get the lowest time interval for MDs.
- EXT_DECL [TRDP_ERR_T tlm_process](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_FDS_T](#) *pRfds, INT32 *pCount)
Message Data Work loop of the TRDP handler.

- `TRDP_ERR_T tlm_notify` (`TRDP_APP_SESSION_T` appHandle, const void *pUserRef, `TRDP_MD_CALLBACK_T` pfCbFunction, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, `TRDP_IP_ADDR_T` srcIpAddr, `TRDP_IP_ADDR_T` destIpAddr, `TRDP_FLAGS_T` pktFlags, const `TRDP_SEND_PARAM_T` *pSendParam, const UINT8 *pData, UINT32 dataSize, const `TRDP_URI_USER_T` sourceURI, const `TRDP_URI_USER_T` destURI)
Initiate sending MD notification message.
- `TRDP_ERR_T tlm_request` (`TRDP_APP_SESSION_T` appHandle, const void *pUserRef, `TRDP_MD_CALLBACK_T` pfCbFunction, `TRDP_UUID_T` *pSessionId, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, `TRDP_IP_ADDR_T` srcIpAddr, `TRDP_IP_ADDR_T` destIpAddr, `TRDP_FLAGS_T` pktFlags, UINT32 numReplies, UINT32 replyTimeout, const `TRDP_SEND_PARAM_T` *pSendParam, const UINT8 *pData, UINT32 dataSize, const `TRDP_URI_USER_T` sourceURI, const `TRDP_URI_USER_T` destURI)
Initiate sending MD request message.
- `EXT_DECL TRDP_ERR_T tlm_addListener` (`TRDP_APP_SESSION_T` appHandle, `TRDP_LIS_T` *pListenHandle, const void *pUserRef, `TRDP_MD_CALLBACK_T` pfCbFunction, BOOL8 comIdListener, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, `TRDP_IP_ADDR_T` srcIpAddr1, `TRDP_IP_ADDR_T` srcIpAddr2, `TRDP_IP_ADDR_T` mcDestIpAddr, `TRDP_FLAGS_T` pktFlags, const `TRDP_URI_USER_T` srcURI, const `TRDP_URI_USER_T` destURI)
Subscribe to MD messages.
- `TRDP_ERR_T tlm_delListener` (`TRDP_APP_SESSION_T` appHandle, `TRDP_LIS_T` listenHandle)
Remove Listener.
- `EXT_DECL TRDP_ERR_T tlm_readdListener` (`TRDP_APP_SESSION_T` appHandle, `TRDP_LIS_T` listenHandle, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, `TRDP_IP_ADDR_T` srcIpAddr1, `TRDP_IP_ADDR_T` srcIpAddr2, `TRDP_IP_ADDR_T` mcDestIpAddr)
Resubscribe to MD messages.
- `TRDP_ERR_T tlm_reply` (`TRDP_APP_SESSION_T` appHandle, const `TRDP_UUID_T` *pSessionId, UINT32 comId, UINT16 userStatus, const `TRDP_SEND_PARAM_T` *pSendParam, const UINT8 *pData, UINT32 dataSize)
Send a MD reply message.
- `TRDP_ERR_T tlm_replyQuery` (`TRDP_APP_SESSION_T` appHandle, const `TRDP_UUID_T` *pSessionId, UINT32 comId, UINT16 userStatus, UINT32 confirmTimeout, const `TRDP_SEND_PARAM_T` *pSendParam, const UINT8 *pData, UINT32 dataSize)
Send a MD reply query message.
- `TRDP_ERR_T tlm_confirm` (`TRDP_APP_SESSION_T` appHandle, const `TRDP_UUID_T` *pSessionId, UINT16 userStatus, const `TRDP_SEND_PARAM_T` *pSendParam)
Initiate sending MD confirm message.
- `EXT_DECL TRDP_ERR_T tlm_abortSession` (`TRDP_APP_SESSION_T` appHandle, const `TRDP_UUID_T` *pSessionId)
Cancel an open session.

5.20.1 Detailed Description

Functions for Message Data Communication.

API implementation of TRDP Light

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013-2019. All rights reserved.

5.20.2 Function Documentation

5.20.2.1 tlm_abortSession()

```
EXT_DECL TRDP_ERR_T tlm_abortSession (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId )
```

Cancel an open session.

Abort an open session; any pending messages will be dropped

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>p↔ SessionId</i>	Session ID returned by request

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOSESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.20.2.2 tlm_addListener()

```
EXT_DECL TRDP_ERR_T tlm_addListener (
    TRDP_APP_SESSION_T appHandle,
    TRDP_LIS_T * pListenHandle,
    const void * pUserRef,
    TRDP_MD_CALLBACK_T pfCbFunction,
    BOOL8 comIdListener,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr1,
    TRDP_IP_ADDR_T srcIpAddr2,
    TRDP_IP_ADDR_T mcDestIpAddr,
    TRDP_FLAGS_T pktFlags,
    const TRDP_URI_USER_T srcURI,
    const TRDP_URI_USER_T destURI )
```

Subscribe to MD messages.

Add a listener to TRDP to get notified when messages are received

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>pListenHandle</i>	Handle for this listener returned
in	<i>pUserRef</i>	user supplied value returned with received message
in	<i>pfCbFunction</i>	Pointer to listener specific callback function, NULL to use default function
in	<i>comIdListener</i>	set TRUE if <code>comId</code> shall be observed
in	<i>comId</i>	<code>comId</code> to be observed
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr1</i>	Source IP address, lower address in case of address range, set to 0 if not used
in	<i>srcIpAddr2</i>	upper address in case of address range, set to 0 if not used
in	<i>mcDestIpAddr</i>	multicast group to listen on
in	<i>pktFlags</i>	OPTION: <code>TRDP_FLAGS_DEFAULT</code> , <code>TRDP_FLAGS_MARSHALL</code>
in	<i>srcURI</i>	only functional group of source URI, set to NULL if not used
in	<i>destURI</i>	only functional group of destination URI, set to NULL if not used

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.20.2.3 tlm_confirm()

```

TRDP_ERR_T tlm_confirm (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId,
    UINT16 userStatus,
    const TRDP_SEND_PARAM_T * pSendParam )

```

Initiate sending MD confirm message.

Send a MD confirmation message User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pSessionId</i>	Session ID returned by request
in	<i>userStatus</i>	Info for requester about application errors
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters

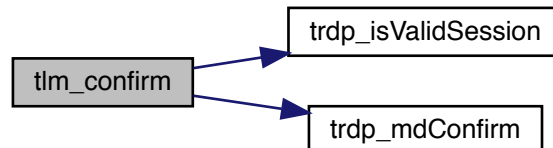
Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error

Return values

<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOSESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:



5.20.2.4 tlm_delListener()

```

TRDP_ERR_T tlm_delListener (
    TRDP_APP_SESSION_T appHandle,
    TRDP_LIS_T listenHandle )
  
```

Remove Listener.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>listenHandle</i>	Handle for this listener

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.20.2.5 tlm_getInterval()

```

EXT_DECL TRDP_ERR_T tlm_getInterval (
    TRDP_APP_SESSION_T appHandle,
    TRDP_TIME_T * pInterval,
  
```

```

TRDP_FDS_T * pFileDesc,
INT32 * pNoDesc )

```

Get the lowest time interval for MDs.

Return the maximum time interval suitable for 'select()' so that we can report time outs to the higher layer.

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
out	<i>pInterval</i>	pointer to needed interval
in, out	<i>pFileDesc</i>	pointer to file descriptor set
out	<i>pNoDesc</i>	pointer to put no of highest used descriptors (for <code>select()</code>)

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.20.2.6 tlm_notify()

```

TRDP_ERR_T tlm_notify (
    TRDP_APP_SESSION_T appHandle,
    const void * pUserRef,
    TRDP_MD_CALLBACK_T pfCbFunction,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr,
    TRDP_IP_ADDR_T destIpAddr,
    TRDP_FLAGS_T pktFlags,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize,
    const TRDP_URI_USER_T sourceURI,
    const TRDP_URI_USER_T destURI )

```

Initiate sending MD notification message.

Send a MD notification message

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pUserRef</i>	user supplied value returned with reply
in	<i>pfCbFunction</i>	Pointer to listener specific callback function, NULL to use default function
in	<i>comId</i>	comId of packet to be sent
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to

Parameters

in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
in	<i>pSendParam</i>	optional pointer to send parameter, NULL - default parameters are used
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data
in	<i>sourceURI</i>	only functional group of source URI
in	<i>destURI</i>	only functional group of destination URI

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.20.2.7 tlm_process()

```
EXT_DECL TRDP_ERR_T tlm_process (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FDS_T * pRfds,
    INT32 * pCount )
```

Message Data Work loop of the TRDP handler.

Search the queue for pending MDs to be sent Search the receive queue for pending MDs (replies, time outs) and incoming requests

Parameters

in	<i>appHandle</i>	The handle returned by tlc_openSession
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.20.2.8 tlm_readListener()

```
EXT_DECL TRDP_ERR_T tlm_readListener (
    TRDP_APP_SESSION_T appHandle,
    TRDP_LIS_T listenHandle,
```

```

UINT32 etbTopoCnt,
UINT32 opTrnTopoCnt,
TRDP_IP_ADDR_T srcIpAddr1,
TRDP_IP_ADDR_T srcIpAddr2,
TRDP_IP_ADDR_T mcDestIpAddr )

```

Resubscribe to MD messages.

Readd a listener after topoCount changes to get notified when messages are received

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>listenHandle</i>	Handle for this listener
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr1</i>	Source IP address, lower address in case of address range, set to 0 if not used
in	<i>srcIpAddr2</i>	upper address in case of address range, set to 0 if not used
in	<i>mcDestIpAddr</i>	multicast group to listen on

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.20.2.9 tlm_reply()

```

TRDP_ERR_T tlm_reply (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId,
    UINT32 comId,
    UINT16 userStatus,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize )

```

Send a MD reply message.

Send a MD reply message after receiving an request User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

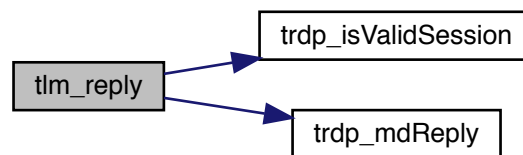
Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pSessionId</i>	Session ID returned by indication
in	<i>comId</i>	comId of packet to be sent
in	<i>userStatus</i>	Info for requester about application errors
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	Out of memory
<i>TRDP_NO_SESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:



5.20.2.10 tlm_replyQuery()

```

TRDP_ERR_T tlm_replyQuery (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId,
    UINT32 comId,
    UINT16 userStatus,
    UINT32 confirmTimeout,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize )
  
```

Send a MD reply query message.

Send a MD reply query message after receiving a request and ask for confirmation. User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

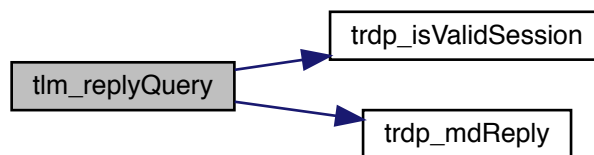
Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pSessionId</i>	Session ID returned by indication
in	<i>comId</i>	comId of packet to be sent
in	<i>userStatus</i>	Info for requester about application errors
in	<i>confirmTimeout</i>	timeout for confirmation
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NO_SESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:



5.20.2.11 tlm_request()

```

TRDP_ERR_T tlm_request (
    TRDP_APP_SESSION_T appHandle,
    const void * pUserRef,
    TRDP_MD_CALLBACK_T pfCbFunction,
    TRDP_UUID_T * pSessionId,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr,
    TRDP_IP_ADDR_T destIpAddr,
    TRDP_FLAGS_T pktFlags,
    UINT32 numReplies,
    UINT32 replyTimeout,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize,
    const TRDP_URI_USER_T sourceURI,
    const TRDP_URI_USER_T destURI )
  
```

Initiate sending MD request message.

Send a MD request message

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pUserRef</i>	user supplied value returned with reply

Parameters

in	<i>pfCbFunction</i>	Pointer to listener specific callback function, NULL to use default function
out	<i>pSessionId</i>	return session ID
in	<i>comId</i>	comId of packet to be sent
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL
in	<i>numReplies</i>	number of expected replies, 0 if unknown
in	<i>replyTimeout</i>	timeout for reply
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data
in	<i>sourceURI</i>	only functional group of source URI
in	<i>destURI</i>	only functional group of destination URI

Return values

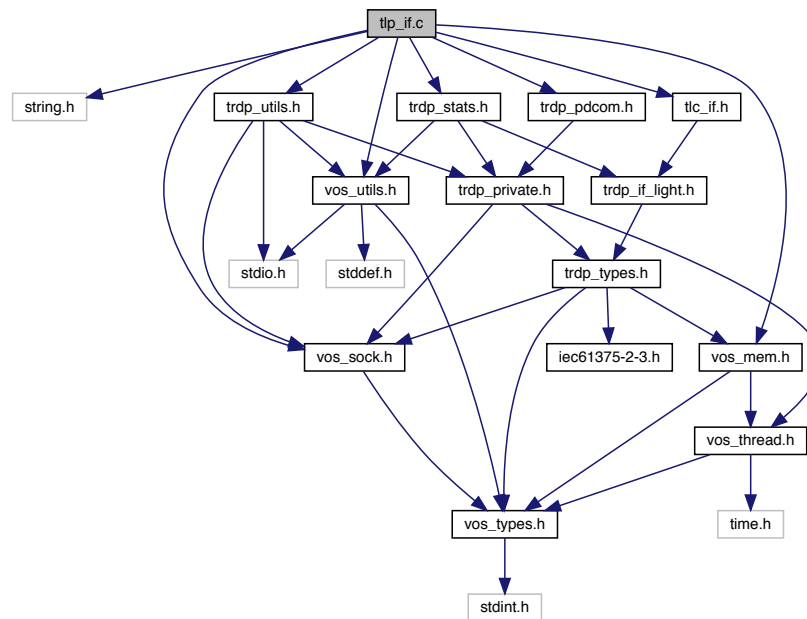
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21 tlp_if.c File Reference

Functions for Process Data Communication.

```
#include <string.h>
#include "tlc_if.h"
#include "trdp_utils.h"
#include "trdp_pdcom.h"
#include "trdp_stats.h"
#include "vos_sock.h"
#include "vos_mem.h"
#include "vos_utils.h"
```

Include dependency graph for tlp_if.c:



Functions

- EXT_DECL [TRDP_ERR_T tlp_getInterval](#) (TRDP_APP_SESSION_T appHandle, [TRDP_TIME_T](#) *pInterval, [TRDP_FDS_T](#) *pFileDesc, INT32 *pNoDesc)
Get the lowest time interval for PDs.
- EXT_DECL [TRDP_ERR_T tlp_processReceive](#) (TRDP_APP_SESSION_T appHandle, [TRDP_FDS_T](#) *p↔ Rfds, INT32 *pCount)
Work loop of the TRDP handler.
- EXT_DECL [TRDP_ERR_T tlp_processSend](#) (TRDP_APP_SESSION_T appHandle)
Work loop of the TRDP handler.
- [TRDP_ERR_T tlp_setRedundant](#) (TRDP_APP_SESSION_T appHandle, UINT32 redId, BOOL8 leader)
Do not send non-redundant PDs when we are follower.
- EXT_DECL [TRDP_ERR_T tlp_getRedundant](#) (TRDP_APP_SESSION_T appHandle, UINT32 redId, BOOL8 *pLeader)
Get status of redundant Comlds.
- EXT_DECL [TRDP_ERR_T tlp_publish](#) (TRDP_APP_SESSION_T appHandle, [TRDP_PUB_T](#) *pPubHandle, const void *pUserRef, [TRDP_PD_CALLBACK_T](#) pfCbFunction, UINT32 serviceId, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr, UINT32 interval, UINT32 redId, [TRDP_FLAGS_T](#) pktFlags, const [TRDP_SEND_PARAM_T](#) *pSendParam, const U↔ INT8 *pData, UINT32 dataSize)
Prepare for sending PD messages.
- EXT_DECL [TRDP_ERR_T tlp_republish](#) (TRDP_APP_SESSION_T appHandle, [TRDP_PUB_T](#) pubHandle, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr)
Prepare for sending PD messages.
- [TRDP_ERR_T tlp_unpublish](#) (TRDP_APP_SESSION_T appHandle, [TRDP_PUB_T](#) pubHandle)
Stop sending PD messages.
- [TRDP_ERR_T tlp_put](#) (TRDP_APP_SESSION_T appHandle, [TRDP_PUB_T](#) pubHandle, const UINT8 *p↔ Data, UINT32 dataSize)

Update the process data to send.

- `TRDP_ERR_T tlp_putImmediate` (`TRDP_APP_SESSION_T` appHandle, `TRDP_PUB_T` pubHandle, const `UINT8 *pData`, `UINT32` dataSize, `VOS_TIMEVAL_T *pTxTime`)

Update and send process data.

- `EXT_DECL TRDP_ERR_T tlp_request` (`TRDP_APP_SESSION_T` appHandle, `TRDP_SUB_T` subHandle, `UINT32` serviceId, `UINT32` comId, `UINT32` etbTopoCnt, `UINT32` opTrnTopoCnt, `TRDP_IP_ADDR_T` srcIpAddr, `TRDP_IP_ADDR_T` destIpAddr, `UINT32` redId, `TRDP_FLAGS_T` pktFlags, const `TRDP_SEND_PARAM_T *pSendParam`, const `UINT8 *pData`, `UINT32` dataSize, `UINT32` replyComId, `TRDP_IP_ADDR_T` replyIpAddr)

Initiate sending PD messages (PULL).

- `EXT_DECL TRDP_ERR_T tlp_subscribe` (`TRDP_APP_SESSION_T` appHandle, `TRDP_SUB_T *pSubHandle`, const void *pUserRef, `TRDP_PD_CALLBACK_T` pfCbFunction, `UINT32` serviceId, `UINT32` comId, `UINT32` etbTopoCnt, `UINT32` opTrnTopoCnt, `TRDP_IP_ADDR_T` srcIpAddr1, `TRDP_IP_ADDR_T` srcIpAddr2, `TRDP_IP_ADDR_T` destIpAddr, `TRDP_FLAGS_T` pktFlags, const `TRDP_COM_PARAM_T *pRecParams`, `UINT32` timeout, `TRDP_TO_BEHAVIOR_T` toBehavior)

Prepare for receiving PD messages.

- `EXT_DECL TRDP_ERR_T tlp_unsubscribe` (`TRDP_APP_SESSION_T` appHandle, `TRDP_SUB_T` subHandle)

Stop receiving PD messages.

- `EXT_DECL TRDP_ERR_T tlp_resubscribe` (`TRDP_APP_SESSION_T` appHandle, `TRDP_SUB_T` subHandle, `UINT32` etbTopoCnt, `UINT32` opTrnTopoCnt, `TRDP_IP_ADDR_T` srcIpAddr1, `TRDP_IP_ADDR_T` srcIpAddr2, `TRDP_IP_ADDR_T` destIpAddr)

Reprepare for receiving PD messages.

- `EXT_DECL TRDP_ERR_T tlp_get` (`TRDP_APP_SESSION_T` appHandle, `TRDP_SUB_T` subHandle, `TRDP_PD_INFO_T *pPdInfo`, `UINT8 *pData`, `UINT32 *pDataSize`)

Get the last valid PD message.

5.21.1 Detailed Description

Functions for Process Data Communication.

API implementation of TRDP Light

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013-2019. All rights reserved.

5.21.2 Function Documentation

5.21.2.1 tlp_get()

```
EXT_DECL TRDP_ERR_T tlp_get (
    TRDP_APP_SESSION_T appHandle,
    TRDP_SUB_T subHandle,
    TRDP_PD_INFO_T * pPdInfo,
    UINT8 * pData,
    UINT32 * pDataSize )
```

Get the last valid PD message.

This allows polling of PDs instead of event driven handling by callbacks

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>subHandle</i>	the handle returned by subscription
in, out	<i>pPdInfo</i>	pointer to application's info buffer
in, out	<i>pData</i>	pointer to application's data buffer
in, out	<i>pDataSize</i>	in: size of buffer, out: size of data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_SUB_ERR</i>	not subscribed
<i>TRDP_TIMEOUT_ERR</i>	packet timed out
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_COMID_ERR</i>	ComID not found when marshalling

5.21.2.2 tlp_getInterval()

```
EXT_DECL TRDP_ERR_T tlp_getInterval (
    TRDP_APP_SESSION_T appHandle,
    TRDP_TIME_T * pInterval,
    TRDP_FDS_T * pFileDesc,
    INT32 * pNoDesc )
```

Get the lowest time interval for PDs.

Return the maximum time interval suitable for 'select()' so that we can send due PD packets in time. If the PD send queue is empty, return zero time

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
out	<i>pInterval</i>	pointer to needed interval
in, out	<i>pFileDesc</i>	pointer to file descriptor set
out	<i>pNoDesc</i>	pointer to put no of highest used descriptors (for select())

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21.2.3 tlp_getRedundant()

```
EXT_DECL TRDP_ERR_T tlp_getRedundant (
    TRDP_APP_SESSION_T appHandle,
    UINT32 redId,
    BOOL8 * pLeader )
```

Get status of redundant ComIds.

Only the status of the first found redundancy group entry will be returned!

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>redId</i>	will be returned for all ComID's with the given redId
in, out	<i>pLeader</i>	TRUE if we're sending this redundancy group (leader)

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	redId invalid or not existing
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21.2.4 tlp_processReceive()

```
EXT_DECL TRDP_ERR_T tlp_processReceive (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FDS_T * pRfds,
    INT32 * pCount )
```

Work loop of the TRDP handler.

Check the sockets for incoming PD telegrams. Search the receive queue for pending PDs (time out) and report them, either by informing the higher layer via the callback mechanism or just by marking the subscriber as timed-out

Parameters

in	<i>appHandle</i>	The handle returned by tlc_openSession
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21.2.5 tlp_processSend()

```
EXT_DECL TRDP_ERR_T tlp_processSend (
    TRDP_APP_SESSION_T appHandle )
```

Work loop of the TRDP handler.

Search the queue for pending PDs to be sent

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
----	------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21.2.6 tlp_publish()

```
EXT_DECL TRDP_ERR_T tlp_publish (
    TRDP_APP_SESSION_T appHandle,
    TRDP_PUB_T * pPubHandle,
    const void * pUserRef,
    TRDP_PD_CALLBACK_T pCbFunction,
    UINT32 serviceId,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr,
    TRDP_IP_ADDR_T destIpAddr,
    UINT32 interval,
    UINT32 redId,
    TRDP_FLAGS_T pktFlags,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize )
```

Prepare for sending PD messages.

Queue a PD message, it will be send when `tlc_publish` has been called

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>pPubHandle</i>	returned handle for related re/unpublish
in	<i>pUserRef</i>	user supplied value returned within the info structure of callback function
in	<i>pfCbFunction</i>	Pointer to pre-send callback function, NULL if not used
in	<i>serviceld</i>	optional serviceld this telegram belongs to (default = 0)
in	<i>comld</i>	comld of packet to send
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>interval</i>	frequency of PD packet (≥ 10 ms) in usec
in	<i>redld</i>	0 - Non-redundant, > 0 valid redundancy group
in	<i>pktFlags</i>	OPTION: <code>TRDP_FLAGS_DEFAULT</code> , <code>TRDP_FLAGS_NONE</code> , <code>TRDP_FLAGS_MARSHALL</code> , <code>TRDP_FLAGS_CALLBACK</code>
in	<i>pSendParam</i>	optional pointer to send parameter, NULL - default parameters are used
in	<i>pData</i>	optional pointer to data packet / dataset, NULL if sending starts later with tlp_put()
in	<i>dataSize</i>	size of data packet ≥ 0 and \leq <code>TRDP_MAX_PD_DATA_SIZE</code>

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not insert (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21.2.7 tlp_put()

```
TRDP_ERR_T tlp_put (
    TRDP_APP_SESSION_T appHandle,
    TRDP_PUB_T pubHandle,
    const UINT8 * pData,
    UINT32 dataSize )
```

Update the process data to send.

Update previously published data. The new telegram will be sent earliest when `tlc_process` is called.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pubHandle</i>	the handle returned by <code>publish</code>
in, out	<i>pData</i>	pointer to application's data buffer
in, out	<i>dataSize</i>	size of data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error on uninitialized parameter or changed dataSize compared to published one
<i>TRDP_NOPUB_ERR</i>	not published
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_COMID_ERR</i>	ComID not found when marshalling

5.21.2.8 tlp_putImmediate()

```
TRDP_ERR_T tlp_putImmediate (
    TRDP_APP_SESSION_T appHandle,
    TRDP_PUB_T pubHandle,
    const UINT8 * pData,
    UINT32 dataSize,
    VOS_TIMEVAL_T * pTxTime )
```

Update and send process data.

Update previously published data. The new telegram will be sent immediatly or at txTime, if txTime != 0 and TSN == 1 Should be used if application (or higher layer, e.g. ara::com and acyclic events) needs full control over process data schedule.

Note: For TSN this function is not protected by any mutexes and should not be called while adding or removing any publishers, subscribers or even sessions! Also: Marshalling is not supported!

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>pubHandle</i>	the handle returned by publish
in, out	<i>pData</i>	pointer to application's data buffer
in, out	<i>dataSize</i>	size of data
in	<i>pTxTime</i>	when to send (absolute time), optional for TSN only

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error on uninitialized parameter or changed dataSize compared to published one
<i>TRDP_NOPUB_ERR</i>	not published
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21.2.9 tlp_republish()

```
EXT_DECL TRDP_ERR_T tlp_republish (
    TRDP_APP_SESSION_T appHandle,
```



```

TRDP_PUB_T pubHandle,
UINT32 etbTopoCnt,
UINT32 opTrnTopoCnt,
TRDP_IP_ADDR_T srcIpAddr,
TRDP_IP_ADDR_T destIpAddr )

```

Prepare for sending PD messages.

Reinitialize and queue a PD message, it will be send when tlc_publish has been called

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>pubHandle</i>	handle for related unpubish
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not insert (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21.2.10 tlp_request()

```

EXT_DECL TRDP_ERR_T tlp_request (
    TRDP_APP_SESSION_T appHandle,
    TRDP_SUB_T subHandle,
    UINT32 serviceId,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr,
    TRDP_IP_ADDR_T destIpAddr,
    UINT32 redId,
    TRDP_FLAGS_T pktFlags,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize,
    UINT32 replyComId,
    TRDP_IP_ADDR_T replyIpAddr )

```

Initiate sending PD messages (PULL).

Send a PD request message

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
----	------------------	--

Parameters

in	<i>subHandle</i>	handle from related subscribe
in	<i>serviceld</i>	optional serviceld this telegram belongs to (default = 0)
in	<i>comId</i>	comId of packet to be sent
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>redId</i>	0 - Non-redundant, > 0 valid redundancy group
in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
in	<i>pSendParam</i>	optional pointer to send parameter, NULL - default parameters are used
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data
in	<i>replyComId</i>	comId of reply (default comID of subscription)
in	<i>replyIpAddr</i>	IP for reply

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not insert (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_NOSUB_ERR</i>	no matching subscription found

5.21.2.11 tlp_resubscribe()

```
EXT_DECL TRDP_ERR_T tlp_resubscribe (
    TRDP_APP_SESSION_T appHandle,
    TRDP_SUB_T subHandle,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr1,
    TRDP_IP_ADDR_T srcIpAddr2,
    TRDP_IP_ADDR_T destIpAddr )
```

Reprepare for receiving PD messages.

Resubscribe to a specific PD ComID and source IP

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>subHandle</i>	handle for this subscription
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr1</i>	Source IP address, lower address in case of address range, set to 0 if not used
in	<i>srcIpAddr2</i>	upper address in case of address range, set to 0 if not used
in	<i>destIpAddr</i>	IP address to join

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not reserve memory (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP SOCK_ERR</i>	Resource (socket) not available, subscription canceled

5.21.2.12 tlp_setRedundant()

```
TRDP_ERR_T tlp_setRedundant (
    TRDP_APP_SESSION_T appHandle,
    UINT32 redId,
    BOOL8 leader )
```

Do not send non-redundant PDs when we are follower.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>redId</i>	will be set for all ComID's with the given redId, 0 to change for all redId
in	<i>leader</i>	TRUE if we send

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error / redId not existing
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21.2.13 tlp_subscribe()

```
EXT_DECL TRDP_ERR_T tlp_subscribe (
    TRDP_APP_SESSION_T appHandle,
    TRDP_SUB_T * pSubHandle,
    const void * pUserRef,
    TRDP_PD_CALLBACK_T pfCbFunction,
    UINT32 serviceId,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr1,
    TRDP_IP_ADDR_T srcIpAddr2,
    TRDP_IP_ADDR_T destIpAddr,
    TRDP_FLAGS_T pktFlags,
    const TRDP_COM_PARAM_T * pRecParams,
    UINT32 timeout,
    TRDP_TO_BEHAVIOR_T toBehavior )
```

Prepare for receiving PD messages.

Subscribe to a specific PD ComID and source IP.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>pSubHandle</i>	return a handle for this subscription
in	<i>pUserRef</i>	user supplied value returned within the info structure
in	<i>pfCbFunction</i>	Pointer to subscriber specific callback function, NULL to use default function
in	<i>serviceld</i>	optional serviceld this telegram belongs to (default = 0)
in	<i>comId</i>	comId of packet to receive
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr1</i>	Source IP address, lower address in case of address range, set to 0 if not used
in	<i>srcIpAddr2</i>	upper address in case of address range, set to 0 if not used
in	<i>destIpAddr</i>	IP address to join
in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
in	<i>pRecParams</i>	optional pointer to send parameter, NULL - default parameters are used
in	<i>timeout</i>	timeout (>= 10ms) in usec
in	<i>toBehavior</i>	timeout behavior

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not reserve memory (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21.2.14 tlp_unpublish()

```
TRDP_ERR_T tlp_unpublish (
    TRDP_APP_SESSION_T appHandle,
    TRDP_PUB_T pubHandle )
```

Stop sending PD messages.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pubHandle</i>	the handle returned by <code>prepare</code>

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_NOPUB_ERR</i>	not published

Return values

<i>TRDP_NOINIT_ERR</i>	handle invalid
------------------------	----------------

5.21.2.15 tlp_unsubscribe()

```
EXT_DECL TRDP_ERR_T tlp_unsubscribe (
    TRDP_APP_SESSION_T appHandle,
    TRDP_SUB_T subHandle )
```

Stop receiving PD messages.

Unsubscribe to a specific PD ComID

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>subHandle</i>	the handle for this subscription

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_NOSUB_ERR</i>	not subscribed
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.22 trdp_dllmain.c File Reference

Windows DLL main function.

5.22.1 Detailed Description

Windows DLL main function.

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss, Bombardier

Remarks

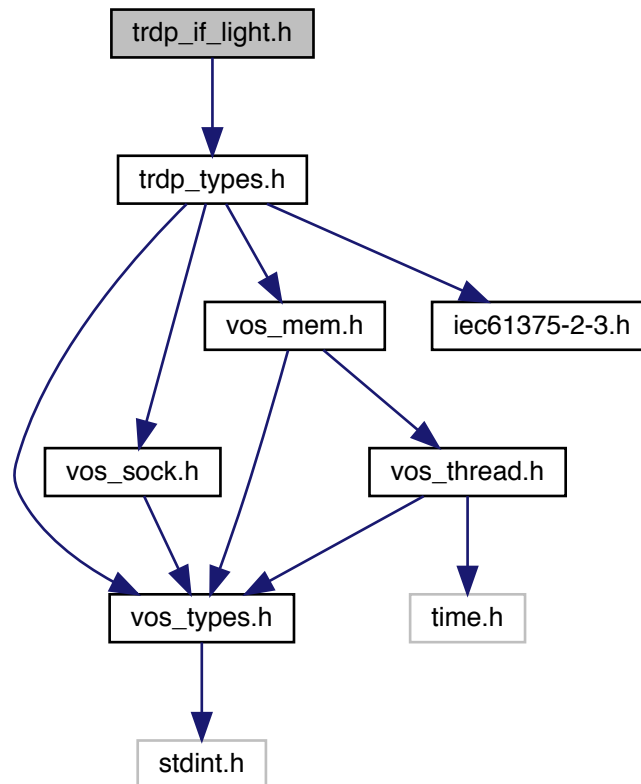
This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.23 trdp_if_light.h File Reference

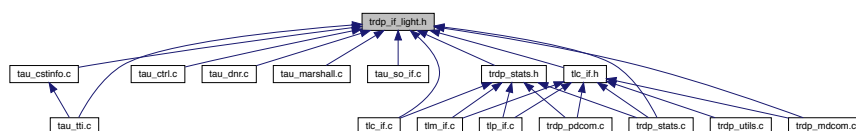
TRDP Light interface functions (API)

```
#include "trdp_types.h"
```

Include dependency graph for trdp_if_light.h:



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



Functions

- EXT_DECL [TRDP_ERR_T](#) [tlc_init](#) (const [TRDP_PRINT_DBG_T](#) pPrintDebugString, void *pRefCon, const [TRDP_MEM_CONFIG_T](#) *pMemConfig)

Support for message data can only be excluded during compile time!

- EXT_DECL `TRDP_ERR_T tlc_openSession` (`TRDP_APP_SESSION_T *pAppHandle`, `TRDP_IP_ADDR_T ownIpAddr`, `TRDP_IP_ADDR_T leaderIpAddr`, const `TRDP_MARSHALL_CONFIG_T *pMarshall`, const `TRDP_PD_CONFIG_T *pPdDefault`, const `TRDP_MD_CONFIG_T *pMdDefault`, const `TRDP_PROCESS_CONFIG_T *pProcessConfig`)

Open a session with the TRDP stack.

- EXT_DECL `TRDP_ERR_T tlc_reinitSession` (`TRDP_APP_SESSION_T appHandle`)

Re-Initialize.

- EXT_DECL `TRDP_ERR_T tlc_configSession` (`TRDP_APP_SESSION_T appHandle`, const `TRDP_MARSHALL_CONFIG_T *pMarshall`, const `TRDP_PD_CONFIG_T *pPdDefault`, const `TRDP_MD_CONFIG_T *pMdDefault`, const `TRDP_PROCESS_CONFIG_T *pProcessConfig`)

(Re-)configure a session.

- EXT_DECL `TRDP_ERR_T tlc_updateSession` (`TRDP_APP_SESSION_T appHandle`)

Update a session.

- EXT_DECL `TRDP_ERR_T tlc_presetIndexSession` (`TRDP_APP_SESSION_T appHandle`, `TRDP_INDEX_TABLE_SIZE_T *pIndexTableSizes`)

Preset the index table sizes of a session.

- EXT_DECL `TRDP_ERR_T tlc_closeSession` (`TRDP_APP_SESSION_T appHandle`)

Close a session.

- EXT_DECL `TRDP_ERR_T tlc_terminate` (void)

Un-Initialize.

- EXT_DECL `TRDP_ERR_T tlc_setETBTopoCount` (`TRDP_APP_SESSION_T appHandle`, `UINT32 etbTopoCnt`)

Set new topocount for trainwide communication.

- EXT_DECL `UINT32 tlc_getETBTopoCount` (`TRDP_APP_SESSION_T appHandle`)

Set new topocount for trainwide communication.

- EXT_DECL `TRDP_ERR_T tlc_setOpTrainTopoCount` (`TRDP_APP_SESSION_T appHandle`, `UINT32 opTrnTopoCnt`)

Set new operational train topocount for direction/orientation sensitive communication.

- EXT_DECL `UINT32 tlc_getOpTrainTopoCount` (`TRDP_APP_SESSION_T appHandle`)

Set new operational train topocount for direction/orientation sensitive communication.

- EXT_DECL `TRDP_ERR_T tlc_getInterval` (`TRDP_APP_SESSION_T appHandle`, `TRDP_TIME_T *pInterval`, `TRDP_FDS_T *pFileDesc`, `INT32 *pNoDesc`)

Get the lowest time interval for PDs.

- EXT_DECL `TRDP_ERR_T tlc_process` (`TRDP_APP_SESSION_T appHandle`, `TRDP_FDS_T *pRfds`, `INT32 *pCount`)

Work loop of the TRDP handler.

- EXT_DECL `TRDP_IP_ADDR_T tlc_getOwnIpAddress` (`TRDP_APP_SESSION_T appHandle`)

Get the interface address.

- EXT_DECL `TRDP_ERR_T tlp_getInterval` (`TRDP_APP_SESSION_T appHandle`, `TRDP_TIME_T *pInterval`, `TRDP_FDS_T *pFileDesc`, `INT32 *pNoDesc`)

Get the lowest time interval for PDs.

- EXT_DECL `TRDP_ERR_T tlp_processSend` (`TRDP_APP_SESSION_T appHandle`)

Work loop of the TRDP handler.

- EXT_DECL `TRDP_ERR_T tlp_processReceive` (`TRDP_APP_SESSION_T appHandle`, `TRDP_FDS_T *pRfds`, `INT32 *pCount`)

Work loop of the TRDP handler.

- EXT_DECL `TRDP_ERR_T tlp_publish` (`TRDP_APP_SESSION_T appHandle`, `TRDP_PUB_T *pPubHandle`, const void `*pUserRef`, `TRDP_PD_CALLBACK_T pfCbFunction`, `UINT32 serviceId`, `UINT32 comId`, `UINT32 etbTopoCnt`, `UINT32 opTrnTopoCnt`, `TRDP_IP_ADDR_T srcIpAddr`, `TRDP_IP_ADDR_T destIpAddr`, `UINT32 interval`, `UINT32 redId`, `TRDP_FLAGS_T pktFlags`, const `TRDP_SEND_PARAM_T *pSendParam`, const `UINT8 *pData`, `UINT32 dataSize`)

Prepare for sending PD messages.

- EXT_DECL [TRDP_ERR_T tlp_republish](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_PUB_T](#) pubHandle, [UINT32](#) etbTopoCnt, [UINT32](#) opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr)
Prepare for sending PD messages.
- EXT_DECL [TRDP_ERR_T tlp_unpublish](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_PUB_T](#) pubHandle)
Stop sending PD messages.
- EXT_DECL [TRDP_ERR_T tlp_put](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_PUB_T](#) pubHandle, const [UINT8](#) *pData, [UINT32](#) dataSize)
Update the process data to send.
- EXT_DECL [TRDP_ERR_T tlp_putImmediate](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_PUB_T](#) pubHandle, const [UINT8](#) *pData, [UINT32](#) dataSize, [VOS_TIMEVAL_T](#) *pTxTime)
Update and send process data.
- EXT_DECL [TRDP_ERR_T tlp_setRedundant](#) ([TRDP_APP_SESSION_T](#) appHandle, [UINT32](#) redId, [BOOL8](#) leader)
Do not send non-redundant PDs when we are follower.
- EXT_DECL [TRDP_ERR_T tlp_getRedundant](#) ([TRDP_APP_SESSION_T](#) appHandle, [UINT32](#) redId, [BOOL8](#) *pLeader)
Get status of redundant Comlds.
- EXT_DECL [TRDP_ERR_T tlp_request](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_SUB_T](#) subHandle, [UINT32](#) servId, [UINT32](#) comId, [UINT32](#) etbTopoCnt, [UINT32](#) opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr, [UINT32](#) redId, [TRDP_FLAGS_T](#) pktFlags, const [TRDP_SEND_PARAM_T](#) *pSendParam, const [UINT8](#) *pData, [UINT32](#) dataSize, [UINT32](#) replyComId, [TRDP_IP_ADDR_T](#) replyIpAddr)
Initiate sending PD messages (PULL).
- EXT_DECL [TRDP_ERR_T tlp_subscribe](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_SUB_T](#) *pSubHandle, const void *pUserRef, [TRDP_PD_CALLBACK_T](#) pfCbFunction, [UINT32](#) servId, [UINT32](#) comId, [UINT32](#) etbTopoCnt, [UINT32](#) opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr1, [TRDP_IP_ADDR_T](#) srcIpAddr2, [TRDP_IP_ADDR_T](#) destIpAddr, [TRDP_FLAGS_T](#) pktFlags, const [TRDP_COM_PARAM_T](#) *pRecParams, [UINT32](#) timeout, [TRDP_TO_BEHAVIOR_T](#) toBehavior)
Prepare for receiving PD messages.
- EXT_DECL [TRDP_ERR_T tlp_resubscribe](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_SUB_T](#) subHandle, [UINT32](#) etbTopoCnt, [UINT32](#) opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr1, [TRDP_IP_ADDR_T](#) srcIpAddr2, [TRDP_IP_ADDR_T](#) destIpAddr)
Reprepare for receiving PD messages.
- EXT_DECL [TRDP_ERR_T tlp_unsubscribe](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_SUB_T](#) subHandle)
Stop receiving PD messages.
- EXT_DECL [TRDP_ERR_T tlp_get](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_SUB_T](#) subHandle, [TRDP_PD_INFO_T](#) *pPdInfo, [UINT8](#) *pData, [UINT32](#) *pDataSize)
Get the last valid PD message.
- EXT_DECL [TRDP_ERR_T tlm_process](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_FDS_T](#) *pRfds, [INT32](#) *pCount)
Message Data Work loop of the TRDP handler.
- EXT_DECL [TRDP_ERR_T tlm_getInterval](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_TIME_T](#) *pInterval, [TRDP_FDS_T](#) *pFileDesc, [INT32](#) *pNoDesc)
Get the lowest time interval for MDs.
- EXT_DECL [TRDP_ERR_T tlm_notify](#) ([TRDP_APP_SESSION_T](#) appHandle, const void *pUserRef, [TRDP_MD_CALLBACK_T](#) pfCbFunction, [UINT32](#) comId, [UINT32](#) etbTopoCnt, [UINT32](#) opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr, [TRDP_FLAGS_T](#) pktFlags, const [TRDP_SEND_PARAM_T](#) *pSendParam, const [UINT8](#) *pData, [UINT32](#) dataSize, const [TRDP_URI_USER_T](#) sourceURI, const [TRDP_URI_USER_T](#) destURI)
Initiate sending MD notification message.

- EXT_DECL [TRDP_ERR_T tlm_request](#) ([TRDP_APP_SESSION_T](#) appHandle, const void *pUserRef, [TRDP_MD_CALLBACK_T](#) pfCbFunction, [TRDP_UUID_T](#) *pSessionId, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr, [TRDP_FLAGS_T](#) pktFlags, UINT32 numReplies, UINT32 replyTimeout, const [TRDP_SEND_PARAM_T](#) *pSendParam, const UINT8 *pData, UINT32 dataSize, const [TRDP_URI_USER_T](#) sourceURI, const [TRDP_URI_USER_T](#) destURI)
Initiate sending MD request message.
- EXT_DECL [TRDP_ERR_T tlm_confirm](#) ([TRDP_APP_SESSION_T](#) appHandle, const [TRDP_UUID_T](#) *pSessionId, UINT16 userStatus, const [TRDP_SEND_PARAM_T](#) *pSendParam)
Initiate sending MD confirm message.
- EXT_DECL [TRDP_ERR_T tlm_abortSession](#) ([TRDP_APP_SESSION_T](#) appHandle, const [TRDP_UUID_T](#) *pSessionId)
Cancel an open session.
- EXT_DECL [TRDP_ERR_T tlm_addListener](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_LIS_T](#) *pListenHandle, const void *pUserRef, [TRDP_MD_CALLBACK_T](#) pfCbFunction, BOOL8 comIdListener, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr1, [TRDP_IP_ADDR_T](#) srcIpAddr2, [TRDP_IP_ADDR_T](#) mcDestIpAddr, [TRDP_FLAGS_T](#) pktFlags, const [TRDP_URI_USER_T](#) srcURI, const [TRDP_URI_USER_T](#) destURI)
Subscribe to MD messages.
- EXT_DECL [TRDP_ERR_T tlm_readdListener](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_LIS_T](#) listenHandle, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) srcIpAddr2, [TRDP_IP_ADDR_T](#) mcDestIpAddr)
Resubscribe to MD messages.
- EXT_DECL [TRDP_ERR_T tlm_delListener](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_LIS_T](#) listenHandle)
Remove Listener.
- [TRDP_ERR_T tlm_reply](#) ([TRDP_APP_SESSION_T](#) appHandle, const [TRDP_UUID_T](#) *pSessionId, UINT32 comId, UINT16 userStatus, const [TRDP_SEND_PARAM_T](#) *pSendParam, const UINT8 *pData, UINT32 dataSize)
Send a MD reply message.
- [TRDP_ERR_T tlm_replyQuery](#) ([TRDP_APP_SESSION_T](#) appHandle, const [TRDP_UUID_T](#) *pSessionId, UINT32 comId, UINT16 userStatus, UINT32 confirmTimeout, const [TRDP_SEND_PARAM_T](#) *pSendParam, const UINT8 *pData, UINT32 dataSize)
Send a MD reply query message.
- EXT_DECL const CHAR8 * [tlc_getVersionString](#) (void)
Return a human readable version representation.
- EXT_DECL const [TRDP_VERSION_T](#) * [tlc_getVersion](#) (void)
Return version.
- EXT_DECL [TRDP_ERR_T tlc_getStatistics](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_STATISTICS_T](#) *pStatistics)
Return statistics.
- EXT_DECL [TRDP_ERR_T tlc_getSubsStatistics](#) ([TRDP_APP_SESSION_T](#) appHandle, UINT16 *pNumSubs, [TRDP_SUBS_STATISTICS_T](#) *pStatistics)
Return PD subscription statistics.
- EXT_DECL [TRDP_ERR_T tlc_getPubStatistics](#) ([TRDP_APP_SESSION_T](#) appHandle, UINT16 *pNumPub, [TRDP_PUB_STATISTICS_T](#) *pStatistics)
Return PD publish statistics.
- EXT_DECL [TRDP_ERR_T tlc_getRedStatistics](#) ([TRDP_APP_SESSION_T](#) appHandle, UINT16 *pNumRed, [TRDP_RED_STATISTICS_T](#) *pStatistics)
Return redundancy group statistics.
- EXT_DECL [TRDP_ERR_T tlc_getJoinStatistics](#) ([TRDP_APP_SESSION_T](#) appHandle, UINT16 *pNumJoin, UINT32 *pIpAddr)
Return join statistics.
- EXT_DECL [TRDP_ERR_T tlc_resetStatistics](#) ([TRDP_APP_SESSION_T](#) appHandle)
Reset statistics.

5.23.1 Detailed Description

TRDP Light interface functions (API)

Low level functions for communicating using the TRDP protocol

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013-2019. All rights reserved.

5.23.2 Function Documentation

5.23.2.1 `tlc_closeSession()`

```
EXT_DECL TRDP_ERR_T tlc_closeSession (
    TRDP_APP_SESSION_T appHandle )
```

Close a session.

Clean up and release all resources of that session

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
----	------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	handle NULL

5.23.2.2 `tlc_configSession()`

```
EXT_DECL TRDP_ERR_T tlc_configSession (
    TRDP_APP_SESSION_T appHandle,
```

```

const TRDP_MARSHALL_CONFIG_T * pMarshall,
const TRDP_PD_CONFIG_T * pPdDefault,
const TRDP_MD_CONFIG_T * pMdDefault,
const TRDP_PROCESS_CONFIG_T * pProcessConfig )

```

(Re-)configure a session.

tlc_configSession is called by openSession, but may also be called later on to change the defaults. Only the supplied settings (pointer != NULL) will be evaluated.

