

TCNOpen TRDP Light
V3.0.0

Generated by Doxygen 1.9.1

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	11
3.1 File List	11
4 Data Structure Documentation	13
4.1 SRM_SERVICE_INFO_T Struct Reference	13
4.1.1 Detailed Description	14
4.1.2 Field Documentation	14
4.1.2.1 fctDev	14
4.2 SRM_SRV_INFO_REQ_T Struct Reference	15
4.2.1 Detailed Description	15
4.3 TAU_DNS_HEADER_T Struct Reference	16
4.3.1 Detailed Description	16
4.4 TAU_MARSHALL_INFO_T Struct Reference	16
4.4.1 Detailed Description	16
4.5 TCN_URI_T Struct Reference	16
4.5.1 Detailed Description	17
4.6 TRDP_CLTR_CST_INFO_T Struct Reference	17
4.6.1 Detailed Description	17
4.7 TRDP_COM_PARAM_T Struct Reference	18
4.7.1 Detailed Description	18
4.8 TRDP_COMID_DSID_MAP_T Struct Reference	18
4.8.1 Detailed Description	19
4.9 TRDP_CONF_VEHICLE_T Struct Reference	19
4.9.1 Detailed Description	19
4.10 TRDP_CONSIST_INFO_T Struct Reference	19
4.10.1 Detailed Description	20
4.10.2 Field Documentation	21
4.10.2.1 cstId	21
4.10.2.2 cstOwner	21
4.11 TRDP_CSTINFOCTRL_T Struct Reference	21

4.11.1 Detailed Description	22
4.11.2 Field Documentation	22
4.11.2.1 cstList	22
4.11.2.2 version	22
4.12 TRDP_DATASET_ELEMENT_T Struct Reference	23
4.12.1 Detailed Description	23
4.13 TRDP_DATASET_T Struct Reference	24
4.13.1 Detailed Description	24
4.14 TRDP_DBG_CONFIG_T Struct Reference	25
4.14.1 Detailed Description	25
4.15 TRDP_DNS_REPLY_T Struct Reference	25
4.15.1 Detailed Description	26
4.15.2 Field Documentation	26
4.15.2.1 tcnUriCnt	26
4.16 TRDP_DNS_REQUEST_T Struct Reference	27
4.16.1 Detailed Description	27
4.16.2 Field Documentation	28
4.16.2.1 tcnUriCnt	28
4.17 TRDP_ETB_CTRL_VDP_T Struct Reference	28
4.17.1 Detailed Description	29
4.18 TRDP_ETB_INFO_T Struct Reference	29
4.18.1 Detailed Description	29
4.18.2 Field Documentation	29
4.18.2.1 cnCnt	29
4.19 TRDP_FUNCTION_INFO_T Struct Reference	30
4.19.1 Detailed Description	30
4.19.2 Field Documentation	30
4.19.2.1 cnId	30
4.19.2.2 cstVehNo	30
4.19.2.3 etbId	31
4.19.2.4 fctId	31
4.20 TRDP_IDX_TABLE_T Struct Reference	31
4.20.1 Detailed Description	32
4.20.2 Field Documentation	32
4.20.2.1 maxNoOfExtPublishers	32
4.20.2.2 maxNoOfHighCatPublishers	32
4.20.2.3 maxNoOfHighCatSubscriptions	32
4.20.2.4 maxNoOfLowCatPublishers	32
4.20.2.5 maxNoOfLowCatSubscriptions	32
4.20.2.6 maxNoOfMidCatPublishers	33
4.20.2.7 maxNoOfMidCatSubscriptions	33
4.21 TRDP_LIST_STATISTICS_T Struct Reference	33

4.21.1 Detailed Description	34
4.22 TRDP_MARSHALL_CONFIG_T Struct Reference	34
4.22.1 Detailed Description	35
4.23 TRDP_MD_CONFIG_T Struct Reference	35
4.23.1 Detailed Description	36
4.24 TRDP_MD_INFO_T Struct Reference	36
4.24.1 Detailed Description	38
4.25 TRDP_MD_STATISTICS_T Struct Reference	38
4.25.1 Detailed Description	39
4.26 TRDP_MEM_CONFIG_T Struct Reference	39
4.26.1 Detailed Description	39
4.27 TRDP_OP_CONSIST_T Struct Reference	39
4.27.1 Detailed Description	40
4.27.2 Field Documentation	40
4.27.2.1 cstUUID	40
4.28 TRDP_OP_TRAIN_DIR_STATE_T Struct Reference	40
4.28.1 Detailed Description	41
4.28.2 Field Documentation	41
4.28.2.1 trnId	41
4.28.2.2 trnOperator	42
4.29 TRDP_OP_TRAIN_DIR_STATUS_INFO_T Struct Reference	42
4.29.1 Detailed Description	42
4.30 TRDP_OP_TRAIN_DIR_T Struct Reference	43
4.30.1 Detailed Description	44
4.30.2 Field Documentation	44
4.30.2.1 opCstList	44
4.30.2.2 opVehList	44
4.31 TRDP_OP_VEHICLE_T Struct Reference	44
4.31.1 Detailed Description	45
4.31.2 Field Documentation	45
4.31.2.1 vehId	45
4.32 TRDP_PD_CONFIG_T Struct Reference	45
4.32.1 Detailed Description	46
4.33 TRDP_PD_INFO_T Struct Reference	46
4.33.1 Detailed Description	47
4.34 TRDP_PD_STATISTICS_T Struct Reference	48
4.34.1 Detailed Description	48
4.35 TRDP_PROCESS_CONFIG_T Struct Reference	49
4.35.1 Detailed Description	49
4.36 TRDP_PROP_T Struct Reference	49
4.36.1 Detailed Description	50
4.37 TRDP_PUB_STATISTICS_T Struct Reference	50

4.37.1 Detailed Description	51
4.38 TRDP_READ_COMPLETE_REPLY_T Struct Reference	51
4.38.1 Detailed Description	52
4.39 TRDP_RED_STATISTICS_T Struct Reference	52
4.39.1 Detailed Description	52
4.40 TRDP_SDT_PAR_T Struct Reference	53
4.40.1 Detailed Description	53
4.41 TRDP_SHORT_VERSION_T Struct Reference	53
4.41.1 Detailed Description	54
4.42 TRDP_STATISTICS_REQUEST_T Struct Reference	54
4.42.1 Detailed Description	54
4.43 TRDP_STATISTICS_T Struct Reference	55
4.43.1 Detailed Description	56
4.44 TRDP_SUBS_STATISTICS_T Struct Reference	56
4.44.1 Detailed Description	56
4.44.2 Field Documentation	56
4.44.2.1 timeout	57
4.44.2.2 toBehav	57
4.45 TRDP_TRAIN_DIR_T Struct Reference	57
4.45.1 Detailed Description	58
4.46 TRDP_TRAIN_NET_DIR_ENTRY_T Struct Reference	58
4.46.1 Detailed Description	58
4.47 TRDP_TRAIN_NET_DIR_T Struct Reference	59
4.47.1 Detailed Description	59
4.48 TRDP_VEHICLE_INFO_T Struct Reference	60
4.48.1 Detailed Description	60
4.48.2 Field Documentation	61
4.48.2.1 vehId	61
4.49 TRDP_XML_DOC_HANDLE_T Struct Reference	61
4.49.1 Detailed Description	61
4.50 VOS_MEM_STATISTICS_T Struct Reference	61
4.50.1 Detailed Description	62
4.51 VOS_QUEUE_T Struct Reference	62
4.51.1 Detailed Description	62
4.52 VOS SOCK_OPT_T Struct Reference	62
4.52.1 Detailed Description	63
4.53 VOS_VERSION_T Struct Reference	63
4.53.1 Detailed Description	64
5 File Documentation	65
5.1 iec61375-2-3.h File Reference	65
5.1.1 Detailed Description	70

5.1.2 Macro Definition Documentation	71
5.1.2.1 ETB_CTRL_COMID	71
5.1.2.2 TRDP_ETBCTRL_DSID	71
5.1.2.3 TRDP_EXTRA_LABEL_LEN	71
5.1.2.4 TRDP_MAX_FILE_NAME_LEN	71
5.1.2.5 TRDP_MAX_LABEL_LEN	72
5.1.2.6 TRDP_MAX_MD_DATA_SIZE	72
5.1.2.7 TRDP_MAX_URI_HOST_LEN	72
5.1.2.8 TRDP_MAX_URI_LEN	72
5.1.2.9 TRDP_MAX_URI_USER_LEN	72
5.1.2.10 TRDP_MD_DEFAULT_REPLY_TIMEOUT	72
5.1.2.11 TRDP_MD_INFINITE_TIME	73
5.1.2.12 TRDP_MIN_PD_HEADER_SIZE	73
5.1.2.13 TRDP_MSG_PT	73
5.1.2.14 TRDP_PD_UDP_PORT	73
5.1.2.15 TRDP_PROCESS_DEFAULT_CYCLE_TIME	73
5.1.2.16 TRDP_PROTOCOL_VERSION_CHECK_MASK	74
5.1.2.17 TRDP_USR_URI_SIZE	74
5.1.2.18 TTDB_NET_DIR_REQ_COMID	74
5.1.2.19 TTDB_OP_DIR_INFO_COMID	74
5.1.2.20 TTDB_STAT_CST_REQ_COMID	74
5.1.2.21 TTDB_TRN_DIR_REQ_COMID	75
5.2 tau_cstinfo.c File Reference	75
5.2.1 Detailed Description	76
5.2.2 Function Documentation	76
5.2.2.1 cstInfoGetPropSize()	76
5.3 tau_ctrl.c File Reference	77
5.3.1 Detailed Description	78
5.3.2 Function Documentation	78
5.3.2.1 tau_getEcspStat()	78
5.3.2.2 tau_initEcspCtrl()	79
5.3.2.3 tau_requestEcspConfirm()	79
5.3.2.4 tau_requestEcspConfirmReply()	80
5.3.2.5 tau_setEcspCtrl()	80
5.3.2.6 tau_terminateEcspCtrl()	81
5.4 tau_ctrl.h File Reference	81
5.4.1 Detailed Description	83
5.4.2 Function Documentation	83
5.4.2.1 tau_getEcspStat()	83
5.4.2.2 tau_initEcspCtrl()	84
5.4.2.3 tau_requestEcspConfirm()	85
5.4.2.4 tau_requestEcspConfirmReply()	85

5.4.2.5 tau_setEcspCtrl()	86
5.4.2.6 tau_terminateEcspCtrl()	86
5.5 tau_ctrl_types.h File Reference	87
5.5.1 Detailed Description	89
5.6 tau_dnr.c File Reference	89
5.6.1 Detailed Description	91
5.6.2 Function Documentation	91
5.6.2.1 tau_addr2Uri()	91
5.6.2.2 tau_delnitDnr()	92
5.6.2.3 tau_DNRstatus()	92
5.6.2.4 tau_getOwnAddr()	93
5.6.2.5 tau_initDnr()	93
5.6.2.6 tau_uri2Addr()	94
5.7 tau_dnr.h File Reference	94
5.7.1 Detailed Description	96
5.7.2 Enumeration Type Documentation	96
5.7.2.1 TRDP_DNR_OPTS_T	96
5.7.3 Function Documentation	97
5.7.3.1 tau_addr2Uri()	97
5.7.3.2 tau_delnitDnr()	98
5.7.3.3 tau_DNRstatus()	98
5.7.3.4 tau_getOwnAddr()	99
5.7.3.5 tau_initDnr()	99
5.7.3.6 tau_uri2Addr()	100
5.8 tau_dnr_types.h File Reference	101
5.8.1 Detailed Description	103
5.9 tau_marshall.c File Reference	103
5.9.1 Detailed Description	104
5.9.2 Function Documentation	104
5.9.2.1 tau_calcDatasetSize()	105
5.9.2.2 tau_calcDatasetSizeByComId()	105
5.9.2.3 tau_initMarshall()	106
5.9.2.4 tau_marshall()	107
5.9.2.5 tau_marshallIDs()	107
5.9.2.6 tau_unmarshall()	108
5.9.2.7 tau_unmarshallIDs()	109
5.10 tau_marshall.h File Reference	109
5.10.1 Detailed Description	111
5.10.2 Function Documentation	111
5.10.2.1 tau_calcDatasetSize()	111
5.10.2.2 tau_calcDatasetSizeByComId()	112
5.10.2.3 tau_initMarshall()	113

5.10.2.4 tau_marshall()	114
5.10.2.5 tau_marshallDs()	115
5.10.2.6 tau_unmarshall()	116
5.10.2.7 tau_unmarshallDs()	118
5.11 tau_so_if.c File Reference	119
5.11.1 Detailed Description	120
5.11.2 Function Documentation	120
5.11.2.1 tau_addService()	120
5.11.2.2 tau_delService()	121
5.11.2.3 tau_freeServicesList()	121
5.11.2.4 tau_getServicesList()	122
5.11.2.5 tau_updService()	122
5.12 tau_so_if.h File Reference	123
5.12.1 Detailed Description	124
5.12.2 Function Documentation	125
5.12.2.1 tau_addService()	125
5.12.2.2 tau_delService()	125
5.12.2.3 tau_freeServicesList()	126
5.12.2.4 tau_getServicesList()	126
5.12.2.5 tau_updService()	127
5.13 tau_tti.c File Reference	127
5.13.1 Detailed Description	129
5.13.2 Function Documentation	130
5.13.2.1 tau_copyCstInfo()	130
5.13.2.2 tau_delInitTTI()	130
5.13.2.3 tau_getCstFctCnt()	130
5.13.2.4 tau_getCstFctInfo()	131
5.13.2.5 tau_getCstInfo()	131
5.13.2.6 tau_getCstVehCnt()	133
5.13.2.7 tau_getOpTrnDirectory()	133
5.13.2.8 tau_getOpTrnDirectoryStatusInfo()	134
5.13.2.9 tau_getOwnIds()	134
5.13.2.10 tau_getOwnOpCstNo()	135
5.13.2.11 tau_getOwnTrnCstNo()	135
5.13.2.12 tau_getStaticCstInfo()	136
5.13.2.13 tau_getTrnCstCnt()	136
5.13.2.14 tau_getTrnDirectory()	137
5.13.2.15 tau_getTrnVehCnt()	137
5.13.2.16 tau_getTTI()	137
5.13.2.17 tau_getVehInfo()	138
5.13.2.18 tau_getVehOrient()	138
5.13.2.19 tau_initTTIaccess()	139

5.14 tau_tti.h File Reference	140
5.14.1 Detailed Description	142
5.14.2 Function Documentation	142
5.14.2.1 tau_delInitTTI()	142
5.14.2.2 tau_getCstFctCnt()	143
5.14.2.3 tau_getCstFctInfo()	143
5.14.2.4 tau_getCstInfo()	144
5.14.2.5 tau_getCstVehCnt()	145
5.14.2.6 tau_getOpTrnDirectory()	146
5.14.2.7 tau_getOpTrnDirectoryStatusInfo()	146
5.14.2.8 tau_getOwnIds()	147
5.14.2.9 tau_getOwnOpCstNo()	148
5.14.2.10 tau_getOwnTrnCstNo()	148
5.14.2.11 tau_getStaticCstInfo()	148
5.14.2.12 tau_getTrnCstCnt()	149
5.14.2.13 tau_getTrnDirectory()	149
5.14.2.14 tau_getTrnVehCnt()	150
5.14.2.15 tau_getTTI()	150
5.14.2.16 tau_getVehInfo()	151
5.14.2.17 tau_getVehOrient()	152
5.14.2.18 tau_initTTIaccess()	153
5.15 tau_tti_types.h File Reference	153
5.15.1 Detailed Description	156
5.16 tau_xml.c File Reference	156
5.16.1 Detailed Description	158
5.16.2 Macro Definition Documentation	158
5.16.2.1 TRDP_SDT_DEFAULT_CMTHR	159
5.16.2.2 TRDP_SDT_DEFAULT_LMIMAX	159
5.16.3 Function Documentation	159
5.16.3.1 tau_freeTelegrams()	159
5.16.3.2 tau_freeXmlDatasetConfig()	159
5.16.3.3 tau_freeXmlDoc()	160
5.16.3.4 tau_prepareXmlDoc()	160
5.16.3.5 tau_prepareXmlMem()	161
5.16.3.6 tau_readXmlDatasetConfig()	161
5.16.3.7 tau_readXmlDeviceConfig()	162
5.16.3.8 tau_readXmlInterfaceConfig()	162
5.16.3.9 tau_readXmlMappedDeviceConfig()	163
5.16.3.10 tau_readXmlMappedDevices()	163
5.16.3.11 tau_readXmlMappedInterfaceConfig()	164
5.16.3.12 tau_readXmlServiceConfig()	164
5.17 tau_xml.h File Reference	165

5.17.1 Detailed Description	168
5.17.2 Macro Definition Documentation	168
5.17.2.1 TRDP_DBG_DEFAULT	168
5.17.3 Enumeration Type Documentation	168
5.17.3.1 TRDP_EXCHG_OPTION_T	168
5.17.4 Function Documentation	169
5.17.4.1 tau_freeTelegrams()	169
5.17.4.2 tau_freeXmlDatasetConfig()	169
5.17.4.3 tau_freeXmlDoc()	170
5.17.4.4 tau_prepareXmlDoc()	170
5.17.4.5 tau_prepareXmlMem()	171
5.17.4.6 tau_readXmlDatasetConfig()	171
5.17.4.7 tau_readXmlDeviceConfig()	172
5.17.4.8 tau_readXmlInterfaceConfig()	173
5.17.4.9 tau_readXmlMappedDeviceConfig()	173
5.17.4.10 tau_readXmlMappedDevices()	174
5.17.4.11 tau_readXmlMappedInterfaceConfig()	174
5.17.4.12 tau_readXmlServiceConfig()	175
5.18 tlc_if.c File Reference	176
5.18.1 Detailed Description	177
5.18.2 Function Documentation	178
5.18.2.1 tlc_closeSession()	178
5.18.2.2 tlc_configSession()	178
5.18.2.3 tlc_getETBTopoCount()	179
5.18.2.4 tlc_getInterval()	179
5.18.2.5 tlc_getOpTrainTopoCount()	180
5.18.2.6 tlc_getOwnIpAddress()	180
5.18.2.7 tlc_getVersion()	180
5.18.2.8 tlc_getVersionString()	181
5.18.2.9 tlc_init()	181
5.18.2.10 tlc_openSession()	182
5.18.2.11 tlc_presetIndexSession()	182
5.18.2.12 tlc_process()	183
5.18.2.13 tlc_reinitSession()	183
5.18.2.14 tlc_setETBTopoCount()	184
5.18.2.15 tlc_setOpTrainTopoCount()	184
5.18.2.16 tlc_terminate()	185
5.18.2.17 tlc_updateSession()	185
5.18.2.18 trdp_getAccess()	186
5.18.2.19 trdp_isValidSession()	186
5.18.2.20 trdp_releaseAccess()	187
5.18.2.21 trdp_sessionQueue()	187

5.19 tlm_if.c File Reference	187
5.19.1 Detailed Description	189
5.19.2 Function Documentation	189
5.19.2.1 tlm_abortSession()	189
5.19.2.2 tlm_addListener()	190
5.19.2.3 tlm_confirm()	191
5.19.2.4 tlm_delListener()	191
5.19.2.5 tlm_getInterval()	192
5.19.2.6 tlm_notify()	192
5.19.2.7 tlm_process()	193
5.19.2.8 tlm_readdListener()	194
5.19.2.9 tlm_reply()	194
5.19.2.10 tlm_replyQuery()	195
5.19.2.11 tlm_request()	196
5.20 tlp_if.c File Reference	197
5.20.1 Detailed Description	199
5.20.2 Function Documentation	199
5.20.2.1 tlp_get()	199
5.20.2.2 tlp_getInterval()	200
5.20.2.3 tlp_getRedundant()	200
5.20.2.4 tlp_processReceive()	201
5.20.2.5 tlp_processSend()	201
5.20.2.6 tlp_publish()	202
5.20.2.7 tlp_put()	203
5.20.2.8 tlp_putImmediate()	204
5.20.2.9 tlp_republish()	204
5.20.2.10 tlp_request()	205
5.20.2.11 tlp_resubscribe()	206
5.20.2.12 tlp_setRedundant()	207
5.20.2.13 tlp_subscribe()	207
5.20.2.14 tlp_unpublish()	208
5.20.2.15 tlp_unsubscribe()	209
5.21 trdp_if_light.h File Reference	209
5.21.1 Detailed Description	212
5.21.2 Function Documentation	213
5.21.2.1 tlc_closeSession()	213
5.21.2.2 tlc_configSession()	213
5.21.2.3 tlc_getETBTopoCount()	214
5.21.2.4 tlc_getInterval()	214
5.21.2.5 tlc_getJoinStatistics()	215
5.21.2.6 tlc_getOpTrainTopoCount()	215
5.21.2.7 tlc_getOwnIpAddress()	216

5.21.2.8 tlc_getPubStatistics()	216
5.21.2.9 tlc_getRedStatistics()	217
5.21.2.10 tlc_getStatistics()	217
5.21.2.11 tlc_getSubsStatistics()	218
5.21.2.12 tlc_getVersion()	218
5.21.2.13 tlc_getVersionString()	219
5.21.2.14 tlc_init()	219
5.21.2.15 tlc_openSession()	219
5.21.2.16 tlc_presetIndexSession()	220
5.21.2.17 tlc_process()	221
5.21.2.18 tlc_reinitSession()	221
5.21.2.19 tlc_resetStatistics()	222
5.21.2.20 tlc_setETBTopoCount()	222
5.21.2.21 tlc_setOpTrainTopoCount()	223
5.21.2.22 tlc_terminate()	223
5.21.2.23 tlc_updateSession()	224
5.21.2.24 tlp_get()	224
5.21.2.25 tlp_getInterval()	225
5.21.2.26 tlp_getRedundant()	225
5.21.2.27 tlp_processReceive()	226
5.21.2.28 tlp_processSend()	226
5.21.2.29 tlp_publish()	227
5.21.2.30 tlp_put()	228
5.21.2.31 tlp_putImmediate()	229
5.21.2.32 tlp_republish()	229
5.21.2.33 tlp_request()	230
5.21.2.34 tlp_resubscribe()	231
5.21.2.35 tlp_setRedundant()	232
5.21.2.36 tlp_subscribe()	232
5.21.2.37 tlp_unpublish()	233
5.21.2.38 tlp_unsubscribe()	234
5.22 trdp_serviceRegistry.h File Reference	234
5.22.1 Detailed Description	237
5.22.2 Macro Definition Documentation	238
5.22.2.1 SOA_SAME_SERVICEID	238
5.22.2.2 SRM_SERVICE_READ_REQ_COMID	238
5.22.2.3 SRM_SRVINFO_NOTIFY_COMID	238
5.23 trdp_stats.c File Reference	239
5.23.1 Detailed Description	240
5.23.2 Function Documentation	240
5.23.2.1 tlc_getJoinStatistics()	240
5.23.2.2 tlc_getPubStatistics()	241

5.23.2.3 tlc_getRedStatistics()	241
5.23.2.4 tlc_getStatistics()	242
5.23.2.5 tlc_getSubsStatistics()	242
5.23.2.6 tlc_resetStatistics()	243
5.23.2.7 trdp_initStats()	243
5.23.2.8 trdp_pdPrepareStats()	244
5.23.2.9 trdp_UpdateStats()	244
5.24 trdp_tsn_def.h File Reference	245
5.24.1 Detailed Description	245
5.24.2 Macro Definition Documentation	246
5.24.2.1 TRDP_PD_DEFAULT_TSN_PRIORITY	246
5.25 trdp_types.h File Reference	246
5.25.1 Detailed Description	251
5.25.2 Macro Definition Documentation	251
5.25.2.1 TRDP_FLAGS_DEFAULT	252
5.25.2.2 USE_HEAP	252
5.25.3 Typedef Documentation	252
5.25.3.1 TRDP_IP_ADDR_T	252
5.25.3.2 TRDP_MARSHALL_T	252
5.25.3.3 TRDP_MD_CALLBACK_T	253
5.25.3.4 TRDP_PD_CALLBACK_T	253
5.25.3.5 TRDP_PRINT_DBG_T	253
5.25.3.6 TRDP_TIME_T	254
5.25.3.7 TRDP_UNMARSHALL_T	254
5.25.4 Enumeration Type Documentation	254
5.25.4.1 TRDP_DATA_TYPE_T	254
5.25.4.2 TRDP_ERR_T	255
5.25.4.3 TRDP_RED_STATE_T	257
5.25.4.4 TRDP_REPLY_STATUS_T	257
5.25.4.5 TRDP_TO_BEHAVIOR_T	257
5.26 vos_mem.c File Reference	258
5.26.1 Detailed Description	259
5.26.2 Function Documentation	260
5.26.2.1 vos_bsearch()	260
5.26.2.2 vos_memAlloc()	260
5.26.2.3 vos_memCount()	261
5.26.2.4 vos_memDelete()	261
5.26.2.5 vos_memFree()	261
5.26.2.6 vos_memInit()	262
5.26.2.7 vos_qsort()	262
5.26.2.8 vos_queueCreate()	263
5.26.2.9 vos_queueDestroy()	263