Parameters

in	<i>appHandle</i>	A handle for further calls to the trdp stack
in	<i>pMarshall</i>	Pointer to marshalling configuration
in	<i>pPdDefault</i>	Pointer to default PD configuration
in	<i>pMdDefault</i>	Pointer to default MD configuration
in	<i>pProcessConfig</i>	Pointer to process configuration only option parameter is used here to define session behavior all other parameters are only used to feed statistics

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	not yet initied
<i>TRDP_PARAM_ERR</i>	parameter error

5.23.2.3 tlc_getETBTopoCount()

```

EXT_DECL UINT32 tlc_getETBTopoCount (
    TRDP_APP_SESSION_T appHandle )

```

Set new topocount for trainwide communication.

This value is used for validating outgoing and incoming packets only!

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
----	------------------	--

Return values

<i>etbTopoCnt</i>	
-------------------	--

5.23.2.4 tlc_getInterval()

```

EXT_DECL TRDP_ERR_T tlc_getInterval (
    TRDP_APP_SESSION_T appHandle,

```

```

    TRDP_TIME_T * pInterval,
    TRDP_FDS_T * pFileDesc,
    INT32 * pNoDesc )

```

Get the lowest time interval for PDs.

Return the maximum time interval suitable for 'select()' so that we can send due PD packets in time. If the PD send queue is empty, return zero time

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
out	<i>pInterval</i>	pointer to needed interval
in, out	<i>pFileDesc</i>	pointer to file descriptor set
out	<i>pNoDesc</i>	pointer to put no of highest used descriptors (for <code>select()</code>)

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.5 tlc_getJoinStatistics()

```

EXT_DECL TRDP_ERR_T tlc_getJoinStatistics (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pNumJoin,
    UINT32 * pIpAddr )

```

Return join statistics.

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pNumJoin</i>	Pointer to the number of joined IP Adresses
out	<i>pIpAddr</i>	Pointer to a list with the joined IP adresses

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more items than requested

5.23.2.6 tlc_getOpTrainTopoCount()

```
EXT_DECL UINT32 tlc_getOpTrainTopoCount (
    TRDP_APP_SESSION_T appHandle )
```

Set new operational train topocount for direction/orientation sensitive communication.

This value is used for validating outgoing and incoming packets only!

Parameters

in	<i>appHandle</i>	The handle returned by tlc_openSession
----	------------------	--

Return values

<i>opTrnTopoCnt</i>	New operational topocount value
---------------------	---------------------------------

5.23.2.7 tlc_getOwnIpAddress()

```
EXT_DECL TRDP_IP_ADDR_T tlc_getOwnIpAddress (
    TRDP_APP_SESSION_T appHandle )
```

Get the interface address.

Parameters

out	<i>appHandle</i>	A handle for further calls to the trdp stack
-----	------------------	--

Return values

<i>real↔ IP</i>	
---------------------	--

5.23.2.8 tlc_getPubStatistics()

```
EXT_DECL TRDP_ERR_T tlc_getPubStatistics (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pNumPub,
    TRDP_PUB_STATISTICS_T * pStatistics )
```

Return PD publish statistics.

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in, out	<i>pNumPub</i>	Pointer to the number of publishers
Generated by Doxygen	<i>pStatistics</i>	Pointer to a list with the publish statistics information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

5.23.2.9 `tlc_getRedStatistics()`

```
EXT_DECL TRDP_ERR_T tlc_getRedStatistics (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pNumRed,
    TRDP_RED_STATISTICS_T * pStatistics )
```

Return redundancy group statistics.

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pNumRed</i>	Pointer to the number of redundancy groups
out	<i>pStatistics</i>	Pointer to a list with the redundancy group information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

5.23.2.10 `tlc_getStatistics()`

```
EXT_DECL TRDP_ERR_T tlc_getStatistics (
    TRDP_APP_SESSION_T appHandle,
    TRDP_STATISTICS_T * pStatistics )
```

Return statistics.

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>pStatistics</i>	Pointer to statistics for this application session

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error

5.23.2.11 tlc_getSubsStatistics()

```
EXT_DECL TRDP_ERR_T tlc_getSubsStatistics (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pNumSubs,
    TRDP_SUBS_STATISTICS_T * pStatistics )
```

Return PD subscription statistics.

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in, out	<i>pNumSubs</i>	In: The number of subscriptions requested Out: Number of subscriptions returned
in, out	<i>pStatistics</i>	Pointer to an array with the subscription statistics information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

5.23.2.12 tlc_getVersion()

```
EXT_DECL const TRDP_VERSION_T* tlc_getVersion (
    void )
```

Return version.

Return pointer to version structure

Return values

<i>TRDP_VERSION_T</i>	
-----------------------	--

5.23.2.13 tlc_getVersionString()

```
EXT_DECL const CHAR8* tlc_getVersionString (
    void )
```

Return a human readable version representation.

Return string in the form 'v.r.u.b'

Return values

<i>const</i>	string
--------------	--------

5.23.2.14 tlc_init()

```
EXT_DECL TRDP_ERR_T tlc_init (
    const TRDP_PRINT_DBG_T pPrintDebugString,
    void * pRefCon,
    const TRDP_MEM_CONFIG_T * pMemConfig )
```

Support for message data can only be excluded during compile time!

Support for message data can only be excluded during compile time!

tlc_init initializes the memory subsystem and takes a function pointer to an output function for logging.

Parameters

in	<i>pPrintDebugString</i>	Pointer to debug print function
in	<i>pRefCon</i>	user context
in	<i>pMemConfig</i>	Pointer to memory configuration

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	memory allocation failed
<i>TRDP_PARAM_ERR</i>	initialization error

5.23.2.15 tlc_openSession()

```
EXT_DECL TRDP_ERR_T tlc_openSession (
    TRDP_APP_SESSION_T * pAppHandle,
    TRDP_IP_ADDR_T ownIpAddr,
    TRDP_IP_ADDR_T leaderIpAddr,
    const TRDP_MARSHALL_CONFIG_T * pMarshall,
    const TRDP_PD_CONFIG_T * pPdDefault,
```



```
const TRDP_MD_CONFIG_T * pMdDefault,
const TRDP_PROCESS_CONFIG_T * pProcessConfig )
```

Open a session with the TRDP stack.

tlc_openSession returns in pAppHandle a unique handle to be used in further calls to the stack.

Parameters

out	<i>pAppHandle</i>	A handle for further calls to the trdp stack
in	<i>ownIpAddr</i>	Own IP address, can be different for each process in multihoming systems, if zero, the default interface / IP will be used.
in	<i>leaderIpAddr</i>	IP address of redundancy leader
in	<i>pMarshall</i>	Pointer to marshalling configuration
in	<i>pPdDefault</i>	Pointer to default PD configuration
in	<i>pMdDefault</i>	Pointer to default MD configuration
in	<i>pProcessConfig</i>	Pointer to process configuration only option parameter is used here to define session behavior all other parameters are only used to feed statistics

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	not yet initied
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP SOCK_ERR</i>	socket error

5.23.2.16 tlc_presetIndexSession()

```
EXT_DECL TRDP_ERR_T tlc_presetIndexSession (
    TRDP_APP_SESSION_T appHandle,
    TRDP_IDX_TABLE_T * pIndexTableSizes )
```

Preset the index table sizes of a session.

tlc_presetIndexSession allows to preallocate the table sizes in HIGH_PERF_INDEXED mode. If no table sizes are provided, the default sizes are used. In normal mode, this is a no-op. This function should be called during initialisation stage, e.g. right after a session has been opened.

Parameters

in	<i>appHandle</i>	The handle returned by tlc_openSession
in	<i>pIndexTableSizes</i>	Pointer to a table of sizes to reserve the memory

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	not yet initied
<i>TRDP_PARAM_ERR</i>	parameter error

5.23.2.17 tlc_process()

```
EXT_DECL TRDP_ERR_T tlc_process (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FDS_T * pRfds,
    INT32 * pCount )
```

Work loop of the TRDP handler.

Search the queue for pending PDs and MDs to be sent Search the receive queue for pending PDs and MDs (time out)

Note: If using `tlc_process()`, do not use `tlp_process*()` and `tlm_process()` calls at the same time! Single thread usage -> use `tlc_getInterval()`, `vos_select()`, `tlc_process()` Multiple threads -> thread 1: use `tlp_getInterval()`, `vos_select()`, `tlp_processReceive()` -> thread 2: cyclically call `tlp_processSend()` -> thread 3: use `tlm_getInterval()`, `vos_select()`, `tlm_process()` for message data

Also see User Manual.

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.18 tlc_reinitSession()

```
EXT_DECL TRDP_ERR_T tlc_reinitSession (
    TRDP_APP_SESSION_T appHandle )
```

Re-Initialize.

Should be called by the application when a link-down/link-up event has occurred during normal operation. We need to re-join the multicast groups...

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
----	------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

Return values

<i>TRDP_PARAM_ERR</i>	handle NULL
-----------------------	-------------

5.23.2.19 tlc_resetStatistics()

```
EXT_DECL TRDP_ERR_T tlc_resetStatistics (
    TRDP_APP_SESSION_T appHandle )
```

Reset statistics.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
----	------------------	--

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error

5.23.2.20 tlc_setETBTopoCount()

```
EXT_DECL TRDP_ERR_T tlc_setETBTopoCount (
    TRDP_APP_SESSION_T appHandle,
    UINT32 etbTopoCnt )
```

Set new topocount for trainwide communication.

This value is used for validating outgoing and incoming packets only!

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>etbTopoCnt</i>	New etbTopoCnt value

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.21 `tlc_setOpTrainTopoCount()`

```
EXT_DECL TRDP_ERR_T tlc_setOpTrainTopoCount (
    TRDP_APP_SESSION_T appHandle,
    UINT32 opTrnTopoCnt )
```

Set new operational train topocount for direction/orientation sensitive communication.

This value is used for validating outgoing and incoming packets only!

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
in	<i>opTrnTopoCnt</i>	New operational topocount value

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.22 `tlc_terminate()`

```
EXT_DECL TRDP_ERR_T tlc_terminate (
    void )
```

Un-Initialize.

Clean up and close all sessions. Mainly used for debugging/test runs. No further calls to library allowed

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	TrafficStore nothing
<i>TRDP_MUTEX_ERR</i>	TrafficStore mutex err

5.23.2.23 `tlc_updateSession()`

```
EXT_DECL TRDP_ERR_T tlc_updateSession (
    TRDP_APP_SESSION_T appHandle )
```

Update a session.

`tlc_updateSession` signals the end of the set-up phase to the stack. It shall be called after the last publisher and subscriber was added and will create the index tables to be used by the high-performance targets. This function is currently a no-op on standard targets.

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
----	------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	not yet initied
<i>TRDP_PARAM_ERR</i>	parameter error

5.23.2.24 tlm_abortSession()

```
EXT_DECL TRDP_ERR_T tlm_abortSession (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId )
```

Cancel an open session.

Abort an open session; any pending messages will be dropped

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pSessionId</i>	Session ID returned by request

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOSESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.25 tlm_addListener()

```
EXT_DECL TRDP_ERR_T tlm_addListener (
    TRDP_APP_SESSION_T appHandle,
    TRDP_LIS_T * pListenHandle,
    const void * pUserRef,
    TRDP_MD_CALLBACK_T pCbFunction,
    BOOL8 comIdListener,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr1,
    TRDP_IP_ADDR_T srcIpAddr2,
    TRDP_IP_ADDR_T mcDestIpAddr,
```

```

TRDP_FLAGS_T pktFlags,
const TRDP_URI_USER_T srcURI,
const TRDP_URI_USER_T destURI )

```

Subscribe to MD messages.

Add a listener to TRDP to get notified when messages are received

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>pListenHandle</i>	Handle for this listener returned
in	<i>pUserRef</i>	user supplied value returned with received message
in	<i>pfCbFunction</i>	Pointer to listener specific callback function, NULL to use default function
in	<i>comIdListener</i>	set TRUE if <code>comId</code> shall be observed
in	<i>comId</i>	<code>comId</code> to be observed
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr1</i>	Source IP address, lower address in case of address range, set to 0 if not used
in	<i>srcIpAddr2</i>	upper address in case of address range, set to 0 if not used
in	<i>mcDestIpAddr</i>	multicast group to listen on
in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_MARSHALL
in	<i>srcURI</i>	only functional group of source URI, set to NULL if not used
in	<i>destURI</i>	only functional group of destination URI, set to NULL if not used

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.26 tlm_confirm()

```

EXT_DECL TRDP_ERR_T tlm_confirm (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId,
    UINT16 userStatus,
    const TRDP_SEND_PARAM_T * pSendParam )

```

Initiate sending MD confirm message.

Send a MD confirmation message User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

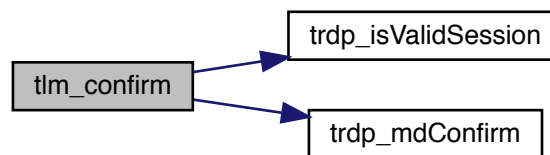
Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pSessionId</i>	Session ID returned by request
in	<i>userStatus</i>	Info for requester about application errors
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOSESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:



5.23.2.27 tlm_delListener()

```
EXT_DECL TRDP_ERR_T tlm_delListener (
    TRDP_APP_SESSION_T appHandle,
    TRDP_LIS_T listenHandle )
```

Remove Listener.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>listenHandle</i>	Handle for this listener

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.28 tlm_getInterval()

```
EXT_DECL TRDP_ERR_T tlm_getInterval (
    TRDP_APP_SESSION_T appHandle,
```

```

TRDP_TIME_T * pInterval,
TRDP_FDS_T * pFileDesc,
INT32 * pNoDesc )

```

Get the lowest time interval for MDs.

Return the maximum time interval suitable for 'select()' so that we can report time outs to the higher layer.

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
out	<i>pInterval</i>	pointer to needed interval
in, out	<i>pFileDesc</i>	pointer to file descriptor set
out	<i>pNoDesc</i>	pointer to put no of highest used descriptors (for <code>select()</code>)

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.29 tlm_notify()

```

EXT_DECL TRDP_ERR_T tlm_notify (
    TRDP_APP_SESSION_T appHandle,
    const void * pUserRef,
    TRDP_MD_CALLBACK_T pfCbFunction,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr,
    TRDP_IP_ADDR_T destIpAddr,
    TRDP_FLAGS_T pktFlags,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize,
    const TRDP_URI_USER_T sourceURI,
    const TRDP_URI_USER_T destURI )

```

Initiate sending MD notification message.

Send a MD notification message

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pUserRef</i>	user supplied value returned with reply
in	<i>pfCbFunction</i>	Pointer to listener specific callback function, NULL to use default function
in	<i>comId</i>	comId of packet to be sent
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack

Parameters

in	<i>destIpAddr</i>	where to send the packet to
in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
in	<i>pSendParam</i>	optional pointer to send parameter, NULL - default parameters are used
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data
in	<i>sourceURI</i>	only functional group of source URI
in	<i>destURI</i>	only functional group of destination URI

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.30 tlm_process()

```
EXT_DECL TRDP_ERR_T tlm_process (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FDS_T * pRfds,
    INT32 * pCount )
```

Message Data Work loop of the TRDP handler.

Search the queue for pending MDs to be sent Search the receive queue for pending MDs (replies, time outs) and incoming requests

Parameters

in	<i>appHandle</i>	The handle returned by tlc_openSession
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.31 tlm_readdListener()

```
EXT_DECL TRDP_ERR_T tlm_readdListener (
    TRDP_APP_SESSION_T appHandle,
```

```

TRDP_LIS_T listenHandle,
UINT32 etbTopoCnt,
UINT32 opTrnTopoCnt,
TRDP_IP_ADDR_T srcIpAddr1,
TRDP_IP_ADDR_T srcIpAddr2,
TRDP_IP_ADDR_T mcDestIpAddr )

```

Resubscribe to MD messages.

Readd a listener after topoCount changes to get notified when messages are received

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>listenHandle</i>	Handle for this listener
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr1</i>	Source IP address, lower address in case of address range, set to 0 if not used
in	<i>srcIpAddr2</i>	upper address in case of address range, set to 0 if not used
in	<i>mcDestIpAddr</i>	multicast group to listen on

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.32 tlm_reply()

```

TRDP_ERR_T tlm_reply (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId,
    UINT32 comId,
    UINT16 userStatus,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize )

```

Send a MD reply message.

Send a MD reply message after receiving an request User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

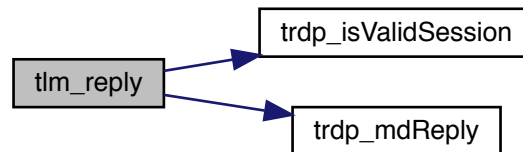
Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pSessionId</i>	Session ID returned by indication
in	<i>comId</i>	comId of packet to be sent
in	<i>userStatus</i>	Info for requester about application errors
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	Out of memory
<i>TRDP_NO_SESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:



5.23.2.33 tlm_replyQuery()

```

TRDP_ERR_T tlm_replyQuery (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId,
    UINT32 comId,
    UINT16 userStatus,
    UINT32 confirmTimeout,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize )
  
```

Send a MD reply query message.

Send a MD reply query message after receiving a request and ask for confirmation. User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

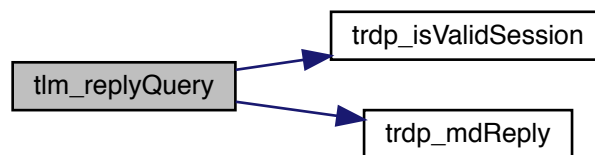
Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pSessionId</i>	Session ID returned by indication
in	<i>comId</i>	comId of packet to be sent
in	<i>userStatus</i>	Info for requester about application errors
in	<i>confirmTimeout</i>	timeout for confirmation
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NO_SESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:



5.23.2.34 tlm_request()

```

EXT_DECL TRDP_ERR_T tlm_request (
    TRDP_APP_SESSION_T appHandle,
    const void * pUserRef,
    TRDP_MD_CALLBACK_T pfCbFunction,
    TRDP_UUID_T * pSessionId,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr,
    TRDP_IP_ADDR_T destIpAddr,
    TRDP_FLAGS_T pktFlags,
    UINT32 numReplies,
    UINT32 replyTimeout,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize,
    const TRDP_URI_USER_T sourceURI,
    const TRDP_URI_USER_T destURI )
  
```

Initiate sending MD request message.

Send a MD request message

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pUserRef</i>	user supplied value returned with reply

Parameters

in	<i>pfCbFunction</i>	Pointer to listener specific callback function, NULL to use default function
out	<i>pSessionId</i>	return session ID
in	<i>comId</i>	comId of packet to be sent
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL
in	<i>numReplies</i>	number of expected replies, 0 if unknown
in	<i>replyTimeout</i>	timeout for reply
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data
in	<i>sourceURI</i>	only functional group of source URI
in	<i>destURI</i>	only functional group of destination URI

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.35 tlp_get()

```
EXT_DECL TRDP_ERR_T tlp_get (
    TRDP_APP_SESSION_T appHandle,
    TRDP_SUB_T subHandle,
    TRDP_PD_INFO_T * pPdInfo,
    UINT8 * pData,
    UINT32 * pDataSize )
```

Get the last valid PD message.

This allows polling of PDs instead of event driven handling by callbacks

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>subHandle</i>	the handle returned by subscription
in, out	<i>pPdInfo</i>	pointer to application's info buffer
in, out	<i>pData</i>	pointer to application's data buffer
in, out	<i>pDataSize</i>	in: size of buffer, out: size of data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_SUB_ERR</i>	not subscribed
<i>TRDP_TIMEOUT_ERR</i>	packet timed out
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_COMID_ERR</i>	ComID not found when marshalling

5.23.2.36 tlp_getInterval()

```
EXT_DECL TRDP_ERR_T tlp_getInterval (
    TRDP_APP_SESSION_T appHandle,
    TRDP_TIME_T * pInterval,
    TRDP_FDS_T * pFileDesc,
    INT32 * pNoDesc )
```

Get the lowest time interval for PDs.

Return the maximum time interval suitable for 'select()' so that we can send due PD packets in time. If the PD send queue is empty, return zero time

Parameters

in	<i>appHandle</i>	The handle returned by tlc_openSession
out	<i>pInterval</i>	pointer to needed interval
in, out	<i>pFileDesc</i>	pointer to file descriptor set
out	<i>pNoDesc</i>	pointer to put no of highest used descriptors (for select())

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.37 tlp_getRedundant()

```
EXT_DECL TRDP_ERR_T tlp_getRedundant (
    TRDP_APP_SESSION_T appHandle,
    UINT32 redId,
    BOOL8 * pLeader )
```

Get status of redundant ComIds.

Only the status of the first found redundancy group entry will be returned!

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>redId</i>	will be returned for all ComID's with the given redId
in, out	<i>pLeader</i>	TRUE if we're sending this redundancy group (leader)

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	redId invalid or not existing
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.38 tlp_processReceive()

```
EXT_DECL TRDP_ERR_T tlp_processReceive (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FDS_T * pRfds,
    INT32 * pCount )
```

Work loop of the TRDP handler.

Check the sockets for incoming PD telegrams. Search the receive queue for pending PDs (time out) and report them, either by informing the higher layer via the callback mechanism or just by marking the subscriber as timed-out

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.39 tlp_processSend()

```
EXT_DECL TRDP_ERR_T tlp_processSend (
    TRDP_APP_SESSION_T appHandle )
```

Work loop of the TRDP handler.

Search the queue for pending PDs to be sent

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
----	------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.40 `tlp_publish()`

```
EXT_DECL TRDP_ERR_T tlp_publish (
    TRDP_APP_SESSION_T appHandle,
    TRDP_PUB_T * pPubHandle,
    const void * pUserRef,
    TRDP_PD_CALLBACK_T pCbFunction,
    UINT32 serviceId,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr,
    TRDP_IP_ADDR_T destIpAddr,
    UINT32 interval,
    UINT32 redId,
    TRDP_FLAGS_T pktFlags,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize )
```

Prepare for sending PD messages.

Queue a PD message, it will be send when `tlc_publish` has been called

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>pPubHandle</i>	returned handle for related re/unpublish
in	<i>pUserRef</i>	user supplied value returned within the info structure of callback function
in	<i>pCbFunction</i>	Pointer to pre-send callback function, NULL if not used
in	<i>serviceId</i>	optional serviceId this telegram belongs to (default = 0)
in	<i>comId</i>	comId of packet to send
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>interval</i>	frequency of PD packet (≥ 10 ms) in usec
in	<i>redId</i>	0 - Non-redundant, > 0 valid redundancy group
in	<i>pktFlags</i>	OPTION: <code>TRDP_FLAGS_DEFAULT</code> , <code>TRDP_FLAGS_NONE</code> , <code>TRDP_FLAGS_MARSHALL</code> , <code>TRDP_FLAGS_CALLBACK</code>
in	<i>pSendParam</i>	optional pointer to send parameter, NULL - default parameters are used
in	<i>pData</i>	optional pointer to data packet / dataset, NULL if sending starts later with <code>tlp_put()</code>
in	<i>dataSize</i>	size of data packet ≥ 0 and \leq <code>TRDP_MAX_PD_DATA_SIZE</code>

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not insert (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.41 tlp_put()

```
EXT_DECL TRDP_ERR_T tlp_put (
    TRDP_APP_SESSION_T appHandle,
    TRDP_PUB_T pubHandle,
    const UINT8 * pData,
    UINT32 dataSize )
```

Update the process data to send.

Update previously published data. The new telegram will be sent earliest when tlc_process is called.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>pubHandle</i>	the handle returned by publish
in, out	<i>pData</i>	pointer to application's data buffer
in, out	<i>dataSize</i>	size of data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error on uninitialized parameter or changed dataSize compared to published one
<i>TRDP_NOPUB_ERR</i>	not published
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_COMID_ERR</i>	ComID not found when marshalling

5.23.2.42 tlp_putImmediate()

```
EXT_DECL TRDP_ERR_T tlp_putImmediate (
    TRDP_APP_SESSION_T appHandle,
    TRDP_PUB_T pubHandle,
    const UINT8 * pData,
    UINT32 dataSize,
    VOS_TIMEVAL_T * pTxTime )
```

Update and send process data.

Update previously published data. The new telegram will be sent immediatly or at txTime, if txTime != 0 and TSN == 1 Should be used if application (or higher layer, e.g. ara::com and acyclic events) needs full control over process data schedule.

Note: For TSN this function is not protected by any mutexes and should not be called while adding or removing any publishers, subscribers or even sessions! Also: Marshalling is not supported!

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>pubHandle</i>	the handle returned by publish
in, out	<i>pData</i>	pointer to application's data buffer
in, out	<i>dataSize</i>	size of data
in	<i>pTxTime</i>	when to send (absolute time), optional for TSN only

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error on uninitialized parameter or changed dataSize compared to published one
<i>TRDP_NOPUB_ERR</i>	not published
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.43 tlp_republish()

```
EXT_DECL TRDP_ERR_T tlp_republish (
    TRDP_APP_SESSION_T appHandle,
    TRDP_PUB_T pubHandle,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr,
    TRDP_IP_ADDR_T destIpAddr )
```

Prepare for sending PD messages.

Reinitialize and queue a PD message, it will be send when tlc_publish has been called

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>pubHandle</i>	handle for related unpublish
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error

Return values

<i>TRDP_MEM_ERR</i>	could not insert (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.44 tlp_request()

```

EXT_DECL TRDP_ERR_T tlp_request (
    TRDP_APP_SESSION_T appHandle,
    TRDP_SUB_T subHandle,
    UINT32 serviceId,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr,
    TRDP_IP_ADDR_T destIpAddr,
    UINT32 redId,
    TRDP_FLAGS_T pktFlags,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize,
    UINT32 replyComId,
    TRDP_IP_ADDR_T replyIpAddr )

```

Initiate sending PD messages (PULL).

Send a PD request message

Parameters

in	<i>appHandle</i>	the handle returned by <i>tlc_openSession</i>
in	<i>subHandle</i>	handle from related subscribe
in	<i>serviceId</i>	optional serviceId this telegram belongs to (default = 0)
in	<i>comId</i>	comId of packet to be sent
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>redId</i>	0 - Non-redundant, > 0 valid redundancy group
in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
in	<i>pSendParam</i>	optional pointer to send parameter, NULL - default parameters are used
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data
in	<i>replyComId</i>	comId of reply (default comID of subscription)
in	<i>replyIpAddr</i>	IP for reply

Return values

<i>TRDP_NO_ERR</i>	no error
--------------------	----------

Return values

<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not insert (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_NOSUB_ERR</i>	no matching subscription found

5.23.2.45 tlp_resubscribe()

```
EXT_DECL TRDP_ERR_T tlp_resubscribe (
    TRDP_APP_SESSION_T appHandle,
    TRDP_SUB_T subHandle,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr1,
    TRDP_IP_ADDR_T srcIpAddr2,
    TRDP_IP_ADDR_T destIpAddr )
```

Reprepare for receiving PD messages.

Resubscribe to a specific PD ComID and source IP

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>subHandle</i>	handle for this subscription
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr1</i>	Source IP address, lower address in case of address range, set to 0 if not used
in	<i>srcIpAddr2</i>	upper address in case of address range, set to 0 if not used
in	<i>destIpAddr</i>	IP address to join

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not reserve memory (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP SOCK_ERR</i>	Resource (socket) not available, subscription canceled

5.23.2.46 tlp_setRedundant()

```
EXT_DECL TRDP_ERR_T tlp_setRedundant (
    TRDP_APP_SESSION_T appHandle,
```

```
    UINT32 redId,  
    BOOL8 leader )
```

Do not send non-redundant PDs when we are follower.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>redId</i>	will be set for all ComID's with the given redId, 0 to change for all redId
in	<i>leader</i>	TRUE if we send

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error / redId not existing
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.47 `tlp_subscribe()`

```
EXT_DECL TRDP_ERR_T tlp_subscribe (
    TRDP_APP_SESSION_T appHandle,
    TRDP_SUB_T * pSubHandle,
    const void * pUserRef,
    TRDP_PD_CALLBACK_T pfCbFunction,
    UINT32 serviceId,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr1,
    TRDP_IP_ADDR_T srcIpAddr2,
    TRDP_IP_ADDR_T destIpAddr,
    TRDP_FLAGS_T pktFlags,
    const TRDP_COM_PARAM_T * pRecParams,
    UINT32 timeout,
    TRDP_TO_BEHAVIOR_T toBehavior )
```

Prepare for receiving PD messages.

Subscribe to a specific PD ComID and source IP.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>pSubHandle</i>	return a handle for this subscription
in	<i>pUserRef</i>	user supplied value returned within the info structure
in	<i>pfCbFunction</i>	Pointer to subscriber specific callback function, NULL to use default function
in	<i>serviceId</i>	optional serviceId this telegram belongs to (default = 0)
in	<i>comId</i>	comId of packet to receive
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr1</i>	Source IP address, lower address in case of address range, set to 0 if not used
in	<i>srcIpAddr2</i>	upper address in case of address range, set to 0 if not used
in	<i>destIpAddr</i>	IP address to join
in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
in	<i>pRecParams</i>	optional pointer to send parameter, NULL - default parameters are used
in	<i>timeout</i>	timeout (>= 10ms) in usec
in	<i>toBehavior</i>	timeout behavior

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not reserve memory (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.48 tlp_unpublish()

```
EXT_DECL TRDP_ERR_T tlp_unpublish (
    TRDP_APP_SESSION_T appHandle,
    TRDP_PUB_T pubHandle )
```

Stop sending PD messages.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pubHandle</i>	the handle returned by <code>prepare</code>

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_NOPUB_ERR</i>	not published
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.23.2.49 tlp_unsubscribe()

```
EXT_DECL TRDP_ERR_T tlp_unsubscribe (
    TRDP_APP_SESSION_T appHandle,
    TRDP_SUB_T subHandle )
```

Stop receiving PD messages.

Unsubscribe to a specific PD ComID

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>subHandle</i>	the handle for this subscription

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error

Return values

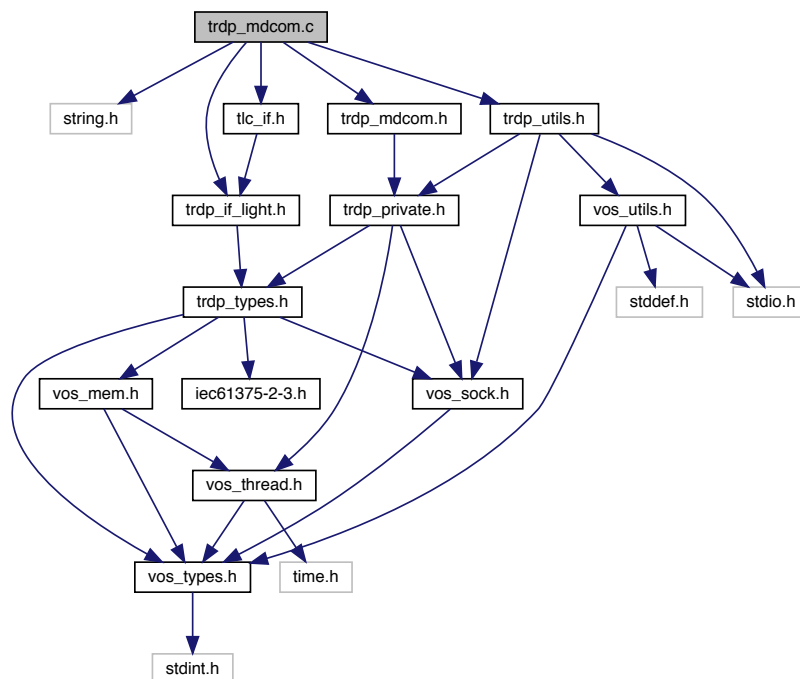
<code>TRDP_NOSUB_ERR</code>	not subscribed
<code>TRDP_NOINIT_ERR</code>	handle invalid

5.24 trdp_mdcom.c File Reference

Functions for MD communication.

```
#include <string.h>
#include "trdp_if_light.h"
#include "tlc_if.h"
#include "trdp_utils.h"
#include "trdp_mdcom.h"
```

Include dependency graph for trdp_mdcom.c:



Functions

- `TRDP_ERR_T trdp_mdGetTCPSocket (TRDP_SESSION_PT pSession)`
Initialize the specific parameters for message data Open a listening socket.
- `void trdp_mdFreeSession (MD_ELE_T *pMDSession)`
Free memory of session.
- `TRDP_ERR_T trdp_mdSend (TRDP_SESSION_PT appHandle)`
Sending MD messages Send the messages stored in the sendQueue Call user's callback if needed.

- void `trdp_mdCheckPending` (`TRDP_APP_SESSION_T` appHandle, `TRDP_FDS_T` *pFileDesc, `INT32` *pNoDesc)

Check for pending packets, set FD if non blocking.
- void `trdp_mdCheckListenSocks` (const `TRDP_SESSION_PT` appHandle, `TRDP_FDS_T` *pRfds, `INT32` *pCount)

Checking receive connection requests and data Call user's callback if needed.
- void `trdp_mdCheckTimeouts` (`TRDP_SESSION_PT` appHandle)

Checking message data timeouts Call user's callback if needed.
- `TRDP_ERR_T` `trdp_mdReply` (const `TRDP_MSG_T` msgType, `TRDP_APP_SESSION_T` appHandle, `TRDP_UUID_T` pSessionId, `UINT32` comId, `UINT32` timeout, `INT32` replyStatus, const `TRDP_SEND_PARAM_T` *pSendParam, const `UINT8` *pData, `UINT32` dataSize)

Send a MD reply/reply query message.
- `TRDP_ERR_T` `trdp_mdCall` (const `TRDP_MSG_T` msgType, `TRDP_APP_SESSION_T` appHandle, const void *pUserRef, `TRDP_MD_CALLBACK_T` pfCbFunction, `TRDP_UUID_T` *pSessionId, `UINT32` comId, `UINT32` etbTopoCnt, `UINT32` opTrnTopoCnt, `TRDP_IP_ADDR_T` srcIpAddr, `TRDP_IP_ADDR_T` destIpAddr, `TRDP_FLAGS_T` pktFlags, `UINT32` numExpReplies, `UINT32` replyTimeout, `INT32` replyStatus, const `TRDP_SEND_PARAM_T` *pSendParam, const `UINT8` *pData, `UINT32` dataSize, const `TRDP_URI_USER_T` srcURI, const `TRDP_URI_USER_T` destURI)

Initiate sending MD request message - private SW level Send a MD request message.
- `TRDP_ERR_T` `trdp_mdConfirm` (`TRDP_APP_SESSION_T` appHandle, const `TRDP_UUID_T` *pSessionId, `UINT16` userStatus, const `TRDP_SEND_PARAM_T` *pSendParam)

Initiate sending MD confirm message - private SW level Send a MD confirmation message User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session.

5.24.1 Detailed Description

Functions for MD communication.

Note

Project: TCNOpen TRDP prototype stack

Author

Simone Pachera, FARsystems Gari Oiarbide, CAF Michael Koch, Bombardier Transportations Bernd Loehr, NewTec

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013-2019. All rights reserved.

5.24.2 Function Documentation

5.24.2.1 trdp_mdCall()

```

TRDP_ERR_T trdp_mdCall (
    const TRDP_MSG_T msgType,
    TRDP_APP_SESSION_T appHandle,
    const void * pUserRef,
    TRDP_MD_CALLBACK_T pCbFunction,
    TRDP_UUID_T * pSessionId,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr,
    TRDP_IP_ADDR_T destIpAddr,
    TRDP_FLAGS_T pktFlags,
    UINT32 numExpReplies,
    UINT32 replyTimeout,
    INT32 replyStatus,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize,
    const TRDP_URI_USER_T srcURI,
    const TRDP_URI_USER_T destURI )

```

Initiate sending MD request message - private SW level Send a MD request message.

Parameters

in	<i>msgType</i>	TRDP_MSG_MN or TRDP_MSG_MR
in	<i>appHandle</i>	the handle returned by tlc_init
in	<i>pUserRef</i>	user supplied value returned with reply
in	<i>pCbFunction</i>	Pointer to listener specific callback function, NULL to use default function
out	<i>pSessionId</i>	return session ID
in	<i>comId</i>	comId of packet to be sent
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL
in	<i>numExpReplies</i>	number of expected replies, 0 if unknown
in	<i>replyTimeout</i>	timeout for reply
in	<i>replyStatus</i>	status to be returned
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data
in	<i>srcURI</i>	only functional group of source URI
in	<i>destURI</i>	only functional group of destination URI

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory

5.24.2.2 trdp_mdCheckListenSocks()

```
void trdp_mdCheckListenSocks (
    const TRDP_SESSION_PT appHandle,
    TRDP_FDS_T * pRfds,
    INT32 * pCount )
```

Checking receive connection requests and data Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

5.24.2.3 trdp_mdCheckPending()

```
void trdp_mdCheckPending (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FDS_T * pFileDesc,
    INT32 * pNoDesc )
```

Check for pending packets, set FD if non blocking.

Parameters

in	<i>appHandle</i>	session pointer
in, out	<i>pFileDesc</i>	pointer to set of ready descriptors
in, out	<i>pNoDesc</i>	pointer to number of ready descriptors

5.24.2.4 trdp_mdCheckTimeouts()

```
void trdp_mdCheckTimeouts (
    TRDP_SESSION_PT appHandle )
```

Checking message data timeouts Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
----	------------------	-----------------

5.24.2.5 trdp_mdConfirm()

```
TRDP_ERR_T trdp_mdConfirm (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId,
    UINT16 userStatus,
    const TRDP_SEND_PARAM_T * pSendParam )
```

Initiate sending MD confirm message - private SW level Send a MD confirmation message User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_init
in	<i>pSessionId</i>	Session ID returned by request
in	<i>userStatus</i>	Info for requester about application errors
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOSESSION_ERR</i>	no such session

5.24.2.6 trdp_mdFreeSession()

```
void trdp_mdFreeSession (
    MD_ELE_T * pMDSession )
```

Free memory of session.

Parameters

in	<i>pMDSession</i>	session pointer
----	-------------------	-----------------

Here is the call graph for this function:



5.24.2.7 trdp_mdGetTCPSocket()

```
TRDP_ERR_T trdp_mdGetTCPSocket (
    TRDP_SESSION_PT pSession )
```

Initialize the specific parameters for message data Open a listening socket.

Parameters

in	<i>pSession</i>	session parameters
----	-----------------	--------------------

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	initialization error

5.24.2.8 trdp_mdReply()

```
TRDP_ERR_T trdp_mdReply (
    const TRDP_MSG_T msgType,
    TRDP_APP_SESSION_T appHandle,
    TRDP_UUID_T pSessionId,
    UINT32 comId,
    UINT32 timeout,
    INT32 replyStatus,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize )
```

Send a MD reply/reply query message.

Send either a MD reply message or a MD reply query message after receiving a request and ask for confirmation. User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

Parameters

in	<i>msgType</i>	TRDP_MSG_MP or TRDP_MSG_MQ
in	<i>appHandle</i>	the handle returned by tlc_init
in	<i>pSessionId</i>	Session ID returned by indication
in	<i>comId</i>	comId of packet to be sent
in	<i>timeout</i>	time out for confirmations (zero for TRDP_MSG_MP)
in	<i>replyStatus</i>	Info for requester about application errors
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data

Return values

<i>TRDP_NO_ERR</i>	no error
--------------------	----------

Return values

<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NO_SESSION_ERR</i>	no such session

5.24.2.9 trdp_mdSend()

```
TRDP_ERR_T trdp_mdSend (
    TRDP_SESSION_PT appHandle )
```

Sending MD messages Send the messages stored in the sendQueue Call user's callback if needed.

Parameters

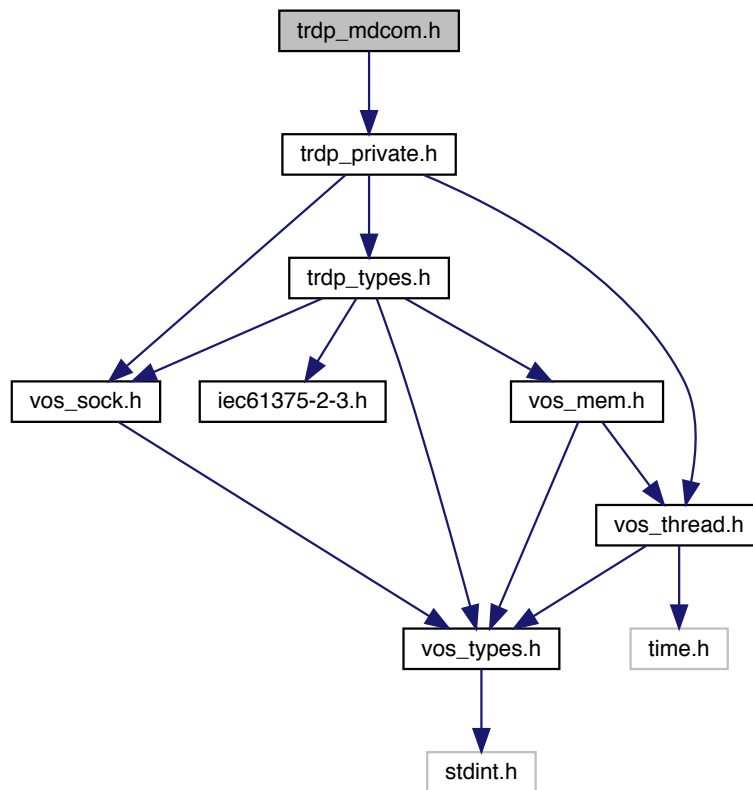
in	<i>appHandle</i>	session pointer
----	------------------	-----------------

5.25 trdp_mdcom.h File Reference

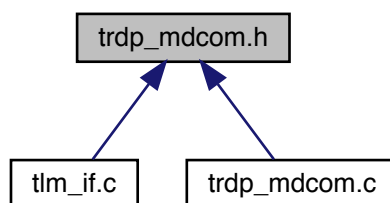
Functions for MD communication.

```
#include "trdp_private.h"
```

Include dependency graph for trdp_mdcom.h:



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



Functions

- [TRDP_ERR_T trdp_mdGetTCPSocket \(TRDP_SESSION_PT pSession\)](#)
Initialize the specific parameters for message data Open a listening socket.

- void `trdp_mdFreeSession` (MD_ELE_T *pMDSession)
Free memory of session.
- TRDP_ERR_T `trdp_mdSend` (TRDP_SESSION_PT appHandle)
Sending MD messages Send the messages stored in the sendQueue Call user's callback if needed.
- void `trdp_mdCheckPending` (TRDP_APP_SESSION_T appHandle, TRDP_FDS_T *pFileDesc, INT32 *pCount, NoDesc)
Check for pending packets, set FD if non blocking.
- void `trdp_mdCheckListenSocks` (const TRDP_SESSION_PT appHandle, TRDP_FDS_T *pRfds, INT32 *pCount)
Checking receive connection requests and data Call user's callback if needed.
- void `trdp_mdCheckTimeouts` (TRDP_SESSION_PT appHandle)
Checking message data timeouts Call user's callback if needed.
- TRDP_ERR_T `trdp_mdConfirm` (TRDP_APP_SESSION_T appHandle, const TRDP_UUID_T *pSessionId, UINT16 userStatus, const TRDP_SEND_PARAM_T *pSendParam)
Initiate sending MD confirm message - private SW level Send a MD confirmation message User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session.
- TRDP_ERR_T `trdp_mdReply` (const TRDP_MSG_T msgType, TRDP_APP_SESSION_T appHandle, TRDP_UUID_T pSessionId, UINT32 comId, UINT32 timeout, INT32 replyStatus, const TRDP_SEND_PARAM_T *pSendParam, const UINT8 *pData, UINT32 dataSize)
Send a MD reply/reply query message.
- TRDP_ERR_T `trdp_mdCall` (const TRDP_MSG_T msgType, TRDP_APP_SESSION_T appHandle, const void *pUserRef, TRDP_MD_CALLBACK_T pfCbFunction, TRDP_UUID_T *pSessionId, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, TRDP_IP_ADDR_T srcIpAddr, TRDP_IP_ADDR_T destIpAddr, TRDP_FLAGS_T pktFlags, UINT32 numExpReplies, UINT32 replyTimeout, INT32 replyStatus, const TRDP_SEND_PARAM_T *pSendParam, const UINT8 *pData, UINT32 dataSize, const TRDP_URI_USER_T srcURI, const TRDP_URI_USER_T destURI)
Initiate sending MD request message - private SW level Send a MD request message.

5.25.1 Detailed Description

Functions for MD communication.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.25.2 Function Documentation

5.25.2.1 trdp_mdCall()

```

TRDP_ERR_T trdp_mdCall (
    const TRDP_MSG_T msgType,
    TRDP_APP_SESSION_T appHandle,
    const void * pUserRef,
    TRDP_MD_CALLBACK_T pCbFunction,
    TRDP_UUID_T * pSessionId,
    UINT32 comId,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    TRDP_IP_ADDR_T srcIpAddr,
    TRDP_IP_ADDR_T destIpAddr,
    TRDP_FLAGS_T pktFlags,
    UINT32 numExpReplies,
    UINT32 replyTimeout,
    INT32 replyStatus,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize,
    const TRDP_URI_USER_T srcURI,
    const TRDP_URI_USER_T destURI )

```

Initiate sending MD request message - private SW level Send a MD request message.

Parameters

in	<i>msgType</i>	TRDP_MSG_MN or TRDP_MSG_MR
in	<i>appHandle</i>	the handle returned by tlc_init
in	<i>pUserRef</i>	user supplied value returned with reply
in	<i>pCbFunction</i>	Pointer to listener specific callback function, NULL to use default function
out	<i>pSessionId</i>	return session ID
in	<i>comId</i>	comId of packet to be sent
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL
in	<i>numExpReplies</i>	number of expected replies, 0 if unknown
in	<i>replyTimeout</i>	timeout for reply
in	<i>replyStatus</i>	status to be returned
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data
in	<i>srcURI</i>	only functional group of source URI
in	<i>destURI</i>	only functional group of destination URI

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory

5.25.2.2 trdp_mdCheckListenSocks()

```
void trdp_mdCheckListenSocks (
    const TRDP_SESSION_PT appHandle,
    TRDP_FDS_T * pRfds,
    INT32 * pCount )
```

Checking receive connection requests and data Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

5.25.2.3 trdp_mdCheckPending()

```
void trdp_mdCheckPending (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FDS_T * pFileDesc,
    INT32 * pNoDesc )
```

Check for pending packets, set FD if non blocking.

Parameters

in	<i>appHandle</i>	session pointer
in, out	<i>pFileDesc</i>	pointer to set of ready descriptors
in, out	<i>pNoDesc</i>	pointer to number of ready descriptors

5.25.2.4 trdp_mdCheckTimeouts()

```
void trdp_mdCheckTimeouts (
    TRDP_SESSION_PT appHandle )
```

Checking message data timeouts Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
----	------------------	-----------------

5.25.2.5 trdp_mdConfirm()

```
TRDP_ERR_T trdp_mdConfirm (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId,
    UINT16 userStatus,
    const TRDP_SEND_PARAM_T * pSendParam )
```

Initiate sending MD confirm message - private SW level Send a MD confirmation message User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_init
in	<i>pSessionId</i>	Session ID returned by request
in	<i>userStatus</i>	Info for requester about application errors
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOSESSION_ERR</i>	no such session

5.25.2.6 trdp_mdFreeSession()

```
void trdp_mdFreeSession (
    MD_ELE_T * pMDSession )
```

Free memory of session.

Parameters

in	<i>pMDSession</i>	session pointer
----	-------------------	-----------------

Here is the call graph for this function:



5.25.2.7 trdp_mdGetTCPSocket()

```
TRDP_ERR_T trdp_mdGetTCPSocket (
    TRDP_SESSION_PT pSession )
```

Initialize the specific parameters for message data Open a listening socket.