5.26.2.10 vos_queueReceive()	264
5.26.2.11 vos_queueSend()	264
5.26.2.12 vos_strncat()	265
5.26.2.13 vos_strncpy()	265
5.26.2.14 vos_strncmp()	266
5.27 vos_mem.h File Reference	266
5.27.1 Detailed Description	268
5.27.2 Macro Definition Documentation	269
5.27.2.1 VOS_MEM_MAX_PREALLOCATE	269
5.27.2.2 VOS_MEM_PREALLOCATE	269
5.27.3 Function Documentation	269
5.27.3.1 vos_bsearch()	269
5.27.3.2 vos_memAlloc()	270
5.27.3.3 vos_memCount()	270
5.27.3.4 vos_memDelete()	271
5.27.3.5 vos_memFree()	271
5.27.3.6 vos_memInit()	272
5.27.3.7 vos_qsort()	272
5.27.3.8 vos_queueCreate()	273
5.27.3.9 vos_queueDestroy()	273
5.27.3.10 vos_queueReceive()	274
5.27.3.11 vos_queueSend()	274
5.27.3.12 vos_strncat()	275
5.27.3.13 vos_strncpy()	275
5.27.3.14 vos_strncmp()	276
5.28 vos_shared_mem.h File Reference	276
5.28.1 Detailed Description	277
5.28.2 Function Documentation	278
5.28.2.1 vos_sharedClose()	278
5.28.2.2 vos_sharedOpen()	278
5.29 vos_sock.h File Reference	279
5.29.1 Detailed Description	282
5.29.2 Macro Definition Documentation	282
5.29.2.1 VOS_MAX_NUM_IF	282
5.29.2.2 VOS_MAX_SOCKET_CNT	282
5.29.2.3 vos_sockCmp	282
5.29.2.4 vos_sockId	283
5.29.2.5 VOS_TTL_MULTICAST	283
5.29.3 Function Documentation	283
5.29.3.1 vos_determineBindAddr()	283
5.29.3.2 vos_dottedIP()	284
5.29.3.3 vos_getInterfaces()	284

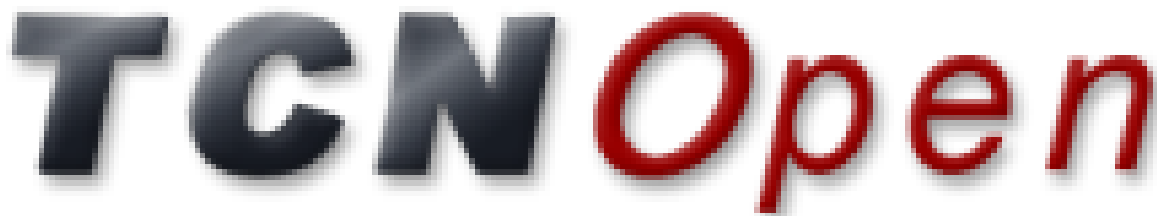
5.29.3.4 vos_htonl()	285
5.29.3.5 vos_htonll()	285
5.29.3.6 vos_htons()	285
5.29.3.7 vos_ifnameFromVlanId()	286
5.29.3.8 vos_ipDotted()	286
5.29.3.9 vos_isMulticast()	287
5.29.3.10 vos_netIfUp()	287
5.29.3.11 vos_ntohl()	287
5.29.3.12 vos_ntohll()	288
5.29.3.13 vos_ntohs()	288
5.29.3.14 vos_select()	288
5.29.3.15 vos_sockAccept()	289
5.29.3.16 vos_sockBind()	290
5.29.3.17 vos_sockClose()	290
5.29.3.18 vos_sockConnect()	291
5.29.3.19 vos_sockGetMAC()	291
5.29.3.20 vos_sockInit()	291
5.29.3.21 vos_sockJoinMC()	292
5.29.3.22 vos_sockLeaveMC()	292
5.29.3.23 vos_sockListen()	293
5.29.3.24 vos_sockOpenTCP()	293
5.29.3.25 vos_sockOpenTSN()	294
5.29.3.26 vos_sockOpenUDP()	294
5.29.3.27 vos_sockPrintOptions()	295
5.29.3.28 vos_sockReceiveTCP()	295
5.29.3.29 vos_sockReceiveTSN()	296
5.29.3.30 vos_sockReceiveUDP()	296
5.29.3.31 vos_sockSendTCP()	297
5.29.3.32 vos_sockSendTSN()	298
5.29.3.33 vos_sockSendUDP()	298
5.29.3.34 vos_sockSetMulticastIf()	299
5.29.3.35 vos_sockSetOptions()	299
5.29.3.36 vos_sockTerm()	300
5.30 vos_thread.h File Reference	300
5.30.1 Detailed Description	303
5.30.2 Function Documentation	303
5.30.2.1 vos_addTime()	303
5.30.2.2 vos_clearTime()	304
5.30.2.3 vos_cmpTime()	304
5.30.2.4 vos_divTime()	304
5.30.2.5 vos_getRealTime()	305
5.30.2.6 vos_getTime()	305

5.30.2.7 vos_getTimeStamp()	305
5.30.2.8 vos_getUuid()	305
5.30.2.9 vos_mulTime()	306
5.30.2.10 vos_mutexCreate()	306
5.30.2.11 vos_mutexDelete()	306
5.30.2.12 vos_mutexLock()	307
5.30.2.13 vos_mutexTryLock()	307
5.30.2.14 vos_mutexUnlock()	308
5.30.2.15 vos_semaCreate()	308
5.30.2.16 vos_semaDelete()	309
5.30.2.17 vos_semaGive()	309
5.30.2.18 vos_semaTake()	309
5.30.2.19 vos_subTime()	310
5.30.2.20 vos_threadCreate()	310
5.30.2.21 vos_threadCreateSync()	311
5.30.2.22 vos_threadDelay()	312
5.30.2.23 vos_threadInit()	312
5.30.2.24 vos_threadIsActive()	312
5.30.2.25 vos_threadSelf()	313
5.30.2.26 vos_threadTerm()	313
5.30.2.27 vos_threadTerminate()	313
5.31 vos_types.h File Reference	314
5.31.1 Detailed Description	316
5.31.2 Typedef Documentation	316
5.31.2.1 VOS_PRINT_DBG_T	316
5.31.2.2 VOS_TIMEVAL_T	316
5.31.3 Enumeration Type Documentation	317
5.31.3.1 VOS_ERR_T	317
5.31.3.2 VOS_LOG_T	318
5.32 vos_utils.c File Reference	318
5.32.1 Detailed Description	319
5.32.2 Function Documentation	319
5.32.2.1 vos_crc32()	319
5.32.2.2 vos_getErrorString()	320
5.32.2.3 vos_getVersion()	320
5.32.2.4 vos_getVersionString()	320
5.32.2.5 vos_hostIsBigEndian()	321
5.32.2.6 vos_init()	321
5.32.2.7 vos_sc32()	322
5.32.2.8 vos_terminate()	322
5.33 vos_utils.h File Reference	322
5.33.1 Detailed Description	324

5.33.2 Macro Definition Documentation	325
5.33.2.1 INITFCS	325
5.33.2.2 VOS_MAX_ERR_STR_SIZE	325
5.33.2.3 VOS_MAX_FRMT_SIZE	325
5.33.2.4 VOS_MAX_PRNT_STR_SIZE	325
5.33.3 Function Documentation	325
5.33.3.1 vos_crc32()	325
5.33.3.2 vos_getErrorString()	326
5.33.3.3 vos_getVersion()	326
5.33.3.4 vos_getVersionString()	327
5.33.3.5 vos_hostIsBigEndian()	327
5.33.3.6 vos_init()	327
5.33.3.7 vos_sc32()	328
5.33.3.8 vos_terminate()	329
Index	331

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:

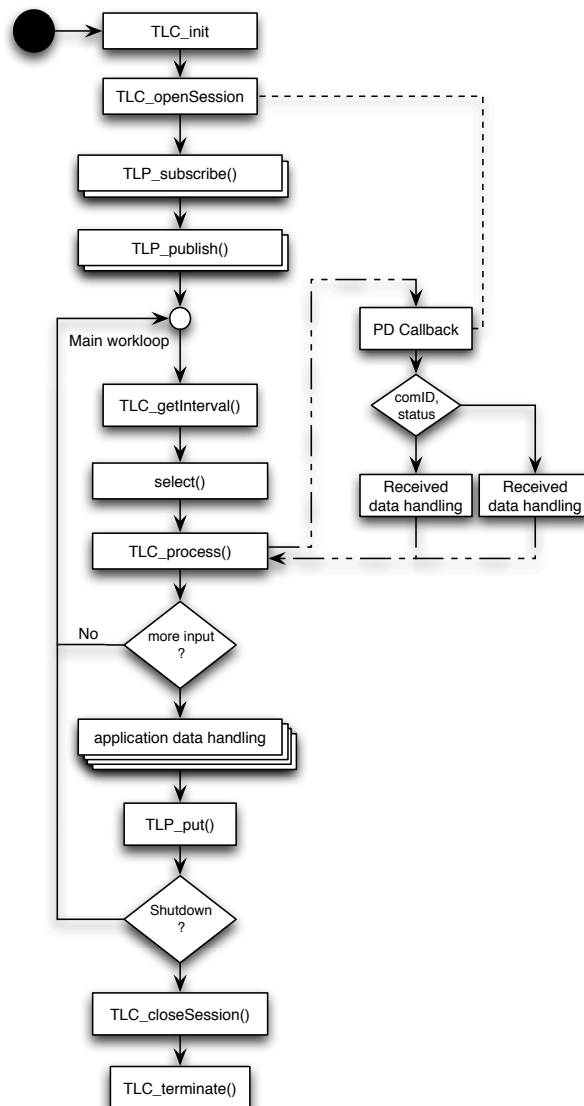


Figure 1.1 Sample client workflow (Single Thread)

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

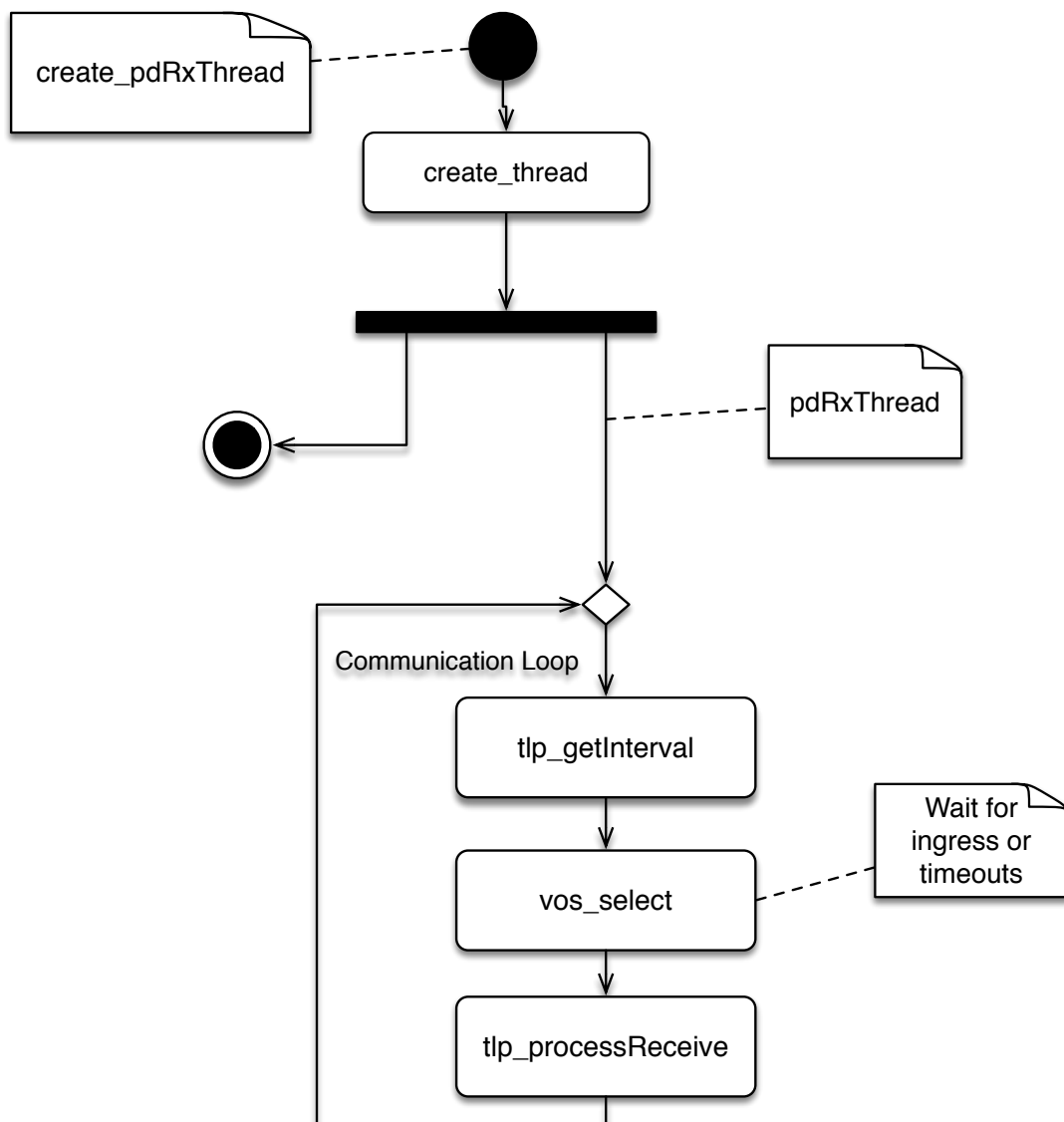


Figure 1.2 Multi-threaded processing of PD Reception

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

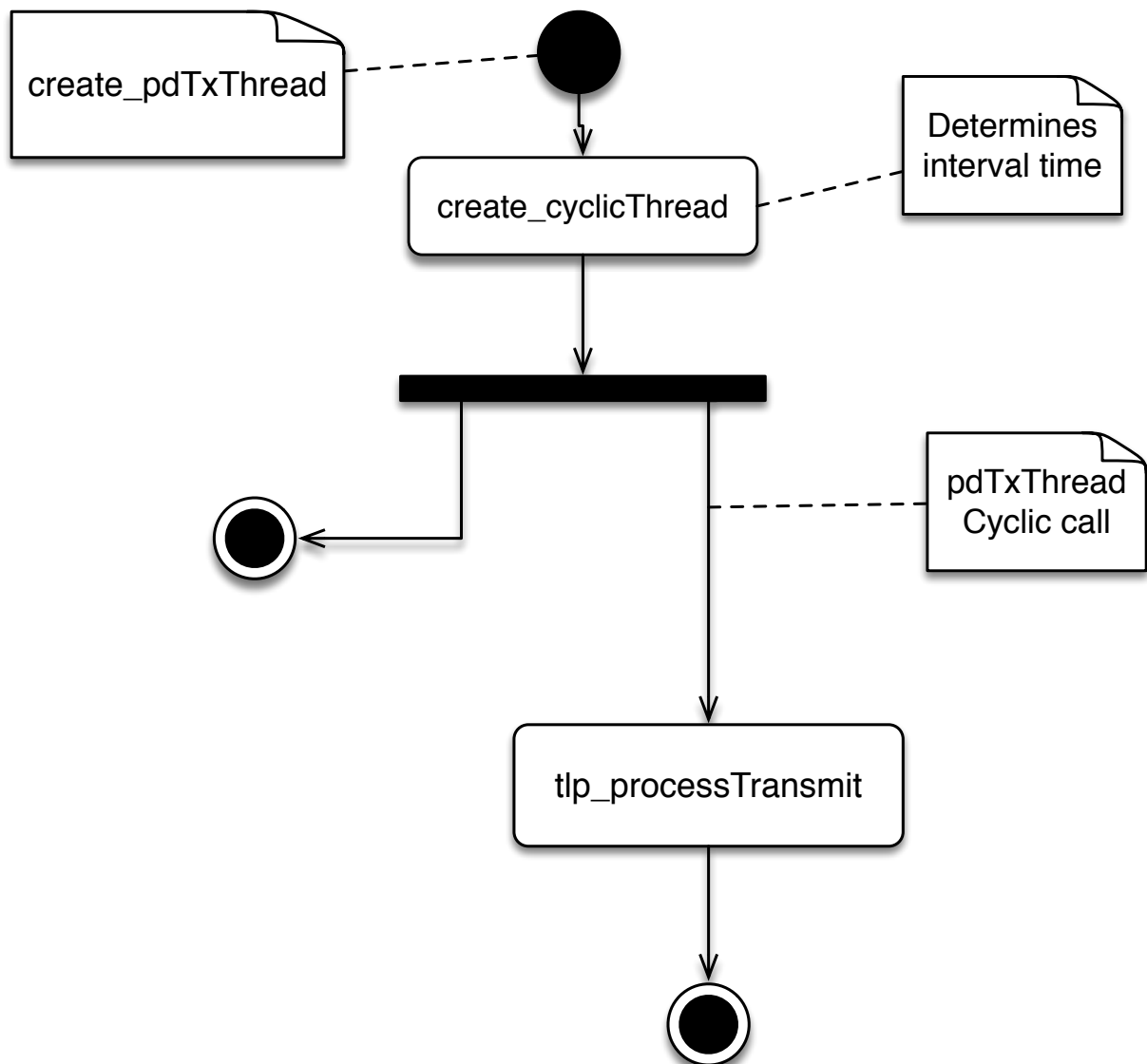


Figure 1.3 Multi-threaded processing of PD Transmit

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

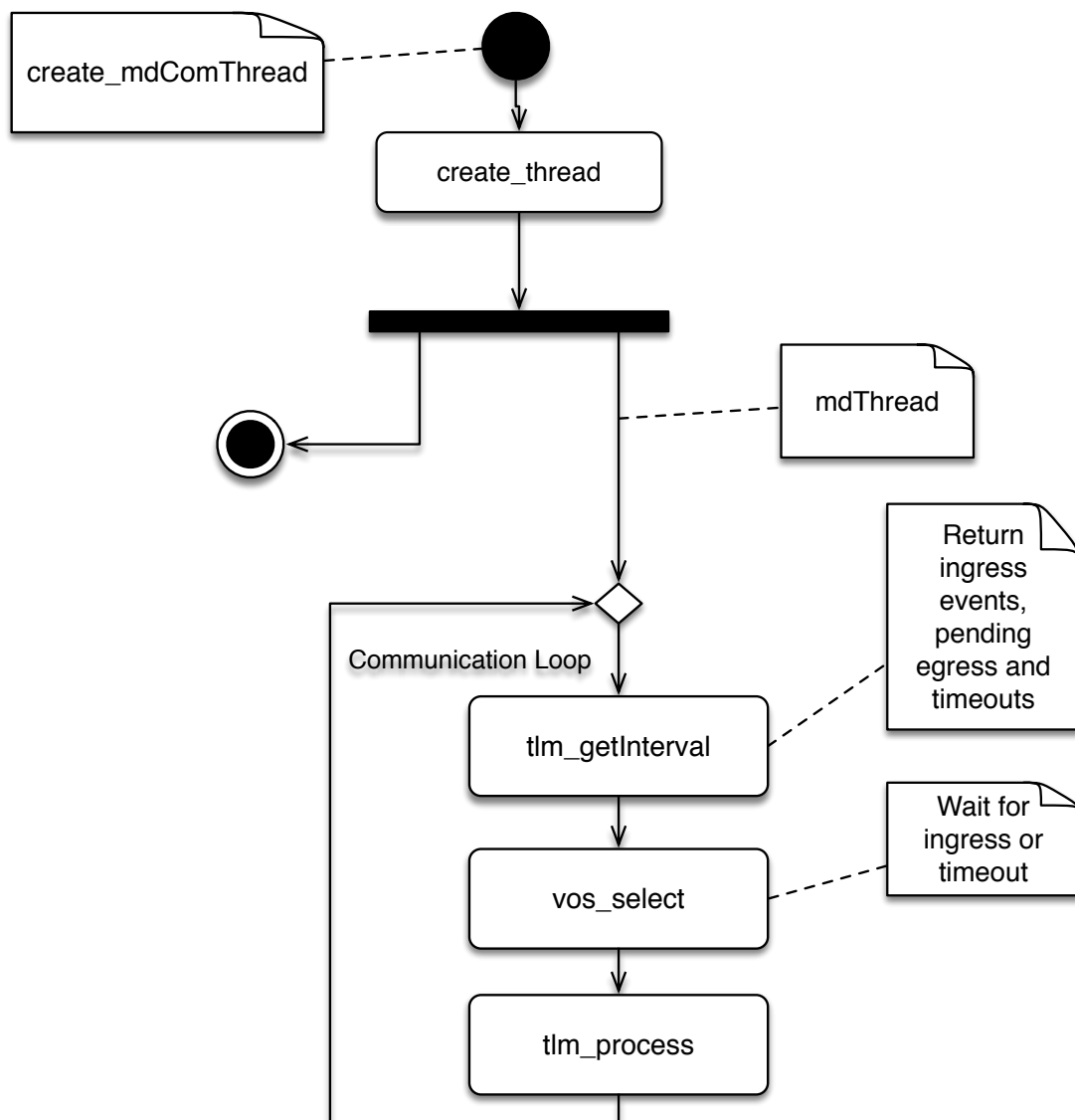


Figure 1.4 Multi-threaded processing of MD

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:

SRM_SERVICE_INFO_T	
Preliminary definition of a service info entry	13
SRM_SRV_INFO_REQ_T	
Preliminary definition of a service info request	15
TAU_DNS_HEADER_T	
DNS header structure	16
TAU_MARSHALL_INFO_T	
Marshalling info, used to and from wire	16
TCN_URI_T	
TCN-DNS simplified header structures	16
TRDP_CLTR_CST_INFO_T	
Closed train consists information	17
TRDP_COM_PARAM_T	
Quality/type of service, time to live , no	18
TRDP_COMID_DSID_MAP_T	
ComId - data set mapping element definition	
18	
TRDP_CONF_VEHICLE_T	
Types for ETB control	19
TRDP_CONSIST_INFO_T	
Consist information structure	19
TRDP_CSTINFOCTRL_T	
CSTINFO Control telegram	21
TRDP_DATASET_ELEMENT_T	
Dataset element definition	
23	
TRDP_DATASET_T	
Dataset definition	
24	
TRDP_DBG_CONFIG_T	
Control for debug output device/file on application level	25
TRDP_DNS_REPLY_T	
TCN-DNS Reply telegram TCN_DNS_REP_DS	25
TRDP_DNS_REQUEST_T	
TCN-DNS Request telegram TCN_DNS_REQ_DS	27

TRDP_ETB_CTRL_VDP_T	
TCN consist structure	28
TRDP_ETB_INFO_T	
Types for train configuration information	29
TRDP_FUNCTION_INFO_T	
Function/device information structure	30
TRDP_IDX_TABLE_T	
Settings for pre-allocation of index tables for application session initialization	31
TRDP_LIST_STATISTICS_T	
Information about a particular MD listener	33
TRDP_MARSHALL_CONFIG_T	
Marshaling/unmarshalling configuration	
34	
TRDP_MD_CONFIG_T	
Default MD configuration	35
TRDP_MD_INFO_T	
Message data info from received telegram; allows the application to generate responses . . .	36
TRDP_MD_STATISTICS_T	
Structure containing all general MD statistics information	38
TRDP_MEM_CONFIG_T	
Enumeration type for memory pre-fragmentation, reuse of VOS definition	39
TRDP_OP_CONSIST_T	
Operational consist structure	39
TRDP_OP_TRAIN_DIR_STATE_T	
Operational train directory state	40
TRDP_OP_TRAIN_DIR_STATUS_INFO_T	
Operational Train directory status info structure	42
TRDP_OP_TRAIN_DIR_T	
Operational train structure	43
TRDP_OP_VEHICLE_T	
Operational vehicle structure	44
TRDP_PD_CONFIG_T	
Default PD configuration	
45	
TRDP_PD_INFO_T	
Process data info from received telegram; allows the application to generate responses	46
TRDP_PD_STATISTICS_T	
Structure containing all general PD statistics information	48
TRDP_PROCESS_CONFIG_T	
Various flags/general TRDP options for library initialization	49
TRDP_PROP_T	
Application defined properties	49
TRDP_PUB_STATISTICS_T	
Table containing particular PD publishing information	50
TRDP_READ_COMPLETE_REPLY_T	
Complete TTDB structure	51
TRDP_RED_STATISTICS_T	
A table containing PD redundant group information	52
TRDP_SDT_PAR_T	
Types to read out the XML configuration	
53	
TRDP_SHORT_VERSION_T	
Version information for communication buffers	53
TRDP_STATISTICS_REQUEST_T	
TRDP statistics type definitions	54
TRDP_STATISTICS_T	
Structure containing all general memory, PD and MD statistics information	55

TRDP_SUBS_STATISTICS_T	
Table containing particular PD subscription information	56
TRDP_TRAIN_DIR_T	
TCN train directory	57
TRDP_TRAIN_NET_DIR_ENTRY_T	
Train network directory entry structure acc	58
TRDP_TRAIN_NET_DIR_T	
Train network directory structure	59
TRDP_VEHICLE_INFO_T	
Vehicle information structure	60
TRDP_XML_DOC_HANDLE_T	
Parsed XML document handle	61
VOS_MEM_STATISTICS_T	
Structure containing all general memory statistics information	61
VOS_QUEUE_T	
Opaque queue define	
62	
VOS_SOCKET_OPT_T	
Common socket options	
62	
VOS_VERSION_T	
Version information	63

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	65
tau_cstinfo.c	Functions for consist information access	75
tau_ctrl.c	Functions for train switch control	77
tau_ctrl.h	TRDP utility interface definitions	81
tau_ctrl_types.h	TRDP utility interface definitions	87
tau_dnr.c	Functions for domain name resolution	89
tau_dnr.h	TRDP utility interface definitions	94
tau_dnr_types.h	TRDP utility interface definitions	101
tau_marshall.c	Marshalling functions for TRDP	103
tau_marshall.h	TRDP utility interface definitions	109
tau_so_if.c	Access to service oriented functions of the SRM	119
tau_so_if.h	Access to the Service Registry	123
tau_tti.c	Functions for train topology information access	127
tau_tti.h	TRDP utility interface definitions	140
tau_tti_types.h	TRDP utility interface definitions	153
tau_xml.c	Functions for XML file parsing	156
tau_xml.h	TRDP utility interface definitions	165
tlc_if.c	Functions for ECN communication	176

tlm_if.c	Functions for Message Data Communication	187
tlp_if.c	Functions for Process Data Communication	197
trdp_if_light.h	TRDP Light interface functions (API)	209
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	234
trdp_stats.c	Statistics functions for TRDP communication	239
trdp_tsn_def.h	Additional definitions for TSN	245
trdp_types.h	Typedefs for TRDP communication	246
vos_mem.c	Memory functions	258
vos_mem.h	Memory and queue functions for OS abstraction	266
vos_shared_mem.h	Shared Memory functions for OS abstraction	276
vos_sock.h	Typedefs for OS abstraction	279
vos_thread.h	Threading functions for OS abstraction	300
vos_types.h	Typedefs for OS abstraction	314
vos_utils.c	Common functions for VOS	318
vos_utils.h	Typedefs for OS abstraction	322

Chapter 4

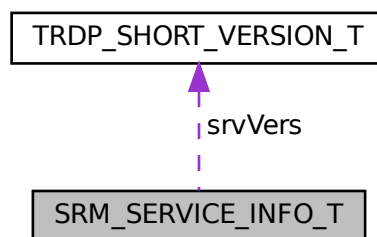
Data Structure Documentation

4.1 SRM_SERVICE_INFO_T Struct Reference

Preliminary definition of a service info entry.

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

Collaboration diagram for SRM_SERVICE_INFO_T:



Data Fields

- [TRDP_NET_LABEL_T](#) `srvName`
– *service short name as defined in X*
- [UINT32](#) `serviceId`
– *High Byte = serviceInstanceId Low 24 Bits = serviceTypeId*
- [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*

4.1.1 Detailed Description

Preliminary definition of a service info entry.

4.1.2 Field Documentation

4.1.2.1 fctDev

`TRDP_NET_LABEL_T SRM_SERVICE_INFO_T::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:

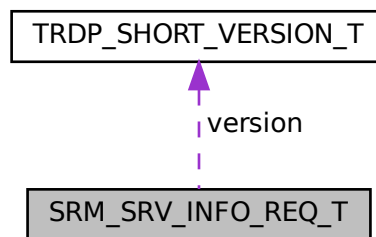
- `trdp_serviceRegistry.h`

4.2 SRM_SRV_INFO_REQ_T Struct Reference

Preliminary definition of a service info request.

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

Collaboration diagram for SRM_SRV_INFO_REQ_T:



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*

4.2.1 Detailed Description

Preliminary definition of a service info request.

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

- [trdp_serviceRegistry.h](#)

4.3 TAU_DNS_HEADER_T Struct Reference

DNS header structure.

4.3.1 Detailed Description

DNS header structure.

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

- [tau_dnr.c](#)

4.4 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

4.4.1 Detailed Description

Marshalling info, used to and from wire.

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

- [tau_marshall.c](#)

4.5 TCN_URI_T 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

4.5.1 Detailed Description

TCN-DNS simplified header structures.

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

- [tau_dnr_types.h](#)

4.6 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)

4.6.1 Detailed Description

Closed train consists information.

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

- [tau_tti_types.h](#)

4.7 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

4.7.1 Detailed Description

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

of retries

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

- [trdp_types.h](#)

4.8 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

4.8.1 Detailed Description

ComId - data set mapping element definition

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

- [trdp_types.h](#)

4.9 TRDP_CONF_VEHICLE_T Struct Reference

Types for ETB control.

```
#include <tau_ctrl_types.h>
```

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

4.9.1 Detailed Description

Types for ETB control.

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

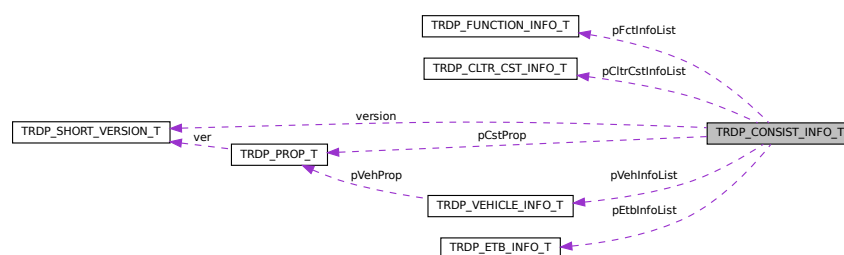
- [tau_ctrl_types.h](#)

4.10 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](#) * [pCstProp](#)
static consist properties #378
- [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 etbId.
- [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

4.10.1 Detailed Description

consist information structure

4.10.2 Field Documentation

4.10.2.1 cstId

[TRDP_NET_LABEL_T](#) TRDP_CONSIST_INFO_T::cstId

application defined consist identifier, e.g.

UIC identifier

4.10.2.2 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:

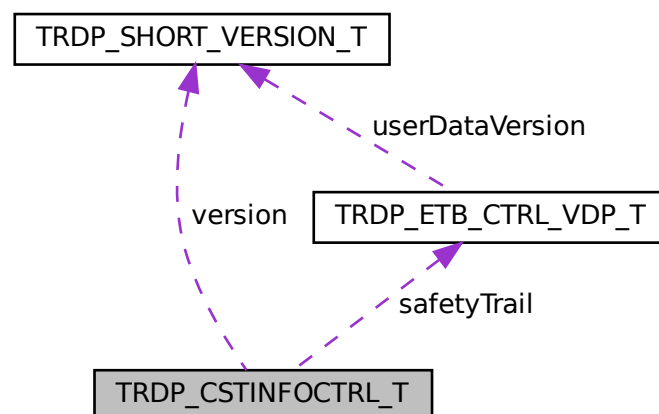
- [tau_tti_types.h](#)

4.11 TRDP_CSTINFOCTRL_T Struct Reference

CSTINFO Control telegram.

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

Collaboration diagram for TRDP_CSTINFOCTRL_T:



Data Fields

- [TRDP_SHORT_VERSION_T version](#)
Consist Info Control structure version parameter 'mainVersion' shall be set to 1.
- [UINT8 trnCstNo](#)
train consist number telegram control type 0 = with trnTopoCnt tracking 1 = without trnTopoCnt tracking
- [UINT8 cstCnt](#)
number of consists in train; range: 1..63
- [TRDP_CONSIST_T cstList \[63u\]](#)
consist list.
- [UINT32 trnTopoCnt](#)
trnTopoCnt value ctrlType == 0: actual value ctrlType == 1: set to 0
- [TRDP_ETB_CTRL_VDP_T safetyTrail](#)
ETBCTRL-VDP trailer, parameter 'safeSequCount' == 0 completely set to 0 == not used

4.11.1 Detailed Description

CSTINFO Control telegram.

4.11.2 Field Documentation

4.11.2.1 cstList

```
TRDP_CONSIST_T TRDP_CSTINFOCTRL_T::cstList[63u]
```

consist list.

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.

4.11.2.2 version

```
TRDP\_SHORT\_VERSION\_T TRDP_CSTINFOCTRL_T::version
```

Consist Info Control structure version parameter 'mainVersion' shall be set to 1.

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

- [tau_tti_types.h](#)

4.12 TRDP_DATASET_ELEMENT_T Struct Reference

Dataset element definition

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

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 marshalling speed-up

4.12.1 Detailed Description

Dataset element definition

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

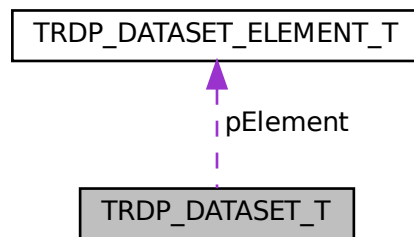
- [trdp_types.h](#)

4.13 TRDP_DATASET_T Struct Reference

Dataset definition

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

Collaboration diagram for TRDP_DATASET_T:



Data Fields

- [UINT32 id](#)
dataset identifier > 1000
- [UINT16 reserved1](#)
Reserved for future use, must be zero
- [UINT16 numElement](#)
Number of elements
- [TRDP_EXTRA_LABEL_T name](#)
Dataset name #349
- [TRDP_DATASET_ELEMENT_T pElement \[\]](#)
Pointer to a dataset element, used as array

4.13.1 Detailed Description

Dataset definition

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

- [trdp_types.h](#)

4.14 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.

4.14.1 Detailed Description

Control for debug output device/file on application level.

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

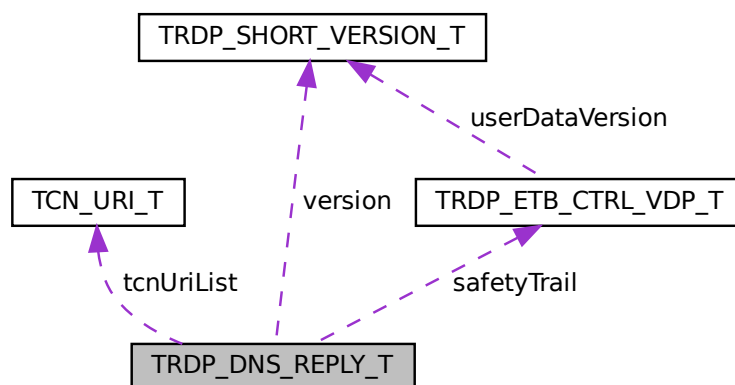
- [tau_xml.h](#)

4.15 TRDP_DNS_REPLY_T Struct Reference

TCN-DNS Reply telegram TCN_DNS_REP_DS.

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

Collaboration diagram for TRDP_DNS_REPLY_T:



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_URL_T](#) [tcnUriList](#) [255]
defined for max size
- [TRDP_ETB_CTRL_VDP_T](#) [safetyTrail](#)
SDT trailer

4.15.1 Detailed Description

TCN-DNS Reply telegram TCN_DNS_REP_DS.

4.15.2 Field Documentation

4.15.2.1 tcnUriCnt

```
UINT8 TRDP_DNS_REPLY_T::tcnUriCnt
```

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

. 255

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

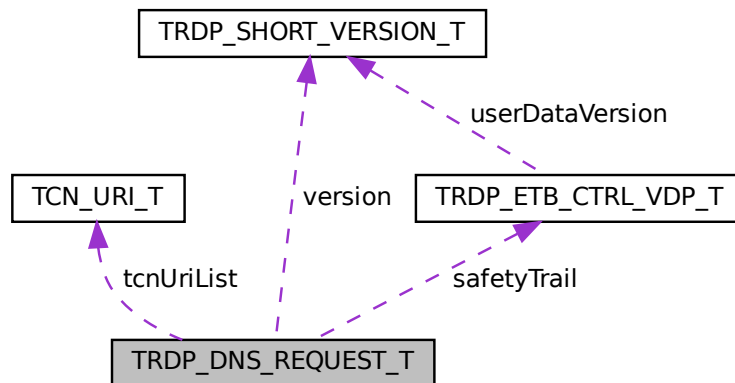
- [tau_dnr_types.h](#)

4.16 TRDP_DNS_REQUEST_T Struct Reference

TCN-DNS Request telegram TCN_DNS_REQ_DS.

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

Collaboration diagram for TRDP_DNS_REQUEST_T:



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)
- `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

4.16.1 Detailed Description

TCN-DNS Request telegram TCN_DNS_REQ_DS.

4.16.2 Field Documentation

4.16.2.1 tcnUriCnt

UINT8 TRDP_DNS_REQUEST_T::tcnUriCnt

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

. 255

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

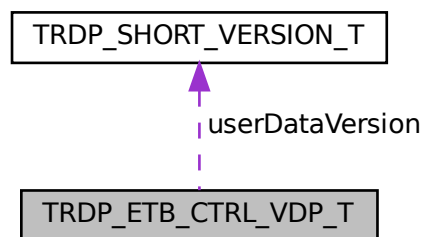
- [tau_dnr_types.h](#)

4.17 TRDP_ETB_CTRL_VDP_T Struct Reference

TCN consist structure.

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

Collaboration diagram for TRDP_ETB_CTRL_VDP_T:



Data Fields

- UINT32 [reserved01](#)
reserved (=0)
- UINT16 [reserved02](#)
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

4.17.1 Detailed Description

TCN consist structure.

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

- [tau_tti_types.h](#)

4.18 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)

4.18.1 Detailed Description

Types for train configuration information.

ETB information

4.18.2 Field Documentation

4.18.2.1 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](#)

4.19 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 etbld](#)
number of connected train backbone.
- [UINT8 cnld](#)
identifier of connected consist network in the consist, related to the etbld.
- [UINT8 reserved02](#)
reserved for future use (= 0)

4.19.1 Detailed Description

function/device information structure

4.19.2 Field Documentation

4.19.2.1 cnld

```
UINT8 TRDP_FUNCTION_INFO_T::cnld
```

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

Value range: 0..31

4.19.2.2 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

4.19.2.3 etbId

UINT8 TRDP_FUNCTION_INFO_T::etbId

number of connected train backbone.

Value range: 0..3

4.19.2.4 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](#)

4.20 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 (base 2: <= 128ms)
- UINT32 [maxNoOfMidCatPublishers](#)
Max.
- UINT32 [maxDepthOfMidCatPublishers](#)
depth / overlapped publishers with intervals <= 1000ms (base 2: <= 1024ms)
- UINT32 [maxNoOfHighCatPublishers](#)
Max.
- UINT32 [maxDepthOfHighCatPublishers](#)
depth / overlapped publishers with intervals <= 10000ms (base 2: <= 8192ms)
- UINT32 [maxNoOfExtPublishers](#)
Max.

4.20.1 Detailed Description

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

4.20.2 Field Documentation

4.20.2.1 maxNoOfExtPublishers

UINT32 TRDP_IDX_TABLE_T::maxNoOfExtPublishers

Max.

number of expected publishers with intervals > 10000ms (base 2: > 8192ms)

4.20.2.2 maxNoOfHighCatPublishers

UINT32 TRDP_IDX_TABLE_T::maxNoOfHighCatPublishers

Max.

number of expected publishers with intervals <= 10000ms (base 2: <= 8192ms)

4.20.2.3 maxNoOfHighCatSubscriptions

UINT32 TRDP_IDX_TABLE_T::maxNoOfHighCatSubscriptions

Max.

number of expected subscriptions with intervals > 1000ms (base 2: > 1024ms)

4.20.2.4 maxNoOfLowCatPublishers

UINT32 TRDP_IDX_TABLE_T::maxNoOfLowCatPublishers

Max.

number of expected publishers with intervals <= 100ms (base 2: <= 128ms)

4.20.2.5 maxNoOfLowCatSubscriptions

UINT32 TRDP_IDX_TABLE_T::maxNoOfLowCatSubscriptions

Max.

number of expected subscriptions with intervals <= 100ms (base 2: <= 128ms)

4.20.2.6 maxNoOfMidCatPublishers

```
UINT32 TRDP_IDX_TABLE_T::maxNoOfMidCatPublishers
```

Max.

number of expected publishers with intervals $\leq 1000\text{ms}$ (base 2: $\leq 1024\text{ms}$)

4.20.2.7 maxNoOfMidCatSubscriptions

```
UINT32 TRDP_IDX_TABLE_T::maxNoOfMidCatSubscriptions
```

Max.

number of expected subscriptions with intervals $\leq 1000\text{ms}$ (base 2: $\leq 1024\text{ms}$)

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

- [trdp_types.h](#)

4.21 TRDP_LIST_STATISTICS_T Struct Reference

Information about a particular MD listener.

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

Data Fields

- [UINT32 comId](#)
ComId to listen to.
- [CHAR8 uri](#) [32]
URI user part to listen to.
- [TRDP_IP_ADDR_T](#) joinedAddr
Joined IP address.
- [UINT32 callBack](#)
Call back function if used.
- [UINT32 queue](#)
Queue reference if used.
- [UINT32 userRef](#)
User reference if used.
- [UINT32 numRecv](#)
Number of received packets

4.21.1 Detailed Description

Information about a particular MD listener.

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

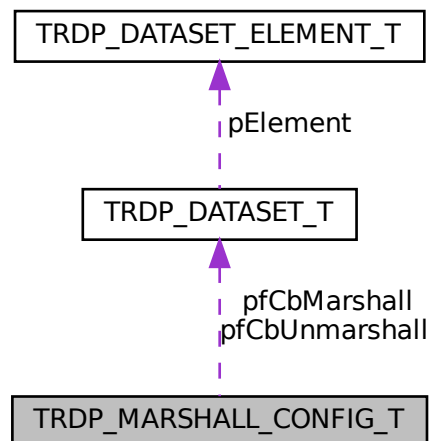
- [trdp_types.h](#)

4.22 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

4.22.1 Detailed Description

Marshaling/unmarshalling configuration

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

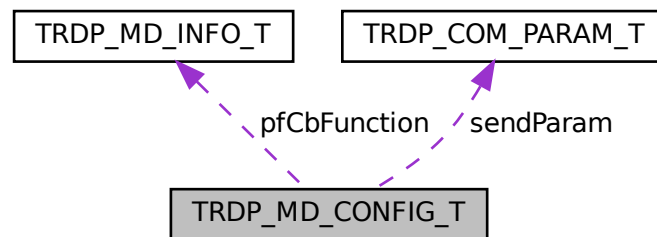
- [trdp_types.h](#)

4.23 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 pfCbFunction](#)
Pointer to MD callback function
- `void *` [pRefCon](#)
Pointer to user context for call back
- [TRDP_COM_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

4.23.1 Detailed Description

Default MD configuration.

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

- [trdp_types.h](#)

4.24 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

4.24.1 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](#)

4.25 TRDP_MD_STATISTICS_T Struct Reference

Structure containing all general MD statistics information.

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

Data Fields

- UINT32 [defQos](#)
default QoS for MD
- UINT32 [defTtl](#)
default TTL for MD
- UINT32 [defReplyTimeout](#)
default reply timeout in us for MD
- UINT32 [defConfirmTimeout](#)
default confirm timeout in us for MD
- UINT32 [numList](#)
number of listeners
- UINT32 [numRcv](#)
number of received MD packets
- UINT32 [numCrcErr](#)
number of received MD packets with CRC err
- UINT32 [numProtErr](#)
number of received MD packets with protocol err
- UINT32 [numTopoErr](#)
number of received MD packets with wrong topo count
- UINT32 [numNoListener](#)
number of received MD packets without listener
- UINT32 [numReplyTimeout](#)
number of reply timeouts
- UINT32 [numConfirmTimeout](#)
number of confirm timeouts
- UINT32 [numSend](#)
number of sent MD packets

4.25.1 Detailed Description

Structure containing all general MD statistics information.

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](#)

4.27 TRDP_OP_CONSIST_T Struct Reference

Operational consist structure.

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

Data Fields

- [TRDP_UUID_T](#) [cstUUID](#)
Reference to static consist attributes, 0 if not available (e.g.
- [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
- [UINT8](#) [trnCstNo](#)
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
- [UINT8](#) [reserved01](#)
reserved for future use (= 0)

4.27.1 Detailed Description

Operational consist structure.

4.27.2 Field Documentation

4.27.2.1 [cstUUID](#)

[TRDP_UUID_T](#) [TRDP_OP_CONSIST_T::cstUUID](#)

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

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

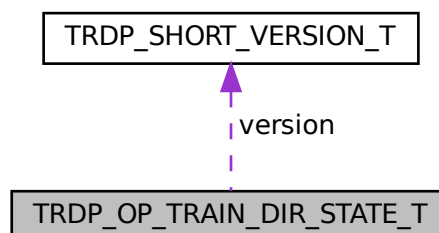
- [tau_tti_types.h](#)

4.28 TRDP_OP_TRAIN_DIR_STATE_T Struct Reference

Operational train directory state.

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

Collaboration diagram for [TRDP_OP_TRAIN_DIR_STATE_T](#):



Data Fields

- [TRDP_SHORT_VERSION_T version](#)
*TrainDirectoryState data structure version
parameter 'mainVersion' shall be set to 1.*
- [UINT8 reserved01](#)
reserved for future use (= 0)
- [UINT8 reserved02](#)
reserved for future use (= 0)
- [UINT8 etbld](#)
*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)*
- [UINT8 trnDirState](#)
TTDB status: '01'B == unconfirmed, '10'B == confirmed.
- [UINT8 opTrnDirState](#)
Operational train directory status: '01'B == invalid, '10'B == valid, '100'B == shared.
- [UINT8 reserved03](#)
reserved for future use (= 0)
- [TRDP_NET_LABEL_T trnId](#)
train identifier, application defined (e.g.
- [TRDP_NET_LABEL_T trnOperator](#)
train operator, e.g.
- [UINT32 opTrnTopoCnt](#)
operational train topology counter set to 0 if opTrnDirState == invalid
- [UINT32 crc](#)
sc-32 computed over record (seed value: 'FFFFFFFF'H)

4.28.1 Detailed Description

Operational train directory state.

4.28.2 Field Documentation

4.28.2.1 trnId

[TRDP_NET_LABEL_T](#) TRDP_OP_TRAIN_DIR_STATE_T::trnId

train identifier, application defined (e.g.

'ICE75', 'IC346'), informal

4.28.2.2 trnOperator

`TRDP_NET_LABEL_T TRDP_OP_TRAIN_DIR_STATE_T::trnOperator`

train operator, e.g.

'trenitalia.it', informal

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

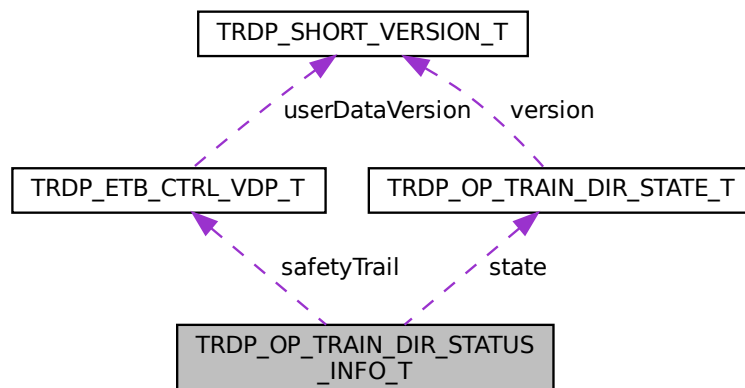
- [tau_tti_types.h](#)

4.29 TRDP_OP_TRAIN_DIR_STATUS_INFO_T Struct Reference

Operational Train directory status info structure.

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

Collaboration diagram for TRDP_OP_TRAIN_DIR_STATUS_INFO_T:



4.29.1 Detailed Description

Operational Train directory status info structure.

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

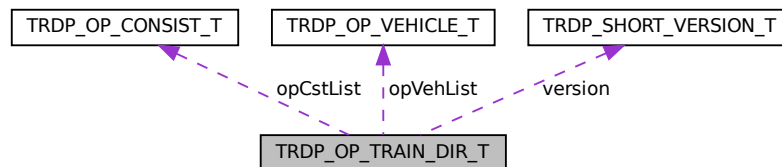
- [tau_tti_types.h](#)

4.30 TRDP_OP_TRAIN_DIR_T Struct Reference

Operational train structure.