Parameters

in	<i>pSession</i>	session parameters
----	-----------------	--------------------

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	initialization error

5.25.2.8 trdp_mdReply()

```
TRDP_ERR_T trdp_mdReply (
    const TRDP_MSG_T msgType,
    TRDP_APP_SESSION_T appHandle,
    TRDP_UUID_T pSessionId,
    UINT32 comId,
    UINT32 timeout,
    INT32 replyStatus,
    const TRDP_SEND_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize )
```

Send a MD reply/reply query message.

Send either a MD reply message or a MD reply query message after receiving a request and ask for confirmation. User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

Parameters

in	<i>msgType</i>	TRDP_MSG_MP or TRDP_MSG_MQ
in	<i>appHandle</i>	the handle returned by tlc_init
in	<i>pSessionId</i>	Session ID returned by indication
in	<i>comId</i>	comId of packet to be sent
in	<i>timeout</i>	time out for confirmations (zero for TRDP_MSG_MP)
in	<i>replyStatus</i>	Info for requester about application errors
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data

Return values

<i>TRDP_NO_ERR</i>	no error
--------------------	----------

Return values

<code>TRDP_PARAM_ERR</code>	parameter error
<code>TRDP_MEM_ERR</code>	out of memory
<code>TRDP_NO_SESSION_ERR</code>	no such session

5.25.2.9 trdp_mdSend()

```
TRDP_ERR_T trdp_mdSend (  
    TRDP_SESSION_PT appHandle )
```

Sending MD messages Send the messages stored in the sendQueue Call user's callback if needed.

Parameters

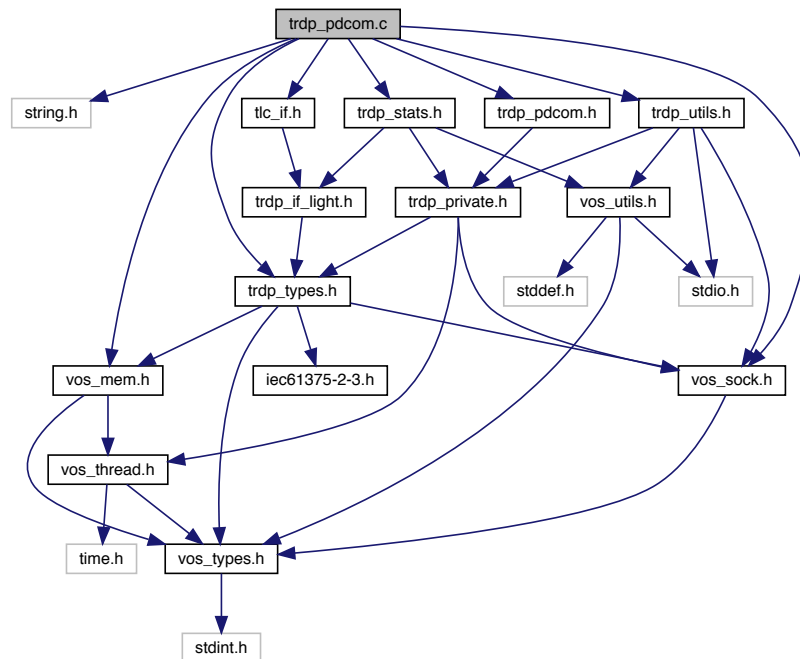
in	<code>appHandle</code>	session pointer
----	------------------------	-----------------

5.26 trdp_pdcom.c File Reference

Functions for PD communication.

```
#include <string.h>  
#include "trdp_types.h"  
#include "trdp_utils.h"  
#include "trdp_pdcom.h"  
#include "tlc_if.h"  
#include "trdp_stats.h"  
#include "vos_sock.h"  
#include "vos_mem.h"
```

Include dependency graph for trdp_pdcom.c:



Functions

- void [trdp_pdlinit](#) ([PD_ELE_T](#) *pPacket, [TRDP_MSG_T](#) type, [UINT32](#) etbTopoCnt, [UINT32](#) opTrnTopoCnt, [UINT32](#) replyComId, [UINT32](#) replyIpAddress, [UINT32](#) servId)

Initialize/construct the packet Set the header infos.
- [TRDP_ERR_T](#) [trdp_pdPut](#) ([PD_ELE_T](#) *pPacket, [TRDP_MARSHALL_T](#) marshall, void *refCon, const [UINT8](#) *pData, [UINT32](#) dataSize)

Copy data Update the data to be sent.
- [TRDP_ERR_T](#) [trdp_pdSendImmediate](#) ([TRDP_SESSION_PT](#) appHandle, [PD_ELE_T](#) *pSendPD)

Send PD message immediately.
- [TRDP_ERR_T](#) [trdp_pdGet](#) ([PD_ELE_T](#) *pPacket, [TRDP_UNMARSHALL_T](#) unmarshall, void *refCon, const [UINT8](#) *pData, [UINT32](#) *pDataSize)

Copy data Set the header infos.
- [TRDP_ERR_T](#) [trdp_pdSendElement](#) ([TRDP_SESSION_PT](#) appHandle, [PD_ELE_T](#) **ppElement)

Send a due PD message.
- [TRDP_ERR_T](#) [trdp_pdSendQueued](#) ([TRDP_SESSION_PT](#) appHandle)

Send all due PD messages.
- [TRDP_ERR_T](#) [trdp_pdReceive](#) ([TRDP_SESSION_PT](#) appHandle, [SOCKET](#) sock)

Receiving PD messages Read the receive socket for arriving PDs, copy the packet to a new PD_ELE_T Check for protocol errors and compare the received data to the data in our receive queue.
- void [trdp_pdCheckPending](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_FDS_T](#) *pFileDesc, [INT32](#) *pNoDesc, int checkSend)

Check for pending packets, set FD if non blocking.
- void [trdp_pdHandleTimeOuts](#) ([TRDP_SESSION_PT](#) appHandle)

Check for time outs.
- void [trdp_handleTimeout](#) ([TRDP_SESSION_PT](#) appHandle, [PD_ELE_T](#) *pPacket)

- Check a packet for timeout packet Call user's callback, if necessary.*
- `TRDP_ERR_T trdp_pdCheckListenSocks` (`TRDP_SESSION_PT` appHandle, `TRDP_FDS_T` *pRfds, INT32 *pCount)
- Checking receive connection requests and data Call user's callback if needed.*
- `void trdp_pdUpdate` (`PD_ELE_T` *pPacket)
- Update the header values.*
- `TRDP_ERR_T trdp_pdCheck` (`PD_HEADER_T` *pPacket, UINT32 packetSize, int *plsTSN)
- Check if the PD header values and the CRCs are sane.*
- `TRDP_ERR_T trdp_pdSend` (`SOCKET` pdSock, `PD_ELE_T` *pPacket, UINT16 port)
- Send one PD packet.*
- `TRDP_ERR_T trdp_pdDistribute` (`PD_ELE_T` *pSndQueue)
- Distribute send time of PD packets over time.*

5.26.1 Detailed Description

Functions for PD communication.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013-2019. All rights reserved.

5.26.2 Function Documentation

5.26.2.1 trdp_handleTimeout()

```
void trdp_handleTimeout (
    TRDP_SESSION_PT appHandle,
    PD_ELE_T * pPacket )
```

Check a packet for timeout packet Call user's callback, if necessary.

Parameters

in	<i>appHandle</i>	Session handle
in	<i>pPacket</i>	pointer to the packet element to check

Here is the call graph for this function:



5.26.2.2 trdp_pdCheck()

```

TRDP_ERR_T trdp_pdCheck (
    PD_HEADER_T * pPacket,
    UINT32 packetSize,
    int * pIsTSN )
  
```

Check if the PD header values and the CRCs are sane.

Parameters

in	<i>pPacket</i>	pointer to the packet to check
in	<i>packetSize</i>	max size to check
out	<i>pIsTSN</i>	set to TRUE on return if PD2 frame

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_CRC_ERR</i>	

5.26.2.3 trdp_pdCheckListenSocks()

```

TRDP_ERR_T trdp_pdCheckListenSocks (
    TRDP_SESSION_PT appHandle,
    TRDP_FDS_T * pRfds,
    INT32 * pCount )
  
```

Checking receive connection requests and data Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

5.26.2.4 trdp_pdCheckPending()

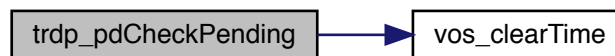
```
void trdp_pdCheckPending (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FDS_T * pFileDesc,
    INT32 * pNoDesc,
    int checkSend )
```

Check for pending packets, set FD if non blocking.

Parameters

in	<i>appHandle</i>	session pointer
in, out	<i>pFileDesc</i>	pointer to set of ready descriptors
in, out	<i>pNoDesc</i>	pointer to number of ready descriptors
in	<i>checkSend</i>	check send queue, too

Here is the call graph for this function:



5.26.2.5 trdp_pdDistribute()

```
TRDP_ERR_T trdp_pdDistribute (
    PD_ELE_T * pSndQueue )
```

Distribute send time of PD packets over time.

The duration of PD packets on a 100MBit/s network ranges from 3us to 150us max. Because a cyclic thread scheduling below 5ms would put a too heavy load on the system, and PD packets cannot get larger than 1432 (+ UDP header), we will not account for differences in packet size. Another factor is the differences in intervals for different packets: We should only change the starting times of the packets within 1/2 the interval time. Otherwise a late addition of packets could lead to timeouts of already queued packets. Scheduling will be computed based on the smallest interval time.

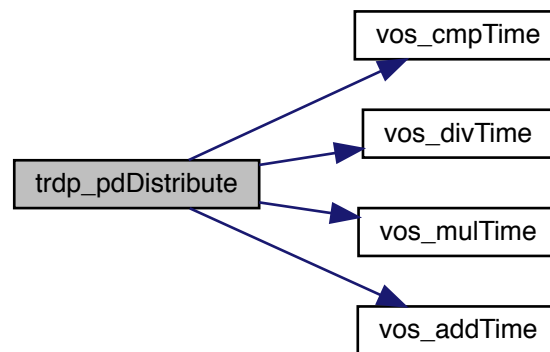
Parameters

in	<i>pSndQueue</i>	pointer to send queue
----	------------------	-----------------------

Return values

<i>TRDP_NO_ERR</i>	
--------------------	--

Here is the call graph for this function:



5.26.2.6 trdp_pdHandleTimeOuts()

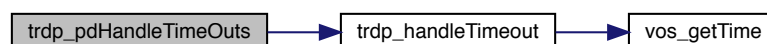
```
void trdp_pdHandleTimeOuts (
    TRDP_SESSION_PT appHandle )
```

Check for time outs.

Parameters

in	<i>appHandle</i>	application handle
----	------------------	--------------------

Here is the call graph for this function:



5.26.2.7 trdp_pdInit()

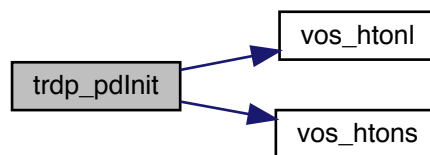
```
void trdp_pdInit (
    PD_ELEMENT * pPacket,
    TRDP_MSG_T type,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    UINT32 replyComId,
    UINT32 replyIpAddress,
    UINT32 serviceId )
```

Initialize/construct the packet Set the header infos.

Parameters

in	<i>pPacket</i>	pointer to the packet element to init
in	<i>type</i>	type the packet
in	<i>etbTopoCnt</i>	topocount to use for PD frame
in	<i>opTrnTopoCnt</i>	topocount to use for PD frame
in	<i>replyComId</i>	Pull request comId
in	<i>replyIpAddress</i>	Pull request Ip
in	<i>serviceId</i>	Service Id

Here is the call graph for this function:



5.26.2.8 trdp_pdPut()

```
TRDP_ERR_T trdp_pdPut (
    PD_ELEMENT * pPacket,
    TRDP_MARSHALL_T marshall,
    void * refCon,
    const UINT8 * pData,
    UINT32 dataSize )
```

Copy data Update the data to be sent.

Parameters

in	<i>pPacket</i>	pointer to the packet element to send
in	<i>marshall</i>	pointer to marshalling function
in	<i>refCon</i>	reference for marshalling function
in	<i>pData</i>	pointer to data
in	<i>dataSize</i>	size of data

Return values

<i>TRDP_NO_ERR</i>	no error other errors
--------------------	-----------------------

5.26.2.9 trdp_pdReceive()

```
TRDP_ERR_T trdp_pdReceive (
    TRDP_SESSION_PT appHandle,
    SOCKET sock )
```

Receiving PD messages Read the receive socket for arriving PDs, copy the packet to a new PD_ELE_T Check for protocol errors and compare the received data to the data in our receive queue.

If it is a new packet, check if it is a PD Request (PULL). If it is an update, exchange the existing entry with the new one Call user's callback if needed

Parameters

in	<i>appHandle</i>	session pointer
in	<i>sock</i>	the socket to read from

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_WIRE_ERR</i>	protocol error (late packet, version mismatch)
<i>TRDP_QUEUE_ERR</i>	not in queue
<i>TRDP_CRC_ERR</i>	header checksum
<i>TRDP_TOPOCOUNT_ERR</i>	invalid topocount

5.26.2.10 trdp_pdSend()

```
TRDP_ERR_T trdp_pdSend (
    SOCKET pdSock,
    PD_ELE_T * pPacket,
    UINT16 port )
```

Send one PD packet.

Parameters

in	<i>pdSock</i>	socket descriptor
in	<i>pPacket</i>	pointer to packet to be sent
in	<i>port</i>	port on which to send

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_IO_ERR</i>	

5.26.2.11 trdp_pdSendElement()

```
TRDP_ERR_T trdp_pdSendElement (
    TRDP_SESSION_PT appHandle,
    PD_ELE_T ** ppElement )
```

Send a due PD message.

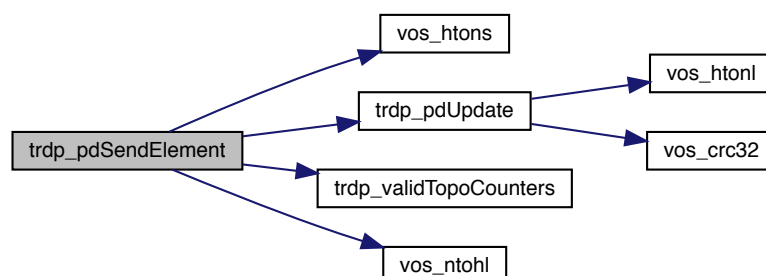
Parameters

in	<i>appHandle</i>	session pointer
in	<i>ppElement</i>	pointer to pointer of the element to send

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_IO_ERR</i>	socket I/O error

Here is the call graph for this function:



5.26.2.12 trdp_pdSendImmediate()

```
TRDP_ERR_T trdp_pdSendImmediate (
    TRDP_SESSION_PT appHandle,
    PD_ELE_T * pSendPD )
```

Send PD message immediately.

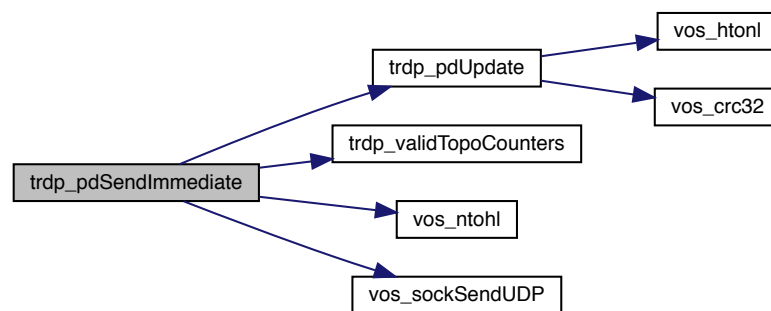
Parameters

in	<i>appHandle</i>	session pointer
in	<i>pSendPD</i>	pointer to element to be sent

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_IO_ERR</i>	socket I/O error

Here is the call graph for this function:



5.26.2.13 trdp_pdSendQueued()

```
TRDP_ERR_T trdp_pdSendQueued (
    TRDP_SESSION_PT appHandle )
```

Send all due PD messages.

Parameters

in	<i>appHandle</i>	session pointer
----	------------------	-----------------

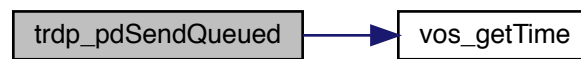
Return values

<i>TRDP_NO_ERR</i>	no error
--------------------	----------

Return values

<code>TRDP_IO_ERR</code>	socket I/O error
--------------------------	------------------

Here is the call graph for this function:



5.26.2.14 trdp_pdUpdate()

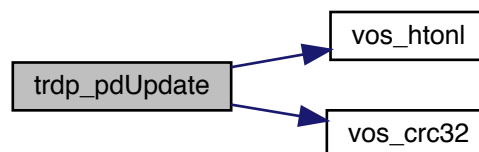
```
void trdp_pdUpdate (
    PD_ELEMENT * pPacket )
```

Update the header values.

Parameters

in	<code>pPacket</code>	pointer to the packet to update
----	----------------------	---------------------------------

Here is the call graph for this function:

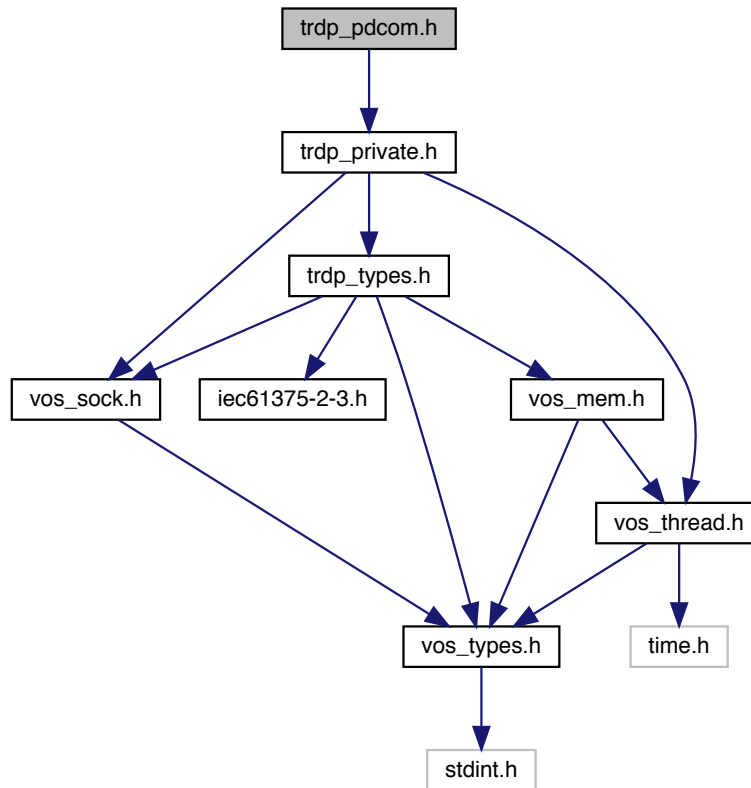


5.27 trdp_pdcom.h File Reference

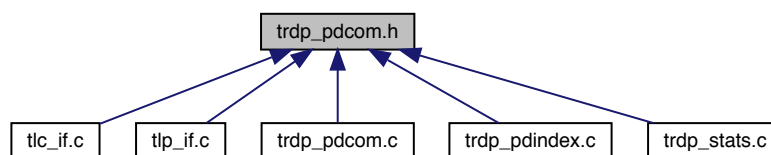
Functions for PD communication.

```
#include "trdp_private.h"
```

Include dependency graph for trdp_pdcom.h:



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



Functions

- void [trdp_pdInit](#) ([PD_ELE_T](#) *, [TRDP_MSG_T](#), [UINT32](#) topoCount, [UINT32](#) optopoCount, [UINT32](#) replyComId, [UINT32](#) replyIpAddress, [UINT32](#) serviceId)
Initialize/construct the packet Set the header infos.
- void [trdp_pdUpdate](#) ([PD_ELE_T](#) *)
Update the header values.

- [TRDP_ERR_T trdp_pdPut](#) ([PD_ELE_T](#) *, [TRDP_MARSHALL_T](#) func, void *refCon, const [UINT8](#) *pData, [UINT32](#) dataSize)
Copy data Update the data to be sent.
- [TRDP_ERR_T trdp_pdCheck](#) ([PD_HEADER_T](#) *pPacket, [UINT32](#) packetSize, int *plsTSN)
Check if the PD header values and the CRCs are sane.
- [TRDP_ERR_T trdp_pdSend](#) ([SOCKET](#) pdSock, [PD_ELE_T](#) *pPacket, [UINT16](#) port)
Send one PD packet.
- [TRDP_ERR_T trdp_pdGet](#) ([PD_ELE_T](#) *pPacket, [TRDP_UNMARSHALL_T](#) unmarshall, void *refCon, const [UINT8](#) *pData, [UINT32](#) *pDataSize)
Copy data Set the header infos.
- [TRDP_ERR_T trdp_pdSendElement](#) ([TRDP_SESSION_PT](#) appHandle, [PD_ELE_T](#) **ppElement)
Send a due PD message.
- [TRDP_ERR_T trdp_pdSendQueued](#) ([TRDP_SESSION_PT](#) appHandle)
Send all due PD messages.
- [TRDP_ERR_T trdp_pdSendImmediate](#) ([TRDP_SESSION_PT](#) appHandle, [PD_ELE_T](#) *pSendPD)
Send PD message immediately.
- [TRDP_ERR_T trdp_pdReceive](#) ([TRDP_SESSION_PT](#) pSessionHandle, [SOCKET](#) sock)
Receiving PD messages Read the receive socket for arriving PDs, copy the packet to a new PD_ELE_T Check for protocol errors and compare the received data to the data in our receive queue.
- void [trdp_pdCheckPending](#) ([TRDP_APP_SESSION_T](#) appHandle, [TRDP_FDS_T](#) *pFileDesc, [INT32](#) *pNoDesc, int checkSending)
Check for pending packets, set FD if non blocking.
- void [trdp_handleTimeout](#) ([TRDP_SESSION_PT](#) appHandle, [PD_ELE_T](#) *plterPD)
Check a packet for timeout packet Call user's callback, if necessary.
- void [trdp_pdHandleTimeOuts](#) ([TRDP_SESSION_PT](#) appHandle)
Check for time outs.
- [TRDP_ERR_T trdp_pdCheckListenSocks](#) ([TRDP_SESSION_PT](#) appHandle, [TRDP_FDS_T](#) *pRfds, [INT32](#) *pCount)
Checking receive connection requests and data Call user's callback if needed.
- [TRDP_ERR_T trdp_pdDistribute](#) ([PD_ELE_T](#) *pSndQueue)
Distribute send time of PD packets over time.

5.27.1 Detailed Description

Functions for PD communication.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013-2019. All rights reserved.

5.27.2 Function Documentation

5.27.2.1 trdp_handleTimeout()

```
void trdp_handleTimeout (
    TRDP_SESSION_PT appHandle,
    PD_ELEMENT * pPacket )
```

Check a packet for timeout packet Call user's callback, if necessary.

Parameters

in	<i>appHandle</i>	Session handle
in	<i>pPacket</i>	pointer to the packet element to check

Here is the call graph for this function:



5.27.2.2 trdp_pdCheck()

```
TRDP_ERR_T trdp_pdCheck (
    PD_HEADER_T * pPacket,
    UINT32 packetSize,
    int * pIsTSN )
```

Check if the PD header values and the CRCs are sane.

Parameters

in	<i>pPacket</i>	pointer to the packet to check
in	<i>packetSize</i>	max size to check
out	<i>pIsTSN</i>	set to TRUE on return if PD2 frame

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_CRC_ERR</i>	

5.27.2.3 trdp_pdCheckListenSocks()

```
TRDP_ERR_T trdp_pdCheckListenSocks (
    TRDP_SESSION_PT appHandle,
    TRDP_FDS_T * pRfds,
    INT32 * pCount )
```

Checking receive connection requests and data Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

5.27.2.4 trdp_pdCheckPending()

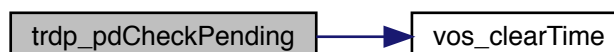
```
void trdp_pdCheckPending (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FDS_T * pFileDesc,
    INT32 * pNoDesc,
    int checkSend )
```

Check for pending packets, set FD if non blocking.

Parameters

in	<i>appHandle</i>	session pointer
in, out	<i>pFileDesc</i>	pointer to set of ready descriptors
in, out	<i>pNoDesc</i>	pointer to number of ready descriptors
in	<i>checkSend</i>	check send queue, too

Here is the call graph for this function:



5.27.2.5 trdp_pdDistribute()

```
TRDP_ERR_T trdp_pdDistribute (
    PD_ELE_T * pSndQueue )
```

Distribute send time of PD packets over time.

The duration of PD packets on a 100MBit/s network ranges from 3us to 150us max. Because a cyclic thread scheduling below 5ms would put a too heavy load on the system, and PD packets cannot get larger than 1432 (+ UDP header), we will not account for differences in packet size. Another factor is the differences in intervals for different packets: We should only change the starting times of the packets within 1/2 the interval time. Otherwise a late addition of packets could lead to timeouts of already queued packets. Scheduling will be computed based on the smallest interval time.

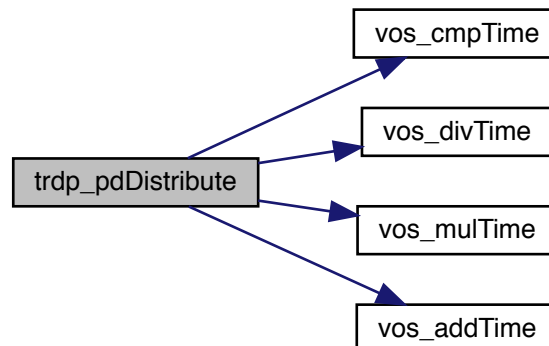
Parameters

in	<i>pSndQueue</i>	pointer to send queue
----	------------------	-----------------------

Return values

<i>TRDP_NO_ERR</i>	
--------------------	--

Here is the call graph for this function:



5.27.2.6 trdp_pdHandleTimeOuts()

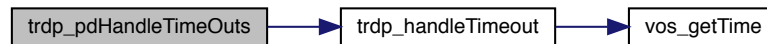
```
void trdp_pdHandleTimeOuts (
    TRDP_SESSION_PT appHandle )
```

Check for time outs.

Parameters

in	<i>appHandle</i>	application handle
----	------------------	--------------------

Here is the call graph for this function:



5.27.2.7 trdp_pdInit()

```

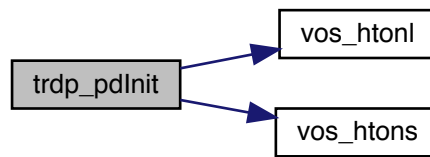
void trdp_pdInit (
    PD_ELE_T * pPacket,
    TRDP_MSG_T type,
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    UINT32 replyComId,
    UINT32 replyIpAddress,
    UINT32 serviceId )
  
```

Initialize/construct the packet Set the header infos.

Parameters

in	<i>pPacket</i>	pointer to the packet element to init
in	<i>type</i>	type the packet
in	<i>etbTopoCnt</i>	topocount to use for PD frame
in	<i>opTrnTopoCnt</i>	topocount to use for PD frame
in	<i>replyComId</i>	Pull request comId
in	<i>replyIpAddress</i>	Pull request Ip
in	<i>serviceId</i>	Service Id

Here is the call graph for this function:



5.27.2.8 trdp_pdPut()

```

TRDP_ERR_T trdp_pdPut (
    PD_ELE_T * pPacket,
    TRDP_MARSHALL_T marshall,
    void * refCon,
    const UINT8 * pData,
    UINT32 dataSize )
  
```

Copy data Update the data to be sent.

Parameters

in	<i>pPacket</i>	pointer to the packet element to send
in	<i>marshall</i>	pointer to marshalling function
in	<i>refCon</i>	reference for marshalling function
in	<i>pData</i>	pointer to data
in	<i>dataSize</i>	size of data

Return values

<i>TRDP_NO_ERR</i>	no error other errors
--------------------	-----------------------

5.27.2.9 trdp_pdReceive()

```

TRDP_ERR_T trdp_pdReceive (
    TRDP_SESSION_PT appHandle,
    SOCKET sock )
  
```

Receiving PD messages Read the receive socket for arriving PDs, copy the packet to a new PD_ELE_T Check for protocol errors and compare the received data to the data in our receive queue.

If it is a new packet, check if it is a PD Request (PULL). If it is an update, exchange the existing entry with the new one Call user's callback if needed

Parameters

in	<i>appHandle</i>	session pointer
in	<i>sock</i>	the socket to read from

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_WIRE_ERR</i>	protocol error (late packet, version mismatch)
<i>TRDP_QUEUE_ERR</i>	not in queue
<i>TRDP_CRC_ERR</i>	header checksum
<i>TRDP_TOPOCOUNT_ERR</i>	invalid topocount

5.27.2.10 trdp_pdSend()

```
TRDP_ERR_T trdp_pdSend (
    SOCKET pdSock,
    PD_ELE_T * pPacket,
    UINT16 port )
```

Send one PD packet.

Parameters

in	<i>pdSock</i>	socket descriptor
in	<i>pPacket</i>	pointer to packet to be sent
in	<i>port</i>	port on which to send

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_IO_ERR</i>	

5.27.2.11 trdp_pdSendElement()

```
TRDP_ERR_T trdp_pdSendElement (
    TRDP_SESSION_PT appHandle,
    PD_ELE_T ** ppElement )
```

Send a due PD message.

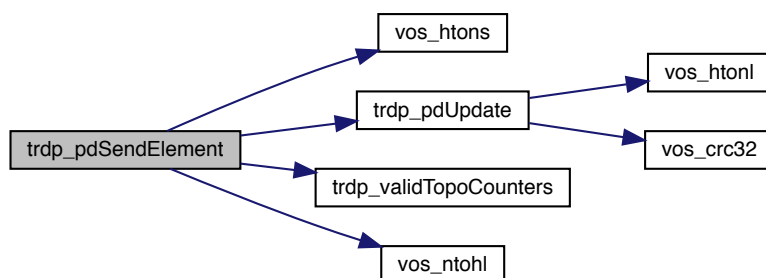
Parameters

in	<i>appHandle</i>	session pointer
in	<i>ppElement</i>	pointer to pointer of the element to send

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_IO_ERR</i>	socket I/O error

Here is the call graph for this function:

5.27.2.12 `trdp_pdSendImmediate()`

```

TRDP_ERR_T trdp_pdSendImmediate (
    TRDP_SESSION_PT appHandle,
    PD_ELE_T * pSendPD )
  
```

Send PD message immediately.

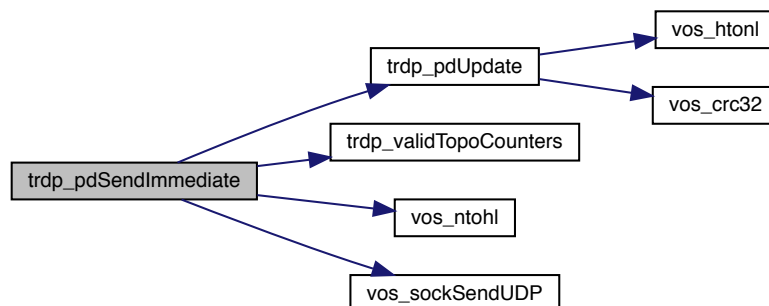
Parameters

in	<i>appHandle</i>	session pointer
in	<i>pSendPD</i>	pointer to element to be sent

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_IO_ERR</i>	socket I/O error

Here is the call graph for this function:



5.27.2.13 trdp_pdSendQueued()

```
TRDP_ERR_T trdp_pdSendQueued (
    TRDP_SESSION_PT appHandle )
```

Send all due PD messages.

Parameters

in	<i>appHandle</i>	session pointer
----	------------------	-----------------

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_IO_ERR</i>	socket I/O error

Here is the call graph for this function:



5.27.2.14 trdp_pdUpdate()

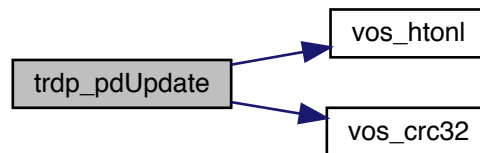
```
void trdp_pdUpdate (
    PD_ELEMENT * pPacket )
```

Update the header values.

Parameters

in	<i>pPacket</i>	pointer to the packet to update
----	----------------	---------------------------------

Here is the call graph for this function:

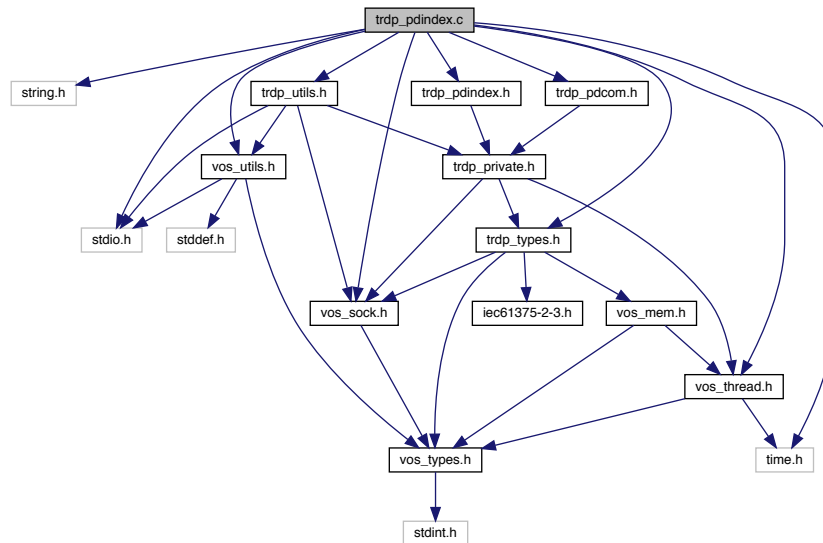


5.28 trdp_pdindex.c File Reference

Functions for indexed PD communication.

```
#include <string.h>
#include <stdio.h>
#include <time.h>
#include "trdp_types.h"
#include "trdp_utils.h"
#include "trdp_pdcom.h"
#include "vos_utils.h"
#include "vos_sock.h"
#include "vos_thread.h"
#include "trdp_pdindex.h"
```

Include dependency graph for trdp_pdindex.c:



5.28.1 Detailed Description

Functions for indexed PD communication.

Faster access to the internal process data telegram send and receive functions

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

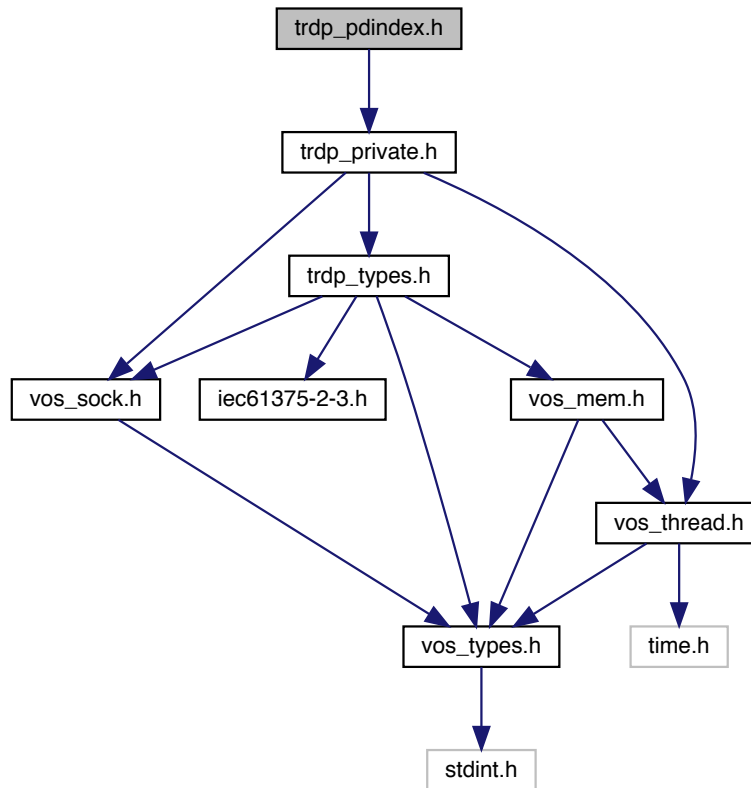
This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2019. All rights reserved.

5.29 trdp_pdindex.h File Reference

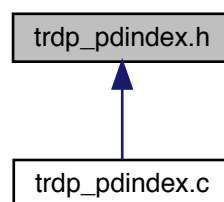
Functions for indexed PD communication.

```
#include "trdp_private.h"
```

Include dependency graph for trdp_pdindex.h:



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



Data Structures

- struct [hp_slot](#)
Low cycle-time slots.
- struct [hp_slots](#)
entry for the application session

Macros

- #define `TRDP_DEFAULT_CYCLE` 1000u
Supported and recommended cycle times for the `tlp_processTransmit` loop.
- #define `TRDP_DEFAULT_INDEX_SIZES`
Default table size settings in `HIGH_PERF_INDEXED` Mode.

Typedefs

- typedef struct `hp_slot` `TRDP_HP_CAT_SLOT_T`
Low cycle-time slots.
- typedef struct `hp_slots` `TRDP_HP_CAT_SLOTS_T`
entry for the application session

5.29.1 Detailed Description

Functions for indexed PD communication.

Faster access to the internal process data telegram send and receive functions

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2019. All rights reserved.

5.29.2 Macro Definition Documentation

5.29.2.1 `TRDP_DEFAULT_INDEX_SIZES`

```
#define TRDP_DEFAULT_INDEX_SIZES
```

Value:

```
{100,      \
           200,  \
           10,   \
           100,  \
           15,   \
           200,  \
           15,   \
           10,   \
           5,    \
           10   }
```

Default table size settings in `HIGH_PERF_INDEXED` Mode.

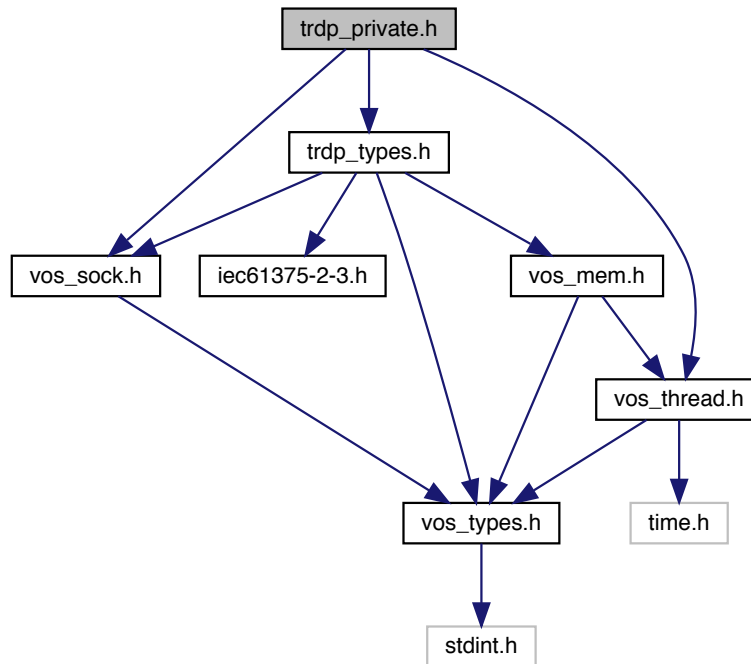
Max. number of expected publishers with intervals > 10000ms

5.30 trdp_private.h File Reference

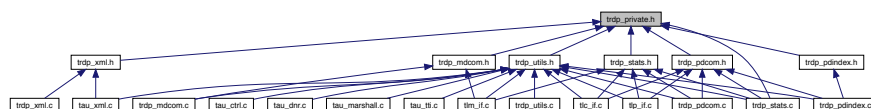
Typedefs for TRDP communication.

```
#include "trdp_types.h"
#include "vos_thread.h"
#include "vos_sock.h"
```

Include dependency graph for trdp_private.h:



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



Data Structures

- struct [TRDP_HANDLE](#)
Hidden handle definition, used as unique addressing item.
- struct [TRDP_SEQ_CNT_ENTRY_T](#)
Tuples of last received sequence counter per comId.
- struct [TRDP_SOCKET_TCP](#)
TCP parameters.

- struct [TRDP_SOCKETS](#)
Socket item.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [PD_ELE](#)
Queue element for PD packets to send or receive.
- struct [TRDP_SESSION](#)
Session/application variables store.

Macros

- #define [TRDP_TIMER_GRANULARITY](#) 5000u
granularity in us - we allow 5ms now!
- #define [TRDP_MAX_PD_SOCKET_CNT](#) VOS_MAX_SOCKET_CNT
Separate PD and MD socket lists.
- #define [TRDP_MD_MAN_CYCLE_TIME](#) 5000u
cycle time [us] = delay for outgoing MD
- #define [TRDP_DEBUG_DEFAULT_FILE_SIZE](#) 65536u
Default maximum size of log file.
- #define [TRDP_SEQ_CNT_START_ARRAY_SIZE](#) 64u
This should be enough for the start.
- #define [TRDP_IF_WAIT_FOR_READY](#) 120u
120 seconds (120 tries each second to bind to an IP address)
- #define [TRDP_PROTO_VER](#) 0x0101u
compatible protocol version with service Id
- #define [TRDP_PRIV_NONE](#) 0u
Internal flags for packets.
- #define [TRDP_TIMED_OUT](#) 0x2u
if set, inform the user
- #define [TRDP_INVALID_DATA](#) 0x4u
if set, inform the user
- #define [TRDP_REQ_2B_SENT](#) 0x8u
if set, the request needs to be sent
- #define [TRDP_REDUNDANT](#) 0x20u
if set, packet should not be sent (redundant)
- #define [TRDP_CHECK_COMID](#) 0x40u
if set, do filter comId (addListener)
- #define [TRDP_IS_TSN](#) 0x80u
if set, PD will be sent on trdp_put() only

Typedefs

- typedef struct [TRDP_HANDLE](#) [TRDP_ADDRESSES_T](#)
Hidden handle definition, used as unique addressing item.
- typedef struct [TRDP_SOCKET_TCP](#) [TRDP_SOCKET_TCP_T](#)
TCP parameters.
- typedef struct [TRDP_SOCKETS](#) [TRDP_SOCKETS_T](#)
Socket item.
- typedef struct [PD_ELE](#) [PD_ELE_T](#)
Queue element for PD packets to send or receive.
- typedef struct [TRDP_SESSION](#) [TRDP_SESSION_T](#)
Session/application variables store.

Enumerations

- enum [TRDP_MD_ELE_ST_T](#) {
[TRDP_ST_NONE](#) = 0u,
[TRDP_ST_TX_NOTIFY_ARM](#) = 1u,
[TRDP_ST_TX_REQUEST_ARM](#) = 2u,
[TRDP_ST_TX_REPLY_ARM](#) = 3u,
[TRDP_ST_TX_REPLYQUERY_ARM](#) = 4u,
[TRDP_ST_TX_CONFIRM_ARM](#) = 5u,
[TRDP_ST_RX_READY](#) = 6,
[TRDP_ST_TX_REQUEST_W4REPLY](#) = 7u,
[TRDP_ST_RX_REPLYQUERY_W4C](#) = 8u,
[TRDP_ST_RX_REQ_W4AP_REPLY](#) = 9u,
[TRDP_ST_TX_REQ_W4AP_CONFIRM](#) = 10u,
[TRDP_ST_RX_REPLY_SENT](#) = 11u,
[TRDP_ST_RX_NOTIFY_RECEIVED](#) = 12u,
[TRDP_ST_TX_REPLY_RECEIVED](#) = 13u,
[TRDP_ST_RX_CONF_RECEIVED](#) = 14u }
Internal MD state.
- enum [TRDP SOCK_TYPE_T](#) {
[TRDP SOCK_INVAL](#) = 0u,
[TRDP SOCK_PD](#) = 1u,
[TRDP SOCK_MD_UDP](#) = 2u,
[TRDP SOCK_MD_TCP](#) = 3u,
[TRDP SOCK_PD_TSN](#) = 4u }
Socket usage.

5.30.1 Detailed Description

Typedefs for TRDP communication.

TRDP internal type definitions

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013-2019. All rights reserved.

5.30.2 Macro Definition Documentation

5.30.2.1 TRDP_MAX_PD_SOCKET_CNT

```
#define TRDP_MAX_PD_SOCKET_CNT VOS_MAX_SOCKET_CNT
```

Separate PD and MD socket lists.

Reserve 1/4 of sockets for MD, if supported all available sockets for PD

5.30.3 Enumeration Type Documentation

5.30.3.1 TRDP_MD_ELE_ST_T

```
enum TRDP_MD_ELE_ST_T
```

Internal MD state.

Enumerator

TRDP_ST_NONE	neutral value
TRDP_ST_TX_NOTIFY_ARM	ready to send notify MD
TRDP_ST_TX_REQUEST_ARM	ready to send request MD
TRDP_ST_TX_REPLY_ARM	ready to send reply MD
TRDP_ST_TX_REPLYQUERY_ARM	ready to send reply with confirm request MD
TRDP_ST_TX_CONFIRM_ARM	ready to send confirm MD
TRDP_ST_RX_READY	armed listener
TRDP_ST_TX_REQUEST_W4REPLY	request sent, wait for reply
TRDP_ST_RX_REPLYQUERY_W4C	reply send, with confirm request MD
TRDP_ST_RX_REQ_W4AP_REPLY	request received, wait for application reply send
TRDP_ST_TX_REQ_W4AP_CONFIRM	reply conf. rq. tx, wait for application conf send
TRDP_ST_RX_REPLY_SENT	reply sent
TRDP_ST_RX_NOTIFY_RECEIVED	notification received, wait for application to accept
TRDP_ST_TX_REPLY_RECEIVED	reply received
TRDP_ST_RX_CONF_RECEIVED	confirmation received

5.30.3.2 TRDP SOCK_TYPE_T

```
enum TRDP SOCK_TYPE_T
```

Socket usage.

Enumerator

TRDP_SOCKET_INVALID	Socket is undefined.
TRDP_SOCKET_PD	Socket is used for UDP process data.
TRDP_SOCKET_MD_UDP	Socket is used for UDP message data.
TRDP_SOCKET_MD_TCP	Socket is used for TCP message data.
TRDP_SOCKET_PD_TSN	Socket is used for TSN process data.

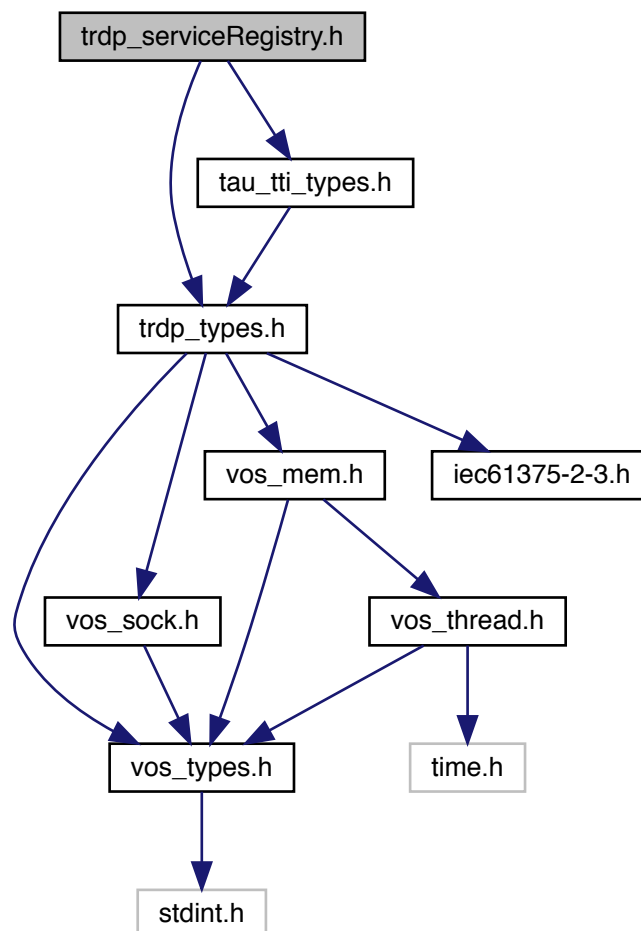
5.31 trdp_serviceRegistry.h File Reference

Additional definitions for IEC 61375-2-3 (Service Discovery) The definitions herein are preliminary and will change with the next major release of the IEC 61375-2-3 standard.