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

Collaboration diagram for TRDP_OP_TRAIN_DIR_T:



Data Fields

- [TRDP_SHORT_VERSION_T version](#)
Train info structure version.
- [UINT8 etbId](#)
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)
- [UINT8 opTrnOrient](#)
operational train orientation '00'B = unknown '01'B = same as train direction '10'B = inverse to train direction
- [UINT8 reserved01](#)
reserved for future use (= 0)
- [UINT8 reserved02](#)
reserved for future use (= 0)
- [UINT8 reserved03](#)
reserved for future use (= 0)
- [UINT8 opCstCnt](#)
number of consists in train (1..63)
- [TRDP_OP_CONSIST_T opCstList](#) [63u]
operational consist list starting with op.
- [UINT8 reserved04](#)
reserved for future use (= 0)
- [UINT8 reserved05](#)
reserved for future use (= 0)
- [UINT8 reserved06](#)
reserved for future use (= 0)
- [UINT8 opVehCnt](#)
number of vehicles in train (1..63)
- [TRDP_OP_VEHICLE_T opVehList](#) [63u]
operational vehicle list starting with op.
- [UINT32 opTrnTopoCnt](#)
operational train topology counter computed as defined in 5.3.3.2.16 (seed value : trnTopoCnt)

4.30.1 Detailed Description

Operational train structure.

4.30.2 Field Documentation

4.30.2.1 opCstList

[TRDP_OP_CONSIST_T](#) [TRDP_OP_TRAIN_DIR_T::opCstList](#) [63u]

operational consist list starting with op.

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

4.30.2.2 opVehList

[TRDP_OP_VEHICLE_T](#) [TRDP_OP_TRAIN_DIR_T::opVehList](#) [63u]

operational vehicle list starting with op.

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

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

- [tau_tti_types.h](#)

4.31 TRDP_OP_VEHICLE_T Struct Reference

Operational vehicle structure.

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

Data Fields

- [TRDP_NET_LABEL_T](#) [vehId](#)
Unique vehicle identifier, application defined (e.g.
- [UINT8](#) [opVehNo](#)
operational vehicle sequence number in train value range 1..63
- [ANTIVALENT8](#) [isLead](#)
vehicle is leading
- [UINT8](#) [leadDir](#)
'vehicle leading direction 0 = not relevant 1 = leading direction 1 2 = leading direction 2
- [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: 1..63, a value of 0 indicates that this vehicle has been inserted by correction
- [UINT8](#) [vehOrient](#)
vehicle orientation, '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction
- [UINT8](#) [ownOpCstNo](#)
operational consist number the vehicle belongs to
- [UINT8](#) [reserved01](#)
reserved for future use (= 0)
- [UINT8](#) [reserved02](#)
reserved for future use (= 0)

4.31.1 Detailed Description

Operational vehicle structure.

4.31.2 Field Documentation

4.31.2.1 vehId

[TRDP_NET_LABEL_T](#) TRDP_OP_VEHICLE_T::vehId

Unique vehicle identifier, application defined (e.g.

UIC Identifier)

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

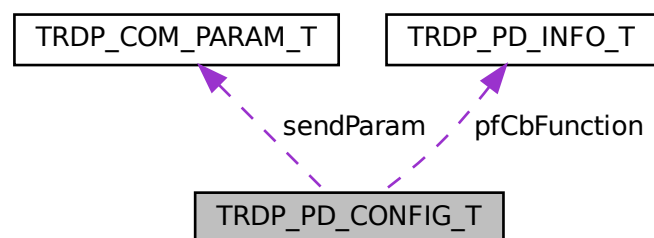
- [tau_tti_types.h](#)

4.32 TRDP_PD_CONFIG_T Struct Reference

Default PD configuration

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

Collaboration diagram for TRDP_PD_CONFIG_T:



Data Fields

- [TRDP_PD_CALLBACK_T pfCbFunction](#)
Pointer to PD callback function
- `void *` [pRefCon](#)
Pointer to user context for call back
- [TRDP_COM_PARAM_T sendParam](#)
Default send parameters
- [TRDP_FLAGS_T flags](#)
Default flags for PD packets
- [UINT32 timeout](#)
Default timeout in us
- [TRDP_TO_BEHAVIOR_T toBehavior](#)
Default timeout behavior
- [UINT16 port](#)
Port to be used for PD communication (default: 17224)

4.32.1 Detailed Description

Default PD configuration

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

- [trdp_types.h](#)

4.33 TRDP_PD_INFO_T Struct Reference

Process 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 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 serviceId](#)
the reserved field of the PD header

4.33.1 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](#)

4.34 TRDP_PD_STATISTICS_T Struct Reference

Structure containing all general PD statistics information.

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

Data Fields

- `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

4.34.1 Detailed Description

Structure containing all general PD statistics information.

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

- [trdp_types.h](#)

4.35 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
- TRDP_LABEL_T [type](#)
type #349
- 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.
- UINT16 [vlanId](#)
0=no VLAN, 1=reserved, 2..4094=VLAN-ID, 4095=wildcard match

4.35.1 Detailed Description

Various flags/general TRDP options for library initialization.

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

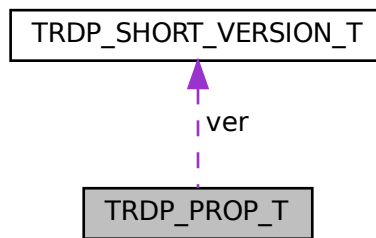
- [trdp_types.h](#)

4.36 TRDP_PROP_T Struct Reference

Application defined properties.

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

Collaboration diagram for TRDP_PROP_T:



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 []`
properties, application defined

4.36.1 Detailed Description

Application defined properties.

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

- [tau_tti_types.h](#)

4.37 TRDP_PUB_STATISTICS_T Struct Reference

Table containing particular PD publishing information.

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

Data Fields

- [UINT32 comId](#)
Published ComId
- [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.
- [UINT32 numSend](#)
Number of packets sent out.

4.37.1 Detailed Description

Table containing particular PD publishing information.

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

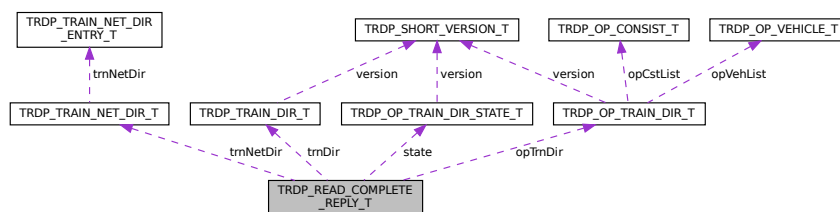
- [trdp_types.h](#)

4.38 TRDP_READ_COMPLETE_REPLY_T Struct Reference

Complete TTDB structure.

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

Collaboration diagram for TRDP_READ_COMPLETE_REPLY_T:



Data Fields

- [TRDP_OP_TRAIN_DIR_STATE_T state](#)
operational state of the train
- [TRDP_OP_TRAIN_DIR_T opTrnDir](#)
operational directory
- [TRDP_TRAIN_DIR_T trnDir](#)
train directory
- [TRDP_TRAIN_NET_DIR_T trnNetDir](#)
network directory

4.38.1 Detailed Description

Complete TTDB structure.

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

- [tau_tti_types.h](#)

4.39 TRDP_RED_STATISTICS_T Struct Reference

A table containing PD redundant group information.

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

Data Fields

- [UINT32 id](#)
Redundant Id.
- [UINT32 state](#)
Redundant state.Leader or Follower.

4.39.1 Detailed Description

A table containing PD redundant group information.

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

- [trdp_types.h](#)

4.40 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 [nrxSafe](#)
Timeout cycles.
- UINT8 [reserved1](#)
Reserved for future use.
- UINT16 [lmiMax](#)
Latency monitoring cycles.

4.40.1 Detailed Description

Types to read out the XML configuration

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

- [tau_xml.h](#)

4.41 TRDP_SHORT_VERSION_T Struct Reference

Version information for communication buffers.

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

Data Fields

- UINT8 [ver](#)
Version - incremented for incompatible changes.
- UINT8 [rel](#)
Release - incremented for compatible changes

4.41.1 Detailed Description

Version information for communication buffers.

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

- [tau_tti_types.h](#)

4.42 TRDP_STATISTICS_REQUEST_T Struct Reference

TRDP statistics type definitions.

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

Data Fields

- UINT32 [comId](#)
ComId to request: 35...41.

4.42.1 Detailed Description

TRDP statistics type definitions.

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 usage Structure containing comId for MD statistics request (ComId 32).

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

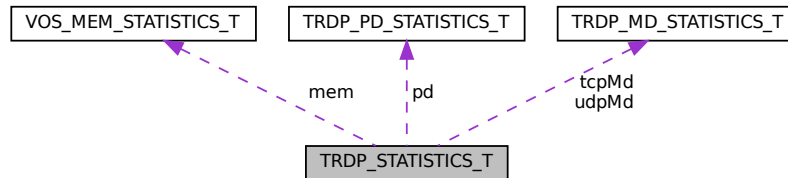
- [trdp_types.h](#)

4.43 TRDP_STATISTICS_T Struct Reference

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

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

Collaboration diagram for TRDP_STATISTICS_T:



Data Fields

- [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.

4.43.1 Detailed Description

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

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

- [trdp_types.h](#)

4.44 TRDP_SUBS_STATISTICS_T Struct Reference

Table containing particular PD subscription information.

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

Data Fields

- [UINT32 comId](#)
Subscribed ComId
- [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.
- [UINT32 numMissed](#)
number of packets skipped for this subscription

4.44.1 Detailed Description

Table containing particular PD subscription information.

4.44.2 Field Documentation

4.44.2.1 timeout

UINT32 TRDP_SUBS_STATISTICS_T::timeout

Time-out value in us.

0 = No time-out supervision

4.44.2.2 toBehav

UINT32 TRDP_SUBS_STATISTICS_T::toBehav

Behavior at time-out.

Set data to zero / keep last value

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

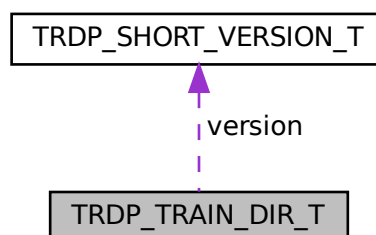
- [trdp_types.h](#)

4.45 TRDP_TRAIN_DIR_T Struct Reference

TCN train directory.

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

Collaboration diagram for TRDP_TRAIN_DIR_T:



Data Fields

- [TRDP_SHORT_VERSION_T version](#)
*TrainDirectory data structure version
parameter 'mainVersion' shall be set to 1.*
- [UINT8 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)*
- [UINT8 cstCnt](#)
number of consists in train; range: 1..63
- [TRDP_CONSIST_T cstList](#) [63u]
*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*
- [UINT32 trnTopoCnt](#)
computed as defined in 5.3.3.2.16 (seed value: etbTopoCnt)

4.45.1 Detailed Description

TCN train directory.

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

- [tau_tti_types.h](#)

4.46 TRDP_TRAIN_NET_DIR_ENTRY_T Struct Reference

Train network directory entry structure acc.

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

Data Fields

- [TRDP_UUID_T cstUUID](#)
unique consist identifier
- [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*

4.46.1 Detailed Description

Train network directory entry structure acc.

to IEC61375-2-5

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

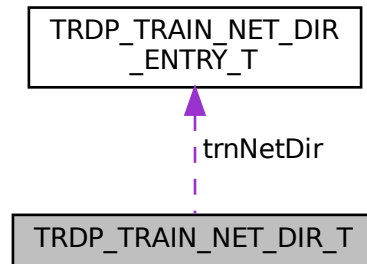
- [tau_tti_types.h](#)

4.47 TRDP_TRAIN_NET_DIR_T Struct Reference

Train network directory structure.

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

Collaboration diagram for TRDP_TRAIN_NET_DIR_T:



Data Fields

- UINT16 [reserved01](#)
reserved for future use (= 0)
- UINT16 [entryCnt](#)
number of entries in train network directory
- [TRDP_TRAIN_NET_DIR_ENTRY_T trnNetDir](#) [63u]
train network directory
- UINT32 [etbTopoCnt](#)
train network directory CRC

4.47.1 Detailed Description

Train network directory structure.

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

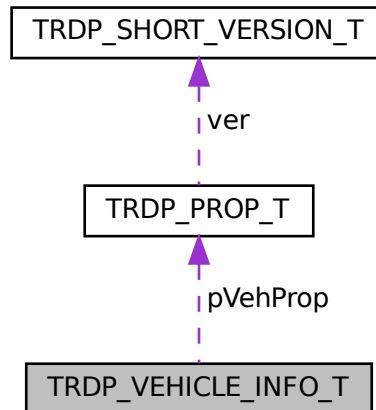
- [tau_tti_types.h](#)

4.48 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](#) * `pVehProp`
static vehicle properties #378

4.48.1 Detailed Description

vehicle information structure

4.48.2 Field Documentation

4.48.2.1 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](#)

4.49 TRDP_XML_DOC_HANDLE_T Struct Reference

Parsed XML document handle.

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

Data Fields

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

4.49.1 Detailed Description

Parsed XML document handle.

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

- [tau_xml.h](#)

4.50 VOS_MEM_STATISTICS_T Struct Reference

Structure containing all general memory statistics information.

```
#include <vos_mem.h>
```

Data Fields

- 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](#) [15u]
preallocated memory blocks
- UINT32 [usedBlockSize](#) [15u]
used memory blocks

4.50.1 Detailed Description

Structure containing all general memory statistics information.

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

- [vos_mem.h](#)

4.51 VOS_QUEUE_T Struct Reference

Opaque queue define

4.51.1 Detailed Description

Opaque queue define

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

- [vos_mem.c](#)

4.52 VOS_SOCKET_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!

4.52.1 Detailed Description

Common socket options

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

- [vos_sock.h](#)

4.53 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

4.53.1 Detailed Description

Version information.

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

- [vos_types.h](#)

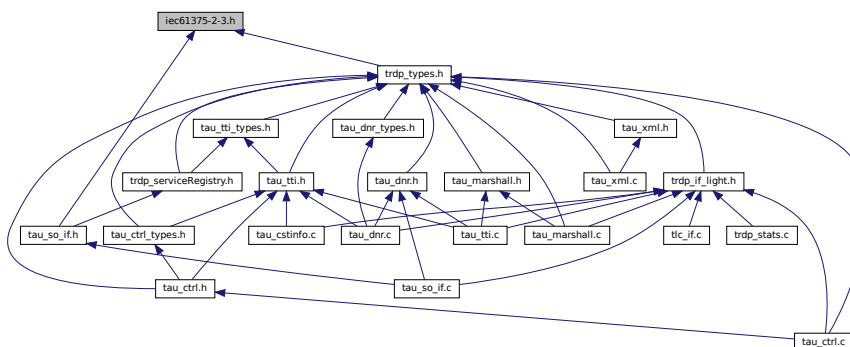
Chapter 5

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_PROTOCOL_VERSION_CHECK_MASK 0xFF00u`
Protocol version is defined in trdp_private.h.

- `#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_EXTRA_LABEL_LEN 100u`
long 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_PT` 0x5074u
- Message Types*
- #define `TRDP_MSG_PD` 0x5064u
- 'Pd' PD Data*
- #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` 3000000u
- [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 `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_REQ_COMID` 103u
MD reply
- #define `TTDB_TRN_DIR_REQ_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_REQ_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/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2015-2021. All rights reserved.

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_EXTRA_LABEL_LEN

```
#define TRDP_EXTRA_LABEL_LEN 100u
```

long label length incl.

terminating '0' #349

5.1.2.4 TRDP_MAX_FILE_NAME_LEN

```
#define TRDP_MAX_FILE_NAME_LEN 128u
```

path and file name length incl.

terminating '0'

5.1.2.5 TRDP_MAX_LABEL_LEN

```
#define TRDP_MAX_LABEL_LEN 16u
```

label length incl.

terminating '0'

5.1.2.6 TRDP_MAX_MD_DATA_SIZE

```
#define TRDP_MAX_MD_DATA_SIZE 65388u
```

MD packet properties

MD payload size

5.1.2.7 TRDP_MAX_URI_HOST_LEN

```
#define TRDP_MAX_URI_HOST_LEN (5u * TRDP\_MAX\_LABEL\_LEN)
```

URI host part incl.

terminating '0'

5.1.2.8 TRDP_MAX_URI_LEN

```
#define TRDP_MAX_URI_LEN (7u * TRDP\_MAX\_LABEL\_LEN)
```

URI length incl.

',' '@' and terminating '0'

5.1.2.9 TRDP_MAX_URI_USER_LEN

```
#define TRDP_MAX_URI_USER_LEN (2u * TRDP\_MAX\_LABEL\_LEN)
```

URI user part incl.

',' and terminating '0'

5.1.2.10 TRDP_MD_DEFAULT_REPLY_TIMEOUT

```
#define TRDP_MD_DEFAULT_REPLY_TIMEOUT 5000000u
```

Default MD communication parameters

[us] default reply timeout 5s

5.1.2.11 TRDP_MD_INFINITE_TIME

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

Definitions for time out behaviour accd.

table A.18

5.1.2.12 TRDP_MIN_PD_HEADER_SIZE

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

PD packet properties

PD header size with FCS

5.1.2.13 TRDP_MSG_PT

```
#define TRDP_MSG_PT 0x5074u
```

Message Types

'Pt' PD Data TSN

5.1.2.14 TRDP_PD_UDP_PORT

```
#define TRDP_PD_UDP_PORT 17224u
```

TRDP defines (from former trpd_proto.h)

IANA assigned process data UDP port

5.1.2.15 TRDP_PROCESS_DEFAULT_CYCLE_TIME

```
#define TRDP_PROCESS_DEFAULT_CYCLE_TIME 10000u
```

Default TRDP process options

[us] 10ms cycle time for TRDP process

5.1.2.16 TRDP_PROTOCOL_VERSION_CHECK_MASK

```
#define TRDP_PROTOCOL_VERSION_CHECK_MASK 0xFF00u
```

Protocol version is defined in trdp_private.h.

Version check, two digits are relevant

5.1.2.17 TRDP_USR_URI_SIZE

```
#define TRDP_USR_URI_SIZE 32u
```

max.

User URI size in MD header

5.1.2.18 TTDB_NET_DIR_REQ_COMID

```
#define TTDB_NET_DIR_REQ_COMID 106u
```

TTDB manager telegram MD: Get the NETWORK_TRAIN_DIRECTORY

MD request

5.1.2.19 TTDB_OP_DIR_INFO_COMID

```
#define TTDB_OP_DIR_INFO_COMID 101u
```

TTDB manager telegram MD: Push the OP_TRAIN_DIRECTORY

MD notification

5.1.2.20 TTDB_STAT_CST_REQ_COMID

```
#define TTDB_STAT_CST_REQ_COMID 104u
```

TTDB manager telegram MD: Get the static consist information

MD request

5.1.2.21 TTDB_TRN_DIR_REQ_COMID

```
#define TTDB_TRN_DIR_REQ_COMID 102u
```

TTDB manager telegram MD: Get the TRAIN_DIRECTORY

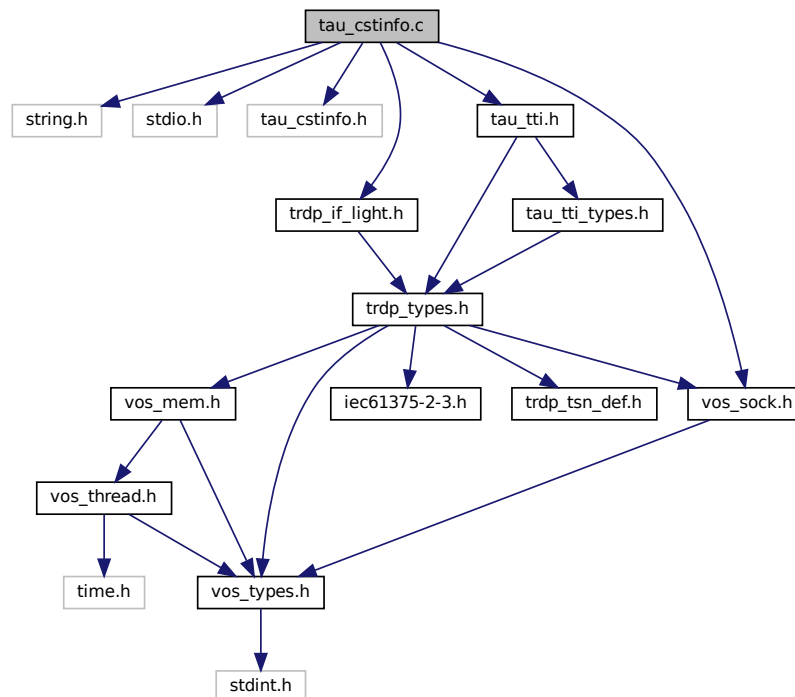
MD request

5.2 tau_cstinfo.c File Reference

Functions for consist information access.

```
#include <string.h>
#include <stdio.h>
#include "tau_cstinfo.h"
#include "trdp_if_light.h"
#include "tau_tti.h"
#include "vos_sock.h"
```

Include dependency graph for tau_cstinfo.c:



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.

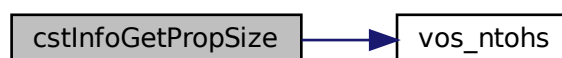
Parameters

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:

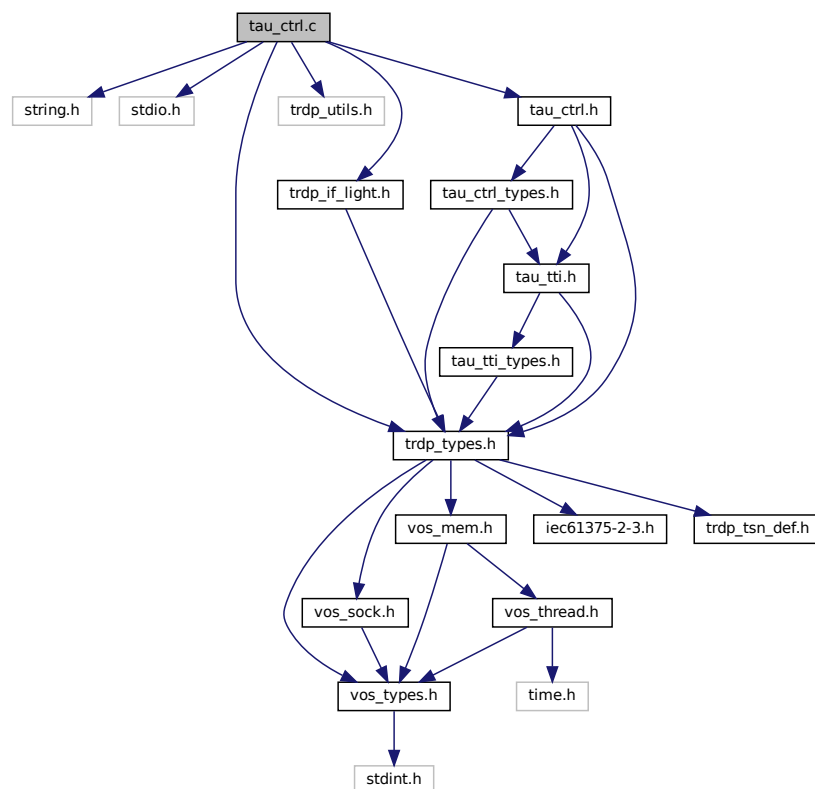


5.3 tau_ctrl.c File Reference

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](#) ecspIpAddr, UINT32 redId)
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](#) pCbFunction, TRDP_ECSP_CONF_REQUEST_T *pEcspConfRequest)

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

- EXT_DECL [TRDP_ERR_T](#) [tau_requestEcspConfirmReply](#) (TRDP_APP_SESSION_T appHandle, const void *pUserRef, [TRDP_MD_INFO_T](#) *pMsg, TRDP_ECSP_CONF_REPLY_T *pEcspConfReply)

Function to retrieve ECSP confirmation/correction reply.

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, 2013-2021. 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
--------------------	----------

Return values

<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 ecspIpAddr,
    UINT32 redId )
```

Function to init ECSP control interface.

Parameters

in	<i>appHandle</i>	Application handle
in	<i>ecspIpAddr</i>	ECSP address
in	<i>redId</i>	redId, if redundancy needed for ECSP control publisher

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 pfCbFunction,
    TRDP_ECSP_CONF_REQUEST_T * pEcspConfRequest )
```

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

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>pfCbFunction</i>	Optional 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_requestEcspConfirmReply()

```
EXT_DECL TRDP_ERR_T tau_requestEcspConfirmReply (
    TRDP_APP_SESSION_T appHandle,
    const void * pUserRef,
    TRDP_MD_INFO_T * pMsg,
    TRDP_ECSP_CONF_REPLY_T * pEcspConfReply )
```

Function to retrieve ECSP confirmation/correction reply.