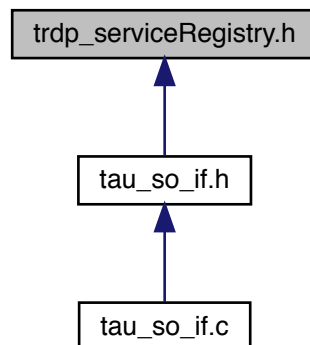
```
#include "trdp_types.h"
```

```
#include "tau_tti_types.h"
```

Include dependency graph for trdp_serviceRegistry.h:



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



Data Structures

- struct [service_info](#)
Preliminary definition of a service info entry.
- struct [srv_info_req](#)
Preliminary definition of a service info request.
- struct [GNU_PACKED](#)
Types for ETB control.

Macros

- #define [SRM_SRVINFO_NOTIFY_COMID](#) 200u
Additional defines to be reserved for SR Manager.
- #define [SRM_SRVINFO_NOTIFY_URI](#) "grpSRM.anyVeh.aCst.aClTrn.ITrn"
multicast group
- #define [SRM_SRVINFO_NOTIFY_DS](#) "CST_SRV_INFO"
SRM_CST_SRV_INFO_T.
- #define [SRM_SRV_REQ_NOTIFY_COMID](#) 201u
SRVINFOREQ request data:
- #define [SRM_SRV_REQ_NOTIFY_URI](#) "grpSRM.anyVeh.aCst.aClTrn.ITrn"
multicast group
- #define [SRM_SRV_REQ_NOTIFY_DS](#) "SRV_INFO_REQ"
SRM_SRV_INFO_REQ_T.
- #define [SRM_SERVICE_READ_REQ_COMID](#) 112u
Additional COMIDs to be reserved for SR Manager.
- #define [SRM_SERVICE_READ_REQ_TO](#) 3000000u
[us] 3s timeout
- #define [SRM_SERVICE_READ_REP_COMID](#) 113u
MD reply.
- #define [SRM_SERVICE_READ_REP_DS](#) "SRM_SERVICE_ENTRIES_T"

- `SRM_SERVICE_ENTRIES_T.`
- `#define SRM_SERVICE_READ_REQ_DSID SRM_SERVICE_DSID`
`SRM_SERVICE_ENTRIES_T.`
- `#define SRM_SERVICE_ADD_REQ_COMID 114u`
SRM manager telegram MD: Add service instance(s) to the Service Registry.
- `#define SRM_SERVICE_ADD_REQ_TO 3000000u`
[us] 3s timeout
- `#define SRM_SERVICE_ADD_REQ_DS "SRM_SERVICE_ENTRIES_T"`
`SRM_SERVICE_ENTRIES_T.`
- `#define SRM_SERVICE_ADD_REQ_DSID SRM_SERVICE_DSID`
`SRM_SERVICE_ENTRIES_T.`
- `#define SRM_SERVICE_ADD_REQ_COMID 115u`
Reply returns instanceld.
- `#define SRM_SERVICE_ADD_REQ_DSID SRM_SERVICE_DSID`
`SRM_SERVICE_ENTRIES_T.`
- `#define SRM_SERVICE_UPD_NOTIFY_COMID 116u`
SRM manager telegram MD: Update service instance(s) to the Service Registry.
- `#define SRM_SERVICE_UPD_NOTIFY_TTL 3000000u`
[us] default time-to-live
- `#define SRM_SERVICE_UPD_NOTIFY_DS "SRM_SERVICE_ENTRIES_T"`
`SRM_SERVICE_ENTRIES_T.`
- `#define SRM_SERVICE_UPD_NOTIFY_DSID SRM_SERVICE_DSID`
`SRM_SERVICE_ENTRIES_T.`
- `#define SRM_SERVICE_DEL_REQ_COMID 117u`
SRM manager telegram MD: Remove Service instance(s) from the Service Registry.
- `#define SRM_SERVICE_DEL_REQ_TO 3000000u`
[us] 3s timeout
- `#define SRM_SERVICE_DEL_REQ_DS "SRM_SERVICE_ENTRIES_T"`
`SRM_SERVICE_ENTRIES_T.`
- `#define SRM_SERVICE_DEL_REQ_DSID SRM_SERVICE_DSID`
`SRM_SERVICE_ENTRIES_T.`
- `#define SRM_SERVICE_DEL_REQ_COMID 118u`
MD reply OK or not.
- `#define SOA_SERVICEID(instId, typeId) ((instId) << 24 | (typeId))`
serviceld from instanceld and typeId
- `#define SOA_TYPE(serviceld) ((serviceld) & 0xFFFFF)`
return 24 Bit service type part of serviceld
- `#define SOA_INST(serviceld) (((serviceld) >> 24) & 0xFF)`
return 8 Bit instance ID part of serviceld
- `#define SOA_SAME_SERVICEID_OR0(a, b) (((a) == 0u) || ((a) == (b)))`
return TRUE if serviceld(a) is 0 or equals the second serviceld (b)
- `#define SOA_SAME_SERVICEID(a, b) ((a) == (b))`
return TRUE if servicelds (incl.
- `#define SOA_SAME_SERVICE_TYPE(a, b) (SOA_TYPE(a) == SOA_TYPE(b))`
return TRUE if service types match

Typedefs

- `typedef struct service_info SRM_SERVICE_INFO_T`
Preliminary definition of a service info entry.
- `typedef struct srv_info_req SRM_SRV_INFO_REQ_T`
Preliminary definition of a service info request.

5.31.1 Detailed Description

Additional definitions for IEC 61375-2-3 (Service Discovery) The definitions herein are preliminary and will change with the next major release of the IEC 61375-2-3 standard.

Note

Project: CTA2 WP3 / TCNOpen TRDP

Author

Bernd Loehr, NewTec GmbH, 2019-04-08

Remarks

Copyright 2019 Bombardier Transportation & NewTec GmbH

5.31.2 Macro Definition Documentation

5.31.2.1 SOA_SAME_SERVICEID

```
#define SOA_SAME_SERVICEID(  
    a,  
    b ) ((a) == (b))
```

return TRUE if serviceIds (incl.

instance) match

5.31.2.2 SRM_SERVICE_READ_REQ_COMID

```
#define SRM_SERVICE_READ_REQ_COMID 112u
```

Additional COMIDs to be reserved for SR Manager.

Transport: MD over TCP preferred for reliability SRM manager telegram MD: Read Services from the Consist-local Service Registry

5.31.2.3 SRM_SRVINFO_NOTIFY_COMID

```
#define SRM_SRVINFO_NOTIFY_COMID 200u
```

Additional defines to be reserved for SR Manager.

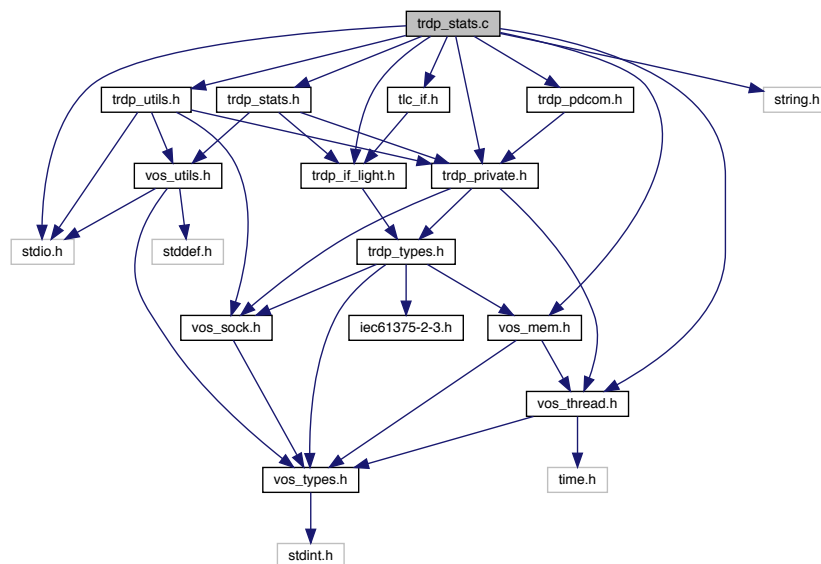
Transport: Trainwide MD over UDP / Multicast SRVINFO notification data:

5.32 trdp_stats.c File Reference

Statistics functions for TRDP communication.

```
#include <stdio.h>
#include <string.h>
#include "trdp_stats.h"
#include "trdp_if_light.h"
#include "tlc_if.h"
#include "trdp_private.h"
#include "trdp_pdcom.h"
#include "trdp_utils.h"
#include "vos_mem.h"
#include "vos_thread.h"
```

Include dependency graph for trdp_stats.c:



Functions

- void [trdp_UpdateStats](#) (TRDP_APP_SESSION_T appHandle)
Update the statistics.
- void [trdp_initStats](#) (TRDP_APP_SESSION_T appHandle)
Init statistics.
- EXT_DECL [TRDP_ERR_T tlc_resetStatistics](#) (TRDP_APP_SESSION_T appHandle)
Reset statistics.
- EXT_DECL [TRDP_ERR_T tlc_getStatistics](#) (TRDP_APP_SESSION_T appHandle, TRDP_STATISTICS_T *pStatistics)
Return statistics.
- EXT_DECL [TRDP_ERR_T tlc_getSubsStatistics](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pNumSubs, TRDP_SUBS_STATISTICS_T *pStatistics)
Return PD subscription statistics.
- EXT_DECL [TRDP_ERR_T tlc_getPubStatistics](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pNumPub, TRDP_PUB_STATISTICS_T *pStatistics)

Return PD publish statistics.

- EXT_DECL `TRDP_ERR_T tlc_getRedStatistics (TRDP_APP_SESSION_T appHandle, UINT16 *pNumRed, TRDP_RED_STATISTICS_T *pStatistics)`

Return redundancy group statistics.

- EXT_DECL `TRDP_ERR_T tlc_getJoinStatistics (TRDP_APP_SESSION_T appHandle, UINT16 *pNumJoin, UINT32 *pIpAddr)`

Return join statistics.

- void `trdp_pdPrepareStats (TRDP_APP_SESSION_T appHandle, PD_ELE_T *pPacket)`

Fill the statistics packet.

5.32.1 Detailed Description

Statistics functions for TRDP communication.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.32.2 Function Documentation

5.32.2.1 tlc_getJoinStatistics()

```
EXT_DECL TRDP_ERR_T tlc_getJoinStatistics (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pNumJoin,
    UINT32 * pIpAddr )
```

Return join statistics.

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pNumJoin</i>	Pointer to the number of joined IP Addresses
out	<i>pIpAddr</i>	Pointer to a list with the joined IP addresses

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more items than requested

5.32.2.2 `tlc_getPubStatistics()`

```
EXT_DECL TRDP_ERR_T tlc_getPubStatistics (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pNumPub,
    TRDP_PUB_STATISTICS_T * pStatistics )
```

Return PD publish statistics.

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pNumPub</i>	Pointer to the number of publishers
out	<i>pStatistics</i>	Pointer to a list with the publish statistics information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

5.32.2.3 `tlc_getRedStatistics()`

```
EXT_DECL TRDP_ERR_T tlc_getRedStatistics (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pNumRed,
    TRDP_RED_STATISTICS_T * pStatistics )
```

Return redundancy group statistics.

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pNumRed</i>	Pointer to the number of redundancy groups
out	<i>pStatistics</i>	Pointer to a list with the redundancy group information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

5.32.2.4 tlc_getStatistics()

```
EXT_DECL TRDP_ERR_T tlc_getStatistics (
    TRDP_APP_SESSION_T appHandle,
    TRDP_STATISTICS_T * pStatistics )
```

Return statistics.

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
out	<i>pStatistics</i>	Pointer to statistics for this application session

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error

5.32.2.5 tlc_getSubsStatistics()

```
EXT_DECL TRDP_ERR_T tlc_getSubsStatistics (
    TRDP_APP_SESSION_T appHandle,
    UINT16 * pNumSubs,
    TRDP_SUBS_STATISTICS_T * pStatistics )
```

Return PD subscription statistics.

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in, out	<i>pNumSubs</i>	In: The number of subscriptions requested Out: Number of subscriptions returned
in, out	<i>pStatistics</i>	Pointer to an array with the subscription statistics information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

5.32.2.6 `tlc_resetStatistics()`

```
EXT_DECL TRDP\_ERR\_T tlc_resetStatistics (
    TRDP\_APP\_SESSION\_T appHandle )
```

Reset statistics.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
----	------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error

5.32.2.7 `trdp_initStats()`

```
void trdp_initStats (
    TRDP\_APP\_SESSION\_T appHandle )
```

Init statistics.

Clear the stats structure for a session.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
----	------------------	---

< leader host name Here is the call graph for this function:



5.32.2.8 trdp_pdPrepareStats()

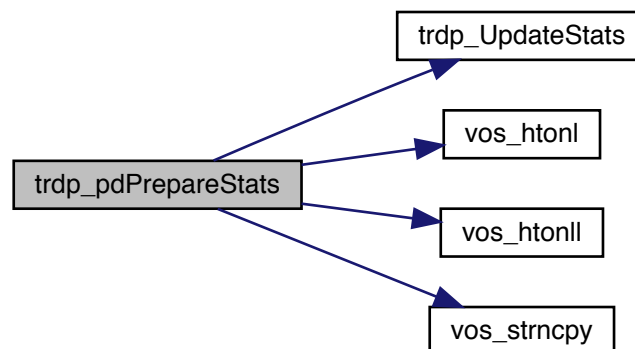
```
void trdp_pdPrepareStats (
    TRDP_APP_SESSION_T appHandle,
    PD_ELE_T * pPacket )
```

Fill the statistics packet.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pPacket</i>	pointer to the packet to fill

Here is the call graph for this function:



5.32.2.9 trdp_UpdateStats()

```
void trdp_UpdateStats (
    TRDP_APP_SESSION_T appHandle )
```

Update the statistics.

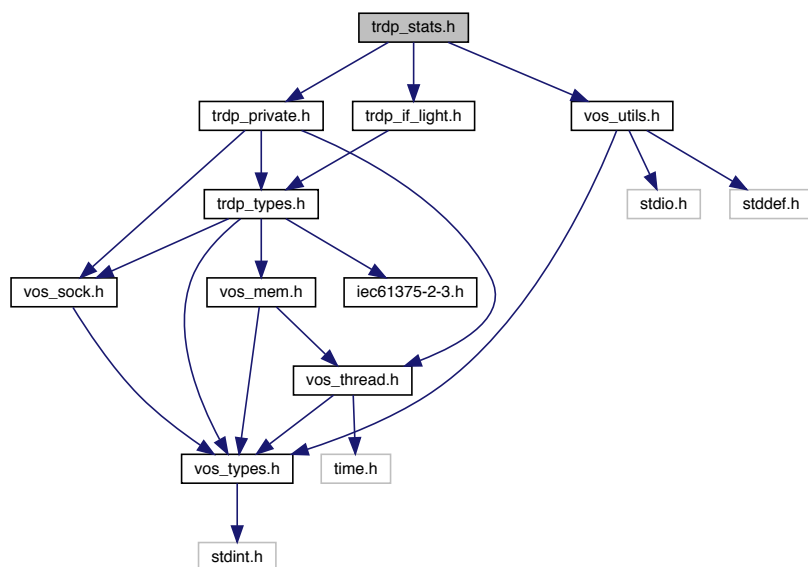
Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
----	------------------	---

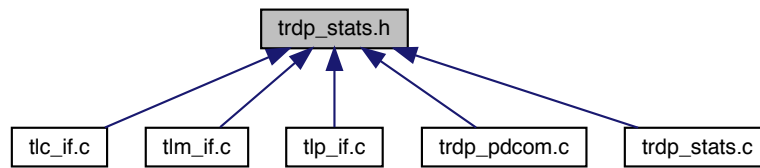
5.33 trdp_stats.h File Reference

Statistics for TRDP communication.

```
#include "trdp_if_light.h"
#include "trdp_private.h"
#include "vos_utils.h"
Include dependency graph for trdp_stats.h:
```



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



Functions

- void `trdp_initStats` (`TRDP_APP_SESSION_T` appHandle)
Init statistics.
- void `trdp_pdPrepareStats` (`TRDP_APP_SESSION_T` appHandle, `PD_ELE_T` *pPacket)
Fill the statistics packet.

5.33.1 Detailed Description

Statistics for TRDP communication.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.33.2 Function Documentation

5.33.2.1 trdp_initStats()

```
void trdp_initStats (  
    TRDP_APP_SESSION_T appHandle )
```

Init statistics.

Clear the stats structure for a session.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
----	------------------	---

< host name

< leader host name Here is the call graph for this function:

**5.33.2.2 trdp_pdPrepareStats()**

```

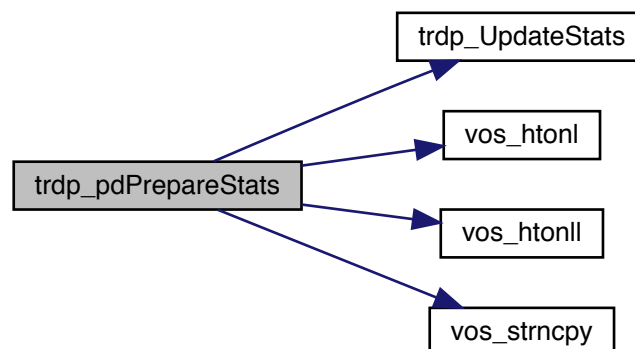
void trdp_pdPrepareStats (
    TRDP_APP_SESSION_T appHandle,
    PD_ELE_T * pPacket )
  
```

Fill the statistics packet.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pPacket</i>	pointer to the packet to fill

Here is the call graph for this function:



5.34 trdp_tsn_def.h File Reference

Additional definitions for TSN.

Macros

- `#define TRDP_MD_DEFAULT_QOS 2u`
matching new proposed priority classes
- `#define TRDP_PD_DEFAULT_QOS 2u`
Default PD communication parameters.
- `#define TRDP_PD_DEFAULT_TSN_PRIORITY 3u`
matching new proposed priority classes
- `#define TRDP_PD_DEFAULT_TSN FALSE`
matching new proposed priority classes
- `#define TRDP_MIN_PD2_HEADER_SIZE sizeof(PD2_HEADER_T)`
PD packet properties.
- `#define TRDP_MAX_PD2_DATA_SIZE 1458u`
PD2 data.
- `#define TRDP_MSG_TSN_PD 0x01u`
New Message Types for TRDP Version 2 TSN-PDU (preliminary)
- `#define TRDP_MSG_TSN_PD_SDT 0x02u`
TSN safe PD Data.
- `#define TRDP_MSG_TSN_PD_MSDT 0x03u`
TSN multiple SDT PD Data.
- `#define TRDP_MSG_TSN_PD_RES 0x04u`
TSN reserved.
- `#define TRDP_VER_TSN_PROTO 0x02u`
Protocol version for TSN.

5.34.1 Detailed Description

Additional definitions for TSN.

This header file defines proposed extensions and additions to IEC61375-2-3:2017 The definitions herein are preliminary and may change with the next major release of the IEC 61375-2-3 standard.

Note

Project: TCNOpen TRDP prototype stack & FDF/DbD

Author

Bernd Loehr, NewTec GmbH, 2019-02-19

Remarks

Copyright 2019, NewTec GmbH

Id

[trdp_tsn_def.h](#) 1932 2019-07-03 15:31:16Z bloehr

5.34.2 Macro Definition Documentation

5.34.2.1 TRDP_MIN_PD2_HEADER_SIZE

```
#define TRDP_MIN_PD2_HEADER_SIZE sizeof(PD2_HEADER_T)
```

PD packet properties.

TSN header size with FCS

5.34.2.2 TRDP_MSG_TSN_PD

```
#define TRDP_MSG_TSN_PD 0x01u
```

New Message Types for TRDP Version 2 TSN-PDU (preliminary)

TSN non safe PD Data

5.34.2.3 TRDP_PD_DEFAULT_QOS

```
#define TRDP_PD_DEFAULT_QOS 2u
```

Default PD communication parameters.

matching new proposed priority classes

5.35 trdp_types.h File Reference

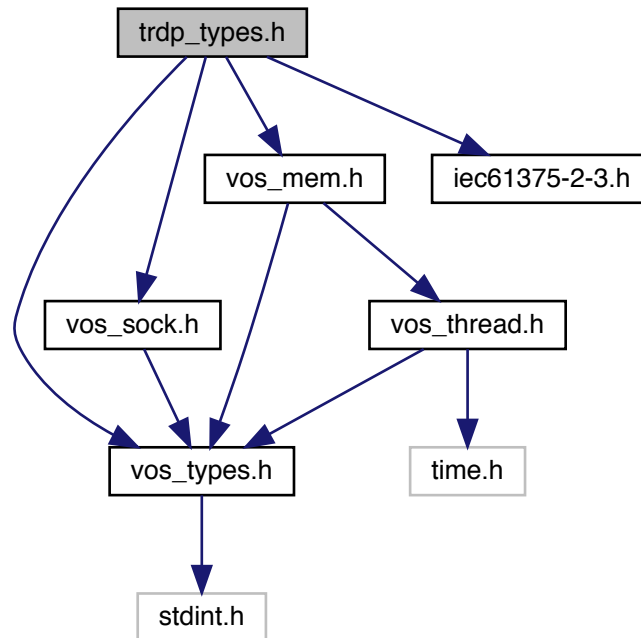
Typedefs for TRDP communication.

```
#include "vos_types.h"  
#include "vos_mem.h"  
#include "vos_sock.h"
```

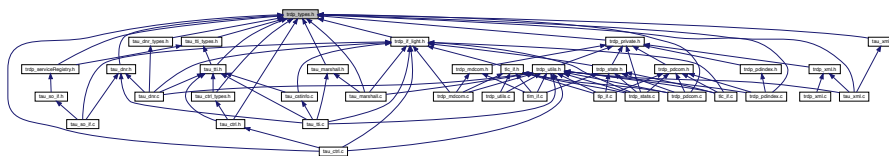


```
#include "iec61375-2-3.h"
```

Include dependency graph for trdp_types.h:



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



Data Structures

- struct [TRDP_PD_INFO_T](#)
Process data info from received telegram; allows the application to generate responses.
- struct [TRDP_MD_INFO_T](#)
Message data info from received telegram; allows the application to generate responses.
- struct [TRDP_COM_PARAM_T](#)
Quality/type of service, time to live , no.
- struct [TRDP_DATASET_ELEMENT_T](#)
Dataset element definition.
- struct [TRDP_DATASET](#)
Dataset definition.
- struct [TRDP_COMID_DSID_MAP_T](#)

- *ComId - data set mapping element definition.*
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [TRDP_MARSHALL_CONFIG_T](#)
Marshaling/unmarshalling configuration.
- struct [TRDP_PD_CONFIG_T](#)
Default PD configuration.
- struct [TRDP_MD_CONFIG_T](#)
Default MD configuration.
- struct [TRDP_MEM_CONFIG_T](#)
Enumeration type for memory pre-fragmentation, reuse of VOS definition.
- struct [TRDP_PROCESS_CONFIG_T](#)
Various flags/general TRDP options for library initialization.
- struct [TRDP_IDX_TABLE_T](#)
Settings for pre-allocation of index tables for application session initialization.

Macros

- #define [USE_HEAP](#) 0
If this is set, we can allocate dynamically memory.
- #define [TRDP_FLAGS_DEFAULT](#) 0u
Various flags for PD and MD packets.
- #define [TRDP_FLAGS_NONE](#) 0x01u
No flags set.
- #define [TRDP_FLAGS_MARSHALL](#) 0x02u
Optional marshalling/unmarshalling in TRDP stack.
- #define [TRDP_FLAGS_CALLBACK](#) 0x04u
Use of callback function.
- #define [TRDP_FLAGS_TCP](#) 0x08u
Use TCP for message data.
- #define [TRDP_FLAGS_FORCE_CB](#) 0x10u
Force a callback for every received packet.
- #define [TRDP_FLAGS_TSN](#) 0x20u
Hard Real Time PD.

- #define [TRDP_FLAGS_TSN_SDT](#) 0x40u
SDT PD.
- #define [TRDP_FLAGS_TSN_MSDT](#) 0x80u
Multi SDT PD.
- #define [TRDP_INFINITE_TIMEOUT](#) 0xffffffffu
Infinite reply timeout.
- #define [TRDP_DEFAULT_PD_TIMEOUT](#) 100000u
Default PD timeout 100ms from 61375-2-3 Table C.7.
- #define [TRDP_BOOL8](#) [TRDP_BITSET8](#)
1 bit relevant (equal to zero = false, not equal to zero = true)
- #define [TRDP_ANTIVALENT8](#) [TRDP_BITSET8](#)
2 bit relevant (0x0 = error, 0x01 = false, 0x02 = true, 0x03 undefined)
- #define [TRDP_OPTION_NONE](#) 0u
Various flags/general TRDP options for library initialization.
- #define [TRDP_OPTION_BLOCK](#) 0x01u
Default: Use nonblocking I/O calls, polling necessary Set: Read calls will block, use select()
- #define [TRDP_OPTION_TRAFFIC_SHAPING](#) 0x02u
Use traffic shaping - distribute packet sending Default: OFF.
- #define [TRDP_OPTION_NO_REUSE_ADDR](#) 0x04u
Do not allow re-use of address/port (-> no multihoming) Default: Allow.
- #define [TRDP_OPTION_NO_MC_LOOP_BACK](#) 0x08u
Do not allow loop back of multicast traffic Default: Allow.
- #define [TRDP_OPTION_NO_UDP_CHK](#) 0x10u
Suppress UDP CRC generation Default: Compute UDP CRC.
- #define [TRDP_OPTION_WAIT_FOR_DNR](#) 0x20u
Wait for DNR Default: Don't wait.
- #define [TRDP_OPTION_NO_PD_STATS](#) 0x40u
Suppress PD statistics \ Default: Don't suppress.
- #define [TRDP_OPTION_DEFAULT_CONFIG](#) 0x80u
no XML process config, defaults were used

Typedefs

- typedef [VOS_IP4_ADDR_T](#) [TRDP_IP_ADDR_T](#)
TRDP general type definitions.
- typedef [CHAR8](#) [TRDP_NET_LABEL_T](#)[[TRDP_MAX_LABEL_LEN](#)]
Definition for usage in network packets, not necessarily \0 terminated!
- typedef [VOS_VERSION_T](#) [TRDP_VERSION_T](#)
Version information.
- typedef [VOS_TIMEVAL_T](#) [TRDP_TIME_T](#)
Timer value compatible with timeval / select.
- typedef [VOS_FDS_T](#) [TRDP_FDS_T](#)
File descriptor set compatible with fd_set / select.
- typedef [VOS_UUID_T](#) [TRDP_UUID_T](#)
UUID definition reuses the VOS definition.
- typedef struct [TRDP_DATASET](#) [TRDP_DATASET_T](#)
Dataset definition.
- typedef [TRDP_DATASET_T](#) * [pTRDP_DATASET_T](#)
Array of pointers to dataset.
- typedef [VOS_PRINT_DBG_T](#) [TRDP_PRINT_DBG_T](#)

TRDP configuration type definitions.

- typedef [VOS_LOG_T](#) [TRDP_LOG_T](#)

Categories for logging, reuse of the VOS definition.

- typedef [TRDP_ERR_T](#)(* [TRDP_MARSHALL_T](#)) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDst, UINT32 *pDstSize, [TRDP_DATASET_T](#) **ppCachedDS)

Function type for marshalling .

- typedef [TRDP_ERR_T](#)(* [TRDP_UNMARSHALL_T](#)) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDst, UINT32 *pDstSize, [TRDP_DATASET_T](#) **ppCachedDS)

Function type for unmarshalling.

- typedef void(* [TRDP_PD_CALLBACK_T](#)) (void *pRefCon, [TRDP_APP_SESSION_T](#) appHandle, const [T↔RDP_PD_INFO_T](#) *pMsg, UINT8 *pData, UINT32 dataSize)

Callback for receiving indications, timeouts, releases, responses.

- typedef void(* [TRDP_MD_CALLBACK_T](#)) (void *pRefCon, [TRDP_APP_SESSION_T](#) appHandle, const [T↔RDP_MD_INFO_T](#) *pMsg, UINT8 *pData, UINT32 dataSize)

Callback for receiving indications, timeouts, releases, responses.

Enumerations

- enum [TRDP_ERR_T](#) {
[TRDP_NO_ERR](#) = 0,
[TRDP_PARAM_ERR](#) = -1,
[TRDP_INIT_ERR](#) = -2,
[TRDP_NOINIT_ERR](#) = -3,
[TRDP_TIMEOUT_ERR](#) = -4,
[TRDP_NODATA_ERR](#) = -5,
[TRDP_SOCKET_ERR](#) = -6,
[TRDP_IO_ERR](#) = -7,
[TRDP_MEM_ERR](#) = -8,
[TRDP_SEMA_ERR](#) = -9,
[TRDP_QUEUE_ERR](#) = -10,
[TRDP_QUEUE_FULL_ERR](#) = -11,
[TRDP_MUTEX_ERR](#) = -12,
[TRDP_THREAD_ERR](#) = -13,
[TRDP_BLOCK_ERR](#) = -14,
[TRDP_INTEGRATION_ERR](#) = -15,
[TRDP_NOCONN_ERR](#) = -16,
[TRDP_NOSESSION_ERR](#) = -30,
[TRDP_SESSION_ABORT_ERR](#) = -31,
[TRDP_NOSUB_ERR](#) = -32,
[TRDP_NOPUB_ERR](#) = -33,
[TRDP_NOLIST_ERR](#) = -34,
[TRDP_CRC_ERR](#) = -35,
[TRDP_WIRE_ERR](#) = -36,
[TRDP_TOPO_ERR](#) = -37,
[TRDP_COMID_ERR](#) = -38,
[TRDP_STATE_ERR](#) = -39,
[TRDP_APP_TIMEOUT_ERR](#) = -40,
[TRDP_APP_REPLYTO_ERR](#) = -41,
[TRDP_APP_CONFIRMTO_ERR](#) = -42,
[TRDP_REPLYTO_ERR](#) = -43,
[TRDP_CONFIRMTO_ERR](#) = -44,
[TRDP_REQCONFIRMTO_ERR](#) = -45,
[TRDP_PACKET_ERR](#) = -46,
[TRDP_UNRESOLVED_ERR](#) = -47,
[TRDP_XML_PARSER_ERR](#) = -48,

```

TRDP_INUSE_ERR = -49,
TRDP_MARSHALLING_ERR = -50,
TRDP_UNKNOWN_ERR = -99 }

```

Return codes for all API functions, -1..-29 taken over from vos.

- enum `TRDP_REPLY_STATUS_T`

TRDP data transfer type definitions.

- enum `TRDP_RED_STATE_T` {
`TRDP_RED_FOLLOWER` = 0u,
`TRDP_RED_LEADER` = 1u }

Redundancy states.

- enum `TRDP_TO_BEHAVIOR_T` {
`TRDP_TO_DEFAULT` = 0u,
`TRDP_TO_SET_TO_ZERO` = 1u,
`TRDP_TO_KEEP_LAST_VALUE` = 2u }

How invalid PD shall be handled.

- enum `TRDP_DATA_TYPE_T` {
`TRDP_INVALID` = 0u,
`TRDP_BITSET8` = 1u,
`TRDP_CHAR8` = 2u,
`TRDP_UTF16` = 3u,
`TRDP_INT8` = 4u,
`TRDP_INT16` = 5u,
`TRDP_INT32` = 6u,
`TRDP_INT64` = 7u,
`TRDP_UINT8` = 8u,
`TRDP_UINT16` = 9u,
`TRDP_UINT32` = 10u,
`TRDP_UINT64` = 11u,
`TRDP_REAL32` = 12u,
`TRDP_REAL64` = 13u,
`TRDP_TIMEDATE32` = 14u,
`TRDP_TIMEDATE48` = 15u,
`TRDP_TIMEDATE64` = 16u,
`TRDP_TYPE_MAX` = 30u }

TRDP dataset description definitions.

5.35.1 Detailed Description

Typedefs for TRDP communication.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2015-2019. All rights reserved.

5.35.2 Macro Definition Documentation

5.35.2.1 TRDP_FLAGS_DEFAULT

```
#define TRDP_FLAGS_DEFAULT 0u
```

Various flags for PD and MD packets.

Default value defined in `tlc_openDession` will be taken

5.35.3 Typedef Documentation

5.35.3.1 TRDP_IP_ADDR_T

```
typedef VOS_IP4_ADDR_T TRDP_IP_ADDR_T
```

TRDP general type definitions.

5.35.3.2 TRDP_MARSHALL_T

```
typedef TRDP_ERR_T(* TRDP_MARSHALL_T) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize,
UINT8 *pDst, UINT32 *pDstSize, TRDP_DATASET_T **ppCachedDS)
```

Function type for marshalling .

The function must know about the dataset's alignment etc.

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>comId</i>	ComId to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	size of the source buffer
in	<i>pDst</i>	pointer to a buffer for the treated message
in, out	<i>pDstSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppCachedDS</i>	pointer to pointer of cached dataset

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_COMID_ERR</i>	comid not existing

5.35.3.3 TRDP_MD_CALLBACK_T

```
typedef void(* TRDP_MD_CALLBACK_T) (void *pRefCon, TRDP_APP_SESSION_T appHandle, const TRDP_MESSAGE_T *pMsg,
UINT8 *pData, UINT32 dataSize)
```

Callback for receiving indications, timeouts, releases, responses.

Parameters

in	<i>appHandle</i>	handle returned also by <code>tlc_init</code>
in	<i>pRefCon</i>	pointer to user context
in	<i>pMsg</i>	pointer to received message information
in	<i>pData</i>	pointer to received data
in	<i>dataSize</i>	size of received data pointer to received data

5.35.3.4 TRDP_PD_CALLBACK_T

```
typedef void(* TRDP_PD_CALLBACK_T) (void *pRefCon, TRDP_APP_SESSION_T appHandle, const TRDP_PACKET_T *pMsg,
UINT8 *pData, UINT32 dataSize)
```

Callback for receiving indications, timeouts, releases, responses.

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>appHandle</i>	application handle returned by <code>tlc_openSession</code>
in	<i>pMsg</i>	pointer to received message information
in	<i>pData</i>	pointer to received data
in	<i>dataSize</i>	size of received data pointer to received data

5.35.3.5 TRDP_PRINT_DBG_T

```
typedef VOS_PRINT_DBG_T TRDP_PRINT_DBG_T
```

TRDP configuration type definitions.

Callback function definition for error/debug output, reuse of the VOS defined function.

5.35.3.6 TRDP_TIME_T

```
typedef VOS_TIMEVAL_T TRDP_TIME_T
```

Timer value compatible with `timeval` / `select`.

Relative or absolute date, depending on usage

5.35.3.7 TRDP_UNMARSHALL_T

```
typedef TRDP_ERR_T(* TRDP_UNMARSHALL_T) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize,
UINT8 *pDst, UINT32 *pDstSize, TRDP_DATASET_T **ppCachedDS)
```

Function type for unmarshalling.

The function must know about the dataset's alignment etc.

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>comId</i>	ComId to identify the structure out of a configuration
in	<i>pSrc</i>	pointer to received original message
in	<i>srcSize</i>	data length from TRDP packet header
in	<i>pDst</i>	pointer to a buffer for the treated message
in, out	<i>pDstSize</i>	size of the provide buffer / size of the treated message
in, out	<i>ppCachedDS</i>	pointer to pointer of cached dataset

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provide buffer to small
<i>TRDP_COMID_ERR</i>	comid not existing

5.35.4 Enumeration Type Documentation

5.35.4.1 TRDP_DATA_TYPE_T

```
enum TRDP_DATA_TYPE_T
```

TRDP dataset description definitions.

Dataset element definition

Enumerator

TRDP_INVALID	Invalid/unknown.
TRDP_BITSET8	=UINT8
TRDP_CHAR8	char, can be used also as UTF8
TRDP_UTF16	Unicode UTF-16 character.
TRDP_INT8	Signed integer, 8 bit.
TRDP_INT16	Signed integer, 16 bit.
TRDP_INT32	Signed integer, 32 bit.
TRDP_INT64	Signed integer, 64 bit.
TRDP_UINT8	Unsigned integer, 8 bit.
TRDP_UINT16	Unsigned integer, 16 bit.

Enumerator

TRDP_UINT32	Unsigned integer, 32 bit.
TRDP_UINT64	Unsigned integer, 64 bit.
TRDP_REAL32	Floating point real, 32 bit.
TRDP_REAL64	Floating point real, 64 bit.
TRDP_TIMEDATE32	32 bit UNIX time
TRDP_TIMEDATE48	48 bit TCN time (32 bit UNIX time and 16 bit ticks)
TRDP_TIMEDATE64	32 bit UNIX time + 32 bit microseconds
TRDP_TYPE_MAX	Values greater are considered nested datasets.

5.35.4.2 TRDP_ERR_T

enum [TRDP_ERR_T](#)

Return codes for all API functions, -1..-29 taken over from vos.

Enumerator

TRDP_NO_ERR	No error.
TRDP_PARAM_ERR	Parameter missing or out of range.
TRDP_INIT_ERR	Call without valid initialization.
TRDP_NOINIT_ERR	Call with invalid handle.
TRDP_TIMEOUT_ERR	Timeout.
TRDP_NODATA_ERR	Non blocking mode: no data received.
TRDP SOCK_ERR	Socket error / option not supported.
TRDP_IO_ERR	Socket IO error, data can't be received/sent.
TRDP_MEM_ERR	No more memory available.
TRDP_SEMA_ERR	Semaphore not available.
TRDP_QUEUE_ERR	Queue empty.
TRDP_QUEUE_FULL_ERR	Queue full.
TRDP_MUTEX_ERR	Mutex not available.
TRDP_THREAD_ERR	Thread error.
TRDP_BLOCK_ERR	System call would have blocked in blocking mode.
TRDP_INTEGRATION_ERR	Alignment or endianness for selected target wrong.
TRDP_NOCONN_ERR	No TCP connection.
TRDP_NOSESSION_ERR	No such session.
TRDP_SESSION_ABORT_ERR	Session aborted.
TRDP_NOSUB_ERR	No subscriber.
TRDP_NOPUB_ERR	No publisher.
TRDP_NOLIST_ERR	No listener.
TRDP_CRC_ERR	Wrong CRC.
TRDP_WIRE_ERR	Wire.
TRDP_TOPO_ERR	Invalid topo count.
TRDP_COMID_ERR	Unknown ComId.
TRDP_STATE_ERR	Call in wrong state.
TRDP_APP_TIMEOUT_ERR	Application Timeout.
TRDP_APP_REPLYTO_ERR	Application Reply Sent Timeout.

Enumerator

TRDP_APP_CONFIRMTO_ERR	Application Confirm Sent Timeout.
TRDP_REPLYTO_ERR	Protocol Reply Timeout.
TRDP_CONFIRMTO_ERR	Protocol Confirm Timeout.
TRDP_REQCONFIRMTO_ERR	Protocol Confirm Timeout (Request sender)
TRDP_PACKET_ERR	Incomplete message data packet.
TRDP_UNRESOLVED_ERR	DNR: address could not be resolved.
TRDP_XML_PARSER_ERR	Returned by the tau_xml subsystem.
TRDP_INUSE_ERR	Resource is still in use.
TRDP_MARSHALLING_ERR	Source size exceeded, dataset mismatch.
TRDP_UNKNOWN_ERR	Unspecified error.

5.35.4.3 TRDP_RED_STATE_T

enum [TRDP_RED_STATE_T](#)

Redundancy states.

Enumerator

TRDP_RED_FOLLOWER	Redundancy follower - redundant PD will be not sent out.
TRDP_RED_LEADER	Redundancy leader - redundant PD will be sent out.

5.35.4.4 TRDP_REPLY_STATUS_T

enum [TRDP_REPLY_STATUS_T](#)

TRDP data transfer type definitions.

Reply status messages

5.35.4.5 TRDP_TO_BEHAVIOR_T

enum [TRDP_TO_BEHAVIOR_T](#)

How invalid PD shall be handled.

Enumerator

TRDP_TO_DEFAULT	Default value defined in tlc_openDession will be taken.
TRDP_TO_SET_TO_ZERO	If set, data will be reset to zero on time out.
TRDP_TO_KEEP_LAST_VALUE	If set, last received values will be returned.

5.36 trdp_utils.c File Reference

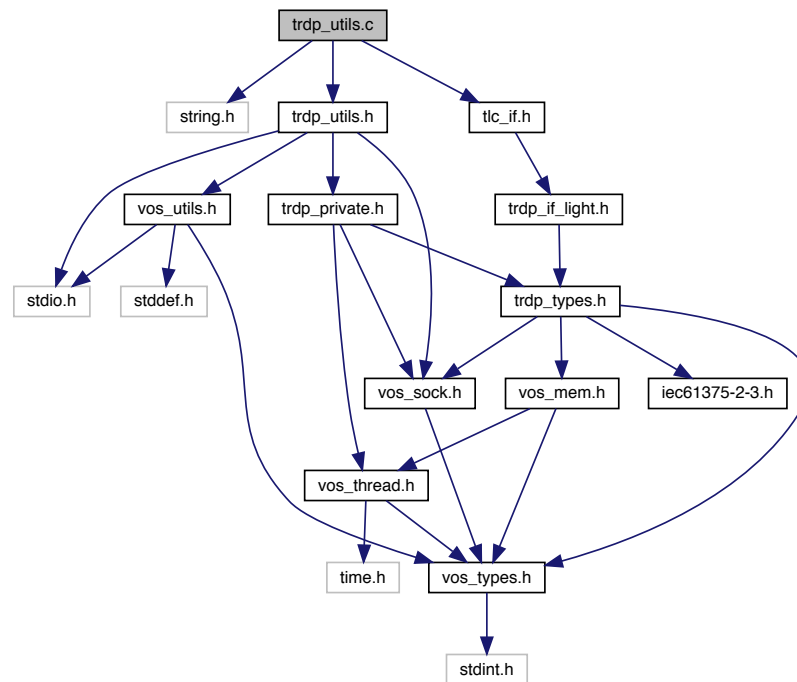
Helper functions for TRDP communication.

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

```
#include "tlc_if.h"
```

```
#include "trdp_utils.h"
```

Include dependency graph for trdp_utils.c:



Functions

- void [printSocketUsage](#) (TRDP_SOCKETS_T iface[])

Debug socket usage output.
- BOOL8 [trdp_SockIsJoined](#) (const TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT], TRDP_IP_ADDR_T mcGroup)

Check if a mc group is in the list.
- BOOL8 [trdp_SockAddJoin](#) (TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT], TRDP_IP_ADDR_T mcGroup)

Add mc group to the list.
- BOOL8 [trdp_SockDelJoin](#) (TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT], TRDP_IP_ADDR_T mcGroup)

remove mc group from the list
- TRDP_IP_ADDR_T [trdp_findMCjoins](#) (TRDP_APP_SESSION_T appHandle, TRDP_IP_ADDR_T mcGroup)

Check an MC group not used by other sockets / subscribers/ listeners.
- UINT32 [trdp_packetSizePD](#) (UINT32 dataSize)

Get the packet size from the raw data size.
- UINT32 [trdp_packetSizeMD](#) (UINT32 dataSize)

- Get the packet size from the raw data size.*

 - `PD_ELE_T * trdp_queueFindComId (PD_ELE_T *pHead, UINT32 comId)`

Return the element with same comId.
- `PD_ELE_T * trdp_queueFindPubAddr (PD_ELE_T *pHead, TRDP_ADDRESSES_T *addr)`

Return the element with same comId, servId and IP addresses.
- `PD_ELE_T * trdp_queueFindSubAddr (PD_ELE_T *pHead, TRDP_ADDRESSES_T *addr)`

Return the element with same comId and IP addresses.
- `PD_ELE_T * trdp_findSubAddr (PD_ELE_T *pHead, TRDP_ADDRESSES_T *addr, UINT32 comId)`

Return the element with same comId and IP addresses.
- `PD_ELE_T * trdp_queueFindExistingSub (PD_ELE_T *pHead, TRDP_ADDRESSES_T *addr)`

Return the element with same comId and IP addresses.
- `void trdp_queueDelElement (PD_ELE_T **ppHead, PD_ELE_T *pDelete)`

Delete an element.
- `BOOL8 trdp_validTopoCounters (UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, UINT32 etbTopoCntFilter, U↵INT32 opTrnTopoCntFilter)`

Check topography counters The applied conformance pattern follows Table A.5/A.21 (positive match): Telegram to be sent Locally stored value (appSession) Case etbTopoCnt opTrnTopoCnt etbTopoCntFilter opTrnTopoCntFilter 1 any any 0 0 2 any equal 0 equal 3 equal any equal 0 4 equal equal equal equal.
- `void trdp_queueAppLast (PD_ELE_T **ppHead, PD_ELE_T *pNew)`

Append an element at end of queue.
- `void trdp_queueInsFirst (PD_ELE_T **ppHead, PD_ELE_T *pNew)`

Insert an element at front of queue.
- `void trdp_initSockets (TRDP_SOCKETS_T iface[], UINT8 noOfEntries)`

Handle the socket pool: Initialize it.
- `TRDP_ERR_T trdp_requestSocket (TRDP_SOCKETS_T iface[], UINT16 port, const TRDP_SEND_PA↵RAM_T *params, TRDP_IP_ADDR_T srcIP, TRDP_IP_ADDR_T mcGroup, TRDP SOCK_TYPE_T type, TRDP_OPTION_T options, BOOL8 rcvMostly, SOCKET useSocket, INT32 *pIndex, TRDP_IP_ADDR_↵T cornerIp)`

Handle the socket pool: Request a socket from our socket pool First we loop through the socket pool and check if there is already a socket which would suit us.
- `void trdp_releaseSocket (TRDP_SOCKETS_T iface[], INT32 lIndex, UINT32 connectTimeout, BOOL8 checkAll, TRDP_IP_ADDR_T mcGroupUsed)`

Handle the socket pool: if a received TCP socket is unused, the socket connection timeout is started.
- `UINT32 trdp_getSeqCnt (UINT32 comId, TRDP_MSG_T msgType, TRDP_IP_ADDR_T srcIPAddr)`

Get the initial sequence counter for the comId/message type and subnet (source IP).
- `void trdp_resetSequenceCounter (PD_ELE_T *pElement, TRDP_IP_ADDR_T srcIP, TRDP_MSG_T msg↵Type)`

remove the sequence counter for the comId/source IP.
- `int trdp_checkSequenceCounter (PD_ELE_T *pElement, UINT32 sequenceCounter, TRDP_IP_ADDR_↵T srcIP, TRDP_MSG_T msgType)`

check and update the sequence counter for the comId/source IP.
- `BOOL8 trdp_isAddressed (const TRDP_URI_USER_T listUri, const TRDP_URI_USER_T destUri)`

Check if listener URI is in addressing range of destination URI.
- `BOOL8 trdp_isInIPrange (TRDP_IP_ADDR_T receivedSrcIP, TRDP_IP_ADDR_T listenedSourceIPlow, T↵RDP_IP_ADDR_T listenedSourceIPhigh)`

Check if received IP is in addressing range of listener's IPs.

5.36.1 Detailed Description

Helper functions for TRDP communication.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013-2019. All rights reserved.