Parameters

in	<i>appHandle</i>	Application Handle
in	<i>pUserRef</i>	user reference returned with reply
	<i>[in/out]</i>	pMsg Pointer to message info data
	<i>[in/out]</i>	pEcspConfReply Pointer to confirmation reply data

Return values

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

5.3.2.5 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
--------------------	----------

Return values

<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

5.3.2.6 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

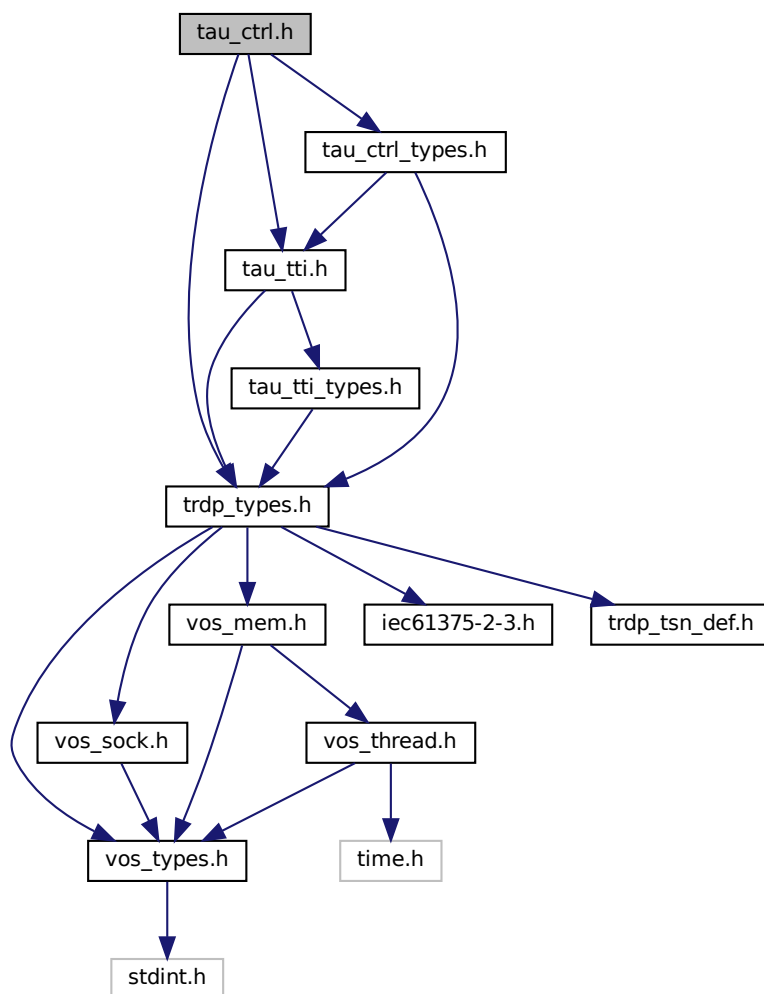
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_UNKNOWN_ERR</i>	undefined error

5.4 tau_ctrl.h File Reference

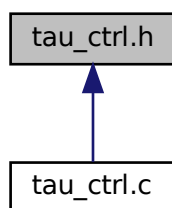
TRDP utility interface definitions.

```
#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

- [TRDP_ERR_T tau_initEcspCtrl](#) (TRDP_APP_SESSION_T appHandle, [TRDP_IP_ADDR_T](#) ecspIpAddr, UINT32 redId)
Function to init ECSP control interface.
- [TRDP_ERR_T tau_terminateEcspCtrl](#) (TRDP_APP_SESSION_T appHandle)
Function to close ECSP control interface.
- [TRDP_ERR_T tau_setEcspCtrl](#) (TRDP_APP_SESSION_T appHandle, TRDP_ECSP_CTRL_T *pEcspCtrl)
Function to set ECSP control information.
- [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.
- [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.
- [TRDP_ERR_T tau_requestEcspConfirmReply](#) (TRDP_APP_SESSION_T appHandle, const void *pUserRef, TRDP_MD_INFO_T *pMsg, TRDP_ECSP_CONF_REPLY_T *pEcspConfReply)
Function to retrieve ECSP confirmation/correction reply.

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()

```
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()

```

TRDP_ERR_T tau_initEcspCtrl (
    TRDP_APP_SESSION_T appHandle,
    TRDP_IP_ADDR_T ecspIpAddr,
    UINT32 redId )

```

Function to init ECSP control interface.

Parameters

in	<i>appHandle</i>	Application handle
in	<i>ecspIpAddr</i>	ECSP address
in	<i>redId</i>	redId, if redundancy needed for ECSP control publisher

Return values

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

5.4.2.3 tau_requestEcspConfirm()

```
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>	Optional 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

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>	Optional 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_requestEcspConfirmReply()

```
TRDP_ERR_T tau_requestEcspConfirmReply (
    TRDP_APP_SESSION_T appHandle,
    const void * pUserRef,
    TRDP_MD_INFO_T * pMsg,
    TRDP_ECSP_CONF_REPLY_T * pEcspConfReply )
```

Function to retrieve ECSP confirmation/correction reply.

Parameters

in	<i>appHandle</i>	Application Handle
in	<i>pUserRef</i>	user reference returned with reply
	[in/out]	pMsg Pointer to message info data
	[in/out]	pEcspConfReply Pointer to confirmation reply data

Return values

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

5.4.2.5 tau_setEcspCtrl()

```
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

5.4.2.6 tau_terminateEcspCtrl()

```
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

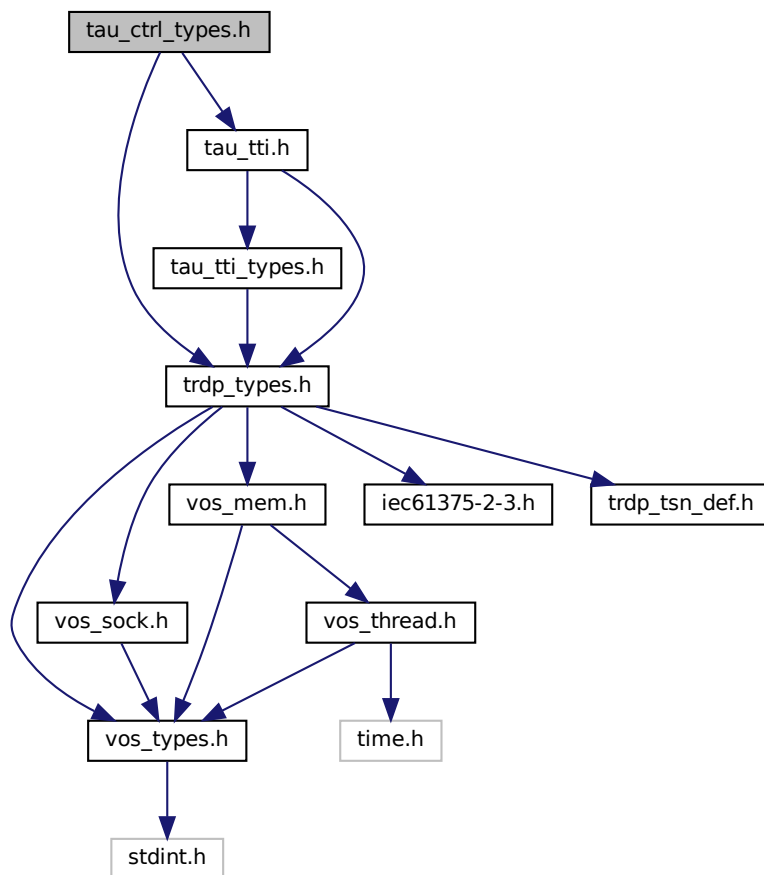
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_UNKNOWN_ERR</i>	undefined error

5.5 tau_ctrl_types.h File Reference

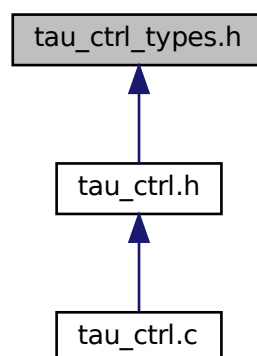
TRDP utility interface definitions.

```
#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 [TRDP_CONF_VEHICLE_T](#)
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.

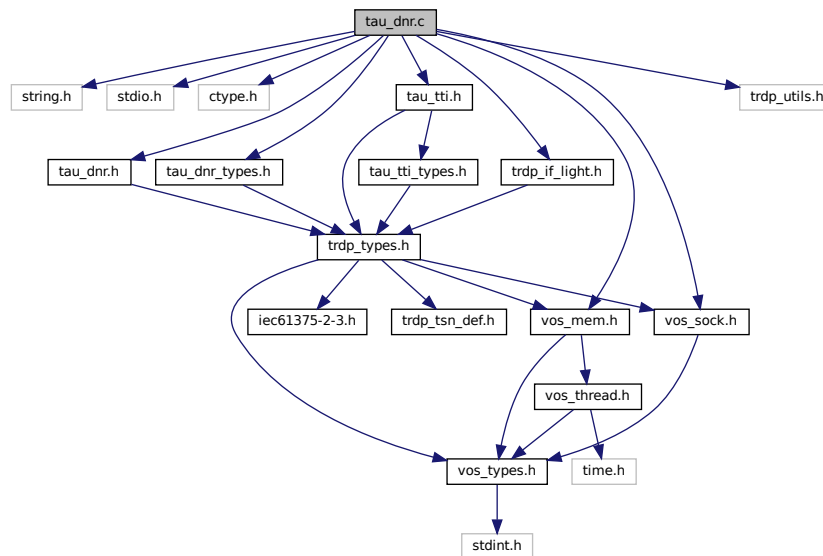
5.6 tau_dnr.c File Reference

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 [TAU_DNS_HEADER_T](#)
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

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 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, 2013-2021. 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)

< Timeout in seconds for DNS server reply, if hosts file was provided

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 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

< Timeout in seconds for DNS server reply, if no hosts file provided

< Timeout in seconds for DNS server reply, if hosts file was provided

< Timeout in seconds for DNS server reply, if hosts file was provided

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

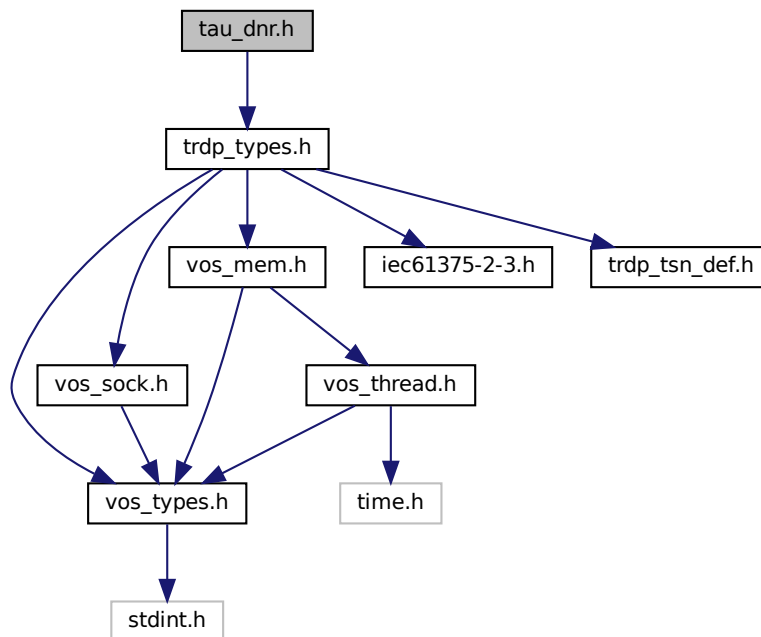
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_UNRESOLVED_ERR</i>	Could not resolve error
<i>TRDP_TOPO_ERR</i>	Cache/DB entry is invalid

5.7 tau_dnr.h File Reference

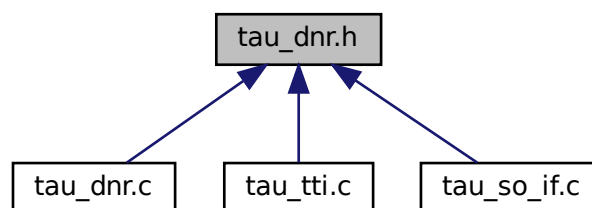
TRDP utility interface definitions.


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

Include dependency graph for tau_dnr.h:



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



Enumerations

- enum [TRDP_DNR_STATE_T](#)
DNR state.
- enum [TRDP_DNR_OPTS_T](#) {}
DNR options.

Functions

- `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.
- void `tau_delinitDnr` (TRDP_APP_SESSION_T appHandle)
Release any resources allocated by DNR.
- `TRDP_DNR_STATE_T tau_DNRstatus` (TRDP_APP_SESSION_T appHandle)
Function to get the status of DNR.
- `TRDP_IP_ADDR_T tau_getOwnAddr` (TRDP_APP_SESSION_T appHandle)
Function to get the own IP address.
- `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.
- `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_T

```
enum TRDP_DNR_OPTS_T
```

DNR options.

Enumerator

TRDP_DNR_OWN_THREAD	For single threaded systems only! Internally call tlc_process()
---------------------	---

5.7.3 Function Documentation

5.7.3.1 tau_addr2Uri()

```
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()

```
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()

```
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)

< Timeout in seconds for DNS server reply, if hosts file was provided

5.7.3.4 tau_getOwnAddr()

```
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()

```
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.
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 <i>tlc_process</i> 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

< Timeout in seconds for DNS server reply, if no hosts file provided

< Timeout in seconds for DNS server reply, if hosts file was provided

< Timeout in seconds for DNS server reply, if hosts file was provided

5.7.3.6 tau_uri2Addr()

```
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

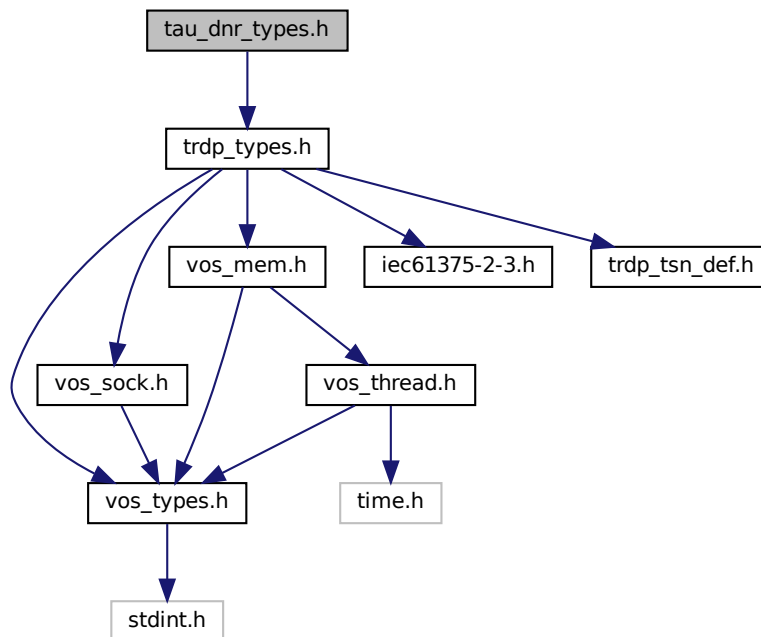
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_UNRESOLVED_ERR</i>	Could not resolve error
<i>TRDP_TOPO_ERR</i>	Cache/DB entry is invalid

5.8 tau_dnr_types.h File Reference

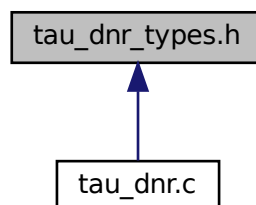
TRDP utility interface definitions.

```
#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_T](#)
TCN-DNS simplified header structures.
- struct [TRDP_DNS_REQUEST_T](#)
TCN-DNS Request telegram TCN_DNS_REQ_DS.
- struct [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

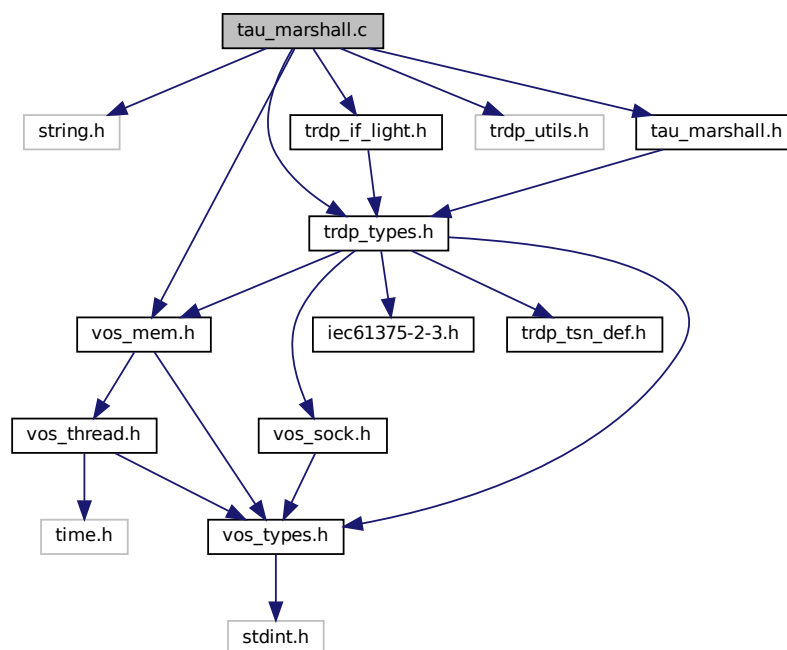
This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright NewTec GmbH, 2017. All rights reserved.

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 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

5.9.2.2 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

Parameters

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

5.9.2.3 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 to 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
<i>TRDP_MARSHALLING_ERR</i>	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

<i>TRDP_INIT_ERR</i>	marshalling not initialised
----------------------	-----------------------------

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_COMID_ERR</i>	comid not existing
<i>TRDP_MARSHALLING_ERR</i>	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

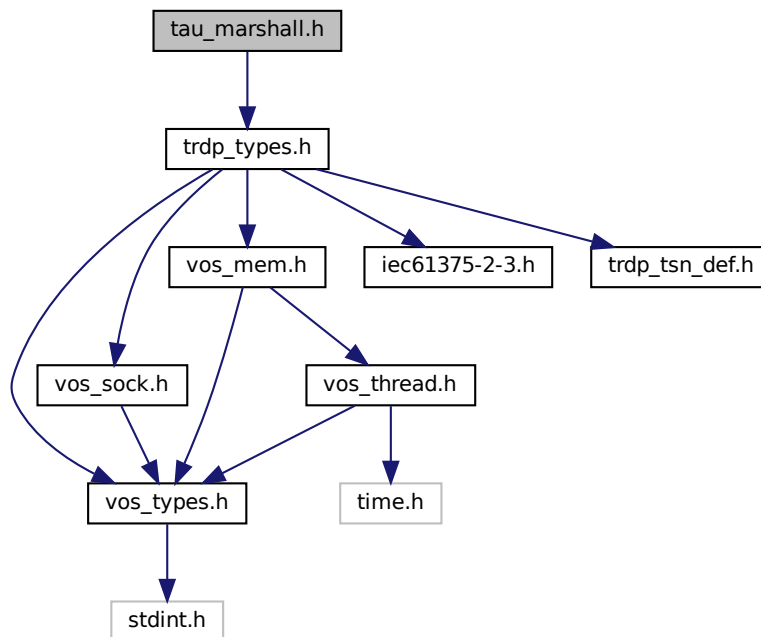
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_STATE_ERR</i>	Too deep recursion
<i>TRDP_COMID_ERR</i>	comid not existing
<i>TRDP_MARSHALLING_ERR</i>	dataset/source size mismatch

5.10 tau_marshall.h File Reference

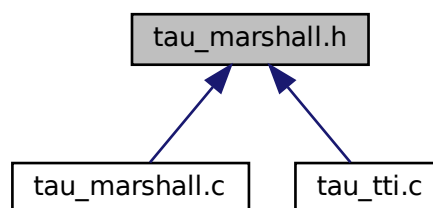
TRDP utility interface definitions.

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

Include dependency graph for tau_marshall.h:



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



Functions

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

marshall function.

- [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.

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

unmarshall function.

- [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.

- [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.

- [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

- marshall/unmarshalling

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()

```
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 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

5.10.2.2 tau_calcDatasetSizeByComId()

```

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()

```

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()

```

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()

```

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()

```
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()

```
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

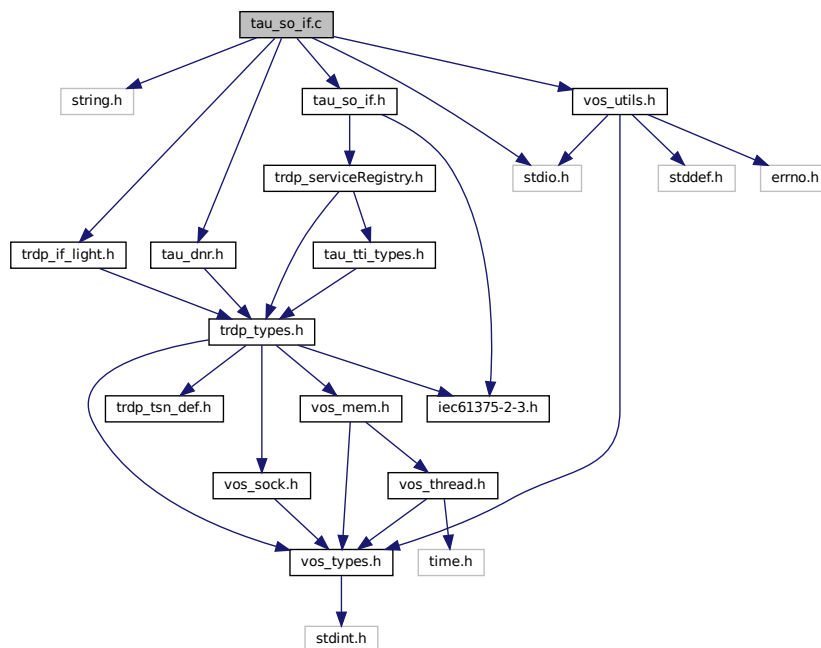
Return values

<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_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_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_↵_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

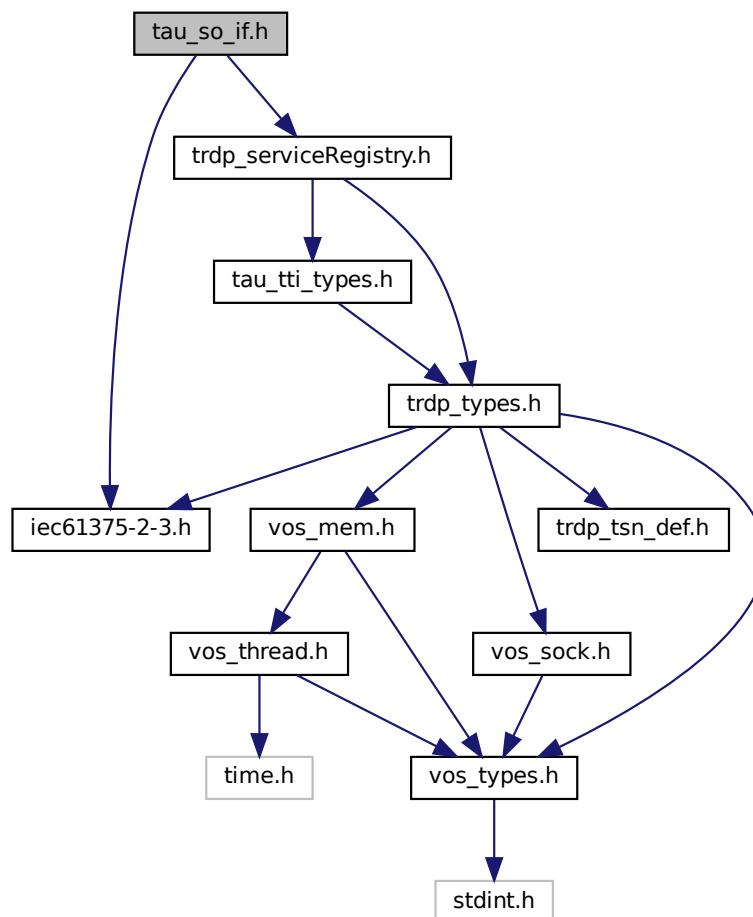
Return values

<i>TRDP_TIMEOUT_ERR</i>	Reply timed out
<i>TRDP_SEMA_ERR</i>	Semaphore could not be aquired

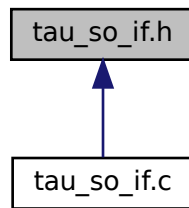
5.12 tau_so_if.h File Reference

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

- [TRDP_ERR_T tau_addService](#) (TRDP_APP_SESSION_T appHandle, [SRM_SERVICE_INFO_T](#) *p↔ ServiceToAdd, BOOL8 waitForCompletion)
Function to add to the service registry of the consist-local SRM.
- [TRDP_ERR_T tau_delService](#) (TRDP_APP_SESSION_T appHandle, [SRM_SERVICE_INFO_T](#) *p↔ ServiceToAdd, BOOL8 waitForCompletion)
Remove the defined service from the service registry of the consist-local SRM.
- [TRDP_ERR_T tau_updService](#) (TRDP_APP_SESSION_T appHandle, [SRM_SERVICE_INFO_T](#) *p↔ ServiceToAdd, BOOL8 waitForCompletion)
Register an update a service.
- [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.
- 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()

```
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

5.12.2.2 tau_delService()

```
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()

```
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()

```
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

5.12.2.5 tau_updService()

```
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

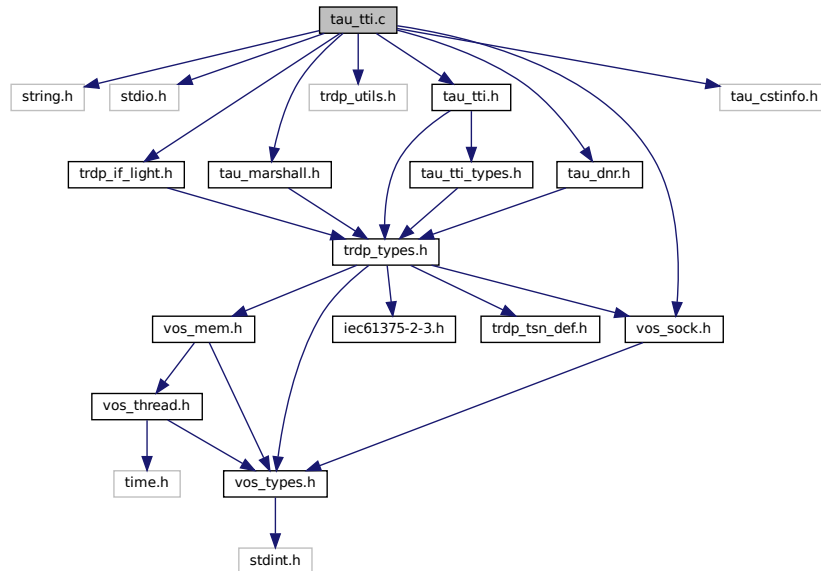
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error
<i>TRDP_NODATA_ERR</i>	Data currently not available, try again later
<i>TRDP_TIMEOUT_ERR</i>	Reply timed out
<i>TRDP_SEMA_ERR</i>	Semaphore could not be aquired

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.h"
Include dependency graph for tau_tti.c:
```



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_getOpTrnDirectory](#) (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.
- 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.
- EXT_DECL [TRDP_ERR_T tau_getTrnDirectory](#) (TRDP_APP_SESSION_T appHandle, [TRDP_TRAIN_DIR_T](#) *pTrnDir)
Function to retrieve the train directory.
- EXT_DECL [TRDP_ERR_T tau_copyCstInfo](#) ([TRDP_CONSIST_INFO_T](#) **ppDstCstInfo, [TRDP_CONSIST_INFO_T](#) *pSrcCstInfo)
Function to allocate memory and to copy the consist info into.
- EXT_DECL [TRDP_ERR_T tau_getStaticCstInfo](#) (TRDP_APP_SESSION_T appHandle, [TRDP_CONSIST_INFO_T](#) **ppCstInfo, [TRDP_UUID_T](#) const cstUUID)
Function to alloc memory and to retrieve the consist information of a train's consist.
- 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.
- 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](#) **ppFctInfo, UINT16 *pFctCnt, const TRDP_LABEL_T pCstLabel)
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](#) **ppVehInfo, 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](#) **ppCstInfo, const TRDP_LABEL_T pCstLabel)
Function to alloc memory and 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.

5.13.1 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_↵ NODATA_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, 2016-2020. All rights reserved.

5.13.2 Function Documentation

5.13.2.1 tau_copyCstInfo()

```
EXT_DECL TRDP_ERR_T tau_copyCstInfo (
    TRDP_CONSIST_INFO_T ** ppDstCstInfo,
    TRDP_CONSIST_INFO_T * pSrcCstInfo )
```

Function to allocate memory and to copy the consist info into.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>pCstInfo</i>	Pointer to a consist info structure to be returned.

Return values

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

5.13.2.2 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>	
-------------	--

5.13.2.3 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

5.13.2.4 tau_getCstFctInfo()

```
EXT_DECL TRDP_ERR_T tau_getCstFctInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FUNCTION_INFO_T ** ppFctInfo,
    UINT16 * pFctCnt,
    const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the function information of the consist.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>ppFctInfo</i>	Pointer to a pointer to function info list to be returned. Memory needs to be freed by application.
out	<i>pFctCnt</i>	Pointer to number of functions returned in provided buffer.
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

5.13.2.5 tau_getCstInfo()

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

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

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>ppCstInfo</i>	Pointer to a pointer to consist info structure to be returned. The memory to copy the consist info will be allocated and has to be freed using vos_memFree() .
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_UNRESOLVED_ERR</i>	address could not be resolved

5.13.2.6 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

5.13.2.7 tau_getOpTrnDirectory()

```
EXT_DECL TRDP_ERR_T tau_getOpTrnDirectory (
    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

5.13.2.8 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

5.13.2.9 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

5.13.2.10 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
-------------------	---

5.13.2.11 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
--------------------	---

5.13.2.12 tau_getStaticCstInfo()

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

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

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>ppCstInfo</i>	Pointer to a pointer to consist info structure to be returned. The memory to copy the consist info will be allocated and hast to be freed using vos_memFree() .
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

5.13.2.13 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
<i>TRDP_NODATA_ERR</i>	Try again

5.13.2.14 tau_getTrnDirectory()

```
EXT_DECL TRDP_ERR_T tau_getTrnDirectory (
    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

5.13.2.15 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

5.13.2.16 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

5.13.2.17 tau_getVehInfo()

```

EXT_DECL TRDP_ERR_T tau_getVehInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_VEHICLE_INFO_T ** ppVehInfo,
    const TRDP_LABEL_T pVehLabel,
    const TRDP_LABEL_T pCstLabel )

```

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

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

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>ppVehInfo</i>	Pointer to the vehicle info to be returned. Memory has to be freed by vos_memAlloc() Call vos_memFree(pVehInfo->pVehProp) to release the property memory.
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

5.13.2.18 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

5.13.2.19 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.

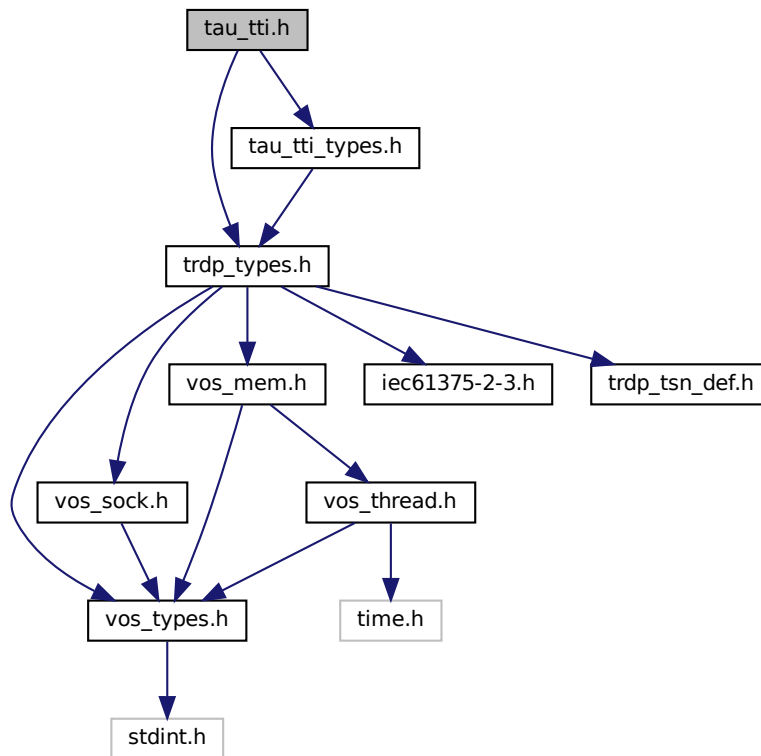
Return values

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

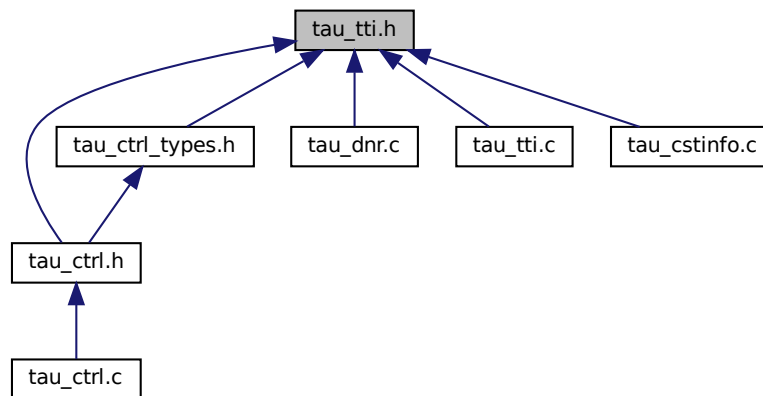
5.14 tau_tti.h File Reference

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

- [TRDP_ERR_T tau_initTTIaccess](#) (TRDP_APP_SESSION_T appHandle, [VOS_SEMA_T](#) userAction, [TRDP_IP_ADDR_T](#) ecsplpAddr, CHAR8 *hostsFileName)
Function to init TTI access.
- void [tau_delInitTTI](#) (TRDP_APP_SESSION_T appHandle)
Function to terminate TTI access.
- [TRDP_ERR_T tau_getOpTrnDirectory](#) (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.
- [TRDP_ERR_T tau_getOpTrnDirectoryStatusInfo](#) (TRDP_APP_SESSION_T appHandle, [TRDP_OP_TRAIN_DIR_STATUS_INF](#) *pOpTrnDirStatusInfo)
Function to retrieve the operational train directory state info.
- [TRDP_ERR_T tau_getTrnDirectory](#) (TRDP_APP_SESSION_T appHandle, [TRDP_TRAIN_DIR_T](#) *pTrnDir)
Function to retrieve the train directory.
- [TRDP_ERR_T tau_getStaticCstInfo](#) (TRDP_APP_SESSION_T appHandle, [TRDP_CONSIST_INFO_T](#) **ppCstInfo, [TRDP_UUID_T](#) const cstUUID)
Function to alloc memory and to retrieve the consist information of a train's consist.
- [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.
- [TRDP_ERR_T tau_getTrnCstCnt](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pTrnCstCnt)
Function to retrieve the total number of consists in the train.
- [TRDP_ERR_T tau_getTrnVehCnt](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pTrnVehCnt)
Function to retrieve the total number of vehicles in the train.
- [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.
- [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.

- [TRDP_ERR_T tau_getCstFctInfo](#) (TRDP_APP_SESSION_T appHandle, [TRDP_FUNCTION_INFO_T](#) **ppFctInfo, UINT16 *pFctCnt, const TRDP_LABEL_T pCstLabel)
Function to retrieve the function information of the consist.
- [TRDP_ERR_T tau_getVehInfo](#) (TRDP_APP_SESSION_T appHandle, [TRDP_VEHICLE_INFO_T](#) **ppVehInfo, const TRDP_LABEL_T pVehLabel, const TRDP_LABEL_T pCstLabel)
Function to allocate memory and to retrieve the vehicle information of a consist's vehicle.
- [TRDP_ERR_T tau_getCstInfo](#) (TRDP_APP_SESSION_T appHandle, [TRDP_CONSIST_INFO_T](#) **ppCstInfo, const TRDP_LABEL_T pCstLabel)
Function to alloc memory and to retrieve the consist information of a train's consist.
- [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.
- [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 ?.
- [UINT8 tau_getOwnOpCstNo](#) (TRDP_APP_SESSION_T appHandle)
Get own operational consist number.
- [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()

```
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()

```
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()

```
TRDP_ERR_T tau_getCstFctInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_FUNCTION_INFO_T ** ppFctInfo,
    UINT16 * pFctCnt,
    const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the function information of the consist.

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>**ppFctInfo</i>	Pointer to pointer to function info list to be returned. Memory needs to be freed by application.
out	<i>*pFctCnt</i>	Pointer to number of functions returned in provided buffer.
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>ppFctInfo</i>	Pointer to a pointer to function info list to be returned. Memory needs to be freed by application.
out	<i>pFctCnt</i>	Pointer to number of functions returned in provided buffer.
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

5.14.2.4 tau_getCstInfo()

```

TRDP_ERR_T tau_getCstInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_CONSIST_INFO_T ** ppCstInfo,
    const TRDP_LABEL_T pCstLabel )

```

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

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>ppCstInfo</i>	Pointer to a pointer to consist info structure to be returned. The memory to copy the consist info will be allocated and has to be freed using vos_memFree() .
in	<i>pCstLabel</i>	Label 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

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>ppCstInfo</i>	Pointer to a pointer to consist info structure to be returned. The memory to copy the consist info will be allocated and has to be freed using vos_memFree() .
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_UNRESOLVED_ERR</i>	address could not be resolved

5.14.2.5 tau_getCstVehCnt()

```
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_getOpTrnDirectory()

```
TRDP_ERR_T tau_getOpTrnDirectory (
    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()

```
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()

```
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()

```
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
-------------------	---

5.14.2.10 tau_getOwnTrnCstNo()

```
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
--------------------	---

5.14.2.11 tau_getStaticCstInfo()

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

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

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>ppCstInfo</i>	Pointer to a pointer to consist info structure to be returned. The memory to copy the consist info will be allocated and has to be freed using vos_memFree() .
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_getTrnCstCnt()

```
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

5.14.2.13 tau_getTrnDirectory()

```
TRDP_ERR_T tau_getTrnDirectory (
    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

5.14.2.14 tau_getTrnVehCnt()

```
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()

```
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()

```

TRDP_ERR_T tau_getVehInfo (
    TRDP_APP_SESSION_T appHandle,
    TRDP_VEHICLE_INFO_T ** ppVehInfo,
    const TRDP_LABEL_T pVehLabel,
    const TRDP_LABEL_T pCstLabel )

```

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

Parameters

in	<i>appHandle</i>	Handle returned by tlc_openSession() .
out	<i>ppVehInfo</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. Memory to be freed by the caller
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

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

Parameters

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

Parameters

out	<i>ppVehInfo</i>	Pointer to the vehicle info to be returned. Memory has to be freed by vos_memAlloc() Call vos_memFree(pVehInfo->pVehProp) to release the property memory.
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()

```

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
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()

```

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

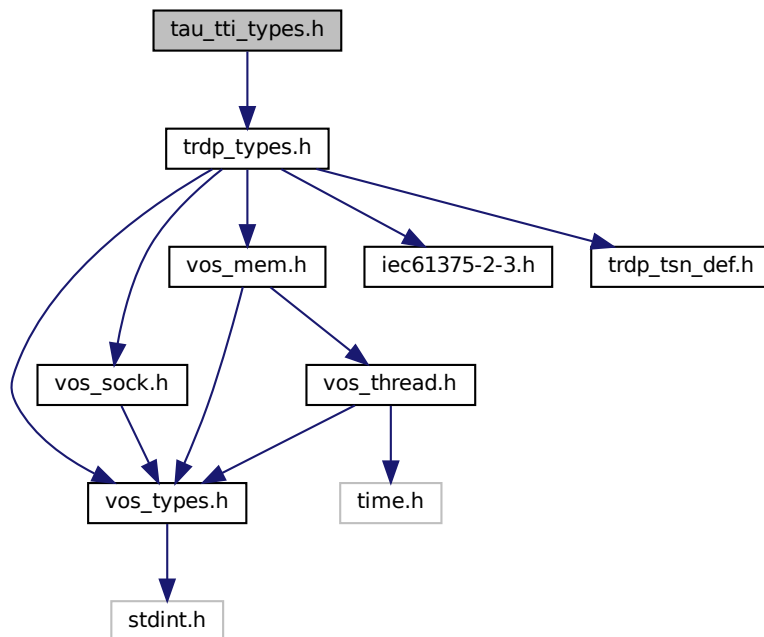
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

5.15 tau_tti_types.h File Reference

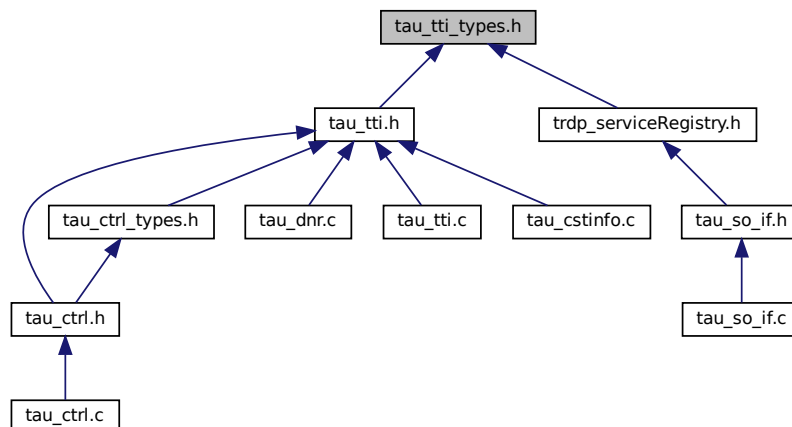
TRDP utility interface definitions.

```
#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 [TRDP_SHORT_VERSION_T](#)
Version information for communication buffers.

- 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 [TRDP_ETB_CTRL_VDP_T](#)
TCN consist structure.
- struct [TRDP_CSTINFOCTRL_T](#)
CSTINFO Control telegram.
- struct [TRDP_TRAIN_DIR_T](#)
TCN train directory.
- struct [TRDP_OP_VEHICLE_T](#)
Operational vehicle structure.
- struct [TRDP_OP_CONSIST_T](#)
Operational consist structure.
- struct [TRDP_OP_TRAIN_DIR_STATE_T](#)
Operational train directory state.
- struct [TRDP_OP_TRAIN_DIR_T](#)
Operational train structure.
- struct [TRDP_OP_TRAIN_DIR_STATUS_INFO_T](#)
Operational Train directory status info structure.
- struct [TRDP_TRAIN_NET_DIR_ENTRY_T](#)
Train network directory entry structure acc.
- struct [TRDP_TRAIN_NET_DIR_T](#)
Train network directory structure.
- struct [TRDP_READ_COMPLETE_REPLY_T](#)
Complete TTDB structure.

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
- #define [TRDP_MAX_PROP_LEN](#) 32768u
maximum length of property information #378

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

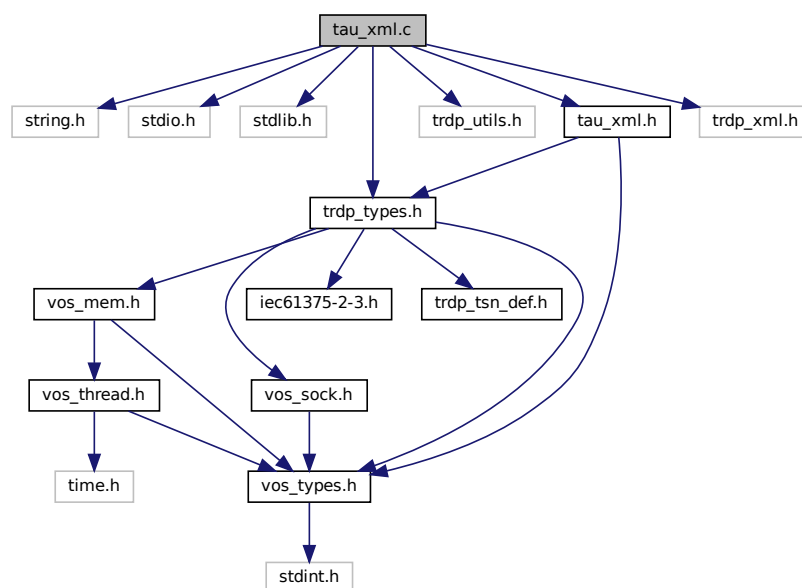
This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2014. All rights reserved.

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.
- #define [TRDP_SDTV4_DEFAULT_UDV_SUB](#) 0u
Default SDTv4 User data subversion
- #define [TRDP_SDTV4_DEFAULT_PROTO_VAR](#) 2u
SDTv4 protocol variant
- #define [TRDP_SDTV4_DEFAULT_SAFE_FUNC_ID](#) 0u
Default SDTv4 Safety Function Identifier
- #define [TRDP_SDTV4_DEFAULT_SAFE_FUNC_VERS](#) 0u
Default SDTv4 Safety Function Version
- #define [TRDP_SDTV4_DEFAULT_SAFE_CHAN_ID](#) 0u
Default SDTv4 Safety Channel Identifier
- #define [TRDP_SDTV4_DEFAULT_SAFE_CHAN_VERS](#) 0u
Default SDTv4 Safety Channel Version

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 *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 .
- 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](#) **ppIfConfig)
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](#) **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.
- 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.
- 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, 2016-2020. 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

Parameters

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

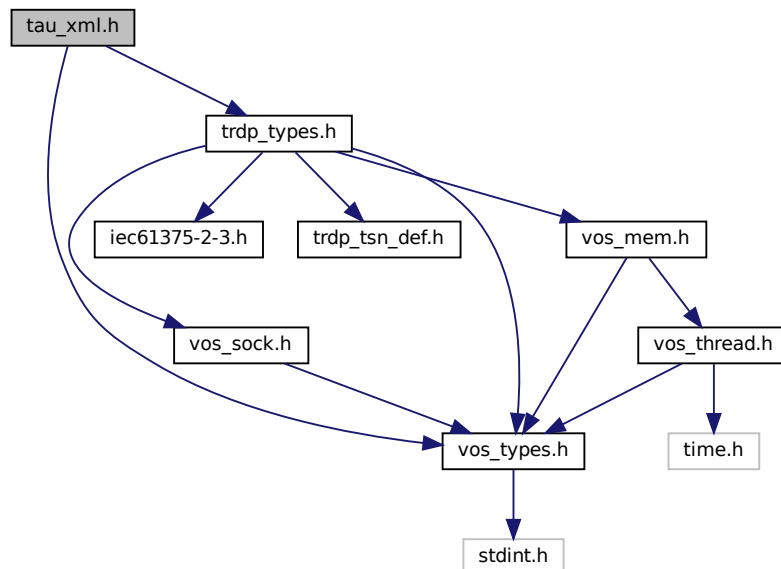
5.17 tau_xml.h File Reference

TRDP utility interface definitions.

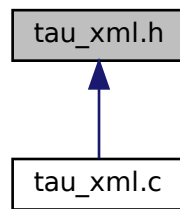
```
#include "vos_types.h"
```

```
#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

- [TRDP_ERR_T tau_prepareXmlDoc](#) (const [CHAR8](#) *pFileName, [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd)
Load XML file into DOM tree, prepare XPath context.
- [TRDP_ERR_T tau_prepareXmlMem](#) (char *pBuffer, size_t bufSize, [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd)
Open XML stream, prepare XPath context.
- void [tau_freeXmlDoc](#) ([TRDP_XML_DOC_HANDLE_T](#) *pDocHnd)
Free all the memory allocated by tau_prepareXmlDoc.
- [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.
- [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 .
- [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](#) **ppDataset)
Function to read the DataSet configuration out of the XML configuration file.
- 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.
- void [tau_freeTelegrams](#) ([UINT32](#) numExchgPar, [TRDP_EXCHG_PAR_T](#) *pExchgPar)
Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.
- [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.
- [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.
- [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.
- [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()

```
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()

```
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()

```
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()

```
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()

```
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()

```
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()

```
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
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()

```
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

5.17.4.9 tau_readXmlMappedDeviceConfig()

```
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()`

```
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 <code>tau_prepareXmlDoc</code>
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()`

```
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 <code>tau_prepareXmlDoc</code>
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()

```
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

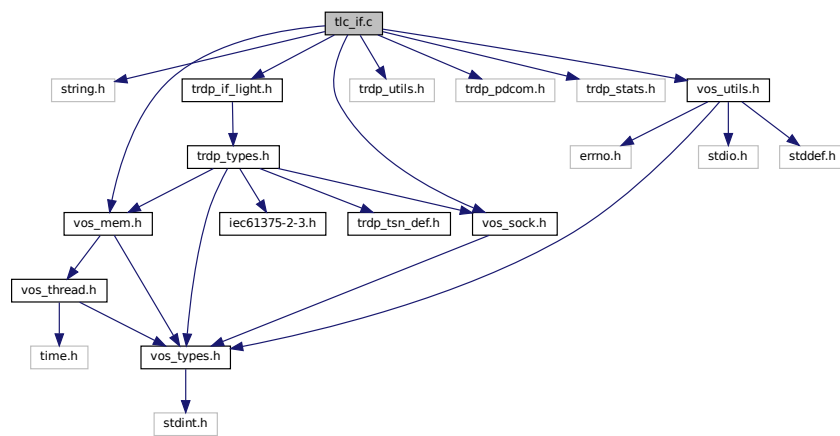
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