5.36.2 Function Documentation**5.36.2.1 printSocketUsage()**

```
void printSocketUsage (
    TRDP_SOCKETS_T iface[] )
```

Debug socket usage output.

Parameters

in	<i>iface</i>	List of sockets
----	--------------	-----------------

5.36.2.2 trdp_checkSequenceCounter()

```
int trdp_checkSequenceCounter (
    PD_ELE_T * pElement,
    UINT32 sequenceCounter,
    TRDP_IP_ADDR_T srcIP,
    TRDP_MSG_T msgType )
```

check and update the sequence counter for the comID/source IP.

If the comID/srcIP is not found, update it and return 0 - else if already received, return 1 On memory error, return -1

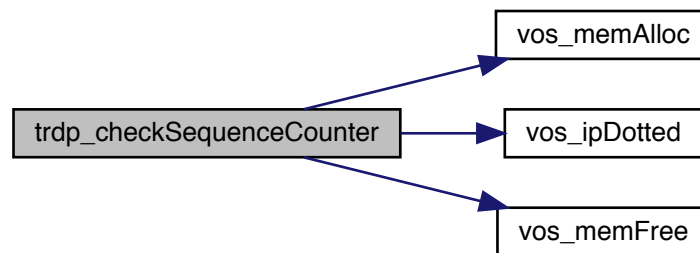
Parameters

in	<i>pElement</i>	subscription element
in	<i>sequenceCounter</i>	sequence counter to check
in	<i>srcIP</i>	Source IP address
in	<i>msgType</i>	type of the message

Return values

0	- no duplicate 1 - duplicate or old sequence counter -1 - memory error
---	--

Here is the call graph for this function:



5.36.2.3 trdp_findMCjoins()

```

TRDP_IP_ADDR_T trdp_findMCjoins (
    TRDP_APP_SESSION_T appHandle,
    TRDP_IP_ADDR_T mcGroup )
  
```

Check an MC group not used by other sockets / subscribers/ listeners.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>mcGroup</i>	multicast group to look for

Return values

<i>multi</i>	cast group if unused VOS_INADDR_ANY if used
--------------	---

5.36.2.4 trdp_findSubAddr()

```

PD_ELE_T* trdp_findSubAddr (
    PD_ELE_T * pHead,
    TRDP_ADDRESSES_T * addr,
    UINT32 comId )
  
```

Return the element with same comId and IP addresses.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>addr</i>	Pub/Sub handle (Address, ComID, srcIP & dest IP, serviceId) to search for
in	<i>comId</i>	ComId to stay on on a sorted search, 0 when searching on unsorted queues

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.36.2.5 trdp_getSeqCnt()

```

UINT32 trdp_getSeqCnt (
    UINT32 comId,
    TRDP_MSG_T msgType,
    TRDP_IP_ADDR_T srcIpAddr )

```

Get the initial sequence counter for the comID/message type and subnet (source IP).

If the comID/srcIP is not found elsewhere, return 0 - else return its current sequence number (the redundant packet needs the same seqNo)

Note: The standard demands that sequenceCounter is managed per comID/msgType at each publisher, but shall be the same for redundant telegrams (subnet/srcIP).

Parameters

in	<i>comId</i>	comID to look for
in	<i>msgType</i>	PD/MD type
in	<i>srcIpAddr</i>	Source IP address

Return values

<i>return</i>	the sequence number
---------------	---------------------

5.36.2.6 trdp_initSockets()

```

void trdp_initSockets (
    TRDP_SOCKETS_T iface[],
    UINT8 noOfEntries )

```

Handle the socket pool: Initialize it.

Parameters

in	<i>iface</i>	pointer to the socket pool
in	<i>noOfEntries</i>	entries in the socket pool

5.36.2.7 trdp_isAddressed()

```
BOOL8 trdp_isAddressed (
    const TRDP_URI_USER_T listUri,
    const TRDP_URI_USER_T destUri )
```

Check if listener URI is in addressing range of destination URI.

Parameters

in	<i>listUri</i>	Null terminated listener URI string to compare
in	<i>destUri</i>	Null terminated destination URI string to compare

Return values

<i>FALSE</i>	- not in addressing range
<i>TRUE</i>	- listener URI is in addressing range of destination URI

5.36.2.8 trdp_isInIPrange()

```
BOOL8 trdp_isInIPrange (
    TRDP_IP_ADDR_T receivedSrcIP,
    TRDP_IP_ADDR_T listenedSourceIPlow,
    TRDP_IP_ADDR_T listenedSourceIPhigh )
```

Check if received IP is in addressing range of listener's IPs.

Parameters

in	<i>receivedSrcIP</i>	Received IP address
in	<i>listenedSourceIPlow</i>	Lower bound IP
in	<i>listenedSourceIPhigh</i>	Upper bound IP

Return values

<i>FALSE</i>	- not in addressing range
<i>TRUE</i>	- received IP is in addressing range of listener

5.36.2.9 trdp_packetSizeMD()

```
UINT32 trdp_packetSizeMD (
    UINT32 dataSize )
```

Get the packet size from the raw data size.

Parameters

in	<i>dataSize</i>	net data size (without padding)
----	-----------------	---------------------------------

Return values

<i>packet</i>	size the size of the complete packet to be sent or received
---------------	---

5.36.2.10 trdp_packetSizePD()

```
UINT32 trdp_packetSizePD (
    UINT32 dataSize )
```

Get the packet size from the raw data size.

Parameters

in	<i>dataSize</i>	net data size (without padding)
----	-----------------	---------------------------------

Return values

<i>packet</i>	size the size of the complete packet to be sent or received
---------------	---

5.36.2.11 trdp_queueAppLast()

```
void trdp_queueAppLast (
    PD_ELEMENT ** ppHead,
    PD_ELEMENT * pNew )
```

Append an element at end of queue.

Parameters

in	<i>ppHead</i>	pointer to pointer to head of queue
in	<i>pNew</i>	pointer to element to append

5.36.2.12 trdp_queueDelElement()

```
void trdp_queueDelElement (
    PD_ELEMENT ** ppHead,
    PD_ELEMENT * pDelete )
```

Delete an element.

Parameters

in	<i>ppHead</i>	pointer to pointer to head of queue
in	<i>pDelete</i>	pointer to element to delete

5.36.2.13 trdp_queueFindComId()

```
PD_ELE_T* trdp_queueFindComId (
    PD_ELE_T * pHead,
    UINT32 comId )
```

Return the element with same comId.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>comId</i>	ComID to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.36.2.14 trdp_queueFindExistingSub()

```
PD_ELE_T* trdp_queueFindExistingSub (
    PD_ELE_T * pHead,
    TRDP_ADDRESSES_T * addr )
```

Return the element with same comId and IP addresses.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>addr</i>	Pub/Sub handle (Address, ComID, srcIP & dest IP) to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.36.2.15 trdp_queueFindPubAddr()

```
PD_ELE_T* trdp_queueFindPubAddr (
    PD_ELE_T * pHead,
    TRDP_ADDRESSES_T * addr )
```

Return the element with same comId, servId and IP addresses.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>addr</i>	Pub/Sub handle (Address, ComID, srcIP & dest IP, servId) to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.36.2.16 trdp_queueFindSubAddr()

```
PD_ELE_T* trdp_queueFindSubAddr (
    PD_ELE_T * pHead,
    TRDP_ADDRESSES_T * addr )
```

Return the element with same comId and IP addresses.

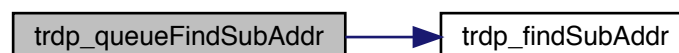
Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>addr</i>	Pub/Sub handle (Address, ComID, srcIP & dest IP, servId) to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

Here is the call graph for this function:



5.36.2.17 trdp_queueInsFirst()

```
void trdp_queueInsFirst (
    PD_ELE_T ** ppHead,
    PD_ELE_T * pNew )
```

Insert an element at front of queue.

Parameters

in	<i>ppHead</i>	pointer to pointer to head of queue
in	<i>pNew</i>	pointer to element to insert

5.36.2.18 trdp_releaseSocket()

```
void trdp_releaseSocket (
    TRDP_SOCKETS_T iface[],
    INT32 lIndex,
    UINT32 connectTimeout,
    BOOL8 checkAll,
    TRDP_IP_ADDR_T mcGroupUsed )
```

Handle the socket pool: if a received TCP socket is unused, the socket connection timeout is started.

In Udp, Release a socket from our socket pool

Parameters

in, out	<i>iface</i>	socket pool
in	<i>lIndex</i>	index of socket to release
in	<i>connectTimeout</i>	time out
in	<i>checkAll</i>	release all TCP pending sockets
in	<i>mcGroupUsed</i>	release MC group subscription

5.36.2.19 trdp_requestSocket()

```
TRDP_ERR_T trdp_requestSocket (
    TRDP_SOCKETS_T iface[],
    UINT16 port,
    const TRDP_SEND_PARAM_T * params,
    TRDP_IP_ADDR_T srcIP,
    TRDP_IP_ADDR_T mcGroup,
    TRDP SOCK_TYPE_T type,
    TRDP_OPTION_T options,
    BOOL8 rcvMostly,
    SOCKET useSocket,
```

```
INT32 * pIndex,
TRDP_IP_ADDR_T cornerIp )
```

Handle the socket pool: Request a socket from our socket pool First we loop through the socket pool and check if there is already a socket which would suit us.

If a multicast group should be joined, we do that on an otherwise suitable socket - up to 20 multicast groups can be joined per socket. If a socket for multicast publishing is requested, we also use the source IP to determine the interface for outgoing multicast traffic.

Parameters

in, out	<i>iface</i>	socket pool
in	<i>port</i>	port to use
in	<i>params</i>	parameters to use
in	<i>srcIP</i>	IP to bind to (0 = any address)
in	<i>mcGroup</i>	MC group to join (0 = do not join)
in	<i>type</i>	type determines port to bind to (PD, MD/UDP, MD/TCP)
in	<i>options</i>	blocking/nonblocking
in	<i>rcvMostly</i>	primarily used for receiving (tbd: bind on sender, too?)
out	<i>useSocket</i>	socket to use, do not open a new one
out	<i>pIndex</i>	returned index of socket pool
in	<i>cornerIp</i>	only used for receiving

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_PARAM_ERR</i>	

5.36.2.20 trdp_resetSequenceCounter()

```
void trdp_resetSequenceCounter (
    PD_ELE_T * pElement,
    TRDP_IP_ADDR_T srcIP,
    TRDP_MSG_T msgType )
```

remove the sequence counter for the comID/source IP.

The sequence counter should be reset if there was a packet time out.

Parameters

in	<i>pElement</i>	subscription element
in	<i>srcIP</i>	Source IP address
in	<i>msgType</i>	message type

Return values

<i>none</i>	
-------------	--

5.36.2.21 trdp_SockAddJoin()

```
BOOL8 trdp_SockAddJoin (
    TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT],
    TRDP_IP_ADDR_T mcGroup )
```

Add mc group to the list.

Parameters

in	<i>mcList</i>	List of multicast groups
in	<i>mcGroup</i>	multicast group

Return values

1	if added 0 if list is full
---	----------------------------

5.36.2.22 trdp_SockDelJoin()

```
BOOL8 trdp_SockDelJoin (
    TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT],
    TRDP_IP_ADDR_T mcGroup )
```

remove mc group from the list

Parameters

in	<i>mcList</i>	List of multicast groups
in	<i>mcGroup</i>	multicast group

Return values

1	if deleted 0 was not in list
---	------------------------------

5.36.2.23 trdp_SockIsJoined()

```
BOOL8 trdp_SockIsJoined (
    const TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT],
    TRDP_IP_ADDR_T mcGroup )
```

Check if a mc group is in the list.

Parameters

in	<i>mcList</i>	List of multicast groups
in	<i>mcGroup</i>	multicast group

Return values

1	if found 0 if not found
---	-------------------------

5.36.2.24 trdp_validTopoCounters()

```

BOOL8 trdp_validTopoCounters (
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    UINT32 etbTopoCntFilter,
    UINT32 opTrnTopoCntFilter )

```

Check topography counters The applied conformance pattern follows Table A.5/A.21 (positive match): Telegram to be sent Locally stored value (appSession) Case etbTopoCnt opTrnTopoCnt etbTopoCntFilter opTrnTopoCntFilter 1 any any 0 0 2 any equal 0 equal 3 equal any equal 0 4 equal equal equal equal.

Parameters

in	<i>etbTopoCnt</i>	ETB topography counter to be checked
in	<i>opTrnTopoCnt</i>	Operational topography counter to be checked
in	<i>etbTopoCntFilter</i>	ETB topography counter filter value
in	<i>opTrnTopoCntFilter</i>	Operational topography counter filter value

Return values

<i>TRUE</i>	Filter criteria matched <i>FALSE</i> Filter criteria not matched
-------------	--

5.37 trdp_utils.h File Reference

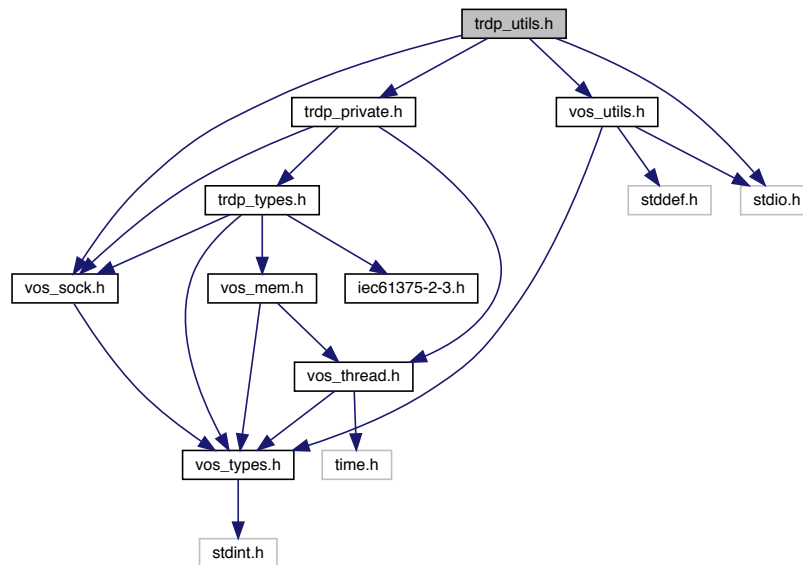
Common utilities for TRDP communication.

```

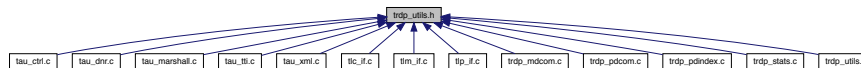
#include <stdio.h>
#include "trdp_private.h"
#include "vos_utils.h"
#include "vos_sock.h"

```

Include dependency graph for trdp_utils.h:



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



Functions

- void [printSocketUsage](#) (TRDP_SOCKETS_T iface[])
Debug socket usage output.
- BOOL8 [trdp_SockIsJoined](#) (const TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT], TRDP_IP_ADDR_T mcGroup)
Check if a mc group is in the list.
- BOOL8 [trdp_SockAddJoin](#) (TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT], TRDP_IP_ADDR_T mcGroup)
Add mc group to the list.
- BOOL8 [trdp_SockDelJoin](#) (TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT], TRDP_IP_ADDR_T mcGroup)
remove mc group from the list
- PD_ELE_T * [trdp_queueFindComId](#) (PD_ELE_T *pHead, UINT32 comId)
Return the element with same comId.
- PD_ELE_T * [trdp_findSubAddr](#) (PD_ELE_T *pHead, TRDP_ADDRESSES_T *pAddr, UINT32 comId)
Return the element with same comId and IP addresses.
- PD_ELE_T * [trdp_queueFindSubAddr](#) (PD_ELE_T *pHead, TRDP_ADDRESSES_T *pAddr)
Return the element with same comId and IP addresses.
- PD_ELE_T * [trdp_queueFindExistingSub](#) (PD_ELE_T *pHead, TRDP_ADDRESSES_T *pAddr)

- Return the element with same comId and IP addresses.*
- `PD_ELE_T * trdp_queueFindPubAddr (PD_ELE_T *pHead, TRDP_ADDRESSES_T *addr)`
- Return the element with same comId, serviceId and IP addresses.*
- `void trdp_queueDelElement (PD_ELE_T **pHead, PD_ELE_T *pDelete)`
- Delete an element.*
- `void trdp_queueAppLast (PD_ELE_T **pHead, PD_ELE_T *pNew)`
- Append an element at end of queue.*
- `void trdp_queueInsFirst (PD_ELE_T **pHead, PD_ELE_T *pNew)`
- Insert an element at front of queue.*
- `void trdp_initSockets (TRDP_SOCKETS_T iface[], UINT8 noOfEntries)`
- Handle the socket pool: Initialize it.*
- `void trdp_resetSequenceCounter (PD_ELE_T *pElement, TRDP_IP_ADDR_T srcIP, TRDP_MSG_T msgType)`
- remove the sequence counter for the comId/source IP.*
- `TRDP_IP_ADDR_T trdp_findMCjoins (TRDP_APP_SESSION_T appHandle, TRDP_IP_ADDR_T mcGroup)`
- Check an MC group not used by other sockets / subscribers/ listeners.*
- `TRDP_ERR_T trdp_requestSocket (TRDP_SOCKETS_T iface[], UINT16 port, const TRDP_SEND_PARAMS_T *params, TRDP_IP_ADDR_T srcIP, TRDP_IP_ADDR_T mcGroup, TRDP_SOCKET_TYPE_T type, TRDP_OPTION_T options, BOOL8 rcvMostly, SOCKET useSocket, INT32 *pIndex, TRDP_IP_ADDR_T cornerIp)`
- Handle the socket pool: Request a socket from our socket pool First we loop through the socket pool and check if there is already a socket which would suit us.*
- `void trdp_releaseSocket (TRDP_SOCKETS_T iface[], INT32 lIndex, UINT32 connectTimeout, BOOL8 checkAll, TRDP_IP_ADDR_T mcGroupUsed)`
- Handle the socket pool: if a received TCP socket is unused, the socket connection timeout is started.*
- `UINT32 trdp_packetSizePD (UINT32 dataSize)`
- Get the packet size from the raw data size.*
- `UINT32 trdp_packetSizeMD (UINT32 dataSize)`
- Get the packet size from the raw data size.*
- `UINT32 trdp_getSeqCnt (UINT32 comId, TRDP_MSG_T msgType, TRDP_IP_ADDR_T srcIPAddr)`
- Get the initial sequence counter for the comId/message type and subnet (source IP).*
- `int trdp_checkSequenceCounter (PD_ELE_T *pElement, UINT32 sequenceCounter, TRDP_IP_ADDR_T srcIP, TRDP_MSG_T msgType)`
- check and update the sequence counter for the comId/source IP.*
- `BOOL8 trdp_isAddressed (const TRDP_URI_USER_T listUri, const TRDP_URI_USER_T destUri)`
- Check if listener URI is in addressing range of destination URI.*
- `BOOL8 trdp_validTopoCounters (UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, UINT32 etbTopoCntFilter, UINT32 opTrnTopoCntFilter)`
- Check topography counters The applied conformance pattern follows Table A.5/A.21 (positive match): Telegram to be sent Locally stored value (appSession) Case etbTopoCnt opTrnTopoCnt etbTopoCntFilter opTrnTopoCntFilter 1 any any 0 0 2 any equal 0 equal 3 equal any equal 0 4 equal equal equal equal.*
- `BOOL8 trdp_isInIPrange (TRDP_IP_ADDR_T receivedSrcIP, TRDP_IP_ADDR_T listenedSourceIPlow, TRDP_IP_ADDR_T listenedSourceIPhigh)`
- Check if received IP is in addressing range of listener's IPs.*

5.37.1 Detailed Description

Common utilities for TRDP communication.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013-2019. All rights reserved.

5.37.2 Function Documentation**5.37.2.1 printSocketUsage()**

```
void printSocketUsage (
    TRDP_SOCKETS_T iface[] )
```

Debug socket usage output.

Parameters

in	<i>iface</i>	List of sockets
----	--------------	-----------------

5.37.2.2 trdp_checkSequenceCounter()

```
int trdp_checkSequenceCounter (
    PD_ELE_T * pElement,
    UINT32 sequenceCounter,
    TRDP_IP_ADDR_T srcIP,
    TRDP_MSG_T msgType )
```

check and update the sequence counter for the comID/source IP.

If the comID/srcIP is not found, update it and return 0 - else if already received, return 1 On memory error, return -1

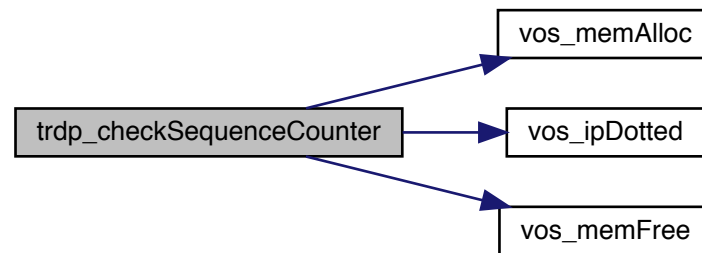
Parameters

in	<i>pElement</i>	subscription element
in	<i>sequenceCounter</i>	sequence counter to check
in	<i>srcIP</i>	Source IP address
in	<i>msgType</i>	type of the message

Return values

0	- no duplicate 1 - duplicate or old sequence counter -1 - memory error
---	--

Here is the call graph for this function:



5.37.2.3 trdp_findMCjoins()

```

TRDP_IP_ADDR_T trdp_findMCjoins (
    TRDP_APP_SESSION_T appHandle,
    TRDP_IP_ADDR_T mcGroup )
  
```

Check an MC group not used by other sockets / subscribers/ listeners.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>mcGroup</i>	multicast group to look for

Return values

<i>multi</i>	cast group if unused VOS_INADDR_ANY if used
--------------	---

5.37.2.4 trdp_findSubAddr()

```

PD_ELE_T* trdp_findSubAddr (
    PD_ELE_T * pHead,
    TRDP_ADDRESSES_T * addr,
    UINT32 comId )
  
```

Return the element with same comId and IP addresses.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>addr</i>	Pub/Sub handle (Address, ComID, srcIP & dest IP, serviceId) to search for
in	<i>comId</i>	ComId to stay on on a sorted search, 0 when searching on unsorted queues

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.37.2.5 trdp_getSeqCnt()

```
UINT32 trdp_getSeqCnt (
    UINT32 comId,
    TRDP_MSG_T msgType,
    TRDP_IP_ADDR_T srcIpAddr )
```

Get the initial sequence counter for the comID/message type and subnet (source IP).

If the comID/srcIP is not found elsewhere, return 0 - else return its current sequence number (the redundant packet needs the same seqNo)

Note: The standard demands that sequenceCounter is managed per comID/msgType at each publisher, but shall be the same for redundant telegrams (subnet/srcIP).

Parameters

in	<i>comId</i>	comID to look for
in	<i>msgType</i>	PD/MD type
in	<i>srcIpAddr</i>	Source IP address

Return values

<i>return</i>	the sequence number
---------------	---------------------

5.37.2.6 trdp_initSockets()

```
void trdp_initSockets (
    TRDP_SOCKETS_T iface[],
    UINT8 noOfEntries )
```

Handle the socket pool: Initialize it.

Parameters

in	<i>iface</i>	pointer to the socket pool
in	<i>noOfEntries</i>	entries in the socket pool

5.37.2.7 trdp_isAddressed()

```

BOOL8 trdp_isAddressed (
    const TRDP_URI_USER_T listUri,
    const TRDP_URI_USER_T destUri )

```

Check if listener URI is in addressing range of destination URI.

Parameters

in	<i>listUri</i>	Null terminated listener URI string to compare
in	<i>destUri</i>	Null terminated destination URI string to compare

Return values

<i>FALSE</i>	- not in addressing range
<i>TRUE</i>	- listener URI is in addressing range of destination URI

5.37.2.8 trdp_isInIPrange()

```

BOOL8 trdp_isInIPrange (
    TRDP_IP_ADDR_T receivedSrcIP,
    TRDP_IP_ADDR_T listenedSourceIPlow,
    TRDP_IP_ADDR_T listenedSourceIPhigh )

```

Check if received IP is in addressing range of listener's IPs.

Parameters

in	<i>receivedSrcIP</i>	Received IP address
in	<i>listenedSourceIPlow</i>	Lower bound IP
in	<i>listenedSourceIPhigh</i>	Upper bound IP

Return values

<i>FALSE</i>	- not in addressing range
<i>TRUE</i>	- received IP is in addressing range of listener

5.37.2.9 trdp_packetSizeMD()

```

UINT32 trdp_packetSizeMD (
    UINT32 dataSize )

```

Get the packet size from the raw data size.

Parameters

in	<i>dataSize</i>	net data size (without padding)
----	-----------------	---------------------------------

Return values

<i>packet</i>	size the size of the complete packet to be sent or received
---------------	---

5.37.2.10 trdp_packetSizePD()

```
UINT32 trdp_packetSizePD (
    UINT32 dataSize )
```

Get the packet size from the raw data size.

Parameters

in	<i>dataSize</i>	net data size (without padding)
----	-----------------	---------------------------------

Return values

<i>packet</i>	size the size of the complete packet to be sent or received
---------------	---

5.37.2.11 trdp_queueAppLast()

```
void trdp_queueAppLast (
    PD_ELE_T ** ppHead,
    PD_ELE_T * pNew )
```

Append an element at end of queue.

Parameters

in	<i>ppHead</i>	pointer to pointer to head of queue
in	<i>pNew</i>	pointer to element to append

5.37.2.12 trdp_queueDelElement()

```
void trdp_queueDelElement (
    PD_ELE_T ** ppHead,
    PD_ELE_T * pDelete )
```

Delete an element.

Parameters

in	<i>ppHead</i>	pointer to pointer to head of queue
in	<i>pDelete</i>	pointer to element to delete

5.37.2.13 trdp_queueFindComId()

```
PD_ELE_T* trdp_queueFindComId (
    PD_ELE_T * pHead,
    UINT32 comId )
```

Return the element with same comId.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>comId</i>	ComID to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.37.2.14 trdp_queueFindExistingSub()

```
PD_ELE_T* trdp_queueFindExistingSub (
    PD_ELE_T * pHead,
    TRDP_ADDRESSES_T * addr )
```

Return the element with same comId and IP addresses.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>addr</i>	Pub/Sub handle (Address, ComID, srcIP & dest IP) to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.37.2.15 trdp_queueFindPubAddr()

```
PD_ELE_T* trdp_queueFindPubAddr (
    PD_ELE_T * pHead,
    TRDP_ADDRESSES_T * addr )
```

Return the element with same comId, serviceId and IP addresses.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>addr</i>	Pub/Sub handle (Address, ComID, srcIP & dest IP, serviceId) to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.37.2.16 trdp_queueFindSubAddr()

```
PD_ELE_T* trdp_queueFindSubAddr (
    PD_ELE_T * pHead,
    TRDP_ADDRESSES_T * addr )
```

Return the element with same comId and IP addresses.

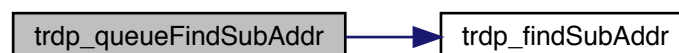
Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>addr</i>	Pub/Sub handle (Address, ComID, srcIP & dest IP, serviceId) to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

Here is the call graph for this function:



5.37.2.17 trdp_queueInsFirst()

```
void trdp_queueInsFirst (
    PD_ELEMENT ** ppHead,
    PD_ELEMENT * pNew )
```

Insert an element at front of queue.

Parameters

in	<i>ppHead</i>	pointer to pointer to head of queue
in	<i>pNew</i>	pointer to element to insert

5.37.2.18 trdp_releaseSocket()

```
void trdp_releaseSocket (
    TRDP_SOCKETS_T iface[],
    INT32 lIndex,
    UINT32 connectTimeout,
    BOOL8 checkAll,
    TRDP_IP_ADDR_T mcGroupUsed )
```

Handle the socket pool: if a received TCP socket is unused, the socket connection timeout is started.

In Udp, Release a socket from our socket pool

Parameters

in, out	<i>iface</i>	socket pool
in	<i>lIndex</i>	index of socket to release
in	<i>connectTimeout</i>	time out
in	<i>checkAll</i>	release all TCP pending sockets
in	<i>mcGroupUsed</i>	release MC group subscription

5.37.2.19 trdp_requestSocket()

```
TRDP_ERR_T trdp_requestSocket (
    TRDP_SOCKETS_T iface[],
    UINT16 port,
    const TRDP_SEND_PARAM_T * params,
    TRDP_IP_ADDR_T srcIP,
    TRDP_IP_ADDR_T mcGroup,
    TRDP SOCK_TYPE_T type,
    TRDP_OPTION_T options,
    BOOL8 rcvMostly,
    SOCKET useSocket,
```

```
INT32 * pIndex,
TRDP_IP_ADDR_T cornerIp )
```

Handle the socket pool: Request a socket from our socket pool First we loop through the socket pool and check if there is already a socket which would suit us.

If a multicast group should be joined, we do that on an otherwise suitable socket - up to 20 multicast groups can be joined per socket. If a socket for multicast publishing is requested, we also use the source IP to determine the interface for outgoing multicast traffic.

Parameters

in, out	<i>iface</i>	socket pool
in	<i>port</i>	port to use
in	<i>params</i>	parameters to use
in	<i>srcIP</i>	IP to bind to (0 = any address)
in	<i>mcGroup</i>	MC group to join (0 = do not join)
in	<i>type</i>	type determines port to bind to (PD, MD/UDP, MD/TCP)
in	<i>options</i>	blocking/nonblocking
in	<i>rcvMostly</i>	primarily used for receiving (tbd: bind on sender, too?)
out	<i>useSocket</i>	socket to use, do not open a new one
out	<i>pIndex</i>	returned index of socket pool
in	<i>cornerIp</i>	only used for receiving

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_PARAM_ERR</i>	

5.37.2.20 trdp_resetSequenceCounter()

```
void trdp_resetSequenceCounter (
    PD_ELE_T * pElement,
    TRDP_IP_ADDR_T srcIP,
    TRDP_MSG_T msgType )
```

remove the sequence counter for the comID/source IP.

The sequence counter should be reset if there was a packet time out.

Parameters

in	<i>pElement</i>	subscription element
in	<i>srcIP</i>	Source IP address
in	<i>msgType</i>	message type

Return values

<i>none</i>	
-------------	--

5.37.2.21 trdp_SockAddJoin()

```
BOOL8 trdp_SockAddJoin (
    TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT],
    TRDP_IP_ADDR_T mcGroup )
```

Add mc group to the list.

Parameters

in	<i>mcList</i>	List of multicast groups
in	<i>mcGroup</i>	multicast group

Return values

1	if added 0 if list is full
---	----------------------------

5.37.2.22 trdp_SockDelJoin()

```
BOOL8 trdp_SockDelJoin (
    TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT],
    TRDP_IP_ADDR_T mcGroup )
```

remove mc group from the list

Parameters

in	<i>mcList</i>	List of multicast groups
in	<i>mcGroup</i>	multicast group

Return values

1	if deleted 0 was not in list
---	------------------------------

5.37.2.23 trdp_SockIsJoined()

```
BOOL8 trdp_SockIsJoined (
    const TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT],
    TRDP_IP_ADDR_T mcGroup )
```

Check if a mc group is in the list.

Parameters

in	<i>mcList</i>	List of multicast groups
in	<i>mcGroup</i>	multicast group

Return values

1	if found 0 if not found
---	-------------------------

5.37.2.24 trdp_validTopoCounters()

```

BOOL8 trdp_validTopoCounters (
    UINT32 etbTopoCnt,
    UINT32 opTrnTopoCnt,
    UINT32 etbTopoCntFilter,
    UINT32 opTrnTopoCntFilter )

```

Check topography counters The applied conformance pattern follows Table A.5/A.21 (positive match): Telegram to be sent Locally stored value (appSession) Case etbTopoCnt opTrnTopoCnt etbTopoCntFilter opTrnTopoCntFilter 1 any any 0 0 2 any equal 0 equal 3 equal any equal 0 4 equal equal equal equal.

Parameters

in	<i>etbTopoCnt</i>	ETB topography counter to be checked
in	<i>opTrnTopoCnt</i>	Operational topography counter to be checked
in	<i>etbTopoCntFilter</i>	ETB topography counter filter value
in	<i>opTrnTopoCntFilter</i>	Operational topography counter filter value

Return values

<i>TRUE</i>	Filter criteria matched <i>FALSE</i> Filter criteria not matched
-------------	--

5.38 trdp_xml.c File Reference

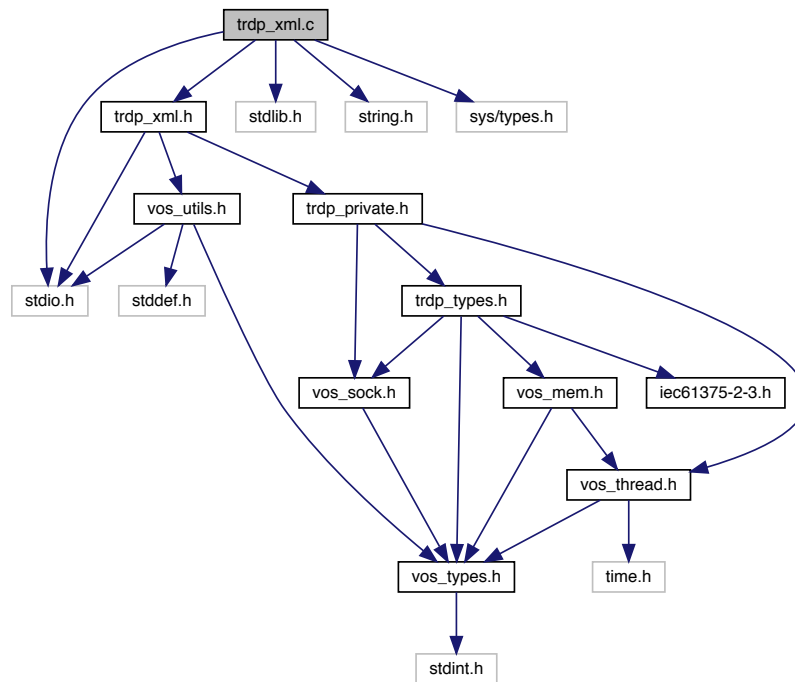
Simple XML parser.

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include "trdp_xml.h"

```

Include dependency graph for trdp_xml.c:



Functions

- **TRDP_ERR_T trdp_XMLOpen** (XML_HANDLE_T *pXML, const char *file)
Opens the XML parsing.
- **TRDP_ERR_T trdp_XMLMemOpen** (XML_HANDLE_T *pXML, char *pBuffer, size_t bufSize)
Opens the XML parsing from a buffer (string stream).
- void **trdp_XMLRewind** (XML_HANDLE_T *pXML)
Rewind to start.
- void **trdp_XMLClose** (XML_HANDLE_T *pXML)
Closes the XML parsng.
- int **trdp_XMLSeekStartTagAny** (XML_HANDLE_T *pXML, char *tag, int maxlen)
Seek next tag on starting depth and return it in provided buffer.
- int **trdp_XMLSeekStartTag** (XML_HANDLE_T *pXML, const char *tag)
Seek a specific tag.
- int **trdp_XMLCountStartTag** (XML_HANDLE_T *pXML, const char *tag)
Count a specific tag.
- void **trdp_XMLEnter** (XML_HANDLE_T *pXML)
Enter level in XML file.
- void **trdp_XMLLeave** (XML_HANDLE_T *pXML)
Leave level in XML file.
- XML_TOKEN_T **trdp_XMLGetAttribute** (XML_HANDLE_T *pXML, CHAR8 *attribute, UINT32 *pValueInt, CHAR8 *value)
Get value of next attribute, as string and if possible as integer.

5.38.1 Detailed Description

Simple XML parser.

Hint: Missing optional elements must be handled using the count-function, otherwise following elements will be following ignored!

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH; based on code by Peter Brander, Bombardier

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>.

5.38.2 Function Documentation

5.38.2.1 trdp_XMLClose()

```
void trdp_XMLClose (
    XML_HANDLE_T * pXML )
```

Closes the XML parsng.

Parameters

in	<i>pXML</i>	Pointer to local data
----	-------------	-----------------------

Return values

<i>none</i>	
-------------	--

5.38.2.2 trdp_XMLCountStartTag()

```
int trdp_XMLCountStartTag (
    XML_HANDLE_T * pXML,
    const char * tag )
```

Count a specific tag.

Parameters

in	<i>pXML</i>	Pointer to local data
in	<i>tag</i>	Tag to count

Return values

0	if found !=0 if not found
---	---------------------------

5.38.2.3 trdp_XMLEnter()

```
void trdp_XMLEnter (
    XML_HANDLE_T * pXML )
```

Enter level in XML file.

Parameters

in	<i>pXML</i>	Pointer to local data
----	-------------	-----------------------

Return values

<i>none</i>	
-------------	--

5.38.2.4 trdp_XMLGetAttribute()

```
XML_TOKEN_T trdp_XMLGetAttribute (
    XML_HANDLE_T * pXML,
    CHAR8 * attribute,
    UINT32 * pValueInt,
    CHAR8 * value )
```

Get value of next attribute, as string and if possible as integer.

Parameters

in	<i>pXML</i>	Pointer to local data
in	<i>attribute</i>	Pointer to attribute
out	<i>pValueInt</i>	Pointer to resulting integer value
out	<i>value</i>	Pointer to resulting string value

Return values

<i>TOK_ATTRIBUTE</i>	if found token if not found
----------------------	-----------------------------

5.38.2.5 trdp_XMLLeave()

```
void trdp_XMLLeave (
    XML_HANDLE_T * pXML )
```

Leave level in XML file.

Parameters

in	<i>pXML</i>	Pointer to local data
----	-------------	-----------------------

Return values

<i>none</i>	
-------------	--

5.38.2.6 trdp_XMLMemOpen()

```
TRDP_ERR_T trdp_XMLMemOpen (
    XML_HANDLE_T * pXML,
    char * pBuffer,
    size_t bufSize )
```

Opens the XML parsing from a buffer (string stream).

Parameters

in	<i>pXML</i>	Pointer to local data
in	<i>pBuffer</i>	Pointer to XML stream buffer
in	<i>bufSize</i>	Size of XML stream buffer

Return values

<i>TRDP_IO_ERR</i>	
--------------------	--

Here is the call graph for this function:



5.38.2.7 trdp_XMLOpen()

```
TRDP_ERR_T trdp_XMLOpen (
    XML_HANDLE_T * pXML,
    const char * file )
```

Opens the XML parsing.

Parameters

in	<i>pXML</i>	Pointer to local data
in	<i>file</i>	Pathname of XML file

Return values

<i>none</i>	
-------------	--

5.38.2.8 trdp_XMLRewind()

```
void trdp_XMLRewind (
    XML_HANDLE_T * pXML )
```

Rewind to start.

Parameters

in	<i>pXML</i>	Pointer to local data
----	-------------	-----------------------

Return values

<i>none</i>	
-------------	--

5.38.2.9 trdp_XMLSeekStartTag()

```
int trdp_XMLSeekStartTag (
    XML_HANDLE_T * pXML,
    const char * tag )
```

Seek a specific tag.

Parameters

in	<i>pXML</i>	Pointer to local data
in	<i>tag</i>	Tag to be found

Return values

<i>0</i>	if found !=0 if not found
----------	---------------------------

5.38.2.10 trdp_XMLSeekStartTagAny()

```
int trdp_XMLSeekStartTagAny (
    XML_HANDLE_T * pXML,
    char * tag,
    int maxlen )
```

Seek next tag on starting depth and return it in provided buffer.

Start tags on deeper depths are ignored.

Parameters

in	<i>pXML</i>	Pointer to local data
in, out	<i>tag</i>	Buffer for found tag
in	<i>maxlen</i>	Length of buffer

Return values

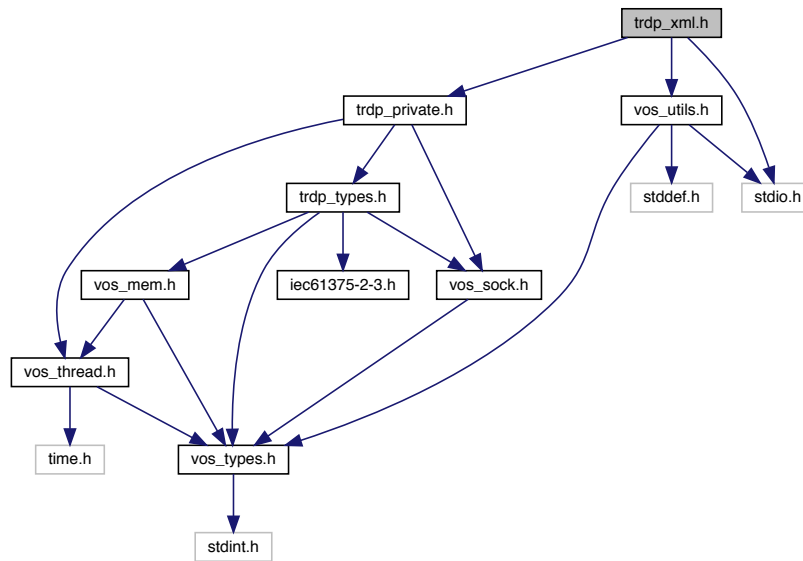
<i>0</i>	if found !=0 if not found
----------	---------------------------

5.39 trdp_xml.h File Reference

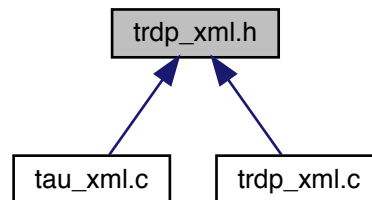
Simple XML parser.

```
#include <stdio.h>
#include "trdp_private.h"
#include "vos_utils.h"
```

Include dependency graph for trdp_xml.h:



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



Functions

- [TRDP_ERR_T trdp_XMLOpen](#) (XML_HANDLE_T *pXML, const char *file)
Opens the XML parsing.
- [TRDP_ERR_T trdp_XMLMemOpen](#) (XML_HANDLE_T *pXML, char *pBuffer, size_t bufSize)
Opens the XML parsing from a buffer (string stream).
- void [trdp_XMLClose](#) (XML_HANDLE_T *pXML)
Closes the XML parsing.
- int [trdp_XMLCountStartTag](#) (XML_HANDLE_T *pXML, const char *tag)
Count a specific tag.
- int [trdp_XMLSeekStartTagAny](#) (XML_HANDLE_T *pXML, char *tag, int maxlen)
Seek next tag on starting depth and return it in provided buffer.

- int [trdp_XMLSeekStartTag](#) (XML_HANDLE_T *pXML, const char *tag)
Seek a specific tag.
- XML_TOKEN_T [trdp_XMLGetAttribute](#) (XML_HANDLE_T *pXML, CHAR8 *attribute, UINT32 *pValueInt, CHAR8 *value)
Get value of next attribute, as string and if possible as integer.
- void [trdp_XMLRewind](#) (XML_HANDLE_T *pXML)
Rewind to start.
- void [trdp_XMLEnter](#) (XML_HANDLE_T *pXML)
Enter level in XML file.
- void [trdp_XMLLeave](#) (XML_HANDLE_T *pXML)
Leave level in XML file.

5.39.1 Detailed Description

Simple XML parser.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright NewTec GmbH or its subsidiaries and others, 2016. All rights reserved.

5.39.2 Function Documentation

5.39.2.1 trdp_XMLClose()

```
void trdp_XMLClose (
    XML_HANDLE_T * pXML )
```

Closes the XML parsng.

Parameters

in	<i>pXML</i>	Pointer to local data
----	-------------	-----------------------

Return values

<i>none</i>	
-------------	--

5.39.2.2 trdp_XMLCountStartTag()

```
int trdp_XMLCountStartTag (
    XML_HANDLE_T * pXML,
    const char * tag )
```

Count a specific tag.

Parameters

in	<i>pXML</i>	Pointer to local data
in	<i>tag</i>	Tag to count

Return values

0	if found !=0 if not found
---	---------------------------

5.39.2.3 trdp_XMLEnter()

```
void trdp_XMLEnter (
    XML_HANDLE_T * pXML )
```

Enter level in XML file.

Parameters

in	<i>pXML</i>	Pointer to local data
----	-------------	-----------------------

Return values

<i>none</i>	
-------------	--

5.39.2.4 trdp_XMLGetAttribute()

```
XML_TOKEN_T trdp_XMLGetAttribute (
    XML_HANDLE_T * pXML,
    CHAR8 * attribute,
    UINT32 * pValueInt,
    CHAR8 * value )
```

Get value of next attribute, as string and if possible as integer.

Parameters

in	<i>pXML</i>	Pointer to local data
in	<i>attribute</i>	Pointer to attribute
out	<i>pValueInt</i>	Pointer to resulting integer value
out	<i>value</i>	Pointer to resulting string value

Return values

<i>TOK_ATTRIBUTE</i>	if found token if not found
----------------------	-----------------------------

5.39.2.5 trdp_XMLLeave()

```
void trdp_XMLLeave (
    XML_HANDLE_T * pXML )
```

Leave level in XML file.

Parameters

in	<i>pXML</i>	Pointer to local data
----	-------------	-----------------------

Return values

<i>none</i>	
-------------	--

5.39.2.6 trdp_XMLMemOpen()

```
TRDP_ERR_T trdp_XMLMemOpen (
    XML_HANDLE_T * pXML,
    char * pBuffer,
    size_t bufSize )
```

Opens the XML parsing from a buffer (string stream).

Parameters

in	<i>pXML</i>	Pointer to local data
in	<i>pBuffer</i>	Pointer to XML stream buffer
in	<i>bufSize</i>	Size of XML stream buffer

Return values

<i>TRDP_IO_ERR</i>	
--------------------	--

Here is the call graph for this function:



5.39.2.7 trdp_XMLOpen()

```
TRDP_ERR_T trdp_XMLOpen (
    XML_HANDLE_T * pXML,
    const char * file )
```

Opens the XML parsing.

Parameters

in	<i>pXML</i>	Pointer to local data
in	<i>file</i>	Pathname of XML file

Return values

<i>none</i>	
-------------	--

5.39.2.8 trdp_XMLRewind()

```
void trdp_XMLRewind (
    XML_HANDLE_T * pXML )
```

Rewind to start.

Parameters

in	<i>pXML</i>	Pointer to local data
----	-------------	-----------------------

Return values

<i>none</i>	
-------------	--

5.39.2.9 trdp_XMLSeekStartTag()

```
int trdp_XMLSeekStartTag (
    XML_HANDLE_T * pXML,
    const char * tag )
```

Seek a specific tag.

Parameters

in	<i>pXML</i>	Pointer to local data
in	<i>tag</i>	Tag to be found

Return values

0	if found !=0 if not found
---	---------------------------

5.39.2.10 trdp_XMLSeekStartTagAny()

```
int trdp_XMLSeekStartTagAny (
    XML_HANDLE_T * pXML,
    char * tag,
    int maxlen )
```

Seek next tag on starting depth and return it in provided buffer.

Start tags on deeper depths are ignored.

Parameters

in	<i>pXML</i>	Pointer to local data
in, out	<i>tag</i>	Buffer for found tag
in	<i>maxlen</i>	Length of buffer

Return values

0	if found !=0 if not found
---	---------------------------

5.40 vos_mem.c File Reference

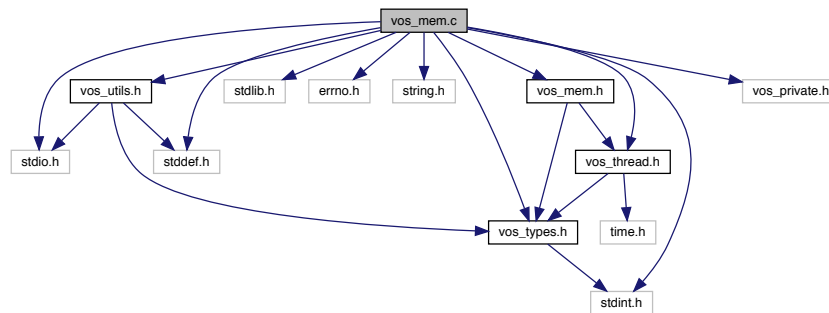
Memory functions.

```
#include <stdio.h>
#include <stddef.h>
#include <stdint.h>
#include <stdlib.h>
#include <errno.h>
```



```
#include <string.h>
#include "vos_types.h"
#include "vos_utils.h"
#include "vos_mem.h"
#include "vos_thread.h"
#include "vos_private.h"
```

Include dependency graph for vos_mem.c:



Functions

- EXT_DECL [VOS_ERR_T vos_memInit](#) (UINT8 *pMemoryArea, UINT32 size, const UINT32 fragMem[[VOS_MEM_NBLOCKSIZES](#)])
Initialize the memory unit.
- EXT_DECL void [vos_memDelete](#) (UINT8 *pMemoryArea)
Delete the memory area.
- EXT_DECL UINT8 * [vos_memAlloc](#) (UINT32 size)
Allocate a block of memory (from memory area above).
- EXT_DECL void [vos_memFree](#) (void *pMemBlock)
Deallocate a block of memory (from memory area above).
- EXT_DECL [VOS_ERR_T vos_memCount](#) (UINT32 *pAllocatedMemory, UINT32 *pFreeMemory, UINT32 *pMinFree, UINT32 *pNumAllocBlocks, UINT32 *pNumAllocErr, UINT32 *pNumFreeErr, UINT32 blockSize[[VOS_MEM_NBLOCKSIZES](#)], UINT32 usedBlockSize[[VOS_MEM_NBLOCKSIZES](#)])
Return used and available memory (of memory area above).
- EXT_DECL void [vos_qsort](#) (void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))
Sort an array.
- EXT_DECL void * [vos_bsearch](#) (const void *pKey, const void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))
Binary search in a sorted array.
- EXT_DECL INT32 [vos_strncmp](#) (const CHAR8 *pStr1, const CHAR8 *pStr2, UINT32 count)
Case insensitive string compare.
- EXT_DECL void [vos_strncpy](#) (CHAR8 *pStrDst, const CHAR8 *pStrSrc, UINT32 count)
String copy with length limitation.
- EXT_DECL void [vos_strncat](#) (CHAR8 *pStrDst, UINT32 count, const CHAR8 *pStrSrc)
String concatenation with length limitation.
- EXT_DECL [VOS_ERR_T vos_queueCreate](#) ([VOS_QUEUE_POLICY_T](#) queueType, UINT32 maxNoOfMsg, [VOS_QUEUE_T](#) *pQueueHandle)
Initialize a message queue.
- EXT_DECL [VOS_ERR_T vos_queueSend](#) ([VOS_QUEUE_T](#) queueHandle, UINT8 *pData, UINT32 size)

Send a message.

- EXT_DECL `VOS_ERR_T vos_queueReceive` (`VOS_QUEUE_T` queueHandle, `UINT8 **ppData`, `UINT32 *pSize`, `UINT32 usTimeout`)

Get a message.

- EXT_DECL `VOS_ERR_T vos_queueDestroy` (`VOS_QUEUE_T` queueHandle)

Destroy a message queue.

5.40.1 Detailed Description

Memory functions.

OS abstraction of memory access and control

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.40.2 Function Documentation

5.40.2.1 vos_bsearch()

```
EXT_DECL void* vos_bsearch (
    const void * pKey,
    const void * pBuf,
    UINT32 num,
    UINT32 size,
    int(*) (const void *, const void *) compare )
```

Binary search in a sorted array.

This is just a wrapper for the standard bsearch function.

Parameters

in	<i>pKey</i>	Key to search for
in	<i>pBuf</i>	Pointer to the array to search
in	<i>num</i>	number of elements
in	<i>size</i>	size of one element
in	<i>compare</i>	Pointer to compare function return -n if arg1 < arg2, return 0 if arg1 == arg2, return +n if arg1 > arg2 where n is an integer != 0

Return values

<i>Pointer</i>	to found element or NULL
----------------	--------------------------

5.40.2.2 vos_memAlloc()

```
EXT_DECL UINT8* vos_memAlloc (
    UINT32 size )
```

Allocate a block of memory (from memory area above).

Parameters

in	size	Size of requested block
----	------	-------------------------

Return values

<i>Pointer</i>	to memory area
<i>NULL</i>	if no memory available

5.40.2.3 vos_memCount()

```
EXT_DECL VOS_ERR_T vos_memCount (
    UINT32 * pAllocatedMemory,
    UINT32 * pFreeMemory,
    UINT32 * pMinFree,
    UINT32 * pNumAllocBlocks,
    UINT32 * pNumAllocErr,
    UINT32 * pNumFreeErr,
    UINT32 blockSize[VOS_MEM_NBLOCKSIZES],
    UINT32 usedBlockSize[VOS_MEM_NBLOCKSIZES] )
```

Return used and available memory (of memory area above).

Parameters

out	<i>pAllocatedMemory</i>	Pointer to allocated memory size
out	<i>pFreeMemory</i>	Pointer to free memory size
out	<i>pMinFree</i>	Pointer to minimal free memory size in statistics interval
out	<i>pNumAllocBlocks</i>	Pointer to number of allocated memory blocks
out	<i>pNumAllocErr</i>	Pointer to number of allocation errors
out	<i>pNumFreeErr</i>	Pointer to number of free errors
out	<i>blockSize</i>	Pointer to list of memory block sizes
out	<i>usedBlockSize</i>	Pointer to list of used memoryblocks

Return values

<code>VOS_NO_ERR</code>	no error
<code>VOS_INIT_ERR</code>	module not initialised

5.40.2.4 `vos_memDelete()`

```
EXT_DECL void vos_memDelete (
    UINT8 * pMemoryArea )
```

Delete the memory area.

This will eventually invalidate any previously allocated memory blocks! It should be called last before the application quits. No further access to the memory blocks is allowed after this call.

Parameters

in	<i>pMemoryArea</i>	Pointer to memory area used
----	--------------------	-----------------------------

5.40.2.5 `vos_memFree()`

```
EXT_DECL void vos_memFree (
    void * pMemBlock )
```

Deallocate a block of memory (from memory area above).

Parameters

in	<i>pMemBlock</i>	Pointer to memory block to be freed
----	------------------	-------------------------------------

5.40.2.6 `vos_memInit()`

```
EXT_DECL VOS_ERR_T vos_memInit (
    UINT8 * pMemoryArea,
    UINT32 size,
    const UINT32 fragMem[VOS_MEM_NBLOCKSIZES] )
```

Initialize the memory unit.

Init a supplied block of memory and prepare it for use with `vos_memAlloc` and `vos_memFree`. The used block sizes can be supplied and will be preallocated. If half of the overall size of the requested memory area would be pre-allocated, either by the default pre-allocation table or a provided one, no pre-allocation takes place.

Parameters

in	<i>pMemoryArea</i>	Pointer to memory area to use
in	<i>size</i>	Size of provided memory area
in	<i>fragMem</i>	Pointer to list of preallocated block sizes, used to fragment memory for large blocks

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_MEM_ERR</i>	no memory available
<i>VOS_MUTEX_ERR</i>	no mutex available

5.40.2.7 vos_qsort()

```
EXT_DECL void vos_qsort (
    void * pBuf,
    UINT32 num,
    UINT32 size,
    int(*) (const void *, const void *) compare )
```

Sort an array.

This is just a wrapper for the standard qsort function.

Parameters

in, out	<i>pBuf</i>	Pointer to the array to sort
in	<i>num</i>	number of elements
in	<i>size</i>	size of one element
in	<i>compare</i>	Pointer to compare function return -n if arg1 < arg2, return 0 if arg1 == arg2, return +n if arg1 > arg2 where n is an integer != 0

Return values

<i>none</i>	
-------------	--

5.40.2.8 vos_queueCreate()

```
EXT_DECL VOS_ERR_T vos_queueCreate (
    VOS_QUEUE_POLICY_T queueType,
    UINT32 maxNoOfMsg,
    VOS_QUEUE_T * pQueueHandle )
```

Initialize a message queue.

Returns a handle for further calls

Parameters

in	<i>queueType</i>	Define queue type (1 = FIFO, 2 = LIFO, 3 = PRIO)
in	<i>maxNoOfMsg</i>	Maximum number of messages
out	<i>pQueueHandle</i>	Handle of created queue

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_INIT_ERR</i>	not supported
<i>VOS_QUEUE_ERR</i>	error creating queue

5.40.2.9 vos_queueDestroy()

```
EXT_DECL VOS_ERR_T vos_queueDestroy (
    VOS_QUEUE_T queueHandle )
```

Destroy a message queue.

Free all resources used by this queue

Parameters

in	<i>queueHandle</i>	Queue handle
----	--------------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid

5.40.2.10 vos_queueReceive()

```
EXT_DECL VOS_ERR_T vos_queueReceive (
    VOS_QUEUE_T queueHandle,
    UINT8 ** ppData,
    UINT32 * pSize,
    UINT32 usTimeout )
```

Get a message.

Parameters

in	<i>queueHandle</i>	Queue handle
out	<i>ppData</i>	Pointer to data pointer to be received
out	<i>pSize</i>	Size of receive data
in	<i>usTimeout</i>	Maximum time to wait for a message (in usec)

Return values

<i>VOSNO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_QUEUE_ERR</i>	queue is empty

5.40.2.11 vos_queueSend()

```
EXT_DECL VOS_ERR_T vos_queueSend (
    VOS_QUEUE_T queueHandle,
    UINT8 * pData,
    UINT32 size )
```

Send a message.

Parameters

in	<i>queueHandle</i>	Queue handle
in	<i>pData</i>	Pointer to data to be sent
in	<i>size</i>	Size of data to be sent

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_INIT_ERR</i>	not supported
<i>VOS_QUEUE_ERR</i>	error creating queue

5.40.2.12 vos_strncat()

```
EXT_DECL void vos_strncat (
    CHAR8 * pStrDst,
    UINT32 count,
    const CHAR8 * pStrSrc )
```

String concatenation with length limitation.

Parameters

in	<i>pStrDst</i>	Destination string
in	<i>count</i>	Size of destination buffer
in	<i>pStrSrc</i>	Null terminated string to append

Return values

<i>none</i>	
-------------	--

5.40.2.13 vos_strncpy()

```
EXT_DECL void vos_strncpy (
    CHAR8 * pStrDst,
    const CHAR8 * pStrSrc,
    UINT32 count )
```

String copy with length limitation.

Parameters

in	<i>pStrDst</i>	Destination string
in	<i>pStrSrc</i>	Null terminated string to copy
in	<i>count</i>	Maximum number of characters to copy

Return values

<i>none</i>	
-------------	--

5.40.2.14 vos_strnicmp()

```
EXT_DECL INT32 vos_strnicmp (
    const CHAR8 * pStr1,
    const CHAR8 * pStr2,
    UINT32 count )
```

Case insensitive string compare.

Parameters

in	<i>pStr1</i>	Null terminated string to compare
in	<i>pStr2</i>	Null terminated string to compare
in	<i>count</i>	Maximum number of characters to compare

Return values

0	- equal
<0	- string1 less than string 2
>0	- string 1 greater than string 2

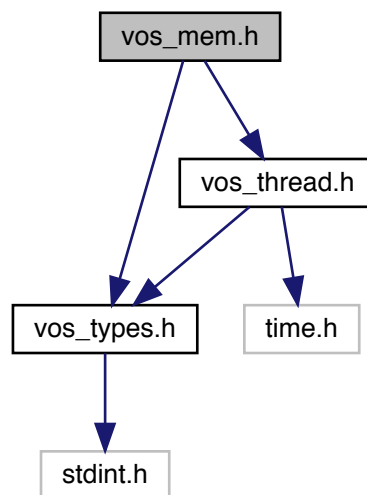
5.41 vos_mem.h File Reference

Memory and queue functions for OS abstraction.

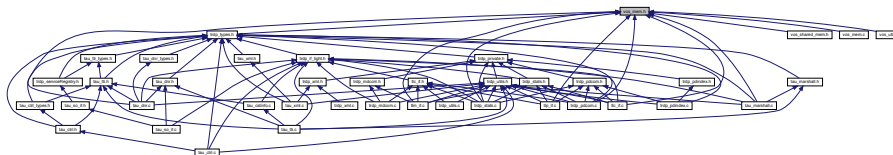
```
#include "vos_types.h"
```

```
#include "vos_thread.h"
```

Include dependency graph for vos_mem.h:



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



Macros

- `#define VOS_MEM_NBLOCKSIZES 15u`
No of pre-defined block sizes.
- `#define VOS_MEM_MAX_PREALLOCATE 10u`
We internally allocate memory always by these block sizes.
- `#define VOS_MEM_PREALLOCATE {0u, 0u, 0u, 0u, 0u, 0u, 0u, 0u, 4u, 0u, 0u, 0u, 0u, 0u, 0u}`
Default pre-allocation of free memory blocks.

Typedefs

- typedef struct VOS_QUEUE * [VOS_QUEUE_T](#)
Opaque queue define.

Enumerations

- enum [VOS_QUEUE_POLICY_T](#)
Queue policy matching pthread/Posix defines.

Functions

- EXT_DECL [VOS_ERR_T vos_memInit](#) (UINT8 *pMemoryArea, UINT32 size, const UINT32 fragMem[[VOS_MEM_NBLOCKSIZES](#)])
Initialize the memory unit.
- EXT_DECL void [vos_memDelete](#) (UINT8 *pMemoryArea)
Delete the memory area.
- EXT_DECL UINT8 * [vos_memAlloc](#) (UINT32 size)
Allocate a block of memory (from memory area above).
- EXT_DECL void [vos_memFree](#) (void *pMemBlock)
Deallocate a block of memory (from memory area above).
- EXT_DECL [VOS_ERR_T vos_memCount](#) (UINT32 *pAllocatedMemory, UINT32 *pFreeMemory, UINT32 *pMinFree, UINT32 *pNumAllocBlocks, UINT32 *pNumAllocErr, UINT32 *pNumFreeErr, UINT32 block↵Size[[VOS_MEM_NBLOCKSIZES](#)], UINT32 usedBlockSize[[VOS_MEM_NBLOCKSIZES](#)])
Return used and available memory (of memory area above).
- EXT_DECL void [vos_qsort](#) (void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))
Sort an array.
- EXT_DECL void * [vos_bsearch](#) (const void *pKey, const void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))
Binary search in a sorted array.
- EXT_DECL INT32 [vos_strncmp](#) (const CHAR8 *pStr1, const CHAR8 *pStr2, UINT32 count)
Case insensitive string compare.
- EXT_DECL void [vos_strncpy](#) (CHAR8 *pStrDst, const CHAR8 *pStrSrc, UINT32 count)
String copy with length limitation.
- EXT_DECL void [vos_strncat](#) (CHAR8 *pStrDst, UINT32 count, const CHAR8 *pStrSrc)
String concatenation with length limitation.
- EXT_DECL [VOS_ERR_T vos_queueCreate](#) ([VOS_QUEUE_POLICY_T](#) queueType, UINT32 maxNoOfMsg, [VOS_QUEUE_T](#) *pQueueHandle)
Initialize a message queue.
- EXT_DECL [VOS_ERR_T vos_queueSend](#) ([VOS_QUEUE_T](#) queueHandle, UINT8 *pData, UINT32 size)
Send a message.
- EXT_DECL [VOS_ERR_T vos_queueReceive](#) ([VOS_QUEUE_T](#) queueHandle, UINT8 **ppData, UINT32 *pSize, UINT32 usTimeout)
Get a message.
- EXT_DECL [VOS_ERR_T vos_queueDestroy](#) ([VOS_QUEUE_T](#) queueHandle)
Destroy a message queue.

5.41.1 Detailed Description

Memory and queue functions for OS abstraction.

This module provides memory control supervision

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH Peter Brander (Memory scheme)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.41.2 Macro Definition Documentation

5.41.2.1 VOS_MEM_MAX_PREALLOCATE

```
#define VOS_MEM_MAX_PREALLOCATE 10u
```

We internally allocate memory always by these block sizes.

The largest available block is 524288 Bytes, provided the overall size of the used memory allocation area is larger.
Max. no. of blocks to pre-allocate

5.41.2.2 VOS_MEM_PREALLOCATE

```
#define VOS_MEM_PREALLOCATE {0u, 0u, 0u, 0u, 0u, 0u, 0u, 0u, 4u, 0u, 0u, 0u, 0u, 0u, 0u}
```

Default pre-allocation of free memory blocks.

To avoid problems with too many small blocks and no large one. Specify how many of each block size that should be pre-allocated (and freed!) to pre-segment the memory area.

5.41.3 Function Documentation

5.41.3.1 vos_bsearch()

```
EXT_DECL void* vos_bsearch (
    const void * pKey,
    const void * pBuf,
    UINT32 num,
    UINT32 size,
    int(*) (const void *, const void *) compare )
```

Binary search in a sorted array.

This is just a wrapper for the standard bsearch function.

Parameters

in	<i>pKey</i>	Key to search for
in	<i>pBuf</i>	Pointer to the array to search
in	<i>num</i>	number of elements
in	<i>size</i>	size of one element
in	<i>compare</i>	Pointer to compare function return -n if arg1 < arg2, return 0 if arg1 == arg2, return +n if arg1 > arg2 where n is an integer != 0

Return values

<i>Pointer</i>	to found element or NULL
----------------	--------------------------

5.41.3.2 vos_memAlloc()

```
EXT_DECL UINT8* vos_memAlloc (
    UINT32 size )
```

Allocate a block of memory (from memory area above).

Parameters

in	<i>size</i>	Size of requested block
----	-------------	-------------------------

Return values

<i>Pointer</i>	to memory area
<i>NULL</i>	if no memory available

5.41.3.3 vos_memCount()

```
EXT_DECL VOS_ERR_T vos_memCount (
    UINT32 * pAllocatedMemory,
    UINT32 * pFreeMemory,
    UINT32 * pMinFree,
    UINT32 * pNumAllocBlocks,
    UINT32 * pNumAllocErr,
    UINT32 * pNumFreeErr,
    UINT32 blockSize[VOS_MEM_NBLOCKSIZES],
    UINT32 usedBlockSize[VOS_MEM_NBLOCKSIZES] )
```

Return used and available memory (of memory area above).

Parameters

out	<i>pAllocatedMemory</i>	Pointer to allocated memory size
-----	-------------------------	----------------------------------

Parameters

out	<i>pFreeMemory</i>	Pointer to free memory size
out	<i>pMinFree</i>	Pointer to minimal free memory size in statistics interval
out	<i>pNumAllocBlocks</i>	Pointer to number of allocated memory blocks
out	<i>pNumAllocErr</i>	Pointer to number of allocation errors
out	<i>pNumFreeErr</i>	Pointer to number of free errors
out	<i>blockSize</i>	Pointer to list of memory block sizes
out	<i>usedBlockSize</i>	Pointer to list of used memory blocks

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised

5.41.3.4 vos_memDelete()

```
EXT_DECL void vos_memDelete (
    UINT8 * pMemoryArea )
```

Delete the memory area.

This will eventually invalidate any previously allocated memory blocks! It should be called last before the application quits. No further access to the memory blocks is allowed after this call.

Parameters

in	<i>pMemoryArea</i>	Pointer to memory area to use
----	--------------------	-------------------------------

This will eventually invalidate any previously allocated memory blocks! It should be called last before the application quits. No further access to the memory blocks is allowed after this call.

Parameters

in	<i>pMemoryArea</i>	Pointer to memory area used
----	--------------------	-----------------------------

5.41.3.5 vos_memFree()

```
EXT_DECL void vos_memFree (
    void * pMemBlock )
```

Deallocate a block of memory (from memory area above).

Parameters

in	<i>pMemBlock</i>	Pointer to memory block to be freed
----	------------------	-------------------------------------

5.41.3.6 vos_memInit()

```
EXT_DECL VOS_ERR_T vos_memInit (
    UINT8 * pMemoryArea,
    UINT32 size,
    const UINT32 fragMem[VOS_MEM_NBLOCKSIZES] )
```

Initialize the memory unit.

Init a supplied block of memory and prepare it for use with vos_alloc and vos_dealloc. The used block sizes can be supplied and will be preallocated.

Parameters

in	<i>pMemoryArea</i>	Pointer to memory area to use
in	<i>size</i>	Size of provided memory area
in	<i>fragMem</i>	Pointer to list of preallocate block sizes, used to fragment memory for large blocks

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_MEM_ERR</i>	no memory available

Init a supplied block of memory and prepare it for use with vos_memAlloc and vos_memFree. The used block sizes can be supplied and will be preallocated. If half of the overall size of the requested memory area would be pre-allocated, either by the default pre-allocation table or a provided one, no pre-allocation takes place.

Parameters

in	<i>pMemoryArea</i>	Pointer to memory area to use
in	<i>size</i>	Size of provided memory area
in	<i>fragMem</i>	Pointer to list of preallocated block sizes, used to fragment memory for large blocks

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_MEM_ERR</i>	no memory available
<i>VOS_MUTEX_ERR</i>	no mutex available

5.41.3.7 vos_qsort()

```
EXT_DECL void vos_qsort (
    void * pBuf,
```

```

UINT32 num,
UINT32 size,
int(*) (const void *, const void *) compare )

```

Sort an array.

This is just a wrapper for the standard qsort function.

Parameters

in, out	<i>pBuf</i>	Pointer to the array to sort
in	<i>num</i>	number of elements
in	<i>size</i>	size of one element
in	<i>compare</i>	Pointer to compare function return -n if arg1 < arg2, return 0 if arg1 == arg2, return +n if arg1 > arg2 where n is an integer != 0

Return values

<i>none</i>	
-------------	--

5.41.3.8 vos_queueCreate()

```

EXT_DECL VOS_ERR_T vos_queueCreate (
    VOS_QUEUE_POLICY_T queueType,
    UINT32 maxNoOfMsg,
    VOS_QUEUE_T * pQueueHandle )

```

Initialize a message queue.

Returns a handle for further calls

Parameters

in	<i>queueType</i>	Define queue type (1 = FIFO, 2 = LIFO, 3 = PRIO)
in	<i>maxNoOfMsg</i>	Maximum number of messages
out	<i>pQueueHandle</i>	Handle of created queue

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_INIT_ERR</i>	not supported
<i>VOS_QUEUE_ERR</i>	error creating queue

5.41.3.9 vos_queueDestroy()

```
EXT_DECL VOS_ERR_T vos_queueDestroy (
    VOS_QUEUE_T queueHandle )
```

Destroy a message queue.

Free all resources used by this queue

Parameters

in	<i>queueHandle</i>	Queue handle
----	--------------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid

5.41.3.10 vos_queueReceive()

```
EXT_DECL VOS_ERR_T vos_queueReceive (
    VOS_QUEUE_T queueHandle,
    UINT8 ** ppData,
    UINT32 * pSize,
    UINT32 usTimeout )
```

Get a message.

Parameters

in	<i>queueHandle</i>	Queue handle
out	<i>ppData</i>	Pointer to data pointer to be received
out	<i>pSize</i>	Size of receive data
in	<i>usTimeout</i>	Maximum time to wait for a message (in usec)

Return values

<i>VOSNO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_QUEUE_ERR</i>	queue is empty

5.41.3.11 vos_queueSend()

```
EXT_DECL VOS_ERR_T vos_queueSend (
    VOS_QUEUE_T queueHandle,
    UINT8 * pData,
    UINT32 size )
```

Send a message.

Parameters

in	<i>queueHandle</i>	Queue handle
in	<i>pData</i>	Pointer to data to be sent
in	<i>size</i>	Size of data to be sent

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_INIT_ERR</i>	not supported
<i>VOS_QUEUE_ERR</i>	error creating queue

5.41.3.12 vos_strncat()

```
EXT_DECL void vos_strncat (
    CHAR8 * pStrDst,
    UINT32 count,
    const CHAR8 * pStrSrc )
```

String concatenation with length limitation.

Parameters

in	<i>pStrDst</i>	Destination string
in	<i>count</i>	Size of destination buffer
in	<i>pStrSrc</i>	Null terminated string to append

Return values

<i>none</i>	
-------------	--

5.41.3.13 vos_strncpy()

```
EXT_DECL void vos_strncpy (
    CHAR8 * pStrDst,
```

```
const CHAR8 * pStrSrc,
UINT32 count )
```

String copy with length limitation.

Parameters

in	<i>pStrDst</i>	Destination string
in	<i>pStrSrc</i>	Null terminated string to copy
in	<i>count</i>	Maximum number of characters to copy

Return values

<i>none</i>	
-------------	--

5.41.3.14 vos_strnicmp()

```
EXT_DECL INT32 vos_strnicmp (
    const CHAR8 * pStr1,
    const CHAR8 * pStr2,
    UINT32 count )
```

Case insensitive string compare.

Parameters

in	<i>pStr1</i>	Null terminated string to compare
in	<i>pStr2</i>	Null terminated string to compare
in	<i>count</i>	Maximum number of characters to compare

Return values

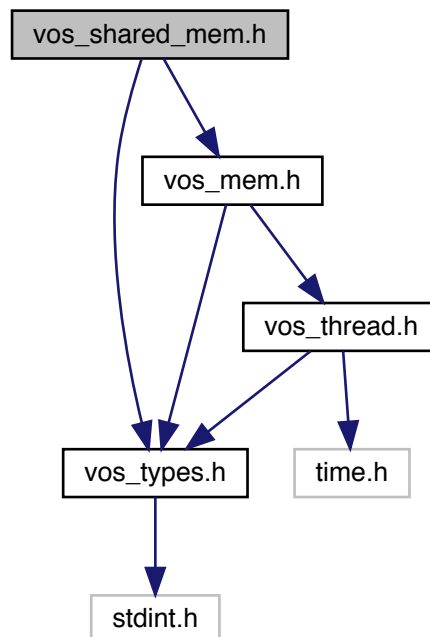
<i>0</i>	- equal
<i><0</i>	- string1 less than string 2
<i>>0</i>	- string 1 greater than string 2

5.42 vos_shared_mem.h File Reference

Shared Memory functions for OS abstraction.

```
#include "vos_types.h"
#include "vos_mem.h"
```

Include dependency graph for vos_shared_mem.h:



Functions

- EXT_DECL [VOS_ERR_T vos_sharedOpen](#) (const CHAR8 *pKey, VOS_SHRD_T *pHandle, UINT8 **ppMemoryArea, UINT32 *pSize)
Create a shared memory area or attach to existing one.
- EXT_DECL [VOS_ERR_T vos_sharedClose](#) (VOS_SHRD_T handle, const UINT8 *pMemoryArea)
Close connection to the shared memory area.

5.42.1 Detailed Description

Shared Memory functions for OS abstraction.

This module provides shared memory control supervision

Note

Project: TCNOpen TRDP prototype stack

Author

Kazumasa Aiba, TOSHIBA

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright TOSHIBA, Japan, 2013.

5.42.2 Function Documentation

5.42.2.1 vos_sharedClose()

```
EXT_DECL VOS_ERR_T vos_sharedClose (
    VOS_SHRD_T handle,
    const UINT8 * pMemoryArea )
```

Close connection to the shared memory area.

If the area was created by the calling process, the area will be closed (freed). If the area was attached, it will be detached. This function is not available in each target implementation.

Parameters

in	<i>handle</i>	Returned handle
in	<i>pMemoryArea</i>	Pointer to memory area

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_MEM_ERR</i>	no memory available

5.42.2.2 vos_sharedOpen()

```
EXT_DECL VOS_ERR_T vos_sharedOpen (
    const CHAR8 * pKey,
    VOS_SHRD_T * pHandle,
    UINT8 ** ppMemoryArea,
    UINT32 * pSize )
```

Create a shared memory area or attach to existing one.

The first call with the a specified key will create a shared memory area with the supplied size and will return a handle and a pointer to that area. If the area already exists, the area will be opened. This function is not available in each target implementation.

Parameters

in	<i>pKey</i>	Unique identifier (file name)
out	<i>pHandle</i>	Pointer to returned handle
out	<i>ppMemoryArea</i>	Pointer to pointer to memory area
in, out	<i>pSize</i>	Pointer to size of area to allocate, on return actual size after attach

Return values

<i>VOS_NO_ERR</i>	no error
-------------------	----------

Macros

- #define `VOS_MAX_SOCKET_CNT` 4
The maximum number of sockets influences memory usage; for small systems we should define a smaller set.
- #define `VOS_MAX_MULTICAST_CNT` 5
The maximum number of multicast groups one socket can join.
- #define `VOS_TTL_MULTICAST` 64
The maximum number of hops a multicast packet can take.
- #define `VOS_MAX_IF_NAME_SIZE` 16
The maximum number of IP interface adapters that can be handled by VOS.
- #define `VOS_MAX_NUM_IF` 8
The maximum number of unicast addresses that can be handled by VOS.
- #define `VOS_MAX_NUM_UNICAST` 10
The MAC size supported by VOS.
- #define `VOS_MAC_SIZE` 6
Size of socket send and receive buffer.
- #define `VOS_INVALID_SOCKET` -1
Invalid socket number.

Functions

- EXT_DECL UINT16 `vos_htons` (UINT16 val)
Byte swapping 2 Bytes.
- EXT_DECL UINT16 `vos_ntohs` (UINT16 val)
Byte swapping 2 Bytes.
- EXT_DECL UINT32 `vos_htonl` (UINT32 val)
Byte swapping 4 Bytes.
- EXT_DECL UINT32 `vos_ntohl` (UINT32 val)
Byte swapping 4 Bytes.
- EXT_DECL UINT64 `voshtonll` (UINT64 val)
Byte swapping 8 Bytes.
- EXT_DECL UINT64 `vosntohl` (UINT64 val)
Byte swapping 8 Bytes.
- EXT_DECL UINT32 `vos_dottedIP` (const CHAR8 *pDottedIP)
Convert IP address from dotted dec.
- EXT_DECL const CHAR8 * `vos_ipDotted` (UINT32 ipAddress)
Convert IP address to dotted dec.
- EXT_DECL BOOL8 `vos_isMulticast` (UINT32 ipAddress)
Check if the supplied address is a multicast group address.
- EXT_DECL `VOS_ERR_T` `vos_getInterfaces` (UINT32 *pAddrCnt, `VOS_IF_REC_T` ifAddrs[])
Get a list of interface addresses The caller has to provide an array of interface records to be filled.
- EXT_DECL BOOL8 `vos_netIfUp` (`VOS_IP4_ADDR_T` ifAddress)
Get the state of an interface.
- EXT_DECL INT32 `vos_select` (SOCKET highDesc, `VOS_FDS_T` *pReadableFD, `VOS_FDS_T` *pWriteableFD, `VOS_FDS_T` *pErrorFD, `VOS_TIMEVAL_T` *pTimeOut)
select function.
- EXT_DECL `VOS_ERR_T` `vos_sockInit` (void)
Initialize the socket library.
- EXT_DECL void `vos_sockTerm` (void)
De-Initialize the socket library.

- EXT_DECL [VOS_ERR_T vos_sockGetMAC](#) (UINT8 pMAC[VOS_MAC_SIZE])
Return the MAC address of the default adapter.
- EXT_DECL [VOS_ERR_T vos_sockOpenUDP](#) (SOCKET *pSock, const [VOS_SOCKET_OPT_T](#) *pOptions)
Create an UDP socket.
- EXT_DECL [VOS_ERR_T vos_sockOpenTCP](#) (SOCKET *pSock, const [VOS_SOCKET_OPT_T](#) *pOptions)
Create a TCP socket.
- EXT_DECL [VOS_ERR_T vos_sockClose](#) (SOCKET sock)
Close a socket.
- EXT_DECL [VOS_ERR_T vos_sockSetOptions](#) (SOCKET sock, const [VOS_SOCKET_OPT_T](#) *pOptions)
Set socket options.
- EXT_DECL [VOS_ERR_T vos_sockJoinMC](#) (SOCKET sock, UINT32 mcAddress, UINT32 ipAddress)
Join a multicast group.
- EXT_DECL [VOS_ERR_T vos_sockLeaveMC](#) (SOCKET sock, UINT32 mcAddress, UINT32 ipAddress)
Leave a multicast group.
- EXT_DECL [VOS_ERR_T vos_sockSendUDP](#) (SOCKET sock, const UINT8 *pBuffer, UINT32 *pSize, UINT32 ipAddress, UINT16 port)
Send UDP data.
- EXT_DECL [VOS_ERR_T vos_sockReceiveUDP](#) (SOCKET sock, UINT8 *pBuffer, UINT32 *pSize, UINT32 *pSrcIPAddr, UINT16 *pSrcIPPort, UINT32 *pDstIPAddr, BOOL8 peek)
Receive UDP data.
- EXT_DECL [VOS_ERR_T vos_sockBind](#) (SOCKET sock, UINT32 ipAddress, UINT16 port)
Bind a socket to an address and port.
- EXT_DECL [VOS_ERR_T vos_sockListen](#) (SOCKET sock, UINT32 backlog)
Listen for incoming TCP connections.
- EXT_DECL [VOS_ERR_T vos_sockAccept](#) (SOCKET sock, SOCKET *pSock, UINT32 *pIPAddr, UINT16 *pPort)
Accept an incoming TCP connection.
- EXT_DECL [VOS_ERR_T vos_sockConnect](#) (SOCKET sock, UINT32 ipAddress, UINT16 port)
Open a TCP connection.
- EXT_DECL [VOS_ERR_T vos_sockSendTCP](#) (SOCKET sock, const UINT8 *pBuffer, UINT32 *pSize)
Send TCP data.
- EXT_DECL [VOS_ERR_T vos_sockReceiveTCP](#) (SOCKET sock, UINT8 *pBuffer, UINT32 *pSize)
Receive TCP data.
- EXT_DECL [VOS_ERR_T vos_sockSetMulticastIf](#) (SOCKET sock, UINT32 mclAddress)
Set Using Multicast I/F.
- EXT_DECL [VOS_IP4_ADDR_T vos_determineBindAddr](#) (VOS_IP4_ADDR_T srcIP, VOS_IP4_ADDR_T mcGroup, VOS_IP4_ADDR_T rcvMostly)
Determines the address to bind to since the behaviour in the different OS is different.

5.43.1 Detailed Description

Typedefs for OS abstraction.

This is the declaration for the OS independend socket interface

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.43.2 Macro Definition Documentation**5.43.2.1 VOS_MAX_SOCKET_CNT**

```
#define VOS_MAX_SOCKET_CNT 4
```

The maximum number of sockets influences memory usage; for small systems we should define a smaller set.

The maximum number of concurrent usable sockets per application session

5.43.2.2 VOS_TTL_MULTICAST

```
#define VOS_TTL_MULTICAST 64
```

The maximum number of hops a multicast packet can take.

The maximum size for the interface name

5.43.3 Function Documentation**5.43.3.1 vos_determineBindAddr()**

```
EXT_DECL VOS_IP4_ADDR_T vos_determineBindAddr (
    VOS_IP4_ADDR_T srcIP,
    VOS_IP4_ADDR_T mcGroup,
    VOS_IP4_ADDR_T rcvMostly )
```

Determines the address to bind to since the behaviour in the different OS is different.

Parameters

in	<i>srcIP</i>	IP to bind to (0 = any address)
in	<i>mcGroup</i>	MC group to join (0 = do not join)
in	<i>rcvMostly</i>	primarily used for receiving (tbd: bind on sender, too?)

Return values

<i>Address</i>	to bind to
----------------	------------

5.43.3.2 vos_dottedIP()

```
EXT_DECL UINT32 vos_dottedIP (
    const CHAR8 * pDottedIP )
```

Convert IP address from dotted dec.

to !host! endianness

Parameters

in	<i>p↔ DottedIP</i>	IP address as dotted decimal.
----	------------------------	-------------------------------

Return values

<i>address</i>	in UINT32 in host endianness
----------------	------------------------------

5.43.3.3 vos_getInterfaces()

```
EXT_DECL VOS_ERR_T vos_getInterfaces (
    UINT32 * pAddrCnt,
    VOS_IF_REC_T ifAddrs[] )
```

Get a list of interface addresses The caller has to provide an array of interface records to be filled.

Parameters

in, out	<i>pAddrCnt</i>	in: pointer to array size of interface record out: pointer to number of interface records read
in, out	<i>ifAddrs</i>	array of interface records

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	<i>pAddrCnt</i> and/or <i>ifAddrs</i> == NULL
<i>VOS_MEM_ERR</i>	memory allocation error
<i>VOS SOCK_ERR</i>	GetAdaptersInfo() error

5.43.3.4 vos_htonl()

```
EXT_DECL UINT32 vos_htonl (
    UINT32 val )
```

Byte swapping 4 Bytes.

Parameters

in	val	Initial value.
----	-----	----------------

Return values

swapped	value
---------	-------

5.43.3.5 vos_htonll()

```
EXT_DECL UINT64 vos_htonll (
    UINT64 val )
```

Byte swapping 8 Bytes.

Parameters

in	val	Initial value.
----	-----	----------------

Return values

swapped	value
---------	-------

5.43.3.6 vos_htons()

```
EXT_DECL UINT16 vos_htons (
    UINT16 val )
```

Byte swapping 2 Bytes.

Parameters

in	val	Initial value.
----	-----	----------------

Return values

swapped	value
---------	-------

5.43.3.7 vos_ipDotted()

```
EXT_DECL const CHAR8* vos_ipDotted (
    UINT32 ipAddress )
```

Convert IP address to dotted dec.

from !host! endianness

Parameters

in	<i>ipAddress</i>	address in UINT32 in host endianness
----	------------------	--------------------------------------

Return values

<i>IP</i>	address as dotted decimal.
-----------	----------------------------

5.43.3.8 vos_isMulticast()

```
EXT_DECL BOOL8 vos_isMulticast (
    UINT32 ipAddress )
```

Check if the supplied address is a multicast group address.

Parameters

in	<i>ipAddress</i>	IP address to check.
----	------------------	----------------------

Return values

<i>TRUE</i>	address is a multicast address
<i>FALSE</i>	address is not a multicast address

5.43.3.9 vos_netIfUp()

```
EXT_DECL BOOL8 vos_netIfUp (
    VOS_IP4_ADDR_T ifAddress )
```

Get the state of an interface.

Parameters

in	<i>ifAddress</i>	address of interface to check
----	------------------	-------------------------------

Return values

<i>TRUE</i>	interface is up and ready
<i>FALSE</i>	interface is down / not ready

5.43.3.10 vos_ntohl()

```
EXT_DECL UINT32 vos_ntohl (  
    UINT32 val )
```

Byte swapping 4 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.43.3.11 vos_ntohll()

```
EXT_DECL UINT64 vos_ntohll (  
    UINT64 val )
```

Byte swapping 8 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.43.3.12 vos_ntohs()

```
EXT_DECL UINT16 vos_ntohs (  
    UINT16 val )
```

Byte swapping 2 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.43.3.13 vos_select()

```
EXT_DECL INT32 vos_select (
    SOCKET highDesc,
    VOS_FDS_T * pReadableFD,
    VOS_FDS_T * pWriteableFD,
    VOS_FDS_T * pErrorFD,
    VOS_TIMEVAL_T * pTimeOut )
```

select function.

Set the ready sockets in the supplied sets. Note: Some target systems might define this function as NOP.

Parameters

<i>in</i>	<i>highDesc</i>	max. socket descriptor + 1
<i>in, out</i>	<i>pReadableFD</i>	pointer to readable socket set
<i>in, out</i>	<i>pWriteableFD</i>	pointer to writeable socket set
<i>in, out</i>	<i>pErrorFD</i>	pointer to error socket set
<i>in</i>	<i>pTimeOut</i>	pointer to time out value

Return values

<i>number</i>	of ready file descriptors
---------------	---------------------------

5.43.3.14 vos_sockAccept()

```
EXT_DECL VOS_ERR_T vos_sockAccept (
    SOCKET sock,
    SOCKET * pSock,
    UINT32 * pIPAddress,
    UINT16 * pPort )
```

Accept an incoming TCP connection.

Accept incoming connections on the provided socket. May block and will return a new socket descriptor when accepting a connection. The original socket *pSock, remains open.

Parameters

in	<i>sock</i>	Socket descriptor
out	<i>pSock</i>	Pointer to socket descriptor, on exit new socket
out	<i>pIPAddress</i>	source IP to receive on, 0 for any
out	<i>pPort</i>	port to receive on, 17224 for PD

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	NULL parameter, parameter error
<i>VOS_UNKNOWN_ERR</i>	socket descriptor unknown error

5.43.3.15 vos_sockBind()

```
EXT_DECL VOS_ERR_T vos_sockBind (
    SOCKET sock,
    UINT32 ipAddress,
    UINT16 port )
```

Bind a socket to an address and port.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>ipAddress</i>	source IP to receive from, 0 for any
in	<i>port</i>	port to receive from

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_IO_ERR</i>	Input/Output error
<i>VOS_MEM_ERR</i>	resource error

5.43.3.16 vos_sockClose()

```
EXT_DECL VOS_ERR_T vos_sockClose (
    SOCKET sock )
```

Close a socket.

Release any resources acquired by this socket

Parameters

in	<i>sock</i>	socket descriptor
----	-------------	-------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	pSock == NULL

5.43.3.17 vos_sockConnect()

```
EXT_DECL VOS_ERR_T vos_sockConnect (
    SOCKET sock,
    UINT32 ipAddress,
    UINT16 port )
```

Open a TCP connection.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>ipAddress</i>	destination IP
in	<i>port</i>	destination port

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_IO_ERR</i>	Input/Output error

5.43.3.18 vos_sockGetMAC()

```
EXT_DECL VOS_ERR_T vos_sockGetMAC (
    UINT8 pMAC[VOS_MAC_SIZE] )
```

Return the MAC address of the default adapter.

Parameters

out	<i>pMAC</i>	return MAC address.
-----	-------------	---------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	pMAC == NULL

Return values

<i>VOS_SOCK_ERR</i>	socket not available or option not supported
---------------------	--

5.43.3.19 vos_sockInit()

```
EXT_DECL VOS_ERR_T vos_sockInit (  
    void )
```

Initialize the socket library.

Must be called once before any other call

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_SOCK_ERR</i>	sockets not supported

5.43.3.20 vos_sockJoinMC()

```
EXT_DECL VOS_ERR_T vos_sockJoinMC (  
    SOCKET sock,  
    UINT32 mcAddress,  
    UINT32 ipAddress )
```

Join a multicast group.

Note: Some target systems might not support this option.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>mcAddress</i>	multicast group to join
in	<i>ipAddress</i>	depicts interface on which to join, default 0 for any

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_SOCK_ERR</i>	option not supported

5.43.3.21 vos_sockLeaveMC()

```
EXT_DECL VOS_ERR_T vos_sockLeaveMC (
```

```

    SOCKET sock,
    UINT32 mcAddress,
    UINT32 ipAddress )

```

Leave a multicast group.

Note: Some target systems might not support this option.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>mcAddress</i>	multicast group to join
in	<i>ipAddress</i>	depicts interface on which to leave, default 0 for any

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS SOCK_ERR</i>	option not supported

5.43.3.22 vos_sockListen()

```

EXT_DECL VOS_ERR_T vos_sockListen (
    SOCKET sock,
    UINT32 backlog )

```

Listen for incoming TCP connections.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>backlog</i>	maximum connection attempts if system is busy

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_IO_ERR</i>	Input/Output error
<i>VOS_MEM_ERR</i>	resource error

5.43.3.23 vos_sockOpenTCP()

```

EXT_DECL VOS_ERR_T vos_sockOpenTCP (
    SOCKET * pSock,
    const VOS_SOCK_OPT_T * pOptions )

```

Create a TCP socket.

Return a socket descriptor for further calls. The socket options are optional and can be applied later.

Parameters

out	<i>pSock</i>	pointer to socket descriptor returned
in	<i>pOptions</i>	pointer to socket options (optional)

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	pSock == NULL
<i>VOS_SOCK_ERR</i>	socket not available or option not supported

5.43.3.24 vos_sockOpenUDP()

```
EXT_DECL VOS_ERR_T vos_sockOpenUDP (
    SOCKET * pSock,
    const VOS_SOCK_OPT_T * pOptions )
```

Create an UDP socket.

Return a socket descriptor for further calls. The socket options are optional and can be applied later. Note: Some target systems might not support every option.

Parameters

out	<i>pSock</i>	pointer to socket descriptor returned
in	<i>pOptions</i>	pointer to socket options (optional)

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	pSock == NULL
<i>VOS_SOCK_ERR</i>	socket not available or option not supported

5.43.3.25 vos_sockReceiveTCP()

```
EXT_DECL VOS_ERR_T vos_sockReceiveTCP (
    SOCKET sock,
    UINT8 * pBuffer,
    UINT32 * pSize )
```

Receive TCP data.

The caller must provide a sufficient sized buffer. If the supplied buffer is smaller than the bytes received, *pSize will reflect the number of copied bytes and the call should be repeated until *pSize is 0 (zero). If the socket was created in blocking-mode (default), then this call will block and will only return if data has been received or the socket was closed or an error occurred. If called in non-blocking mode, and no data is available, VOS_NODATA_ERR will be returned.

Parameters

in	<i>sock</i>	socket descriptor
out	<i>pBuffer</i>	pointer to applications data buffer
in, out	<i>pSize</i>	pointer to the received data size

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be read
<i>VOS_NODATA_ERR</i>	no data in non-blocking
<i>VOS_BLOCK_ERR</i>	call would have blocked in blocking mode

5.43.3.26 vos_sockReceiveUDP()

```
EXT_DECL VOS_ERR_T vos_sockReceiveUDP (
    SOCKET sock,
    UINT8 * pBuffer,
    UINT32 * pSize,
    UINT32 * pSrcIPAddr,
    UINT16 * pSrcIPPort,
    UINT32 * pDstIPAddr,
    BOOL8 peek )
```

Receive UDP data.

The caller must provide a sufficient sized buffer. If the supplied buffer is smaller than the bytes received, *pSize will reflect the number of copied bytes and the call should be repeated until *pSize is 0 (zero). If the socket was created in blocking-mode (default), then this call will block and will only return if data has been received or the socket was closed or an error occurred. If called in non-blocking mode, and no data is available, VOS_NODATA_ERR will be returned. If pointers are provided, source IP, source port and destination IP will be reported on return.

Parameters

in	<i>sock</i>	socket descriptor
out	<i>pBuffer</i>	pointer to applications data buffer
in, out	<i>pSize</i>	pointer to the received data size
out	<i>pSrcIPAddr</i>	pointer to source IP
out	<i>pSrcIPPort</i>	pointer to source port
out	<i>pDstIPAddr</i>	pointer to dest IP
in	<i>peek</i>	if true, leave data in queue

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be read
<i>VOS_NODATA_ERR</i>	no data
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

5.43.3.27 vos_sockSendTCP()

```
EXT_DECL VOS_ERR_T vos_sockSendTCP (
    SOCKET sock,
    const UINT8 * pBuffer,
    UINT32 * pSize )
```

Send TCP data.

Send data to the supplied address and port.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>pBuffer</i>	pointer to data to send
in, out	<i>pSize</i>	In: size of the data to send, Out: no of bytes sent

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be sent
<i>VOS_NOCONN_ERR</i>	no TCP connection
<i>VOS_BLOCK_ERR</i>	call would have blocked in blocking mode, data partially sent

5.43.3.28 vos_sockSendUDP()