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_TABLE_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, [TRDP SOCK_T](#) *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 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 etbTopoCnt)
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_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
<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 tlc_openSession
----	------------------	--

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,
    TRDP SOCK_T * 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

5.18.2.5 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 <code>tlc_openSession</code>
----	------------------	---

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()

```
EXT_DECL 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.

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

< all available sockets for PD

< 120 seconds (120 tries each second to bind to an IP address)

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 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

5.18.2.13 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 <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

< all available sockets for PD

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 <code>tlc_openSession</code>
in	<i>etbTopoCnt</i>	New <code>etbTopoCnt</code> 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 tlc_openSession
----	------------------	--

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

5.18.2.18 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

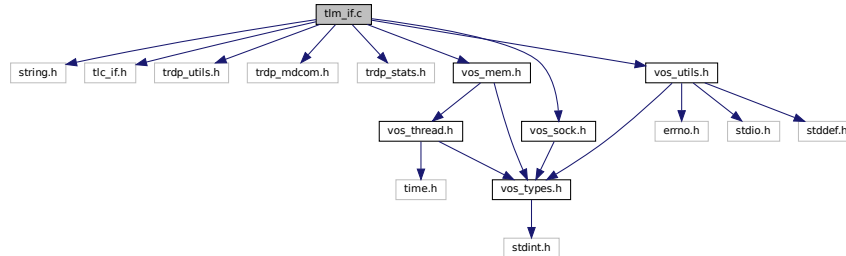
<i>&sSession</i>	
----------------------	--

5.19 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, [TRDP SOCK_T](#) *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.
- 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_COM_PARAM_T](#) *pSendParam, const UINT8 *pData, UINT32 dataSize, const TRDP_URI_USER_T srcURI, 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_COM_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.
- 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_delListener](#) (TRDP_APP_SESSION_T appHandle, TRDP_LIS_T listenHandle)
Remove Listener.
- 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.
- EXT_DECL [TRDP_ERR_T tlm_reply](#) (TRDP_APP_SESSION_T appHandle, const [TRDP_UUID_T](#) *pSessionId, UINT32 comId, UINT32 userStatus, const [TRDP_COM_PARAM_T](#) *pSendParam, const UINT8 *pData, UINT32 dataSize, const TRDP_URI_USER_T srcURI)

Send a MD reply message.

- EXT_DECL `TRDP_ERR_T tlm_replyQuery` (TRDP_APP_SESSION_T appHandle, const TRDP_UUID_T *pSessionId, UINT32 comId, UINT32 userStatus, UINT32 confirmTimeout, const TRDP_COM_PARAM_T *pSendParam, const UINT8 *pData, UINT32 dataSize, const TRDP_URI_USER_T srcURI)

Send a MD reply query message.

- EXT_DECL `TRDP_ERR_T tlm_confirm` (TRDP_APP_SESSION_T appHandle, const TRDP_UUID_T *pSessionId, UINT16 userStatus, const TRDP_COM_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.

5.19.1 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, 2013-2021. All rights reserved.

5.19.2 Function Documentation

5.19.2.1 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>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

5.19.2.2 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 tlc_openSession
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 comId shall be observed
in	<i>comId</i>	comId 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

Return values

<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

< if set, do filter comId (addListener)

5.19.2.3 tlm_confirm()

```
EXT_DECL TRDP_ERR_T tlm_confirm (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId,
    UINT16 userStatus,
    const TRDP_COM_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 tlc_openSession
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

5.19.2.4 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 tlc_openSession
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

5.19.2.5 tlm_getInterval()

```
EXT_DECL TRDP_ERR_T tlm_getInterval (
    TRDP_APP_SESSION_T appHandle,
    TRDP_TIME_T * pInterval,
    TRDP_FDS_T * pFileDesc,
    TRDP SOCK_T * 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 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

< cycle time [us] = delay for outgoing MD

5.19.2.6 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_COM_PARAM_T * pSendParam,
    const UINT8 * pData,
```

```

    UINT32 dataSize,
    const TRDP_URI_USER_T srcURI,
    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
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>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
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.19.2.7 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 <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

5.19.2.8 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 <i>tlc_openSession</i>
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

5.19.2.9 tlm_reply()

```
EXT_DECL TRDP_ERR_T tlm_reply (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId,
    UINT32 comId,
    UINT32 userStatus,
    const TRDP_COM_PARAM_T * pSendParam,
```

```

const UINT8 * pData,
UINT32 dataSize,
const TRDP_URI_USER_T srcURI )

```

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
in	<i>srcURI</i>	only functional group of source 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_NO_SESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.19.2.10 tlm_replyQuery()

```

EXT_DECL TRDP_ERR_T tlm_replyQuery (
    TRDP_APP_SESSION_T appHandle,
    const TRDP_UUID_T * pSessionId,
    UINT32 comId,
    UINT32 userStatus,
    UINT32 confirmTimeout,
    const TRDP_COM_PARAM_T * pSendParam,
    const UINT8 * pData,
    UINT32 dataSize,
    const TRDP_URI_USER_T srcURI )

```

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

Parameters

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
in	<i>srcURI</i>	only functional group of source 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_NO_SESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.19.2.11 tlm_request()

```
EXT_DECL TRDP_ERR_T tlm_request (
    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 numReplies,
    UINT32 replyTimeout,
    const TRDP_COM_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.

Send a MD request message

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
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

Parameters

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>srcURI</i>	only functional group of source URI
in	<i>destURI</i>	only functional group of destination URI

Return values

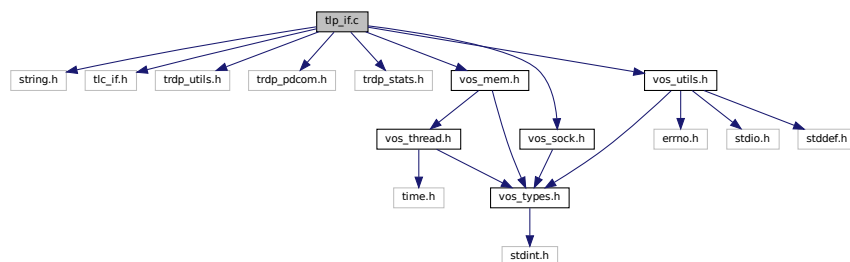
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.20 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, [TRDP SOCK_T](#) *pNoDesc)
Get the lowest time interval for PDs.
- 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_processSend](#) (TRDP_APP_SESSION_T appHandle)
Work loop of the TRDP handler.
- 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 ComIds.
- 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 servId, 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 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_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 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, 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.

5.20.1 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, 2013-2021. All rights reserved.

5.20.2 Function Documentation

5.20.2.1 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 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

Return values

<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

5.20.2.2 tlp_getInterval()

```
EXT_DECL TRDP_ERR_T tlp_getInterval (
    TRDP_APP_SESSION_T appHandle,
    TRDP_TIME_T * pInterval,
    TRDP_FDS_T * pFileDesc,
    TRDP SOCK_T * 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

5.20.2.3 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

< if set, packet should not be sent (redundant)

5.20.2.4 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 <i>tlc_openSession</i>
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

5.20.2.5 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 <i>tlc_openSession</i>
----	------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.20.2.6 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 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 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 tlc_openSession
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>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 (>= 10ms) in usec
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>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 <= TRDP_MAX_PD_DATA_SIZE

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

< if set, inform the user

< if set, PD will be sent on trdp_put() only

< if set, packet should not be sent (redundant)

< if set, PD will be sent on trdp_put() only

5.20.2.7 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

5.20.2.8 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
<i>TRDP_MEM_ERR</i>	out of memory

< if set, inform the user

< if set, inform the user

5.20.2.9 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 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
<i>TRDP_MEM_ERR</i>	could not insert (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.20.2.10 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 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
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

Parameters

in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
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

< if set, the request needs to be sent

< if set, inform the user

5.20.2.11 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

5.20.2.12 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 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

< if set, packet should not be sent (redundant)

< if set, packet should not be sent (redundant)

5.20.2.13 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,
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>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

< if set, inform the user

5.20.2.14 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

5.20.2.15 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 tlc_openSession
in	<i>subHandle</i>	the handle for this subscription

Return values

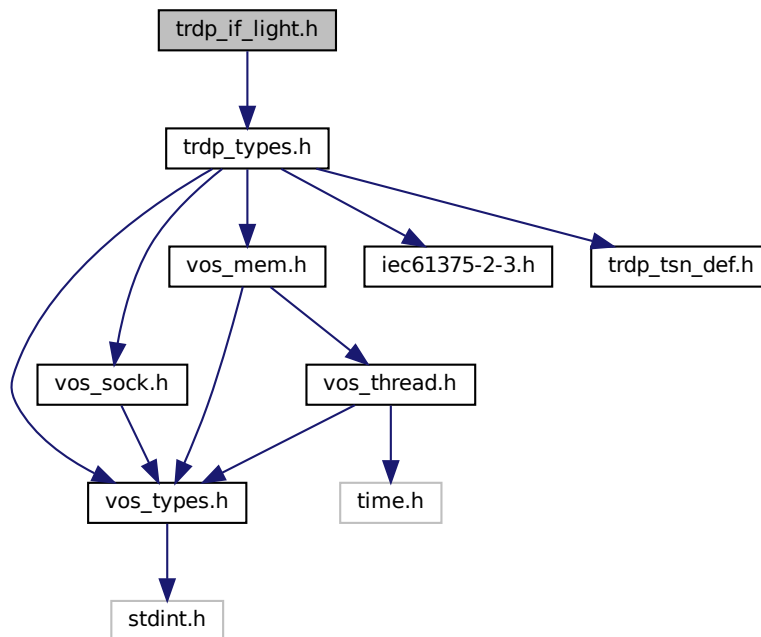
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_NOSUB_ERR</i>	not subscribed
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21 trdp_if_light.h File Reference

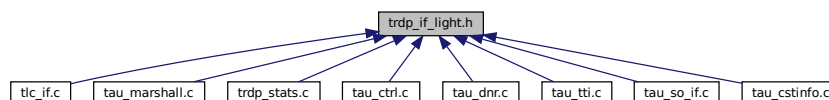
TRDP Light interface functions (API)

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

Include dependency graph for trdp_if_light.h:



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



Functions

- [TRDP_ERR_T tlc_init](#) (const [TRDP_PRINT_DBG_T](#) pPrintDebugString, void *pRefCon, const [TRDP_MEM_CONFIG_T](#) *pMemConfig)
Initialize the TRDP stack.
- [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.
- [TRDP_ERR_T tlc_reinitSession](#) ([TRDP_APP_SESSION_T](#) appHandle)
Re-Initialize.
- [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.*

 - [TRDP_ERR_T tlc_updateSession](#) (TRDP_APP_SESSION_T appHandle)

Update a session.

 - [TRDP_ERR_T tlc_presetIndexSession](#) (TRDP_APP_SESSION_T appHandle, [TRDP_IDX_TABLE_T](#) *pIndexTableSizes)

Preset the index table sizes of a session.

 - [TRDP_ERR_T tlc_closeSession](#) (TRDP_APP_SESSION_T appHandle)

Close a session.

 - [TRDP_ERR_T tlc_terminate](#) (void)

Un-Initialize.

 - [TRDP_ERR_T tlc_setETBTopoCount](#) (TRDP_APP_SESSION_T appHandle, UINT32 etbTopoCnt)

Set new topocount for trainwide communication.

 - [UINT32 tlc_getETBTopoCount](#) (TRDP_APP_SESSION_T appHandle)

Set new topocount for trainwide communication.

 - [TRDP_ERR_T tlc_setOpTrainTopoCount](#) (TRDP_APP_SESSION_T appHandle, UINT32 opTrnTopoCnt)

Set new operational train topocount for direction/orientation sensitive communication.

 - [UINT32 tlc_getOpTrainTopoCount](#) (TRDP_APP_SESSION_T appHandle)

Set new operational train topocount for direction/orientation sensitive communication.

 - [TRDP_ERR_T tlc_getInterval](#) (TRDP_APP_SESSION_T appHandle, [TRDP_TIME_T](#) *pInterval, [TRDP_FDS_T](#) *pFileDesc, [TRDP SOCK_T](#) *pNoDesc)

Get the lowest time interval for PDs.

 - [TRDP_ERR_T tlc_process](#) (TRDP_APP_SESSION_T appHandle, [TRDP_FDS_T](#) *pRfds, INT32 *pCount)

Work loop of the TRDP handler.

 - [TRDP_IP_ADDR_T tlc_getOwnIpAddress](#) (TRDP_APP_SESSION_T appHandle)

Get the interface address.

 - [TRDP_ERR_T tlp_getInterval](#) (TRDP_APP_SESSION_T appHandle, [TRDP_TIME_T](#) *pInterval, [TRDP_FDS_T](#) *pFileDesc, [TRDP SOCK_T](#) *pNoDesc)

Get the lowest time interval for PDs.

 - [TRDP_ERR_T tlp_processSend](#) (TRDP_APP_SESSION_T appHandle)

Work loop of the TRDP handler.

 - [TRDP_ERR_T tlp_processReceive](#) (TRDP_APP_SESSION_T appHandle, [TRDP_FDS_T](#) *pRfds, INT32 *pCount)

Work loop of the TRDP handler.

 - [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 UINT8 *pData, UINT32 dataSize)

Prepare for sending PD messages.

 - [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 *pData, 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.

 - [TRDP_ERR_T tlp_setRedundant](#) (TRDP_APP_SESSION_T appHandle, UINT32 redId, BOOL8 leader)

Do not send non-redundant PDs when we are follower.

 - [TRDP_ERR_T tlp_getRedundant](#) (TRDP_APP_SESSION_T appHandle, UINT32 redId, BOOL8 *pLeader)

Get status of redundant Comlds.

- [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 UINT8 *pData, UINT32 dataSize, UINT32 replyComId, [TRDP_IP_ADDR_T](#) replyIpAddr)

Initiate sending PD messages (PULL).

- [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, UINT32 timeout, [TRDP_TO_BEHAVIOR_T](#) toBehavior)

Prepare for receiving PD messages.

- [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.

- [TRDP_ERR_T tlp_unsubscribe](#) (TRDP_APP_SESSION_T appHandle, TRDP_SUB_T subHandle)

Stop receiving PD messages.

- [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.

- const CHAR8 * [tlc_getVersionString](#) (void)

Return a human readable version representation.

- const [TRDP_VERSION_T](#) * [tlc_getVersion](#) (void)

Return version.

- [TRDP_ERR_T tlc_getStatistics](#) (TRDP_APP_SESSION_T appHandle, [TRDP_STATISTICS_T](#) *pStatistics)

Return statistics.

- [TRDP_ERR_T tlc_getSubsStatistics](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pNumSubs, [TRDP_SUBS_STATISTICS_T](#) *pStatistics)

Return PD subscription statistics.

- [TRDP_ERR_T tlc_getPubStatistics](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pNumPub, [TRDP_PUB_STATISTICS_T](#) *pStatistics)

Return PD publish statistics.

- [TRDP_ERR_T tlc_getRedStatistics](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pNumRed, [TRDP_RED_STATISTICS_T](#) *pStatistics)

Return redundancy group statistics.

- [TRDP_ERR_T tlc_getJoinStatistics](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pNumJoin, UINT32 *pIpAddr)

Return join statistics.

- [TRDP_ERR_T tlc_resetStatistics](#) (TRDP_APP_SESSION_T appHandle)

Reset statistics.

5.21.1 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, 2013-2020. All rights reserved.

5.21.2 Function Documentation**5.21.2.1 tlc_closeSession()**

```
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

5.21.2.2 tlc_configSession()

```
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

5.21.2.3 **tlc_getETBTopoCount()**

```
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>	
-------------------	--

5.21.2.4 **tlc_getInterval()**

```
TRDP_ERR_T tlc_getInterval (
    TRDP_APP_SESSION_T appHandle,
    TRDP_TIME_T * pInterval,
    TRDP_FDS_T * pFileDesc,
    TRDP_SOCKET_T * 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

5.21.2.5 `tlc_getJoinStatistics()`

```
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

5.21.2.6 `tlc_getOpTrainTopoCount()`

```
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 <code>tlc_openSession</code>
----	------------------	---

Return values

<i>opTrnTopoCnt</i>	New operational topocount value
---------------------	---------------------------------

5.21.2.7 `tlc_getOwnIpAddress()`

```
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>	
---------------------	--

5.21.2.8 `tlc_getPubStatistics()`

```
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
--------------------	----------

Return values

<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

< if set, packet should not be sent (redundant)

5.21.2.9 tlc_getRedStatistics()

```
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 tlc_openSession
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

< if set, packet should not be sent (redundant)

5.21.2.10 tlc_getStatistics()

```
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

5.21.2.11 tlc_getSubsStatistics()

```
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

5.21.2.12 tlc_getVersion()

```
const TRDP_VERSION_T* tlc_getVersion (
    void )
```

Return version.

Return pointer to version structure

Return values

<i>TRDP_VERSION</i> _↔ <i>_T</i>	
---	--

5.21.2.13 tlc_getVersionString()

```
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
--------------	--------

5.21.2.14 tlc_init()

```
TRDP_ERR_T tlc_init (
    const TRDP_PRINT_DBG_T pPrintDebugString,
    void * pRefCon,
    const TRDP_MEM_CONFIG_T * pMemConfig )
```

Initialize the TRDP stack.

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

5.21.2.15 tlc_openSession()

```
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

< all available sockets for PD

< 120 seconds (120 tries each second to bind to an IP address)

5.21.2.16 tlc_presetIndexSession()

```

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

5.21.2.17 tlc_process()

```
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 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

5.21.2.18 tlc_reinitSession()

```
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
<i>TRDP_PARAM_ERR</i>	handle NULL

< all available sockets for PD

5.21.2.19 tlc_resetStatistics()

```
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

5.21.2.20 tlc_setETBTopoCount()

```
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 <code>tlc_openSession</code>
in	<i>etbTopoCnt</i>	New <code>etbTopoCnt</code> value

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21.2.21 tlc_setOpTrainTopoCount()

```
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

5.21.2.22 tlc_terminate()

```
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

5.21.2.23 tlc_updateSession()

```
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 tlc_openSession
----	------------------	--

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	not yet initied
<i>TRDP_PARAM_ERR</i>	parameter error

5.21.2.24 tlp_get()

```
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

Return values

<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_COMID_ERR</i>	ComID not found when marshalling

5.21.2.25 tlp_getInterval()

```
TRDP_ERR_T tlp_getInterval (
    TRDP_APP_SESSION_T appHandle,
    TRDP_TIME_T * pInterval,
    TRDP_FDS_T * pFileDesc,
    TRDP SOCK_T * 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

5.21.2.26 tlp_getRedundant()

```
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

< if set, packet should not be sent (redundant)

5.21.2.27 **tlp_processReceive()**

```
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

5.21.2.28 **tlp_processSend()**

```
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

5.21.2.29 tlp_publish()

```

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 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 tlc_openSession
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>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 (>= 10ms) in usec
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>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 <= TRDP_MAX_PD_DATA_SIZE

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

< if set, inform the user

< if set, PD will be sent on trdp_put() only

< if set, packet should not be sent (redundant)

< if set, PD will be sent on trdp_put() only

5.21.2.30 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 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

5.21.2.31 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
<i>TRDP_MEM_ERR</i>	out of memory

< if set, inform the user

< if set, inform the user

5.21.2.32 tlp_republish()

```
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 <code>tlc_openSession</code>
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
<i>TRDP_MEM_ERR</i>	could not insert (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21.2.33 `tlp_request()`

```

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 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 <code>tlc_openSession</code>
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

Parameters

in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
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

< if set, the request needs to be sent

< if set, inform the user

5.21.2.34 tlp_resubscribe()

```
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

5.21.2.35 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

< if set, packet should not be sent (redundant)

< if set, packet should not be sent (redundant)

5.21.2.36 tlp_subscribe()

```
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,
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>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

< if set, inform the user

5.21.2.37 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
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.21.2.38 tlp_unsubscribe()

```
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 tlc_openSession
in	<i>subHandle</i>	the handle for this subscription

Return values

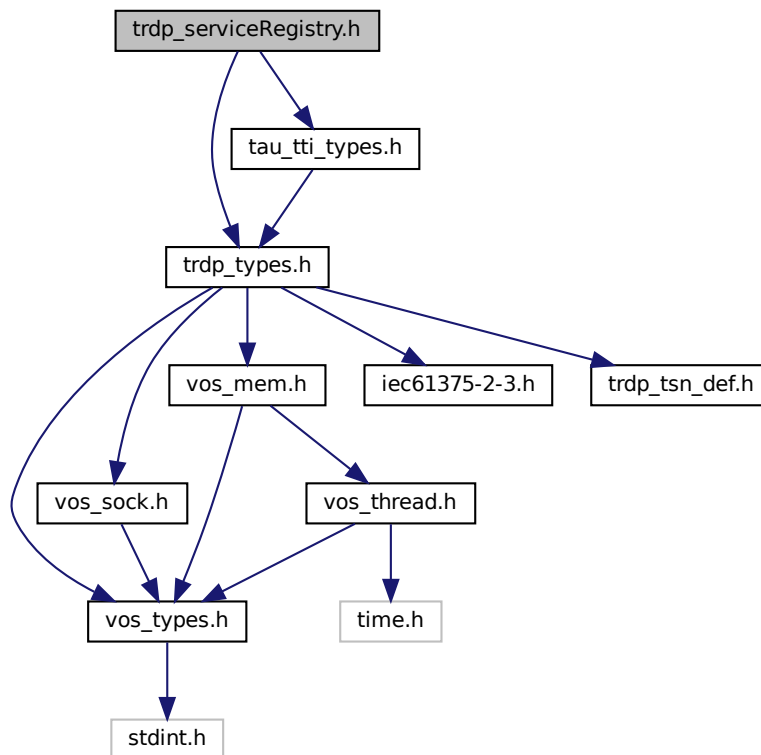
<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_NOSUB_ERR</i>	not subscribed
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.22 trdp_serviceRegistry.h File Reference

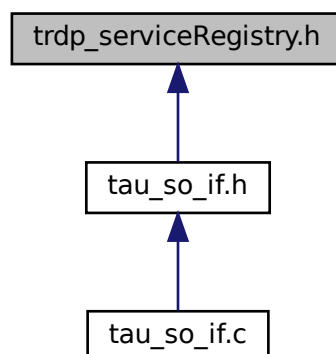
Additional definitions for IEC 61375-2-3 (Service Discovery) The definitions herein are preliminary and will change with the next major release of the IEC 61375-2-3 standard.

```
#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 [SRM_SERVICE_INFO_T](#)

Preliminary definition of a service info entry.

- struct [SRM_SRV_INFO_REQ_T](#)

Preliminary definition of a service info request.

Macros

- #define [SRM_SRVINFO_NOTIFY_COMID](#) 200u
Additional defines to be reserved for SR Manager
- #define [SRM_SRVINFO_NOTIFY_URI](#) "grpSRM.anyVeh.aCst.aCITrn.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.aCITrn.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_REP_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_REP_COMID](#) 115u

Reply returns instanceld

- #define [SRM_SERVICE_ADD_REP_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_REP_COMID](#) 118u
MD reply OK or not
- #define [SOA_SERVICEID](#)(instld, typeld) ((instld) << 24 | (typeld))
serviceld from instanceld and typeld
- #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

5.22.1 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

5.22.2 Macro Definition Documentation

5.22.2.1 SOA_SAME_SERVICEID

```
#define SOA_SAME_SERVICEID(  
    a,  
    b ) ((a) == (b))
```

return TRUE if servicelds (incl.

instance) match

5.22.2.2 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

5.22.2.3 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:

5.23.1 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.

5.23.2 Function Documentation

5.23.2.1 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 tlc_openSession
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

5.23.2.2 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 tlc_openSession
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

< if set, packet should not be sent (redundant)

5.23.2.3 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 tlc_openSession
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

< if set, packet should not be sent (redundant)

5.23.2.4 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

5.23.2.5 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

5.23.2.6 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 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

5.23.2.7 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 tlc_openSession
----	------------------	--

< host name

< leader host nameHere is the call graph for this function:



5.23.2.8 trdp_pdPrepareStats()

```
void trdp_pdPrepareStats (
    TRDP_APP_SESSION_T appHandle,
    PD_ELE_T * pPacket )
```

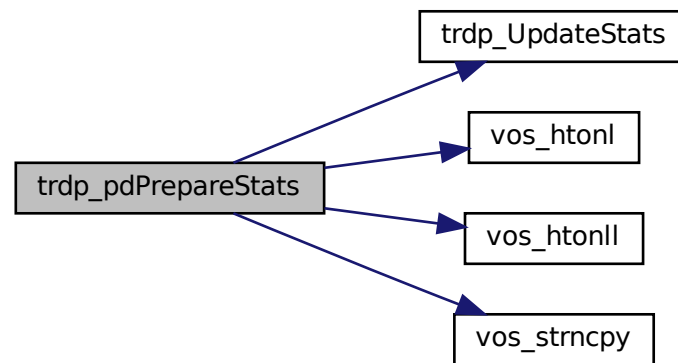
Fill the statistics packet.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pPacket</i>	pointer to the packet to fill

< if set, inform the user

Here is the call graph for this function:



5.23.2.9 trdp_UpdateStats()

```
void trdp_UpdateStats (
    TRDP_APP_SESSION_T appHandle )
```

Update the statistics.