```
EXT_DECL VOS_ERR_T vos_sockSendUDP (
    SOCKET sock,
    const UINT8 * pBuffer,
    UINT32 * pSize,
    UINT32 ipAddress,
    UINT16 port )
```

Send UDP data.

Send data to the given address and port.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>pBuffer</i>	pointer to data to send
in, out	<i>pSize</i>	In: size of the data to send, Out: no of bytes sent
in	<i>ipAddress</i>	destination IP
in	<i>port</i>	destination port

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_IO_ERR</i>	data could not be sent
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

5.43.3.29 vos_sockSetMulticastIf()

```
EXT_DECL VOS_ERR_T vos_sockSetMulticastIf (
    SOCKET sock,
    UINT32 mcIfAddress )
```

Set Using Multicast I/F.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>mcIfAddress</i>	using Multicast I/F Address

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error

5.43.3.30 vos_sockSetOptions()

```
EXT_DECL VOS_ERR_T vos_sockSetOptions (
    SOCKET sock,
    const VOS_SOCKET_OPT_T * pOptions )
```

Set socket options.

Note: Some target systems might not support each option.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>pOptions</i>	pointer to socket options (optional)

Return values

<code>VOS_NO_ERR</code>	no error
<code>VOS_PARAM_ERR</code>	parameter out of range/invalid

5.43.3.31 vos_sockTerm()

```
EXT_DECL void vos_sockTerm (
    void )
```

De-Initialize the socket library.

Must be called after last socket call

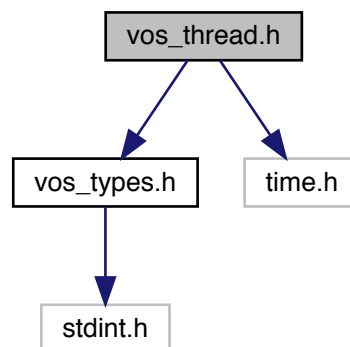
5.44 vos_thread.h File Reference

Threading functions for OS abstraction.

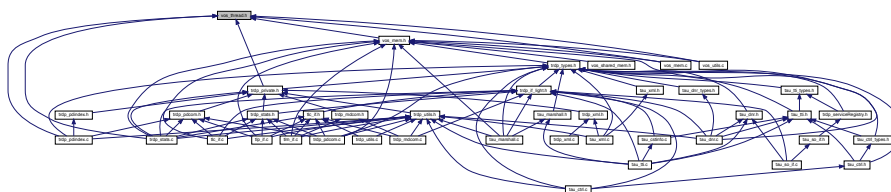
```
#include "vos_types.h"
```

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

Include dependency graph for vos_thread.h:



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



Macros

- `#define VOS_MAX_THREAD_CNT 100`
The maximum number of concurrent usable threads.
- `#define VOS_SEMA_WAIT_FOREVER 0xFFFFFFFFU`
Timeout value to wait forever for a semaphore.

Typedefs

- `typedef uint8_t VOS_THREAD_PRIORITY_T`
Thread priority range from 1 (lowest) to 255 (highest), 0 default of the target system.
- `typedef void (__cdecl * VOS_THREAD_FUNC_T) (void *pArg)`
Thread function definition.
- `typedef struct VOS_MUTEX * VOS_MUTEX_T`
Hidden mutex handle definition.
- `typedef struct VOS_SEMA * VOS_SEMA_T`
Hidden semaphore handle definition.
- `typedef void * VOS_THREAD_T`
Hidden thread handle definition.

Enumerations

- `enum VOS_THREAD_POLICY_T`
Thread policy matching pthread/Posix defines.
- `enum VOS_SEMA_STATE_T`
State of the semaphore.

Functions

- `EXT_DECL VOS_ERR_T vos_threadInit (void)`
Initialize the thread library.
- `EXT_DECL void vos_threadTerm (void)`
De-Initialize the thread library.
- `EXT_DECL VOS_ERR_T vos_threadCreateSync (VOS_THREAD_T *pThread, const CHAR8 *pName, VOS_THREAD_POLICY_T policy, VOS_THREAD_PRIORITY_T priority, uint32_t interval, VOS_TIMEVAL_T *pStartTime, uint32_t stackSize, VOS_THREAD_FUNC_T pFunction, void *pArguments)`
Create a thread.
- `EXT_DECL VOS_ERR_T vos_threadCreate (VOS_THREAD_T *pThread, const CHAR8 *pName, VOS_THREAD_POLICY_T policy, VOS_THREAD_PRIORITY_T priority, uint32_t interval, uint32_t stackSize, VOS_THREAD_FUNC_T pFunction, void *pArguments)`
Create a thread.
- `EXT_DECL VOS_ERR_T vos_threadTerminate (VOS_THREAD_T thread)`
Terminate a thread.
- `EXT_DECL VOS_ERR_T vos_threadIsActive (VOS_THREAD_T thread)`
Is the thread still active? This call will return VOS_NO_ERR if the thread is still active, VOS_PARAM_ERR in case it ran out.
- `EXT_DECL VOS_ERR_T vos_threadDelay (uint32_t delay)`
Delay the execution of the current thread by the given delay in us.
- `EXT_DECL VOS_ERR_T vos_threadSelf (VOS_THREAD_T *pThread)`

- Return thread handle of calling task.*
- EXT_DECL void [vos_getTime](#) (VOS_TIMEVAL_T *pTime)
Return the current monotonic time in sec and us.
- EXT_DECL void [vos_getRealTime](#) (VOS_TIMEVAL_T *pTime)
Return the current real time in sec and us.
- EXT_DECL const CHAR8 * [vos_getTimeStamp](#) (void)
Get a time-stamp string.
- EXT_DECL void [vos_clearTime](#) (VOS_TIMEVAL_T *pTime)
Clear the time stamp.
- EXT_DECL void [vos_addTime](#) (VOS_TIMEVAL_T *pTime, const VOS_TIMEVAL_T *pAdd)
Add the second to the first time stamp, return sum in first.
- EXT_DECL void [vos_subTime](#) (VOS_TIMEVAL_T *pTime, const VOS_TIMEVAL_T *pSub)
Subtract the second from the first time stamp, return diff in first.
- EXT_DECL INT32 [vos_cmpTime](#) (const VOS_TIMEVAL_T *pTime, const VOS_TIMEVAL_T *pCmp)
Compare the second from the first time stamp, return diff in first.
- EXT_DECL void [vos_divTime](#) (VOS_TIMEVAL_T *pTime, UINT32 divisor)
Divide the first time by the second, return quotient in first.
- EXT_DECL void [vos_mulTime](#) (VOS_TIMEVAL_T *pTime, UINT32 mul)
Multiply the first time by the second, return product in first.
- EXT_DECL void [vos_getUuid](#) (VOS_UUID_T pUuid)
Get a universal unique identifier according to RFC 4122 time based version.
- EXT_DECL VOS_ERR_T [vos_mutexCreate](#) (VOS_MUTEX_T *pMutex)
Create a mutex.
- EXT_DECL void [vos_mutexDelete](#) (VOS_MUTEX_T pMutex)
Delete a mutex.
- EXT_DECL VOS_ERR_T [vos_mutexLock](#) (VOS_MUTEX_T pMutex)
Take a mutex.
- EXT_DECL VOS_ERR_T [vos_mutexTryLock](#) (VOS_MUTEX_T pMutex)
Try to take a mutex.
- EXT_DECL VOS_ERR_T [vos_mutexUnlock](#) (VOS_MUTEX_T pMutex)
Release a mutex.
- EXT_DECL VOS_ERR_T [vos_semaCreate](#) (VOS_SEMA_T *pSema, VOS_SEMA_STATE_T initialState)
Create a semaphore.
- EXT_DECL void [vos_semaDelete](#) (VOS_SEMA_T sema)
Delete a semaphore.
- EXT_DECL VOS_ERR_T [vos_semaTake](#) (VOS_SEMA_T sema, UINT32 timeout)
Take a semaphore.
- EXT_DECL void [vos_semaGive](#) (VOS_SEMA_T sema)
Give a semaphore.

5.44.1 Detailed Description

Threading functions for OS abstraction.

Thread-, semaphore- and time-handling functions

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2014. All rights reserved.

5.44.2 Function Documentation**5.44.2.1 vos_addTime()**

```
EXT_DECL void vos_addTime (
    VOS_TIMEVAL_T * pTime,
    const VOS_TIMEVAL_T * pAdd )
```

Add the second to the first time stamp, return sum in first.

Parameters

in, out	<i>pTime</i>	Pointer to time value
in	<i>pAdd</i>	Pointer to time value

5.44.2.2 vos_clearTime()

```
EXT_DECL void vos_clearTime (
    VOS_TIMEVAL_T * pTime )
```

Clear the time stamp.

Parameters

out	<i>pTime</i>	Pointer to time value
-----	--------------	-----------------------

5.44.2.3 vos_cmpTime()

```
EXT_DECL INT32 vos_cmpTime (
    const VOS_TIMEVAL_T * pTime,
    const VOS_TIMEVAL_T * pCmp )
```

Compare the second from the first time stamp, return diff in first.

Parameters

in, out	<i>pTime</i>	Pointer to time value
in	<i>pCmp</i>	Pointer to time value to compare

Return values

0	pTime == pCmp
-1	pTime < pCmp
1	pTime > pCmp

5.44.2.4 vos_divTime()

```
EXT_DECL void vos_divTime (
    VOS_TIMEVAL_T * pTime,
    UINT32 divisor )
```

Divide the first time by the second, return quotient in first.

Parameters

in, out	<i>pTime</i>	Pointer to time value
in	<i>divisor</i>	Divisor

5.44.2.5 vos_getRealTime()

```
EXT_DECL void vos_getRealTime (
    VOS_TIMEVAL_T * pTime )
```

Return the current real time in sec and us.

Parameters

out	<i>pTime</i>	Pointer to time value
-----	--------------	-----------------------

5.44.2.6 vos_getTime()

```
EXT_DECL void vos_getTime (
    VOS_TIMEVAL_T * pTime )
```

Return the current monotonic time in sec and us.

Parameters

out	<i>pTime</i>	Pointer to time value
-----	--------------	-----------------------

5.44.2.7 vos_getTimeStamp()

```
EXT_DECL const CHAR8* vos_getTimeStamp (
    void )
```

Get a time-stamp string.

Get a time-stamp string for debugging in the form "yyyymmdd-hh:mm:ss.ms" Depending on the used OS / hardware the time might not be a real-time stamp but relative from start of system.

Return values

<i>timestamp</i>	"yyyymmdd-hh:mm:ss.ms"
------------------	------------------------

5.44.2.8 vos_getUuid()

```
EXT_DECL void vos_getUuid (
    VOS_UUID_T pUuid )
```

Get a universal unique identifier according to RFC 4122 time based version.

Parameters

out	<i>pUuid</i>	Pointer to a universal unique identifier
-----	--------------	--

5.44.2.9 vos_mulTime()

```
EXT_DECL void vos_mulTime (
    VOS_TIMEVAL_T * pTime,
    UINT32 mul )
```

Multiply the first time by the second, return product in first.

Parameters

in, out	<i>pTime</i>	Pointer to time value
in	<i>mul</i>	Factor

5.44.2.10 vos_mutexCreate()

```
EXT_DECL VOS_ERR_T vos_mutexCreate (
    VOS_MUTEX_T * pMutex )
```

Create a mutex.

Return a mutex handle. The mutex will be available at creation.

Parameters

out	<i>pMutex</i>	Pointer to mutex handle
-----	---------------	-------------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_PARAM_ERR</i>	pMutex == NULL
<i>VOS_MUTEX_ERR</i>	no mutex available

5.44.2.11 vos_mutexDelete()

```
EXT_DECL void vos_mutexDelete (
    VOS_MUTEX_T pMutex )
```

Delete a mutex.

Release the resources taken by the mutex.

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
-------------------	----------

5.44.2.12 vos_mutexLock()

```
EXT_DECL VOS_ERR_T vos_mutexLock (
    VOS_MUTEX_T pMutex )
```

Take a mutex.

Wait for the mutex to become available (lock).

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle

5.44.2.13 vos_mutexTryLock()

```
EXT_DECL VOS_ERR_T vos_mutexTryLock (  
    VOS_MUTEX_T pMutex )
```

Try to take a mutex.

If mutex is can't be taken VOS_MUTEX_ERR is returned.

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_MUTEX_ERR</i>	no mutex available

5.44.2.14 vos_mutexUnlock()

```
EXT_DECL VOS_ERR_T vos_mutexUnlock (  
    VOS_MUTEX_T pMutex )
```

Release a mutex.

Unlock the mutex.

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

5.44.2.15 vos_semaCreate()

```
EXT_DECL VOS_ERR_T vos_semaCreate (
    VOS_SEMA_T * pSema,
    VOS_SEMA_STATE_T initialState )
```

Create a semaphore.

Return a semaphore handle. Depending on the initial state the semaphore will be available on creation or not.

Parameters

out	<i>pSema</i>	Pointer to semaphore handle
in	<i>initialState</i>	The initial state of the semaphore

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_SEMA_ERR</i>	no semaphore available

5.44.2.16 vos_semaDelete()

```
EXT_DECL void vos_semaDelete (
    VOS_SEMA_T sema )
```

Delete a semaphore.

This will eventually release any processes waiting for the semaphore.

Parameters

in	<i>sema</i>	semaphore handle
----	-------------	------------------

5.44.2.17 vos_semaGive()

```
EXT_DECL void vos_semaGive (
    VOS_SEMA_T sema )
```

Give a semaphore.

Release (increase) a semaphore.

Parameters

in	<i>sema</i>	semaphore handle
----	-------------	------------------

5.44.2.18 vos_semaTake()

```
EXT_DECL VOS_ERR_T vos_semaTake (
    VOS_SEMA_T sema,
    UINT32 timeout )
```

Take a semaphore.

Try to get (decrease) a semaphore.

Parameters

in	<i>sema</i>	semaphore handle
in	<i>timeout</i>	Max. time in us to wait, 0 means no wait

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_SEMA_ERR</i>	could not get semaphore in time

5.44.2.19 vos_subTime()

```
EXT_DECL void vos_subTime (
    VOS_TIMEVAL_T * pTime,
    const VOS_TIMEVAL_T * pSub )
```

Subtract the second from the first time stamp, return diff in first.

Parameters

in, out	<i>pTime</i>	Pointer to time value
in	<i>pSub</i>	Pointer to time value

5.44.2.20 vos_threadCreate()

```
EXT_DECL VOS_ERR_T vos_threadCreate (
    VOS_THREAD_T * pThread,
    const CHAR8 * pName,
    VOS_THREAD_POLICY_T policy,
```



```

VOS_THREAD_PRIORITY_T priority,
UINT32 interval,
UINT32 stackSize,
VOS_THREAD_FUNC_T pFunction,
void * pArguments )

```

Create a thread.

Create a thread and return a thread handle for further requests. Not each parameter may be supported by all target systems!

Parameters

out	<i>pThread</i>	Pointer to returned thread handle
in	<i>pName</i>	Pointer to name of the thread (optional)
in	<i>policy</i>	Scheduling policy (FIFO, Round Robin or other)
in	<i>priority</i>	Scheduling priority (1...255 (highest), default 0)
in	<i>interval</i>	Interval for cyclic threads in us (optional)
in	<i>stackSize</i>	Minimum stacksize, default 0: 16kB
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	Pointer to the thread function parameters

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid

5.44.2.21 vos_threadCreateSync()

```

EXT_DECL VOS_ERR_T vos_threadCreateSync (
    VOS_THREAD_T * pThread,
    const CHAR8 * pName,
    VOS_THREAD_POLICY_T policy,
    VOS_THREAD_PRIORITY_T priority,
    UINT32 interval,
    VOS_TIMEVAL_T * pStartTime,
    UINT32 stackSize,
    VOS_THREAD_FUNC_T pFunction,
    void * pArguments )

```

Create a thread.

Create a thread and return a thread handle for further requests. Not each parameter may be supported by all target systems!

Parameters

out	<i>pThread</i>	Pointer to returned thread handle
in	<i>pName</i>	Pointer to name of the thread (optional)

Parameters

in	<i>policy</i>	Scheduling policy (FIFO, Round Robin or other)
in	<i>priority</i>	Scheduling priority (1...255 (highest), default 0)
in	<i>interval</i>	Interval for cyclic threads in us (optional)
in	<i>pStartTime</i>	Starting time for cyclic threads
in	<i>stackSize</i>	Minimum stacksize, default 0: 16kB
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	Pointer to the thread function parameters

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid

5.44.2.22 vos_threadDelay()

```
EXT_DECL VOS_ERR_T vos_threadDelay (
    UINT32 delay )
```

Delay the execution of the current thread by the given delay in us.

Parameters

in	<i>delay</i>	Delay in us
----	--------------	-------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised

5.44.2.23 vos_threadInit()

```
EXT_DECL VOS_ERR_T vos_threadInit (
    void )
```

Initialize the thread library.

Must be called once before any other call

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	threading not supported

5.44.2.24 vos_threadIsActive()

```
EXT_DECL VOS_ERR_T vos_threadIsActive (
    VOS_THREAD_T thread )
```

Is the thread still active? This call will return VOS_NO_ERR if the thread is still active, VOS_PARAM_ERR in case it ran out.

Parameters

in	<i>thread</i>	Thread handle
----	---------------	---------------

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid

5.44.2.25 vos_threadSelf()

```
EXT_DECL VOS_ERR_T vos_threadSelf (
    VOS_THREAD_T * pThread )
```

Return thread handle of calling task.

Parameters

out	<i>pThread</i>	pointer to thread handle
-----	----------------	--------------------------

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	parameter out of range/invalid

5.44.2.26 vos_threadTerm()

```
EXT_DECL void vos_threadTerm (
    void )
```

De-Initialize the thread library.

Must be called after last thread/timer call

5.44.2.27 vos_threadTerminate()

```
EXT_DECL VOS_ERR_T vos_threadTerminate (
    VOS_THREAD_T thread )
```

Terminate a thread.

This call will terminate the thread with the given threadId and release all resources. Depending on the underlying architectures, it may just block until the thread ran out.

Parameters

in	<i>thread</i>	Thread handle (or NULL if current thread)
----	---------------	---

Return values

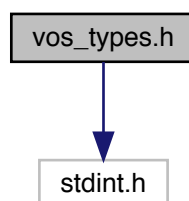
<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid

5.45 vos_types.h File Reference

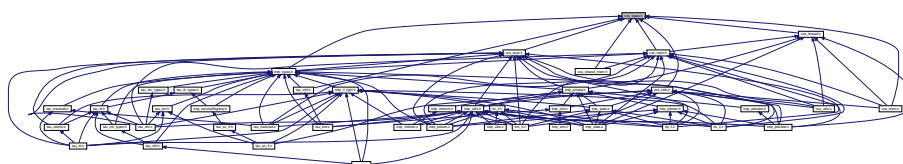
Typedefs for OS abstraction.

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

Include dependency graph for vos_types.h:



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



Data Structures

- struct [VOS_VERSION_T](#)
Version information.

Macros

- #define [INLINE](#) inline
inline macros
- #define [AV_ERROR](#) 0x00
ANTIVALENT8 values.
- #define [TR_DIR1](#) 0x01
Directions/Orientations.

Typedefs

- typedef UINT8 [VOS_UUID_T](#)[16]
universal unique identifier according to RFC 4122, time based version
- typedef struct timeval [VOS_TIMEVAL_T](#)
Timer value compatible with timeval / select.
- typedef void(* [VOS_PRINT_DBG_T](#)) (void *pRefCon, [VOS_LOG_T](#) category, const CHAR8 *pTime, const CHAR8 *pFile, UINT16 LineNumber, const CHAR8 *pMsgStr)
Function definition for error/debug output.

Enumerations

- enum [VOS_ERR_T](#) {
[VOS_NO_ERR](#) = 0,
[VOS_PARAM_ERR](#) = -1,
[VOS_INIT_ERR](#) = -2,
[VOS_NOINIT_ERR](#) = -3,
[VOS_TIMEOUT_ERR](#) = -4,
[VOS_NODATA_ERR](#) = -5,
[VOS SOCK_ERR](#) = -6,
[VOS_IO_ERR](#) = -7,
[VOS_MEM_ERR](#) = -8,
[VOS_SEMA_ERR](#) = -9,
[VOS_QUEUE_ERR](#) = -10,
[VOS_QUEUE_FULL_ERR](#) = -11,
[VOS_MUTEX_ERR](#) = -12,
[VOS_THREAD_ERR](#) = -13,
[VOS_BLOCK_ERR](#) = -14,
[VOS_INTEGRATION_ERR](#) = -15,
[VOS_NOCONN_ERR](#) = -16,
[VOS_INUSE_ERR](#) = -49,
[VOS_UNKNOWN_ERR](#) = -99 }
Return codes for all VOS API functions.
- enum [VOS_LOG_T](#) {
[VOS_LOG_ERROR](#) = 0,
[VOS_LOG_WARNING](#) = 1,
[VOS_LOG_INFO](#) = 2,
[VOS_LOG_DBG](#) = 3,
[VOS_LOG_USR](#) = 4 }
Categories for logging.

5.45.1 Detailed Description

Typedefs for OS abstraction.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.45.2 Typedef Documentation

5.45.2.1 VOS_PRINT_DBG_T

```
typedef void(* VOS_PRINT_DBG_T) (void *pRefCon, VOS_LOG_T category, const CHAR8 *pTime, const CHAR8 *pFile, UINT16 LineNumber, const CHAR8 *pMsgStr)
```

Function definition for error/debug output.

The function will be called for logging and error message output. The user can decide, what kind of info will be logged by filtering the category.

Parameters

in	<i>pRefCon</i>	pointer to user context
in	<i>category</i>	Log category (Error, Warning, Info etc.)
in	<i>pTime</i>	pointer to NULL-terminated string of time stamp
in	<i>pFile</i>	pointer to NULL-terminated string of source module
in	<i>LineNumber</i>	Line number
in	<i>pMsgStr</i>	pointer to NULL-terminated string

5.45.2.2 VOS_TIMEVAL_T

```
typedef struct timeval VOS_TIMEVAL_T
```

Timer value compatible with timeval / select.

Relative or absolute date, depending on usage Assume 32 Bit system, if not defined

5.45.3 Enumeration Type Documentation

5.45.3.1 VOS_ERR_T

enum [VOS_ERR_T](#)

Return codes for all VOS API functions.

Enumerator

VOS_NO_ERR	No error.
VOS_PARAM_ERR	Necessary parameter missing or out of range.
VOS_INIT_ERR	Call without valid initialization.
VOS_NOINIT_ERR	The supplied handle/reference is not valid.
VOS_TIMEOUT_ERR	Timeout.
VOS_NODATA_ERR	Non blocking mode: no data received.
VOS_SOCKET_ERR	Socket option not supported.
VOS_IO_ERR	Socket IO error, data can't be received/sent.
VOS_MEM_ERR	No more memory available.
VOS_SEMA_ERR	Semaphore not available.
VOS_QUEUE_ERR	Queue empty.
VOS_QUEUE_FULL_ERR	Queue full.
VOS_MUTEX_ERR	Mutex not available.
VOS_THREAD_ERR	Thread creation error.
VOS_BLOCK_ERR	System call would have blocked in blocking mode.
VOS_INTEGRATION_ERR	Alignment or endianness for selected target wrong.
VOS_NOCONN_ERR	No TCP connection.
VOS_INUSE_ERR	Resource is still in use.
VOS_UNKNOWN_ERR	Unknown error.

5.45.3.2 VOS_LOG_T

enum [VOS_LOG_T](#)

Categories for logging.

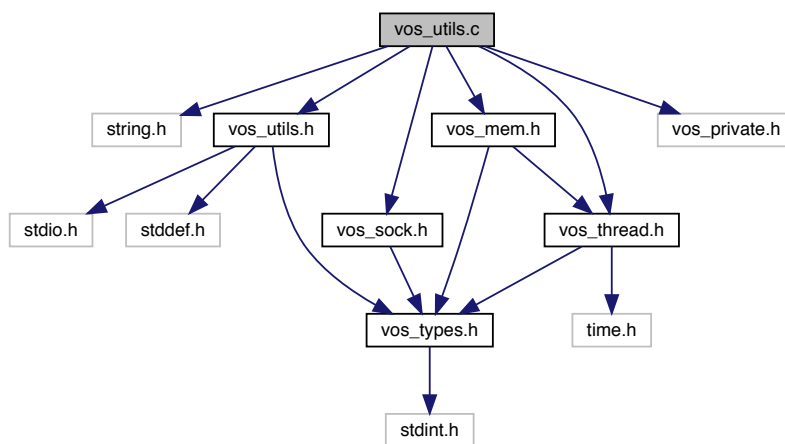
Enumerator

VOS_LOG_ERROR	This is a critical error.
VOS_LOG_WARNING	This is a warning.
VOS_LOG_INFO	This is an info.
VOS_LOG_DBG	This is a debug info.
VOS_LOG_USR	This is a user info.

5.46 vos_utils.c File Reference

Common functions for VOS.

```
#include <string.h>
#include "vos_utils.h"
#include "vos_sock.h"
#include "vos_thread.h"
#include "vos_mem.h"
#include "vos_private.h"
Include dependency graph for vos_utils.c:
```



Functions

- int [vos_hostIsBigEndian](#) ()
Return 1 if this is a big endian machine.
- [VOS_ERR_T vos_init](#) (void *pRefCon, [VOS_PRINT_DBG_T](#) pDebugOutput)
Initialize the virtual operating system.
- EXT_DECL void [vos_terminate](#) (void)
DeInitialize the vos library.
- UINT32 [vos_crc32](#) (UINT32 crc, const UINT8 *pData, UINT32 dataLen)
Compute crc32 according to IEEE802.3.
- UINT32 [vos_sc32](#) (UINT32 crc, const UINT8 *pData, UINT32 dataLen)
Compute crc32 according to IEC 61375-2-3 B.7.
- const char * [vos_getVersionString](#) (void)
Return a human readable version representation.
- EXT_DECL const [VOS_VERSION_T](#) * [vos_getVersion](#) (void)
Return version.
- EXT_DECL const CHAR8 * [vos_getErrorString](#) ([VOS_ERR_T](#) error)
Return a human readable error representation.

5.46.1 Detailed Description

Common functions for VOS.

Common functions of the abstraction layer. Mainly debugging support.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

5.46.2 Function Documentation

5.46.2.1 vos_crc32()

```
UINT32 vos_crc32 (
    UINT32 crc,
    const UINT8 * pData,
    UINT32 dataLen )
```

Compute crc32 according to IEEE802.3.

Calculate CRC for the given buffer and length.

/ to IEC 61375-2-3 A.3 Note: Returned CRC is inverted

Parameters

in	<i>crc</i>	Initial value.
in, out	<i>pData</i>	Pointer to data.
in	<i>dataLen</i>	length in bytes of data.

Return values

<i>crc32</i>	according to IEEE802.3
--------------	------------------------

5.46.2.2 vos_getErrorString()

```
EXT_DECL const CHAR8* vos_getErrorString (
    VOS_ERR_T error )
```

Return a human readable error representation.

Parameters

in	<i>error</i>	The TRDP or VOS error code
----	--------------	----------------------------

Return values

<i>const</i>	string pointer to error string
--------------	--------------------------------

5.46.2.3 vos_getVersion()

```
EXT_DECL const VOS_VERSION_T* vos_getVersion (
    void )
```

Return version.

Return pointer to version structure

Return values

<i>VOS_VERSION_T</i>	
----------------------	--

5.46.2.4 vos_getVersionString()

```
const char* vos_getVersionString (
    void )
```

Return a human readable version representation.

Return string in the form 'v.r.u.b'

Return values

<i>const</i>	string
--------------	--------

5.46.2.5 vos_hostIsBigEndian()

```
int vos_hostIsBigEndian (
    void )
```

Return 1 if this is a big endian machine.

Return values

0	if machine is little endian
1	if machine is big endian

5.46.2.6 vos_init()

```
VOS_ERR_T vos_init (
    void * pRefCon,
    VOS_PRINT_DBG_T pDebugOutput )
```

Initialize the virtual operating system.

Initialize the vos library.

Parameters

in	<i>pRefCon</i>	context for debug output function
in	<i>pDebugOutput</i>	Pointer to debug output function.

Return values

<i>VOS_NO_ERR</i>	no error VOS_INTEGRATION_ERR if endianness/alignment mismatch VOS_SOCK_ERR sockets not supported VOS_UNKNOWN_ERR initialisation error
-------------------	--

5.46.2.7 vos_sc32()

```
UINT32 vos_sc32 (
    UINT32 crc,
    const UINT8 * pData,
    UINT32 dataLen )
```

Compute crc32 according to IEC 61375-2-3 B.7.

Compute crc32 according to IEC 61375-2-3 B.7 Note: Returned CRC is inverted.

Parameters

in	<i>crc</i>	Initial value.
in, out	<i>pData</i>	Pointer to data.
in	<i>dataLen</i>	length in bytes of data.

Return values

sc32	according to IEC 61375-2-3
------	----------------------------

5.46.2.8 vos_terminate()

```
EXT_DECL void vos_terminate (
    void )
```

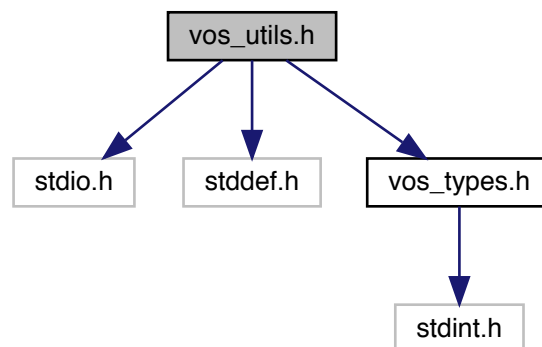
Deinitialize the vos library.

Should be called last after TRDP stack/application does not use any VOS function anymore.

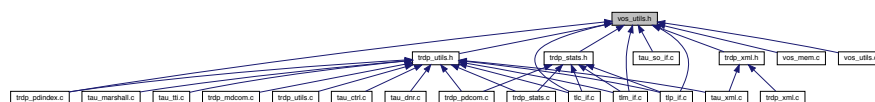
5.47 vos_utils.h File Reference

Typedefs for OS abstraction.

```
#include <stdio.h>
#include <stddef.h>
#include "vos_types.h"
Include dependency graph for vos_utils.h:
```



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



Macros

- #define `VOS_MAX_PRNT_STR_SIZE` 256u
String size definitions for the debug output functions.
- #define `VOS_MAX_FRMT_SIZE` 64u
Max.
- #define `VOS_MAX_ERR_STR_SIZE` (`VOS_MAX_PRNT_STR_SIZE` - `VOS_MAX_FRMT_SIZE`)
Max.
- #define `VOS_DIR_SEP` '/'
This is a helper define for separating a path in debug output.
- #define `vos_snprintf`(str, size, format, args ...) `snprintf`(str, size, format, ## args) /*lint !e586 logging output needed */
Safe printf function.
- #define `vos_printLogStr`(level, string)
Debug output macro without formatting options.
- #define `vos_printLog`(level, format, args ...)
Debug output macro with formatting options.
- #define `ALIGNOF`(type) ((UINT32)offsetof(struct { char c; type member; }, member))
Alignment macros.
- #define `INITFCS` 0xffffffffu
CRC/FCS constants.
- #define `SIZE_OF_FCS` 4u
for better understanding of address calculations
- #define `L_ENDIAN`
Define endianness if not already done by compiler.

Functions

- EXT_DECL int `vos_hostIsBigEndian` (void)
Return 1 if this is a big endian machine.
- EXT_DECL UINT32 `vos_crc32` (UINT32 crc, const UINT8 *pData, UINT32 dataLen)
Calculate CRC for the given buffer and length.
- EXT_DECL UINT32 `vos_sc32` (UINT32 crc, const UINT8 *pData, UINT32 dataLen)
Compute crc32 according to IEC 61375-2-3 B.7 Note: Returned CRC is inverted.
- EXT_DECL `VOS_ERR_T` `vos_init` (void *pRefCon, `VOS_PRINT_DBG_T` pDebugOutput)
Initialize the vos library.
- EXT_DECL void `vos_terminate` (void)
DeInitialize the vos library.
- EXT_DECL const CHAR8 * `vos_getVersionString` (void)
Return a human readable version representation.
- EXT_DECL const `VOS_VERSION_T` * `vos_getVersion` (void)
Return version.
- EXT_DECL const CHAR8 * `vos_getErrorString` (`VOS_ERR_T` error)
Return a human readable error representation.

5.47.1 Detailed Description

Typedefs for OS abstraction.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2018. All rights reserved.

5.47.2 Macro Definition Documentation

5.47.2.1 INITFCS

```
#define INITFCS 0xffffffffu
```

CRC/FCS constants.

Initial FCS value

5.47.2.2 VOS_MAX_ERR_STR_SIZE

```
#define VOS_MAX_ERR_STR_SIZE (VOS_MAX_PRNT_STR_SIZE - VOS_MAX_FRMT_SIZE)
```

Max.

size of the error part

5.47.2.3 VOS_MAX_FRMT_SIZE

```
#define VOS_MAX_FRMT_SIZE 64u
```

Max.

size of the 'format' part

5.47.2.4 VOS_MAX_PRNT_STR_SIZE

```
#define VOS_MAX_PRNT_STR_SIZE 256u
```

String size definitions for the debug output functions.

Max. size of the debug/error string of debug function

5.47.3 Function Documentation

5.47.3.1 vos_crc32()

```
EXT_DECL UINT32 vos_crc32 (
    UINT32 crc,
    const UINT8 * pData,
    UINT32 dataLen )
```

Calculate CRC for the given buffer and length.

For TRDP FCS CRC calculation the CRC32 according to IEEE802.3 with start value 0xffffffff is used. Note↔
: Returned CRC is inverted

Parameters

in	<i>crc</i>	Initial value.
in, out	<i>pData</i>	Pointer to data.
in	<i>dataLen</i>	length in bytes of data.

Return values

<i>crc32</i>	according to IEEE802.3
--------------	------------------------

Calculate CRC for the given buffer and length.

/ to IEC 61375-2-3 A.3 Note: Returned CRC is inverted

Parameters

in	<i>crc</i>	Initial value.
in, out	<i>pData</i>	Pointer to data.
in	<i>dataLen</i>	length in bytes of data.

Return values

<i>crc32</i>	according to IEEE802.3
--------------	------------------------

5.47.3.2 vos_getErrorString()

```
EXT_DECL const CHAR8* vos_getErrorString (
    VOS_ERR_T error )
```

Return a human readable error representation.

Parameters

in	<i>error</i>	The TRDP or VOS error code
----	--------------	----------------------------

Return values

<i>const</i>	string pointer to error string
--------------	--------------------------------

5.47.3.3 vos_getVersion()

```
EXT_DECL const VOS_VERSION_T* vos_getVersion (
    void )
```

Return version.

Return pointer to version structure

Return values

<i>const</i>	VOS_VERSION↵ _T
--------------	--------------------

Return pointer to version structure

Return values

VOS_VERSION↵ _T	
--------------------	--

5.47.3.4 vos_getVersionString()

```
EXT_DECL const CHAR8* vos_getVersionString (
    void )
```

Return a human readable version representation.

Return string in the form 'v.r.u.b'

Return values

<i>const</i>	string
--------------	--------

5.47.3.5 vos_hostIsBigEndian()

```
EXT_DECL int vos_hostIsBigEndian (
    void )
```

Return 1 if this is a big endian machine.

Return values

0	if machine is little endian
1	if machine is big endian

5.47.3.6 vos_init()

```
EXT_DECL VOS_ERR_T vos_init (
    void * pRefCon,
    VOS_PRINT_DBG_T pDebugOutput )
```

Initialize the vos library.

This is used to set the output function for all VOS error and debug output.

Parameters

in	<i>pRefCon</i>	user context
in	<i>pDebugOutput</i>	pointer to debug output function

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	unsupported

Initialize the vos library.

Parameters

in	<i>pRefCon</i>	context for debug output function
in	<i>pDebugOutput</i>	Pointer to debug output function.

Return values

<i>VOS_NO_ERR</i>	no error	<i>VOS_INTEGRATION_ERR</i> if endianness/alignment mismatch	<i>VOS_SOCK_ERR</i> sockets not supported	<i>VOS_UNKNOWN_ERR</i> initialisation error
-------------------	----------	---	---	---

5.47.3.7 vos_sc32()

```
EXT_DECL UINT32 vos_sc32 (
    UINT32 crc,
    const UINT8 * pData,
    UINT32 dataLen )
```

Compute crc32 according to IEC 61375-2-3 B.7 Note: Returned CRC is inverted.

Parameters

in	<i>crc</i>	Initial value.
in, out	<i>pData</i>	Pointer to data.
in	<i>dataLen</i>	length in bytes of data.

Return values

<i>crc32</i>	according to IEC 61375-2-3
--------------	----------------------------

Compute crc32 according to IEC 61375-2-3 B.7 Note: Returned CRC is inverted.

Parameters

in	<i>crc</i>	Initial value.
in, out	<i>pData</i>	Pointer to data.
in	<i>dataLen</i>	length in bytes of data.

Return values

<i>sc32</i>	according to IEC 61375-2-3
-------------	----------------------------

5.47.3.8 vos_terminate()

```
EXT_DECL void vos_terminate (
    void )
```

DeInitialize the vos library.

Should be called last after TRDP stack/application does not use any VOS function anymore.

Index

callBack
 GNU_PACKED, 22

cnCnt
 TRDP_ETB_INFO_T, 48

cnId
 TRDP_FUNCTION_INFO_T, 49

comId
 GNU_PACKED, 22

confVehCnt
 GNU_PACKED, 22

confVehList
 GNU_PACKED, 23

cstId
 TRDP_CONSIST_INFO_T, 42

cstInfoGetPropSize
 tau_cstinfo.c, 78

cstList
 GNU_PACKED, 23

cstOwner
 TRDP_CONSIST_INFO_T, 42

cstUUID
 GNU_PACKED, 23

cstVehNo
 TRDP_FUNCTION_INFO_T, 49

DNS_HEADER, 13

datasetLength
 GNU_PACKED, 23

defQos
 GNU_PACKED, 23

defTtl
 GNU_PACKED, 24

destAddr
 GNU_PACKED, 24

deviceName
 GNU_PACKED, 24

ETB_CTRL_COMID
 iec61375-2-3.h, 73

etbId
 GNU_PACKED, 24

etbTopoCnt
 TRDP_FUNCTION_INFO_T, 49

etbTopoCnt
 GNU_PACKED, 24

fctDev
 service_info, 37

fctId
 TRDP_FUNCTION_INFO_T, 50

filterAddr
 GNU_PACKED, 24

GNU_PACKED, 13

callBack, 22

comId, 22

confVehCnt, 22

confVehList, 23

cstList, 23

cstUUID, 23

datasetLength, 23

defQos, 23

defTtl, 24

destAddr, 24

deviceName, 24

etbId, 24

etbTopoCnt, 24

filterAddr, 24

inhibit, 25

isLead, 25

leadDir, 25

leadVehOfCst, 25

lifesign, 25

msgType, 25

numCrcErr, 26

numMissed, 26

numProtErr, 26

numRcv, 26

numRecv, 26

numSend, 26

numTopoErr, 27

opCstList, 27

opTrnDirState, 27

opTrnTopoCnt, 27

opVehList, 27

ownOpCstNo, 28

protocolVersion, 28

reserved01, 28

reserved02, 28

reserved03, 29

reserved04, 29

reserved06, 29

safetyTrail, 29

serviceEntry, 29

timeout, 29

toBehav, 30

trnCstNo, 30

trnDirState, 30

trnId, 30

trnNetDir, 30

trnOperator, 30

- trnTopoCnt, [31](#)
- trnVehNo, [31](#)
- vehId, [31](#)
- vehOrient, [31](#)
- version, [31](#)
- hp_slot, [32](#)
- hp_slots, [33](#)
- INITFCS
 - vos_utils.h, [416](#)
- iec61375-2-3.h, [69](#)
 - ETB_CTRL_COMID, [73](#)
 - TRDP_ETBCTRL_DSID, [73](#)
 - TRDP_MAX_FILE_NAME_LEN, [74](#)
 - TRDP_MAX_LABEL_LEN, [74](#)
 - TRDP_MAX_MD_DATA_SIZE, [74](#)
 - TRDP_MAX_URI_HOST_LEN, [74](#)
 - TRDP_MAX_URI_LEN, [74](#)
 - TRDP_MAX_URI_USER_LEN, [74](#)
 - TRDP_MD_DEFAULT_REPLY_TIMEOUT, [75](#)
 - TRDP_MD_INFINITE_TIME, [75](#)
 - TRDP_MIN_PD_HEADER_SIZE, [75](#)
 - TRDP_MSG_PD, [75](#)
 - TRDP_PD_UDP_PORT, [75](#)
 - TRDP_PROCESS_DEFAULT_CYCLE_TIME, [75](#)
 - TRDP_USR_URI_SIZE, [76](#)
 - TTDB_NET_DIR_REQ_COMID, [76](#)
 - TTDB_OP_DIR_INFO_COMID, [76](#)
 - TTDB_STAT_CST_REQ_COMID, [76](#)
 - TTDB_TRN_DIR_REQ_COMID, [76](#)
- inhibit
 - GNU_PACKED, [25](#)
- isLead
 - GNU_PACKED, [25](#)
- leadDir
 - GNU_PACKED, [25](#)
- leadVehOfCst
 - GNU_PACKED, [25](#)
- lifesign
 - GNU_PACKED, [25](#)
- maxNoOfExtPublishers
 - TRDP_IDX_TABLE_T, [51](#)
- maxNoOfHighCatPublishers
 - TRDP_IDX_TABLE_T, [52](#)
- maxNoOfHighCatSubscriptions
 - TRDP_IDX_TABLE_T, [52](#)
- maxNoOfLowCatPublishers
 - TRDP_IDX_TABLE_T, [52](#)
- maxNoOfLowCatSubscriptions
 - TRDP_IDX_TABLE_T, [52](#)
- maxNoOfMidCatPublishers
 - TRDP_IDX_TABLE_T, [52](#)
- maxNoOfMidCatSubscriptions
 - TRDP_IDX_TABLE_T, [52](#)
- msgType
 - GNU_PACKED, [25](#)
- numCrcErr
 - GNU_PACKED, [26](#)
- numMissed
 - GNU_PACKED, [26](#)
- numProtErr
 - GNU_PACKED, [26](#)
- numRcv
 - GNU_PACKED, [26](#)
- numRecv
 - GNU_PACKED, [26](#)
- numSend
 - GNU_PACKED, [26](#)
- numTopoErr
 - GNU_PACKED, [27](#)
- opCstList
 - GNU_PACKED, [27](#)
- opTrnDirState
 - GNU_PACKED, [27](#)
- opTrnTopoCnt
 - GNU_PACKED, [27](#)
- opVehList
 - GNU_PACKED, [27](#)
- ownOpCstNo
 - GNU_PACKED, [28](#)
- PD_ELE, [34](#)
 - pFrame, [35](#)
- pFrame
 - PD_ELE, [35](#)
- printSocketUsage
 - trdp_utils.c, [319](#)
 - trdp_utils.h, [332](#)
- protocolVersion
 - GNU_PACKED, [28](#)
- reserved01
 - GNU_PACKED, [28](#)
- reserved02
 - GNU_PACKED, [28](#)
- reserved03
 - GNU_PACKED, [29](#)
- reserved04
 - GNU_PACKED, [29](#)
- reserved06
 - GNU_PACKED, [29](#)
- SOA_SAME_SERVICEID
 - trdp_serviceRegistry.h, [295](#)
- SRM_SERVICE_READ_REQ_COMID
 - trdp_serviceRegistry.h, [295](#)
- SRM_SRVINFO_NOTIFY_COMID
 - trdp_serviceRegistry.h, [295](#)
- safetyTrail
 - GNU_PACKED, [29](#)
- service_info, [36](#)
 - fctDev, [37](#)
- serviceEntry
 - GNU_PACKED, [29](#)

srv_info_req, [37](#)
 TAU_MARSHALL_INFO_T, [38](#)
 TCN_URI, [38](#)
 TRDP_CLTR_CST_INFO_T, [39](#)
 TRDP_COM_PARAM_T, [39](#)
 TRDP_COMID_DSID_MAP_T, [40](#)
 TRDP_CONSIST_INFO_T, [41](#)
 cstId, [42](#)
 cstOwner, [42](#)
 TRDP_DATA_TYPE_T
 trdp_types.h, [314](#)
 TRDP_DATASET_ELEMENT_T, [44](#)
 TRDP_DATASET, [43](#)
 TRDP_DBG_CONFIG_T, [45](#)
 TRDP_DBG_DEFAULT
 tau_xml.h, [169](#)
 TRDP_DEFAULT_INDEX_SIZES
 trdp_pdindex.h, [287](#)
 TRDP_DNR_OPTS
 tau_dnr.h, [98](#)
 TRDP_DNS_REPLY, [45](#)
 tcnUriCnt, [46](#)
 TRDP_DNS_REQUEST, [47](#)
 tcnUriCnt, [47](#)
 TRDP_ERR_T
 trdp_types.h, [315](#)
 TRDP_ETB_INFO_T, [48](#)
 cnCnt, [48](#)
 TRDP_ETBCTRL_DSID
 iec61375-2-3.h, [73](#)
 TRDP_EXCHG_OPTION_T
 tau_xml.h, [169](#)
 TRDP_FLAGS_DEFAULT
 trdp_types.h, [312](#)
 TRDP_FUNCTION_INFO_T, [49](#)
 cnId, [49](#)
 cstVehNo, [49](#)
 etbId, [49](#)
 fctId, [50](#)
 TRDP_HANDLE, [50](#)
 TRDP_IDX_TABLE_T, [51](#)
 maxNoOfExtPublishers, [51](#)
 maxNoOfHighCatPublishers, [52](#)
 maxNoOfHighCatSubscriptions, [52](#)
 maxNoOfLowCatPublishers, [52](#)
 maxNoOfLowCatSubscriptions, [52](#)
 maxNoOfMidCatPublishers, [52](#)
 maxNoOfMidCatSubscriptions, [52](#)
 TRDP_IP_ADDR_T
 trdp_types.h, [312](#)
 TRDP_MARSHALL_CONFIG_T, [53](#)
 TRDP_MARSHALL_T
 trdp_types.h, [312](#)
 TRDP_MAX_FILE_NAME_LEN
 iec61375-2-3.h, [74](#)
 TRDP_MAX_LABEL_LEN
 iec61375-2-3.h, [74](#)
 TRDP_MAX_MD_DATA_SIZE
 iec61375-2-3.h, [74](#)
 TRDP_MAX_PD_SOCKET_CNT
 trdp_private.h, [291](#)
 TRDP_MAX_URI_HOST_LEN
 iec61375-2-3.h, [74](#)
 TRDP_MAX_URI_LEN
 iec61375-2-3.h, [74](#)
 TRDP_MAX_URI_USER_LEN
 iec61375-2-3.h, [74](#)
 TRDP_MD_CALLBACK_T
 trdp_types.h, [313](#)
 TRDP_MD_CONFIG_T, [54](#)
 TRDP_MD_DEFAULT_REPLY_TIMEOUT
 iec61375-2-3.h, [75](#)
 TRDP_MD_ELE_ST_T
 trdp_private.h, [291](#)
 TRDP_MD_INFINITE_TIME
 iec61375-2-3.h, [75](#)
 TRDP_MD_INFO_T, [55](#)
 TRDP_MEM_CONFIG_T, [56](#)
 TRDP_MIN_PD2_HEADER_SIZE
 trdp_tsn_def.h, [306](#)
 TRDP_MIN_PD_HEADER_SIZE
 iec61375-2-3.h, [75](#)
 TRDP_MSG_PD
 iec61375-2-3.h, [75](#)
 TRDP_MSG_TSN_PD
 trdp_tsn_def.h, [306](#)
 TRDP_PD_CALLBACK_T
 trdp_types.h, [313](#)
 TRDP_PD_CONFIG_T, [57](#)
 TRDP_PD_DEFAULT_QOS
 trdp_tsn_def.h, [306](#)
 TRDP_PD_INFO_T, [57](#)
 TRDP_PD_UDP_PORT
 iec61375-2-3.h, [75](#)
 TRDP_PRINT_DBG_T
 trdp_types.h, [313](#)
 TRDP_PROCESS_CONFIG_T, [58](#)
 TRDP_PROCESS_DEFAULT_CYCLE_TIME
 iec61375-2-3.h, [75](#)
 TRDP_PROP_T, [59](#)
 TRDP_RED_STATE_T
 trdp_types.h, [316](#)
 TRDP_REPLY_STATUS_T
 trdp_types.h, [316](#)
 TRDP_SDT_DEFAULT_CMTHR
 tau_xml.c, [160](#)
 TRDP_SDT_DEFAULT_LMIMAX
 tau_xml.c, [160](#)
 TRDP_SDT_PAR_T, [60](#)
 TRDP_SEQ_CNT_ENTRY_T, [60](#)
 TRDP_SESSION, [61](#)
 TRDP SOCK_TYPE_T
 trdp_private.h, [291](#)
 TRDP_SOCKET_TCP, [62](#)
 TRDP_SOCKETS, [63](#)
 usage, [64](#)

- TRDP_TIME_T
 - trdp_types.h, 313
- TRDP_TO_BEHAVIOR_T
 - trdp_types.h, 316
- TRDP_UNMARSHALL_T
 - trdp_types.h, 313
- TRDP_USR_URI_SIZE
 - iec61375-2-3.h, 76
- TRDP_VEHICLE_INFO_T, 65
 - vehId, 65
- TRDP_XML_DOC_HANDLE_T, 66
- TTDB_NET_DIR_REQ_COMID
 - iec61375-2-3.h, 76
- TTDB_OP_DIR_INFO_COMID
 - iec61375-2-3.h, 76
- TTDB_STAT_CST_REQ_COMID
 - iec61375-2-3.h, 76
- TTDB_TRN_DIR_REQ_COMID
 - iec61375-2-3.h, 76
- TTI_CACHED_CONSISTS
 - tau_tti.c, 133
- tau_DNRstatus
 - tau_dnr.c, 93
 - tau_dnr.h, 100
- tau_addService
 - tau_so_if.c, 123
 - tau_so_if.h, 128
- tau_addr2Uri
 - tau_dnr.c, 93
 - tau_dnr.h, 98
- tau_calcDatasetSize
 - tau_marshall.c, 106
 - tau_marshall.h, 113
- tau_calcDatasetSizeByComId
 - tau_marshall.c, 107
 - tau_marshall.h, 114
- tau_cstinfo.c, 77
 - cstInfoGetPropSize, 78
- tau_ctrl.c, 79
 - tau_getEcspStat, 81
 - tau_initEcspCtrl, 81
 - tau_requestEcspConfirm, 82
 - tau_setEcspCtrl, 82
 - tau_terminateEcspCtrl, 83
- tau_ctrl.h, 83
 - tau_getEcspStat, 86
 - tau_initEcspCtrl, 86
 - tau_requestEcspConfirm, 87
 - tau_setEcspCtrl, 87
 - tau_terminateEcspCtrl, 88
- tau_ctrl_types.h, 88
- tau_delnitDnr
 - tau_dnr.c, 93
 - tau_dnr.h, 99
- tau_delnitTTI
 - tau_tti.c, 133
 - tau_tti.h, 144
- tau_delService
 - tau_so_if.c, 124
 - tau_so_if.h, 128
- tau_dnr.c, 91
 - tau_DNRstatus, 93
 - tau_addr2Uri, 93
 - tau_delnitDnr, 93
 - tau_getOwnAddr, 94
 - tau_initDnr, 94
 - tau_uri2Addr, 95
- tau_dnr.h, 96
 - TRDP_DNR_OPTS, 98
 - tau_DNRstatus, 100
 - tau_addr2Uri, 98
 - tau_delnitDnr, 99
 - tau_getOwnAddr, 100
 - tau_initDnr, 101
 - tau_uri2Addr, 102
- tau_dnr_types.h, 103
- tau_freeServicesList
 - tau_so_if.c, 124
 - tau_so_if.h, 129
- tau_freeTelegrams
 - tau_xml.c, 160
 - tau_xml.h, 169
- tau_freeXmlDatasetConfig
 - tau_xml.c, 160
 - tau_xml.h, 170
- tau_freeXmlDoc
 - tau_xml.c, 161
 - tau_xml.h, 170
- tau_getCstFctCnt
 - tau_tti.c, 133
 - tau_tti.h, 145
- tau_getCstFctInfo
 - tau_tti.c, 134
 - tau_tti.h, 145
- tau_getCstInfo
 - tau_tti.c, 134
 - tau_tti.h, 146
- tau_getCstVehCnt
 - tau_tti.c, 135
 - tau_tti.h, 146
- tau_getEcspStat
 - tau_ctrl.c, 81
 - tau_ctrl.h, 86
- tau_getOpTrDirectory
 - tau_tti.c, 135
 - tau_tti.h, 147
- tau_getOpTrnDirectoryStatusInfo
 - tau_tti.c, 135
 - tau_tti.h, 148
- tau_getOwnAddr
 - tau_dnr.c, 94
 - tau_dnr.h, 100
- tau_getOwnIds
 - tau_tti.c, 136
 - tau_tti.h, 148
- tau_getOwnOpCstNo

- [tau_tti.c](#), 136
 - [tau_tti.h](#), 149
- [tau_getOwnTrnCstNo](#)
 - [tau_tti.c](#), 137
 - [tau_tti.h](#), 149
- [tau_getServicesList](#)
 - [tau_so_if.c](#), 124
 - [tau_so_if.h](#), 129
- [tau_getStaticCstInfo](#)
 - [tau_tti.c](#), 137
 - [tau_tti.h](#), 149
- [tau_getTTI](#)
 - [tau_tti.c](#), 139
 - [tau_tti.h](#), 152
- [tau_getTrDirectory](#)
 - [tau_tti.c](#), 138
 - [tau_tti.h](#), 150
- [tau_getTrnCstCnt](#)
 - [tau_tti.c](#), 138
 - [tau_tti.h](#), 151
- [tau_getTrnVehCnt](#)
 - [tau_tti.c](#), 138
 - [tau_tti.h](#), 151
- [tau_getVehInfo](#)
 - [tau_tti.c](#), 139
 - [tau_tti.h](#), 152
- [tau_getVehOrient](#)
 - [tau_tti.c](#), 140
 - [tau_tti.h](#), 153
- [tau_initDnr](#)
 - [tau_dnr.c](#), 94
 - [tau_dnr.h](#), 101
- [tau_initEcspCtrl](#)
 - [tau_ctrl.c](#), 81
 - [tau_ctrl.h](#), 86
- [tau_initMarshall](#)
 - [tau_marshall.c](#), 108
 - [tau_marshall.h](#), 115
- [tau_initTTIaccess](#)
 - [tau_tti.c](#), 140
 - [tau_tti.h](#), 154
- [tau_marshall](#)
 - [tau_marshall.c](#), 108
 - [tau_marshall.h](#), 116
- [tau_marshall.c](#), 105
 - [tau_calcDatasetSize](#), 106
 - [tau_calcDatasetSizeByComId](#), 107
 - [tau_initMarshall](#), 108
 - [tau_marshall](#), 108
 - [tau_marshallDs](#), 109
 - [tau_unmarshall](#), 110
 - [tau_unmarshallDs](#), 110
- [tau_marshall.h](#), 111
 - [tau_calcDatasetSize](#), 113
 - [tau_calcDatasetSizeByComId](#), 114
 - [tau_initMarshall](#), 115
 - [tau_marshall](#), 116
 - [tau_marshallDs](#), 117
 - [tau_unmarshall](#), 118
 - [tau_unmarshallDs](#), 120
- [tau_marshallDs](#)
 - [tau_marshall.c](#), 109
 - [tau_marshall.h](#), 117
- [tau_prepareXmlDoc](#)
 - [tau_xml.c](#), 161
 - [tau_xml.h](#), 171
- [tau_prepareXmlMem](#)
 - [tau_xml.c](#), 162
 - [tau_xml.h](#), 171
- [tau_readXmlDatasetConfig](#)
 - [tau_xml.c](#), 162
 - [tau_xml.h](#), 172
- [tau_readXmlDeviceConfig](#)
 - [tau_xml.c](#), 162
 - [tau_xml.h](#), 172
- [tau_readXmlInterfaceConfig](#)
 - [tau_xml.c](#), 163
 - [tau_xml.h](#), 173
- [tau_readXmlMappedDeviceConfig](#)
 - [tau_xml.c](#), 164
 - [tau_xml.h](#), 174
- [tau_readXmlMappedDevices](#)
 - [tau_xml.c](#), 164
 - [tau_xml.h](#), 174
- [tau_readXmlMappedInterfaceConfig](#)
 - [tau_xml.c](#), 165
 - [tau_xml.h](#), 176
- [tau_readXmlServiceConfig](#)
 - [tau_xml.c](#), 165
 - [tau_xml.h](#), 176
- [tau_requestEcspConfirm](#)
 - [tau_ctrl.c](#), 82
 - [tau_ctrl.h](#), 87
- [tau_setEcspCtrl](#)
 - [tau_ctrl.c](#), 82
 - [tau_ctrl.h](#), 87
- [tau_so_if.c](#), 121
 - [tau_addService](#), 123
 - [tau_delService](#), 124
 - [tau_freeServicesList](#), 124
 - [tau_getServicesList](#), 124
 - [tau_updService](#), 125
- [tau_so_if.h](#), 126
 - [tau_addService](#), 128
 - [tau_delService](#), 128
 - [tau_freeServicesList](#), 129
 - [tau_getServicesList](#), 129
 - [tau_updService](#), 130
- [tau_terminateEcspCtrl](#)
 - [tau_ctrl.c](#), 83
 - [tau_ctrl.h](#), 88
- [tau_tti.c](#), 130
 - [TTI_CACHED_CONSISTS](#), 133
 - [tau_delInitTTI](#), 133
 - [tau_getCstFctCnt](#), 133
 - [tau_getCstFctInfo](#), 134

- tau_getCstInfo, 134
- tau_getCstVehCnt, 135
- tau_getOpTrDirectory, 135
- tau_getOpTrnDirectoryStatusInfo, 135
- tau_getOwnIds, 136
- tau_getOwnOpCstNo, 136
- tau_getOwnTrnCstNo, 137
- tau_getStaticCstInfo, 137
- tau_getTTI, 139
- tau_getTrDirectory, 138
- tau_getTrnCstCnt, 138
- tau_getTrnVehCnt, 138
- tau_getVehInfo, 139
- tau_getVehOrient, 140
- tau_initTTIaccess, 140
- tau_tti.h, 141
 - tau_delnitTTI, 144
 - tau_getCstFctCnt, 145
 - tau_getCstFctInfo, 145
 - tau_getCstInfo, 146
 - tau_getCstVehCnt, 146
 - tau_getOpTrDirectory, 147
 - tau_getOpTrnDirectoryStatusInfo, 148
 - tau_getOwnIds, 148
 - tau_getOwnOpCstNo, 149
 - tau_getOwnTrnCstNo, 149
 - tau_getStaticCstInfo, 149
 - tau_getTTI, 152
 - tau_getTrDirectory, 150
 - tau_getTrnCstCnt, 151
 - tau_getTrnVehCnt, 151
 - tau_getVehInfo, 152
 - tau_getVehOrient, 153
 - tau_initTTIaccess, 154
- tau_tti_types.h, 155
- tau_unmarshall
 - tau_marshall.c, 110
 - tau_marshall.h, 118
- tau_unmarshallDs
 - tau_marshall.c, 110
 - tau_marshall.h, 120
- tau_updService
 - tau_so_if.c, 125
 - tau_so_if.h, 130
- tau_uri2Addr
 - tau_dnr.c, 95
 - tau_dnr.h, 102
- tau_xml.c, 158
 - TRDP_SDT_DEFAULT_CMTHR, 160
 - TRDP_SDT_DEFAULT_LMIMAX, 160
 - tau_freeTelegrams, 160
 - tau_freeXmlDatasetConfig, 160
 - tau_freeXmlDoc, 161
 - tau_prepareXmlDoc, 161
 - tau_prepareXmlMem, 162
 - tau_readXmlDatasetConfig, 162
 - tau_readXmlDeviceConfig, 162
 - tau_readXmlInterfaceConfig, 163
 - tau_readXmlMappedDeviceConfig, 164
 - tau_readXmlMappedDevices, 164
 - tau_readXmlMappedInterfaceConfig, 165
 - tau_readXmlServiceConfig, 165
- tau_xml.h, 166
 - TRDP_DBG_DEFAULT, 169
 - TRDP_EXCHG_OPTION_T, 169
 - tau_freeTelegrams, 169
 - tau_freeXmlDatasetConfig, 170
 - tau_freeXmlDoc, 170
 - tau_prepareXmlDoc, 171
 - tau_prepareXmlMem, 171
 - tau_readXmlDatasetConfig, 172
 - tau_readXmlDeviceConfig, 172
 - tau_readXmlInterfaceConfig, 173
 - tau_readXmlMappedDeviceConfig, 174
 - tau_readXmlMappedDevices, 174
 - tau_readXmlMappedInterfaceConfig, 176
 - tau_readXmlServiceConfig, 176
- tcnUriCnt
 - TRDP_DNS_REPLY, 46
 - TRDP_DNS_REQUEST, 47
- timeout
 - GNU_PACKED, 29
- tlc_closeSession
 - tlc_if.c, 181
 - trdp_if_light.h, 220
- tlc_configSession
 - tlc_if.c, 181
 - trdp_if_light.h, 220
- tlc_getETBTopoCount
 - tlc_if.c, 182
 - trdp_if_light.h, 221
- tlc_getInterval
 - tlc_if.c, 182
 - trdp_if_light.h, 221
- tlc_getJoinStatistics
 - trdp_if_light.h, 222
 - trdp_stats.c, 297
- tlc_getOpTrainTopoCount
 - tlc_if.c, 183
 - trdp_if_light.h, 222
- tlc_getOwnIpAddress
 - tlc_if.c, 183
 - trdp_if_light.h, 223
- tlc_getPubStatistics
 - trdp_if_light.h, 223
 - trdp_stats.c, 298
- tlc_getRedStatistics
 - trdp_if_light.h, 224
 - trdp_stats.c, 298
- tlc_getStatistics
 - trdp_if_light.h, 224
 - trdp_stats.c, 299
- tlc_getSubsStatistics
 - trdp_if_light.h, 225
 - trdp_stats.c, 299
- tlc_getVersion

- tlc_if.c, 183
- trdp_if_light.h, 225
- tlc_getVersionString
 - tlc_if.c, 184
 - trdp_if_light.h, 225
- tlc_if.c, 178
 - tlc_closeSession, 181
 - tlc_configSession, 181
 - tlc_getETBTopoCount, 182
 - tlc_getInterval, 182
 - tlc_getOpTrainTopoCount, 183
 - tlc_getOwnIpAddress, 183
 - tlc_getVersion, 183
 - tlc_getVersionString, 184
 - tlc_init, 184
 - tlc_openSession, 184
 - tlc_presetIndexSession, 185
 - tlc_process, 186
 - tlc_reinitSession, 186
 - tlc_setETBTopoCount, 187
 - tlc_setOpTrainTopoCount, 187
 - tlc_terminate, 188
 - tlc_updateSession, 188
 - trdp_getAccess, 188
 - trdp_isValidSession, 189
 - trdp_releaseAccess, 189
 - trdp_sessionQueue, 190
- tlc_if.h, 190
 - trdp_isValidSession, 192
 - trdp_sessionQueue, 192
- tlc_init
 - tlc_if.c, 184
 - trdp_if_light.h, 226
- tlc_openSession
 - tlc_if.c, 184
 - trdp_if_light.h, 226
- tlc_presetIndexSession
 - tlc_if.c, 185
 - trdp_if_light.h, 227
- tlc_process
 - tlc_if.c, 186
 - trdp_if_light.h, 228
- tlc_reinitSession
 - tlc_if.c, 186
 - trdp_if_light.h, 228
- tlc_resetStatistics
 - trdp_if_light.h, 229
 - trdp_stats.c, 300
- tlc_setETBTopoCount
 - tlc_if.c, 187
 - trdp_if_light.h, 229
- tlc_setOpTrainTopoCount
 - tlc_if.c, 187
 - trdp_if_light.h, 229
- tlc_terminate
 - tlc_if.c, 188
 - trdp_if_light.h, 230
- tlc_updateSession
 - tlc_if.c, 188
 - trdp_if_light.h, 230
- tlim_abortSession
 - tlim_if.c, 195
 - trdp_if_light.h, 231
- tlim_addListener
 - tlim_if.c, 195
 - trdp_if_light.h, 231
- tlim_confirm
 - tlim_if.c, 196
 - trdp_if_light.h, 232
- tlim_delListener
 - tlim_if.c, 197
 - trdp_if_light.h, 233
- tlim_getInterval
 - tlim_if.c, 197
 - trdp_if_light.h, 233
- tlim_if.c, 193
 - tlim_abortSession, 195
 - tlim_addListener, 195
 - tlim_confirm, 196
 - tlim_delListener, 197
 - tlim_getInterval, 197
 - tlim_notify, 198
 - tlim_process, 199
 - tlim_readdListener, 199
 - tlim_reply, 200
 - tlim_replyQuery, 201
 - tlim_request, 202
- tlim_notify
 - tlim_if.c, 198
 - trdp_if_light.h, 234
- tlim_process
 - tlim_if.c, 199
 - trdp_if_light.h, 235
- tlim_readdListener
 - tlim_if.c, 199
 - trdp_if_light.h, 235
- tlim_reply
 - tlim_if.c, 200
 - trdp_if_light.h, 236
- tlim_replyQuery
 - tlim_if.c, 201
 - trdp_if_light.h, 237
- tlim_request
 - tlim_if.c, 202
 - trdp_if_light.h, 238
- tlim_get
 - tlim_if.c, 205
 - trdp_if_light.h, 239
- tlim_getInterval
 - tlim_if.c, 206
 - trdp_if_light.h, 240
- tlim_getRedundant
 - tlim_if.c, 207
 - trdp_if_light.h, 240
- tlim_if.c, 203
 - tlim_get, 205

- tlp_getInterval, 206
 - tlp_getRedundant, 207
 - tlp_processReceive, 207
 - tlp_processSend, 208
 - tlp_publish, 208
 - tlp_put, 209
 - tlp_putImmediate, 210
 - tlp_republish, 210
 - tlp_request, 211
 - tlp_resubscribe, 212
 - tlp_setRedundant, 213
 - tlp_subscribe, 213
 - tlp_unpublish, 214
 - tlp_unsubscribe, 215
- tlp_processReceive
 - tlp_if.c, 207
 - trdp_if_light.h, 241
- tlp_processSend
 - tlp_if.c, 208
 - trdp_if_light.h, 241
- tlp_publish
 - tlp_if.c, 208
 - trdp_if_light.h, 242
- tlp_put
 - tlp_if.c, 209
 - trdp_if_light.h, 243
- tlp_putImmediate
 - tlp_if.c, 210
 - trdp_if_light.h, 243
- tlp_republish
 - tlp_if.c, 210
 - trdp_if_light.h, 244
- tlp_request
 - tlp_if.c, 211
 - trdp_if_light.h, 245
- tlp_resubscribe
 - tlp_if.c, 212
 - trdp_if_light.h, 246
- tlp_setRedundant
 - tlp_if.c, 213
 - trdp_if_light.h, 246
- tlp_subscribe
 - tlp_if.c, 213
 - trdp_if_light.h, 248
- tlp_unpublish
 - tlp_if.c, 214
 - trdp_if_light.h, 249
- tlp_unsubscribe
 - tlp_if.c, 215
 - trdp_if_light.h, 249
- toBehav
 - GNU_PACKED, 30
- trdp_SockAddJoin
 - trdp_utils.c, 328
 - trdp_utils.h, 341
- trdp_SockDelJoin
 - trdp_utils.c, 328
 - trdp_utils.h, 341
- trdp_SockIsJoined
 - trdp_utils.c, 328
 - trdp_utils.h, 341
- trdp_UpdateStats
 - trdp_stats.c, 301
- trdp_XMLClose
 - trdp_xml.c, 344
 - trdp_xml.h, 350
- trdp_XMLCountStartTag
 - trdp_xml.c, 344
 - trdp_xml.h, 351
- trdp_XMLEnter
 - trdp_xml.c, 345
 - trdp_xml.h, 351
- trdp_XMLGetAttribute
 - trdp_xml.c, 345
 - trdp_xml.h, 351
- trdp_XMLLeave
 - trdp_xml.c, 346
 - trdp_xml.h, 352
- trdp_XMLMemOpen
 - trdp_xml.c, 346
 - trdp_xml.h, 352
- trdp_XMLOpen
 - trdp_xml.c, 346
 - trdp_xml.h, 353
- trdp_XMLRewind
 - trdp_xml.c, 347
 - trdp_xml.h, 353
- trdp_XMLSeekStartTag
 - trdp_xml.c, 347
 - trdp_xml.h, 353
- trdp_XMLSeekStartTagAny
 - trdp_xml.c, 348
 - trdp_xml.h, 354
- trdp_checkSequenceCounter
 - trdp_utils.c, 319
 - trdp_utils.h, 332
- trdp_dllmain.c, 215
- trdp_findMCjoins
 - trdp_utils.c, 320
 - trdp_utils.h, 333
- trdp_findSubAddr
 - trdp_utils.c, 320
 - trdp_utils.h, 333
- trdp_getAccess
 - tlc_if.c, 188
- trdp_getSeqCnt
 - trdp_utils.c, 321
 - trdp_utils.h, 334
- trdp_handleTimeout
 - trdp_pdcom.c, 265
 - trdp_pdcom.h, 276
- trdp_if_light.h, 216
 - tlc_closeSession, 220
 - tlc_configSession, 220
 - tlc_getETBTopoCount, 221
 - tlc_getInterval, 221

- tlc_getJoinStatistics, 222
- tlc_getOpTrainTopoCount, 222
- tlc_getOwnIpAddress, 223
- tlc_getPubStatistics, 223
- tlc_getRedStatistics, 224
- tlc_getStatistics, 224
- tlc_getSubsStatistics, 225
- tlc_getVersion, 225
- tlc_getVersionString, 225
- tlc_init, 226
- tlc_openSession, 226
- tlc_presetIndexSession, 227
- tlc_process, 228
- tlc_reinitSession, 228
- tlc_resetStatistics, 229
- tlc_setETBTopoCount, 229
- tlc_setOpTrainTopoCount, 229
- tlc_terminate, 230
- tlc_updateSession, 230
- tlim_abortSession, 231
- tlim_addListener, 231
- tlim_confirm, 232
- tlim_delListener, 233
- tlim_getInterval, 233
- tlim_notify, 234
- tlim_process, 235
- tlim_readdListener, 235
- tlim_reply, 236
- tlim_replyQuery, 237
- tlim_request, 238
- tllp_get, 239
- tllp_getInterval, 240
- tllp_getRedundant, 240
- tllp_processReceive, 241
- tllp_processSend, 241
- tllp_publish, 242
- tllp_put, 243
- tllp_putImmediate, 243
- tllp_republish, 244
- tllp_request, 245
- tllp_resubscribe, 246
- tllp_setRedundant, 246
- tllp_subscribe, 248
- tllp_unpublish, 249
- tllp_unsubscribe, 249
- trdp_initSockets
 - trdp_utils.c, 321
 - trdp_utils.h, 334
- trdp_initStats
 - trdp_stats.c, 300
 - trdp_stats.h, 303
- trdp_isAddressed
 - trdp_utils.c, 322
 - trdp_utils.h, 334
- trdp_isInIPrange
 - trdp_utils.c, 322
 - trdp_utils.h, 335
- trdp_isValidSession
 - tlc_if.c, 189
 - tlc_if.h, 192
- trdp_mdCall
 - trdp_mdcom.c, 251
 - trdp_mdcom.h, 258
- trdp_mdCheckListenSocks
 - trdp_mdcom.c, 253
 - trdp_mdcom.h, 260
- trdp_mdCheckPending
 - trdp_mdcom.c, 253
 - trdp_mdcom.h, 260
- trdp_mdCheckTimeouts
 - trdp_mdcom.c, 253
 - trdp_mdcom.h, 260
- trdp_mdConfirm
 - trdp_mdcom.c, 253
 - trdp_mdcom.h, 260
- trdp_mdFreeSession
 - trdp_mdcom.c, 254
 - trdp_mdcom.h, 261
- trdp_mdGetTCPSocket
 - trdp_mdcom.c, 254
 - trdp_mdcom.h, 261
- trdp_mdReply
 - trdp_mdcom.c, 255
 - trdp_mdcom.h, 262
- trdp_mdSend
 - trdp_mdcom.c, 256
 - trdp_mdcom.h, 263
- trdp_mdcom.c, 250
 - trdp_mdCall, 251
 - trdp_mdCheckListenSocks, 253
 - trdp_mdCheckPending, 253
 - trdp_mdCheckTimeouts, 253
 - trdp_mdConfirm, 253
 - trdp_mdFreeSession, 254
 - trdp_mdGetTCPSocket, 254
 - trdp_mdReply, 255
 - trdp_mdSend, 256
- trdp_mdcom.h, 256
 - trdp_mdCall, 258
 - trdp_mdCheckListenSocks, 260
 - trdp_mdCheckPending, 260
 - trdp_mdCheckTimeouts, 260
 - trdp_mdConfirm, 260
 - trdp_mdFreeSession, 261
 - trdp_mdGetTCPSocket, 261
 - trdp_mdReply, 262
 - trdp_mdSend, 263
- trdp_packetSizeMD
 - trdp_utils.c, 322
 - trdp_utils.h, 335
- trdp_packetSizePD
 - trdp_utils.c, 323
 - trdp_utils.h, 336
- trdp_pdCheck
 - trdp_pdcom.c, 266
 - trdp_pdcom.h, 276

- trdp_pdCheckListenSocks
 - trdp_pdcom.c, [266](#)
 - trdp_pdcom.h, [277](#)
- trdp_pdCheckPending
 - trdp_pdcom.c, [267](#)
 - trdp_pdcom.h, [277](#)
- trdp_pdDistribute
 - trdp_pdcom.c, [267](#)
 - trdp_pdcom.h, [277](#)
- trdp_pdHandleTimeOuts
 - trdp_pdcom.c, [268](#)
 - trdp_pdcom.h, [278](#)
- trdp_pdInit
 - trdp_pdcom.c, [268](#)
 - trdp_pdcom.h, [279](#)
- trdp_pdPrepareStats
 - trdp_stats.c, [301](#)
 - trdp_stats.h, [304](#)
- trdp_pdPut
 - trdp_pdcom.c, [269](#)
 - trdp_pdcom.h, [280](#)
- trdp_pdReceive
 - trdp_pdcom.c, [270](#)
 - trdp_pdcom.h, [280](#)
- trdp_pdSend
 - trdp_pdcom.c, [270](#)
 - trdp_pdcom.h, [281](#)
- trdp_pdSendElement
 - trdp_pdcom.c, [271](#)
 - trdp_pdcom.h, [281](#)
- trdp_pdSendImmediate
 - trdp_pdcom.c, [271](#)
 - trdp_pdcom.h, [282](#)
- trdp_pdSendQueued
 - trdp_pdcom.c, [272](#)
 - trdp_pdcom.h, [283](#)
- trdp_pdUpdate
 - trdp_pdcom.c, [273](#)
 - trdp_pdcom.h, [283](#)
- trdp_pdcom.c, [263](#)
 - trdp_handleTimeout, [265](#)
 - trdp_pdCheck, [266](#)
 - trdp_pdCheckListenSocks, [266](#)
 - trdp_pdCheckPending, [267](#)
 - trdp_pdDistribute, [267](#)
 - trdp_pdHandleTimeOuts, [268](#)
 - trdp_pdInit, [268](#)
 - trdp_pdPut, [269](#)
 - trdp_pdReceive, [270](#)
 - trdp_pdSend, [270](#)
 - trdp_pdSendElement, [271](#)
 - trdp_pdSendImmediate, [271](#)
 - trdp_pdSendQueued, [272](#)
 - trdp_pdUpdate, [273](#)
- trdp_pdcom.h, [273](#)
 - trdp_handleTimeout, [276](#)
 - trdp_pdCheck, [276](#)
 - trdp_pdCheckListenSocks, [277](#)
 - trdp_pdCheckPending, [277](#)
 - trdp_pdDistribute, [277](#)
 - trdp_pdHandleTimeOuts, [278](#)
 - trdp_pdInit, [279](#)
 - trdp_pdPut, [280](#)
 - trdp_pdReceive, [280](#)
 - trdp_pdSend, [281](#)
 - trdp_pdSendElement, [281](#)
 - trdp_pdSendImmediate, [282](#)
 - trdp_pdSendQueued, [283](#)
 - trdp_pdUpdate, [283](#)
- trdp_pindex.c, [284](#)
- trdp_pindex.h, [285](#)
 - TRDP_DEFAULT_INDEX_SIZES, [287](#)
- trdp_private.h, [288](#)
 - TRDP_MAX_PD_SOCKET_CNT, [291](#)
 - TRDP_MD_ELE_ST_T, [291](#)
 - TRDP SOCK_TYPE_T, [291](#)
- trdp_queueAppLast
 - trdp_utils.c, [323](#)
 - trdp_utils.h, [336](#)
- trdp_queueDelElement
 - trdp_utils.c, [323](#)
 - trdp_utils.h, [336](#)
- trdp_queueFindComId
 - trdp_utils.c, [324](#)
 - trdp_utils.h, [337](#)
- trdp_queueFindExistingSub
 - trdp_utils.c, [324](#)
 - trdp_utils.h, [337](#)
- trdp_queueFindPubAddr
 - trdp_utils.c, [324](#)
 - trdp_utils.h, [337](#)
- trdp_queueFindSubAddr
 - trdp_utils.c, [325](#)
 - trdp_utils.h, [338](#)
- trdp_queueInsFirst
 - trdp_utils.c, [325](#)
 - trdp_utils.h, [338](#)
- trdp_releaseAccess
 - tlc_if.c, [189](#)
- trdp_releaseSocket
 - trdp_utils.c, [326](#)
 - trdp_utils.h, [339](#)
- trdp_requestSocket
 - trdp_utils.c, [326](#)
 - trdp_utils.h, [339](#)
- trdp_resetSequenceCounter
 - trdp_utils.c, [327](#)
 - trdp_utils.h, [340](#)
- trdp_serviceRegistry.h, [292](#)
 - SOA_SAME_SERVICEID, [295](#)
 - SRM_SERVICE_READ_REQ_COMID, [295](#)
 - SRM_SRVINFO_NOTIFY_COMID, [295](#)
- trdp_sessionQueue
 - tlc_if.c, [190](#)
 - tlc_if.h, [192](#)
- trdp_stats.c, [296](#)

- tlc_getJoinStatistics, 297
- tlc_getPubStatistics, 298
- tlc_getRedStatistics, 298
- tlc_getStatistics, 299
- tlc_getSubsStatistics, 299
- tlc_resetStatistics, 300
- trdp_UpdateStats, 301
- trdp_initStats, 300
- trdp_pdPrepareStats, 301
- trdp_stats.h, 302
 - trdp_initStats, 303
 - trdp_pdPrepareStats, 304
- trdp_tsn_def.h, 305
 - TRDP_MIN_PD2_HEADER_SIZE, 306
 - TRDP_MSG_TSN_PD, 306
 - TRDP_PD_DEFAULT_QOS, 306
- trdp_types.h, 306
 - TRDP_DATA_TYPE_T, 314
 - TRDP_ERR_T, 315
 - TRDP_FLAGS_DEFAULT, 312
 - TRDP_IP_ADDR_T, 312
 - TRDP_MARSHALL_T, 312
 - TRDP_MD_CALLBACK_T, 313
 - TRDP_PD_CALLBACK_T, 313
 - TRDP_PRINT_DBG_T, 313
 - TRDP_RED_STATE_T, 316
 - TRDP_REPLY_STATUS_T, 316
 - TRDP_TIME_T, 313
 - TRDP_TO_BEHAVIOR_T, 316
 - TRDP_UNMARSHALL_T, 313
- trdp_utils.c, 317
 - printSocketUsage, 319
 - trdp_SockAddJoin, 328
 - trdp_SockDelJoin, 328
 - trdp_SockIsJoined, 328
 - trdp_checkSequenceCounter, 319
 - trdp_findMCjoins, 320
 - trdp_findSubAddr, 320
 - trdp_getSeqCnt, 321
 - trdp_initSockets, 321
 - trdp_isAddressed, 322
 - trdp_isInIPrange, 322
 - trdp_packetSizeMD, 322
 - trdp_packetSizePD, 323
 - trdp_queueAppLast, 323
 - trdp_queueDelElement, 323
 - trdp_queueFindComId, 324
 - trdp_queueFindExistingSub, 324
 - trdp_queueFindPubAddr, 324
 - trdp_queueFindSubAddr, 325
 - trdp_queueInsFirst, 325
 - trdp_releaseSocket, 326
 - trdp_requestSocket, 326
 - trdp_resetSequenceCounter, 327
 - trdp_validTopoCounters, 329
- trdp_utils.h, 329
 - printSocketUsage, 332
 - trdp_SockAddJoin, 341
 - trdp_SockDelJoin, 341
 - trdp_SockIsJoined, 341
 - trdp_checkSequenceCounter, 332
 - trdp_findMCjoins, 333
 - trdp_findSubAddr, 333
 - trdp_getSeqCnt, 334
 - trdp_initSockets, 334
 - trdp_isAddressed, 334
 - trdp_isInIPrange, 335
 - trdp_packetSizeMD, 335
 - trdp_packetSizePD, 336
 - trdp_queueAppLast, 336
 - trdp_queueDelElement, 336
 - trdp_queueFindComId, 337
 - trdp_queueFindExistingSub, 337
 - trdp_queueFindPubAddr, 337
 - trdp_queueFindSubAddr, 338
 - trdp_queueInsFirst, 338
 - trdp_releaseSocket, 339
 - trdp_requestSocket, 339
 - trdp_resetSequenceCounter, 340
 - trdp_validTopoCounters, 342
- trdp_validTopoCounters
 - trdp_utils.c, 329
 - trdp_utils.h, 342
- trdp_xml.c, 342
 - trdp_XMLClose, 344
 - trdp_XMLCountStartTag, 344
 - trdp_XMLEnter, 345
 - trdp_XMLGetAttribute, 345
 - trdp_XMLLeave, 346
 - trdp_XMLMemOpen, 346
 - trdp_XMLOpen, 346
 - trdp_XMLRewind, 347
 - trdp_XMLSeekStartTag, 347
 - trdp_XMLSeekStartTagAny, 348
- trdp_xml.h, 348
 - trdp_XMLClose, 350
 - trdp_XMLCountStartTag, 351
 - trdp_XMLEnter, 351
 - trdp_XMLGetAttribute, 351
 - trdp_XMLLeave, 352
 - trdp_XMLMemOpen, 352
 - trdp_XMLOpen, 353
 - trdp_XMLRewind, 353
 - trdp_XMLSeekStartTag, 353
 - trdp_XMLSeekStartTagAny, 354
- trnCstNo
 - GNU_PACKED, 30
- trnDirState
 - GNU_PACKED, 30
- trnId
 - GNU_PACKED, 30
- trnNetDir
 - GNU_PACKED, 30
- trnOperator
 - GNU_PACKED, 30
- trnTopoCnt

- GNU_PACKED, 31
- trnVehNo
 - GNU_PACKED, 31
- usage
 - TRDP_SOCKETS, 64
- VOS_ERR_T
 - vos_types.h, 409
- VOS_LOG_T
 - vos_types.h, 409
- VOS_MAX_ERR_STR_SIZE
 - vos_utils.h, 416
- VOS_MAX_FRMT_SIZE
 - vos_utils.h, 416
- VOS_MAX_PRNT_STR_SIZE
 - vos_utils.h, 416
- VOS_MAX_SOCKET_CNT
 - vos_sock.h, 379
- VOS_MEM_MAX_PREALLOCATE
 - vos_mem.h, 366
- VOS_MEM_PREALLOCATE
 - vos_mem.h, 366
- VOS_PRINT_DBG_T
 - vos_types.h, 408
- VOS_SOCKET_OPT_T, 66
- VOS_TIMEVAL_T
 - vos_types.h, 408
- VOS_TTL_MULTICAST
 - vos_sock.h, 379
- VOS_VERSION_T, 67
- vehId
 - GNU_PACKED, 31
 - TRDP_VEHICLE_INFO_T, 65
- vehOrient
 - GNU_PACKED, 31
- version
 - GNU_PACKED, 31
- vos_addTime
 - vos_thread.h, 396
- vos_bsearch
 - vos_mem.c, 356
 - vos_mem.h, 366
- vos_clearTime
 - vos_thread.h, 396
- vos_cmpTime
 - vos_thread.h, 396
- vos_crc32
 - vos_utils.c, 411
 - vos_utils.h, 417
- vos_determineBindAddr
 - vos_sock.h, 379
- vos_divTime
 - vos_thread.h, 397
- vos_dottedIP
 - vos_sock.h, 380
- vos_getErrorString
 - vos_utils.c, 411
 - vos_utils.h, 418
- vos_getInterfaces
 - vos_sock.h, 380
- vos_getRealTime
 - vos_thread.h, 397
- vos_getTime
 - vos_thread.h, 397
- vos_getTimeStamp
 - vos_thread.h, 398
- vos_getUuid
 - vos_thread.h, 398
- vos_getVersion
 - vos_utils.c, 412
 - vos_utils.h, 418
- vos_getVersionString
 - vos_utils.c, 412
 - vos_utils.h, 418
- vos_hostIsBigEndian
 - vos_utils.c, 412
 - vos_utils.h, 419
- vos_htonl
 - vos_sock.h, 380
- vos_htonll
 - vos_sock.h, 381
- vos_htons
 - vos_sock.h, 381
- vos_init
 - vos_utils.c, 413
 - vos_utils.h, 419
- vos_ipDotted
 - vos_sock.h, 382
- vos_isMulticast
 - vos_sock.h, 382
- vos_mem.c, 354
 - vos_bsearch, 356
 - vos_memAlloc, 357
 - vos_memCount, 357
 - vos_memDelete, 358
 - vos_memFree, 358
 - vos_memInit, 358
 - vos_qsort, 359
 - vos_queueCreate, 359
 - vos_queueDestroy, 360
 - vos_queueReceive, 360
 - vos_queueSend, 361
 - vos_strncat, 361
 - vos_strncpy, 363
 - vos_strncmp, 363
- vos_mem.h, 364
 - VOS_MEM_MAX_PREALLOCATE, 366
 - VOS_MEM_PREALLOCATE, 366
 - vos_bsearch, 366
 - vos_memAlloc, 367
 - vos_memCount, 367
 - vos_memDelete, 368
 - vos_memFree, 368
 - vos_memInit, 369
 - vos_qsort, 369
 - vos_queueCreate, 370

- vos_queueDestroy, [370](#)
- vos_queueReceive, [371](#)
- vos_queueSend, [371](#)
- vos_strncat, [372](#)
- vos_strncpy, [372](#)
- vos_strncmp, [373](#)
- vos_memAlloc
 - vos_mem.c, [357](#)
 - vos_mem.h, [367](#)
- vos_memCount
 - vos_mem.c, [357](#)
 - vos_mem.h, [367](#)
- vos_memDelete
 - vos_mem.c, [358](#)
 - vos_mem.h, [368](#)
- vos_memFree
 - vos_mem.c, [358](#)
 - vos_mem.h, [368](#)
- vos_memInit
 - vos_mem.c, [358](#)
 - vos_mem.h, [369](#)
- vos_mulTime
 - vos_thread.h, [398](#)
- vos_mutexCreate
 - vos_thread.h, [398](#)
- vos_mutexDelete
 - vos_thread.h, [399](#)
- vos_mutexLock
 - vos_thread.h, [399](#)
- vos_mutexTryLock
 - vos_thread.h, [400](#)
- vos_mutexUnlock
 - vos_thread.h, [400](#)
- vos_netIfUp
 - vos_sock.h, [382](#)
- vos_ntohl
 - vos_sock.h, [383](#)
- vos_ntohll
 - vos_sock.h, [383](#)
- vos_ntohs
 - vos_sock.h, [383](#)
- vos_qsort
 - vos_mem.c, [359](#)
 - vos_mem.h, [369](#)
- vos_queueCreate
 - vos_mem.c, [359](#)
 - vos_mem.h, [370](#)
- vos_queueDestroy
 - vos_mem.c, [360](#)
 - vos_mem.h, [370](#)
- vos_queueReceive
 - vos_mem.c, [360](#)
 - vos_mem.h, [371](#)
- vos_queueSend
 - vos_mem.c, [361](#)
 - vos_mem.h, [371](#)
- vos_sc32
 - vos_utils.c, [413](#)
- vos_utils.h, [420](#)
- vos_select
 - vos_sock.h, [384](#)
- vos_semaCreate
 - vos_thread.h, [400](#)
- vos_semaDelete
 - vos_thread.h, [401](#)
- vos_semaGive
 - vos_thread.h, [401](#)
- vos_semaTake
 - vos_thread.h, [402](#)
- vos_shared_mem.h, [373](#)
 - vos_sharedClose, [375](#)
 - vos_sharedOpen, [375](#)
- vos_sharedClose
 - vos_shared_mem.h, [375](#)
- vos_sharedOpen
 - vos_shared_mem.h, [375](#)
- vos_sock.h, [376](#)
 - VOS_MAX_SOCKET_CNT, [379](#)
 - VOS_TTL_MULTICAST, [379](#)
 - vos_determineBindAddr, [379](#)
 - vos_dottedIP, [380](#)
 - vos_getInterfaces, [380](#)
 - vos_htonl, [380](#)
 - vos_htonll, [381](#)
 - vos_htons, [381](#)
 - vos_ipDotted, [382](#)
 - vos_isMulticast, [382](#)
 - vos_netIfUp, [382](#)
 - vos_ntohl, [383](#)
 - vos_ntohll, [383](#)
 - vos_ntohs, [383](#)
 - vos_select, [384](#)
 - vos_sockAccept, [384](#)
 - vos_sockBind, [385](#)
 - vos_sockClose, [385](#)
 - vos_sockConnect, [386](#)
 - vos_sockGetMAC, [386](#)
 - vos_sockInit, [387](#)
 - vos_sockJoinMC, [387](#)
 - vos_sockLeaveMC, [387](#)
 - vos_sockListen, [388](#)
 - vos_sockOpenTCP, [388](#)
 - vos_sockOpenUDP, [389](#)
 - vos_sockReceiveTCP, [389](#)
 - vos_sockReceiveUDP, [390](#)
 - vos_sockSendTCP, [391](#)
 - vos_sockSendUDP, [391](#)
 - vos_sockSetMulticastIf, [392](#)
 - vos_sockSetOptions, [392](#)
 - vos_sockTerm, [393](#)
- vos_sockAccept
 - vos_sock.h, [384](#)
- vos_sockBind
 - vos_sock.h, [385](#)
- vos_sockClose
 - vos_sock.h, [385](#)

- vos_sockConnect
 - vos_sock.h, [386](#)
- vos_sockGetMAC
 - vos_sock.h, [386](#)
- vos_sockInit
 - vos_sock.h, [387](#)
- vos_sockJoinMC
 - vos_sock.h, [387](#)
- vos_sockLeaveMC
 - vos_sock.h, [387](#)
- vos_sockListen
 - vos_sock.h, [388](#)
- vos_sockOpenTCP
 - vos_sock.h, [388](#)
- vos_sockOpenUDP
 - vos_sock.h, [389](#)
- vos_sockReceiveTCP
 - vos_sock.h, [389](#)
- vos_sockReceiveUDP
 - vos_sock.h, [390](#)
- vos_sockSendTCP
 - vos_sock.h, [391](#)
- vos_sockSendUDP
 - vos_sock.h, [391](#)
- vos_sockSetMulticastIf
 - vos_sock.h, [392](#)
- vos_sockSetOptions
 - vos_sock.h, [392](#)
- vos_sockTerm
 - vos_sock.h, [393](#)
- vos_strncat
 - vos_mem.c, [361](#)
 - vos_mem.h, [372](#)
- vos_strncpy
 - vos_mem.c, [363](#)
 - vos_mem.h, [372](#)
- vos_strnicmp
 - vos_mem.c, [363](#)
 - vos_mem.h, [373](#)
- vos_subTime
 - vos_thread.h, [402](#)
- vos_terminate
 - vos_utils.c, [414](#)
 - vos_utils.h, [420](#)
- vos_thread.h, [393](#)
 - vos_addTime, [396](#)
 - vos_clearTime, [396](#)
 - vos_cmpTime, [396](#)
 - vos_divTime, [397](#)
 - vos_getRealTime, [397](#)
 - vos_getTime, [397](#)
 - vos_getTimeStamp, [398](#)
 - vos_getUuid, [398](#)
 - vos_mulTime, [398](#)
 - vos_mutexCreate, [398](#)
 - vos_mutexDelete, [399](#)
 - vos_mutexLock, [399](#)
 - vos_mutexTryLock, [400](#)
 - vos_mutexUnlock, [400](#)
 - vos_semaCreate, [400](#)
 - vos_semaDelete, [401](#)
 - vos_semaGive, [401](#)
 - vos_semaTake, [402](#)
 - vos_subTime, [402](#)
 - vos_threadCreate, [402](#)
 - vos_threadCreateSync, [403](#)
 - vos_threadDelay, [404](#)
 - vos_threadInit, [404](#)
 - vos_threadIsActive, [405](#)
 - vos_threadSelf, [405](#)
 - vos_threadTerm, [405](#)
 - vos_threadTerminate, [405](#)
- vos_threadCreate
 - vos_thread.h, [402](#)
- vos_threadCreateSync
 - vos_thread.h, [403](#)
- vos_threadDelay
 - vos_thread.h, [404](#)
- vos_threadInit
 - vos_thread.h, [404](#)
- vos_threadIsActive
 - vos_thread.h, [405](#)
- vos_threadSelf
 - vos_thread.h, [405](#)
- vos_threadTerm
 - vos_thread.h, [405](#)
- vos_threadTerminate
 - vos_thread.h, [405](#)
- vos_types.h, [406](#)
 - VOS_ERR_T, [409](#)
 - VOS_LOG_T, [409](#)
 - VOS_PRINT_DBG_T, [408](#)
 - VOS_TIMEVAL_T, [408](#)
- vos_utils.c, [410](#)
 - vos_crc32, [411](#)
 - vos_getErrorString, [411](#)
 - vos_getVersion, [412](#)
 - vos_getVersionString, [412](#)
 - vos_hostIsBigEndian, [412](#)
 - vos_init, [413](#)
 - vos_sc32, [413](#)
 - vos_terminate, [414](#)
- vos_utils.h, [414](#)
 - INITFCS, [416](#)
 - VOS_MAX_ERR_STR_SIZE, [416](#)
 - VOS_MAX_FRMT_SIZE, [416](#)
 - VOS_MAX_PRNT_STR_SIZE, [416](#)
 - vos_crc32, [417](#)
 - vos_getErrorString, [418](#)
 - vos_getVersion, [418](#)
 - vos_getVersionString, [418](#)
 - vos_hostIsBigEndian, [419](#)
 - vos_init, [419](#)
 - vos_sc32, [420](#)
 - vos_terminate, [420](#)