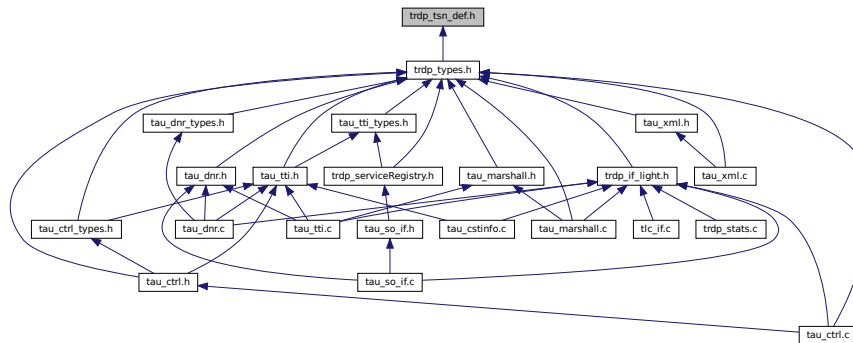
Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
----	------------------	---

5.24 trdp_tsn_def.h File Reference

Additional definitions for TSN.

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



Macros

- `#define TRDP_PD_DEFAULT_TSN_PRIORITY 7u /* #435*/`
Default PD communication parameters

5.24.1 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](#) 2400 2023-07-13 16:30:12Z peter-liesner

PL 2023-07-13: Ticket #435 Cleanup VLAN and TSN for options for Linux systems
 AHW 2023-06-08: Ticket #435 Cleanup VLAN and TSN options at different places
 PL 2023-05-19: Ticket #434 Code adaption due to TSN header version 2 removal.

5.24.2 Macro Definition Documentation

5.24.2.1 TRDP_PD_DEFAULT_TSN_PRIORITY

```
#define TRDP_PD_DEFAULT_TSN_PRIORITY 7u /* #435*/
```

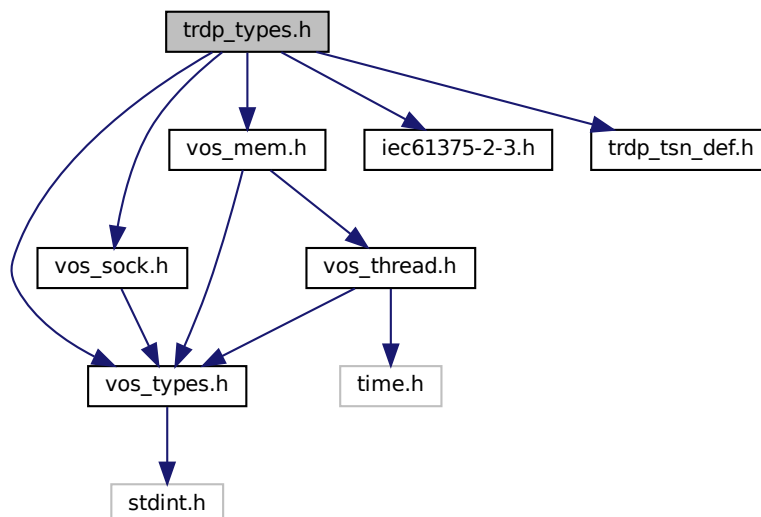
Default PD communication parameters

matching new proposed priority classes

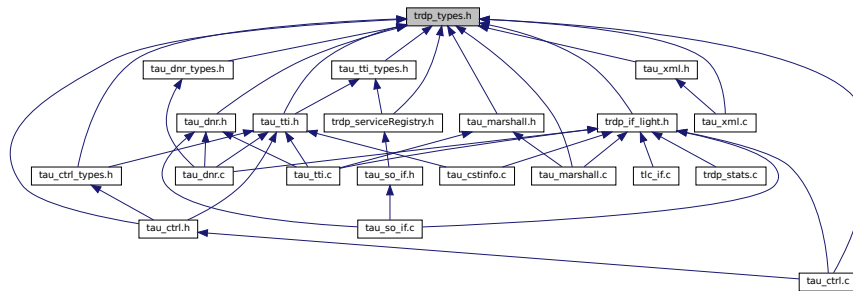
5.25 trdp_types.h File Reference

Typedefs for TRDP communication.

```
#include "vos_types.h"
#include "vos_mem.h"
#include "vos_sock.h"
#include "iec61375-2-3.h"
#include "trdp_tsn_def.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_T](#)
Dataset definition
- struct [TRDP_COMID_DSID_MAP_T](#)
ComId - data set mapping element definition
- struct [TRDP_STATISTICS_REQUEST_T](#)
TRDP statistics type definitions.
- struct [TRDP_PD_STATISTICS_T](#)
Structure containing all general PD statistics information.
- struct [TRDP_MD_STATISTICS_T](#)
Structure containing all general MD statistics information.
- struct [TRDP_STATISTICS_T](#)
Structure containing all general memory, PD and MD statistics information.
- struct [TRDP_SUBS_STATISTICS_T](#)
Table containing particular PD subscription information.
- struct [TRDP_PUB_STATISTICS_T](#)
Table containing particular PD publishing information.
- struct [TRDP_LIST_STATISTICS_T](#)
Information about a particular MD listener.
- struct [TRDP_RED_STATISTICS_T](#)
A table containing PD redundant group information.
- 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
< Real time extensions (preliminary)
- #define [TRDP_INVALID_SOCKET](#) [VOS_INVALID_SOCKET](#)
Invalid socket number.
- #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_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_TIMER_GRANULARITY](#) 500u
granularity in us - we allow 0.5ms now!
- #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[16u]`
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_SOCKET_T TRDP_SOCKET_T`
Socket descriptor set compatible with fd_set / select.
- `typedef VOS_UUID_T TRDP_UUID_T`
UUID definition reuses the VOS definition.
- `typedef TRDP_DATASET_T * pTRDP_DATASET_T`
Array of pointers to dataset
- `typedef VOS_MEM_STATISTICS_T TRDP_MEM_STATISTICS_T`
Structure containing all general memory statistics information.
- `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 marshallng .
- 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 [TRDP_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 [TRDP_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.

5.25.1 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, 2015-2021. All rights reserved.

5.25.2 Macro Definition Documentation

5.25.2.1 TRDP_FLAGS_DEFAULT

```
#define TRDP_FLAGS_DEFAULT 0u
```

Various flags for PD and MD packets

Default value defined in `tlc_openDession` will be taken

5.25.2.2 USE_HEAP

```
#define USE_HEAP 0
```

< Real time extensions (preliminary)

If this is set, we can allocate dynamically memory

5.25.3 Typedef Documentation

5.25.3.1 TRDP_IP_ADDR_T

```
typedef VOS_IP4_ADDR_T TRDP_IP_ADDR_T
```

TRDP general type definitions.

5.25.3.2 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 marshallng .

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 too small
<i>TRDP_COMID_ERR</i>	comid not existing

5.25.3.3 TRDP_MD_CALLBACK_T

```
typedef void(* TRDP_MD_CALLBACK_T) (void *pRefCon, TRDP_APP_SESSION_T appHandle, const TRDP_MD_INFO_T *pMsg, UINT8 *pData, UINT32 dataSize)
```

Callback for receiving indications, timeouts, releases, responses.

Parameters

in	<i>appHandle</i>	handle returned also by <i>tlc_init</i>
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

5.25.3.4 TRDP_PD_CALLBACK_T

```
typedef void(* TRDP_PD_CALLBACK_T) (void *pRefCon, TRDP_APP_SESSION_T appHandle, const TRDP_PD_INFO_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 <i>tlc_openSession</i>
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

5.25.3.5 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.

5.25.3.6 TRDP_TIME_T

```
typedef VOS_TIMEVAL_T TRDP_TIME_T
```

Timer value compatible with timeval / select.

Relative or absolute date, depending on usage

5.25.3.7 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

5.25.4 Enumeration Type Documentation

5.25.4.1 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
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

5.25.4.2 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

Enumerator

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
TRDP_APP_CONFIRMTO_ERR	Application Confirm Sent Timeout
TRDP_REPLYTO_ERR	Protocol Reply Timeout

Enumerator

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

5.25.4.3 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

5.25.4.4 TRDP_REPLY_STATUS_T

enum [TRDP_REPLY_STATUS_T](#)

TRDP data transfer type definitions.

Reply status messages

5.25.4.5 TRDP_TO_BEHAVIOR_T

enum [TRDP_TO_BEHAVIOR_T](#)

How invalid PD shall be handled

Enumerator

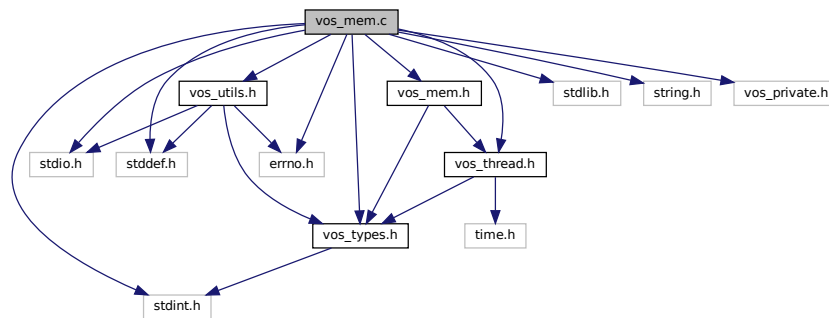
TRDP_TO_DEFAULT	Default value defined in tlc_openDession will be taken
TRDP_TO_SET_TO_ZERO	If set, data will be reset to zero on time out
TRDP_TO_KEEP_LAST_VALUE	If set, last received values will be returned

5.26 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:



Data Structures

- struct [VOS_QUEUE_T](#)
Opaque queue define

Functions

- EXT_DECL [VOS_ERR_T vos_memInit](#) (UINT8 *pMemoryArea, UINT32 size, const UINT32 frag←
Mem[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](#) (VOS_MEM_STATISTICS_T *pMemCount)
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_strnicmp](#) (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.

5.26.1 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.

5.26.2 Function Documentation

5.26.2.1 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
----------------	--------------------------

5.26.2.2 vos_memAlloc()

```
EXT_DECL UINT8* vos_memAlloc (
    UINT32 size )
```

Allocate a block of memory (from memory area above).

Always clears returned memory area

Parameters

in	<i>size</i>	Size of requested block
----	-------------	-------------------------

Return values

<i>Pointer</i>	to memory area
<i>NULL</i>	if no memory available

5.26.2.3 vos_memCount()

```
EXT_DECL VOS_ERR_T vos_memCount (
    VOS_MEM_STATISTICS_T * pMemCount )
```

Return used and available memory (of memory area above).

Parameters

out	<i>pMemCount</i>	Pointer to memory statistics structure
-----	------------------	--

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter error (nullpointer)
<i>VOS_INIT_ERR</i>	module not initialised

5.26.2.4 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
----	--------------------	-----------------------------

5.26.2.5 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
----	------------------	-------------------------------------

5.26.2.6 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

5.26.2.7 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>	
-------------	--

5.26.2.8 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

5.26.2.9 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

5.26.2.10 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

5.26.2.11 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

5.26.2.12 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>	
-------------	--

5.26.2.13 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>	
-------------	--

5.26.2.14 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

<i>0</i>	- equal
<i><0</i>	- string1 less than string 2
<i>>0</i>	- string 1 greater than string 2

5.27 vos_mem.h File Reference

Memory and queue functions for OS abstraction.

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

```
graph TD; vos_mem.h --> vos_thread.h; vos_mem.h --> vos_types.h; vos_thread.h --> vos_types.h; vos_thread.h --> time.h; vos_types.h --> stdint.h
```

[illegible]

- struct `VOS_MEM_STATISTICS_T`
Structure containing all general memory statistics information.

- #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.

Enumerations

- enum [VOS_QUEUE_POLICY_T](#)
Queue policy matching pthread/Posix defines

Functions

- [VOS_ERR_T vos_memInit](#) (UINT8 *pMemoryArea, UINT32 size, const UINT32 fragMem[15u])
Initialize the memory unit.
- void [vos_memDelete](#) (UINT8 *pMemoryArea)
Delete the memory area.
- UINT8 * [vos_memAlloc](#) (UINT32 size)
Allocate a block of memory (from memory area above).
- void [vos_memFree](#) (void *pMemBlock)
Deallocate a block of memory (from memory area above).
- [VOS_ERR_T vos_memCount](#) ([VOS_MEM_STATISTICS_T](#) *pMemCount)
Return used and available memory (of memory area above).
- void [vos_qsort](#) (void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))
Sort an array.
- 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.
- INT32 [vos_strncmp](#) (const CHAR8 *pStr1, const CHAR8 *pStr2, UINT32 count)
Case insensitive string compare.
- void [vos_strncpy](#) (CHAR8 *pStrDst, const CHAR8 *pStrSrc, UINT32 count)
String copy with length limitation.
- void [vos_strncat](#) (CHAR8 *pStrDst, UINT32 count, const CHAR8 *pStrSrc)
String concatenation with length limitation.
- [VOS_ERR_T vos_queueCreate](#) ([VOS_QUEUE_POLICY_T](#) queueType, UINT32 maxNoOfMsg, [VOS_QUEUE_T](#) *pQueueHandle)
Initialize a message queue.
- [VOS_ERR_T vos_queueSend](#) ([VOS_QUEUE_T](#) queueHandle, UINT8 *pData, UINT32 size)
Send a message.
- [VOS_ERR_T vos_queueReceive](#) ([VOS_QUEUE_T](#) queueHandle, UINT8 **ppData, UINT32 *pSize, UINT32 usTimeout)
Get a message.
- [VOS_ERR_T vos_queueDestroy](#) ([VOS_QUEUE_T](#) queueHandle)
Destroy a message queue.

5.27.1 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.

5.27.2 Macro Definition Documentation

5.27.2.1 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

5.27.2.2 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.

5.27.3 Function Documentation

5.27.3.1 vos_bsearch()

```
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
----------------	--------------------------

5.27.3.2 vos_memAlloc()

```
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

Always clears returned memory area

Parameters

in	<i>size</i>	Size of requested block
----	-------------	-------------------------

Return values

<i>Pointer</i>	to memory area
<i>NULL</i>	if no memory available

5.27.3.3 vos_memCount()

```
VOS_ERR_T vos_memCount (
    VOS_MEM_STATISTICS_T * pMemCount )
```


Return used and available memory (of memory area above).

Parameters

out	<i>pMemCount</i>	Pointer to memory statistics structure
-----	------------------	--

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter error (nullpointer)
<i>VOS_INIT_ERR</i>	module not initialised

5.27.3.4 vos_memDelete()

```
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
----	--------------------	-----------------------------

5.27.3.5 vos_memFree()

```
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
----	------------------	-------------------------------------

5.27.3.6 vos_memInit()

```
VOS_ERR_T vos_memInit (
    UINT8 * pMemoryArea,
    UINT32 size,
    const UINT32 fragMem[15u] )
```

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

5.27.3.7 vos_qsort()

```
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>	
-------------	--

5.27.3.8 vos_queueCreate()

```
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

5.27.3.9 vos_queueDestroy()

```
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

5.27.3.10 vos_queueReceive()

```
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

5.27.3.11 vos_queueSend()

```
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

5.27.3.12 vos_strncat()

```
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>	
-------------	--

5.27.3.13 vos_strncpy()

```
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>	
-------------	--

5.27.3.14 vos_strnicmp()

```
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

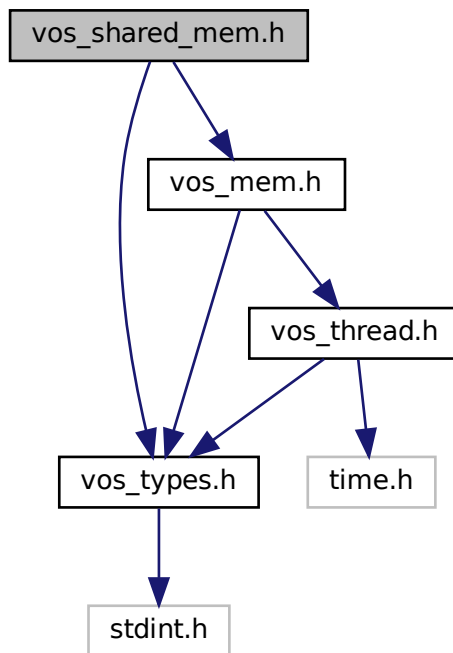
<i>0</i>	- equal
<i><0</i>	- string1 less than string 2
<i>>0</i>	- string 1 greater than string 2

5.28 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

- **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.
- **VOS_ERR_T vos_sharedClose** (VOS_SHRD_T handle, const UINT8 *pMemoryArea)
Close connection to the shared memory area.

5.28.1 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.

5.28.2 Function Documentation

5.28.2.1 vos_sharedClose()

```
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

5.28.2.2 vos_sharedOpen()

```
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

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](#) 80
The maximum number of IP interface adapters that can be handled by VOS.
- #define [VOS_MAX_NUM_IF](#) 64
TSN uses VLAN and each VLAN creates its own IP interface.
- #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.
- #define [vos_sockId](#)(sock) (INT32)(sock)
Return the socket ID.
- #define [vos_sockCmp](#)(sockA, sockB) (sockA < sockB ? -1 : sockA > sockB ? 1 : 0)
Compare sockets.

Functions

- void [vos_reCollectIpInterfaces](#) ()
Use after creating or deleting a network interface: to update the stored IP, VLAN and MAC addresses of local network interfaces.
- UINT16 [vos_htons](#) (UINT16 val)
Byte swapping 2 Bytes.
- UINT16 [vos_ntohs](#) (UINT16 val)
Byte swapping 2 Bytes.
- UINT32 [vos_htonl](#) (UINT32 val)
Byte swapping 4 Bytes.
- UINT32 [vos_ntohl](#) (UINT32 val)
Byte swapping 4 Bytes.
- UINT64 [vos_htonll](#) (UINT64 val)
Byte swapping 8 Bytes.
- UINT64 [vos_ntohll](#) (UINT64 val)
Byte swapping 8 Bytes.
- UINT32 [vos_dottedIP](#) (const CHAR8 *pDottedIP)
Convert IP address from dotted dec.
- const CHAR8 * [vos_ipDotted](#) (UINT32 ipAddress)
Convert IP address to dotted dec.
- BOOL8 [vos_isMulticast](#) (UINT32 ipAddress)
Check if the supplied address is a multicast group address.
- [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.
- BOOL8 [vos_netIfUp](#) (VOS_IP4_ADDR_T ifAddress)

- Get the state of an interface.*
- INT32 [vos_select](#) (VOS_SOCKET_T highDesc, VOS_FDS_T *pReadableFD, VOS_FDS_T *pWriteableFD, VOS_FDS_T *pErrorFD, [VOS_TIMEVAL_T](#) *pTimeOut)
select function.
 - [VOS_ERR_T vos_sockInit](#) (void)
Initialize the socket library.
 - void [vos_sockTerm](#) (void)
De-Initialize the socket library.
 - [VOS_ERR_T vos_sockGetMAC](#) (UINT8 pMAC[6])
Return the MAC address of the default adapter.
 - [VOS_ERR_T vos_sockOpenUDP](#) (VOS_SOCKET_T *pSock, const [VOS_SOCKET_OPT_T](#) *pOptions)
Create an UDP socket.
 - [VOS_ERR_T vos_sockOpenTCP](#) (VOS_SOCKET_T *pSock, const [VOS_SOCKET_OPT_T](#) *pOptions)
Create a TCP socket.
 - [VOS_ERR_T vos_sockClose](#) (VOS_SOCKET_T sock)
Close a socket.
 - [VOS_ERR_T vos_sockSetOptions](#) (VOS_SOCKET_T sock, const [VOS_SOCKET_OPT_T](#) *pOptions)
Set socket options.
 - [VOS_ERR_T vos_sockJoinMC](#) (VOS_SOCKET_T sock, UINT32 mcAddress, UINT32 ipAddress)
Join a multicast group.
 - [VOS_ERR_T vos_sockLeaveMC](#) (VOS_SOCKET_T sock, UINT32 mcAddress, UINT32 ipAddress)
Leave a multicast group.
 - [VOS_ERR_T vos_sockSendUDP](#) (VOS_SOCKET_T sock, const UINT8 *pBuffer, UINT32 *pSize, UINT32 ip↵Address, UINT16 port)
Send UDP data.
 - [VOS_ERR_T vos_sockReceiveUDP](#) (VOS_SOCKET_T sock, UINT8 *pBuffer, UINT32 *pSize, UINT32 *pSrc↵IPAddr, UINT16 *pSrcIPPort, UINT32 *pDstIPAddr, UINT32 *pSrcIFAddr, BOOL8 peek)
Receive UDP data.
 - [VOS_ERR_T vos_sockBind](#) (VOS_SOCKET_T sock, UINT32 ipAddress, UINT16 port)
Bind a socket to an address and port.
 - [VOS_ERR_T vos_sockListen](#) (VOS_SOCKET_T sock, UINT32 backlog)
Listen for incoming TCP connections.
 - [VOS_ERR_T vos_sockAccept](#) (VOS_SOCKET_T sock, VOS_SOCKET_T *pSock, UINT32 *pIPAddress, UINT16 *pPort)
Accept an incoming TCP connection.
 - [VOS_ERR_T vos_sockConnect](#) (VOS_SOCKET_T sock, UINT32 ipAddress, UINT16 port)
Open a TCP connection.
 - [VOS_ERR_T vos_sockSendTCP](#) (VOS_SOCKET_T sock, const UINT8 *pBuffer, UINT32 *pSize)
Send TCP data.
 - [VOS_ERR_T vos_sockReceiveTCP](#) (VOS_SOCKET_T sock, UINT8 *pBuffer, UINT32 *pSize)
Receive TCP data.
 - [VOS_ERR_T vos_sockSetMulticastIf](#) (VOS_SOCKET_T sock, UINT32 mclfAddress)
Set Using Multicast I/F.
 - 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.
 - [VOS_ERR_T vos_ifnameFromVlanId](#) (UINT16 vlanId, VOS_IP4_ADDR_T ipAddr)
Get the interface name for a given VLAN ID (and IP address, if given)
 - [VOS_ERR_T vos_sockOpenTSN](#) (VOS_SOCKET_T *pSock, const [VOS_SOCKET_OPT_T](#) *pOptions)
Create a TSN socket.
 - [VOS_ERR_T vos_sockSendTSN](#) (VOS_SOCKET_T sock, const UINT8 *pBuffer, UINT32 *pSize, VOS_IP4_↵ADDR_T srcIpAddress, VOS_IP4_ADDR_T dstIpAddress, UINT16 port, [VOS_TIMEVAL_T](#) *pTxTime)

Send TSN over UDP data.

- `VOS_ERR_T vos_sockReceiveTSN` (VOS_SOCK_T sock, UINT8 *pBuffer, UINT32 *pSize, UINT32 *pSrcIPAddr, UINT16 *pSrcIPPort, UINT32 *pDstIPAddr, BOOL8 peek)

Receive TSN (UDP) data.

- void `vos_sockPrintOptions` (VOS_SOCK_T sock)

Debug output main socket options.

5.29.1 Detailed Description

Typedefs for OS abstraction.

This is the declaration for the OS independent 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, 2013-2021. All rights reserved.

5.29.2 Macro Definition Documentation

5.29.2.1 VOS_MAX_NUM_IF

```
#define VOS_MAX_NUM_IF 64
```

TSN uses VLAN and each VLAN creates its own IP interface.

The maximum number of unicast addresses that can be handled by VOS

5.29.2.2 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

5.29.2.3 vos_sockCmp

```
#define vos_sockCmp(  
    sockA,  
    sockB ) (sockA < sockB ? -1 : sockA > sockB ? 1 : 0)
```

Compare sockets.

Parameters

in	<i>sockA</i>	Socket A
in	<i>sockB</i>	Socket B

Return values

-1	A < B
0	A = B
1	A > B

5.29.2.4 vos_sockId

```
#define vos_sockId(  
    sock ) (INT32) (sock)
```

Return the socket ID.

Parameters

in	<i>sock</i>	socket descriptor
----	-------------	-------------------

Return values

<i>Socket</i>	ID.
---------------	-----

5.29.2.5 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

5.29.3 Function Documentation

5.29.3.1 vos_determineBindAddr()

```
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
----------------	------------

5.29.3.2 vos_dottedIP()

```
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
----------------	------------------------------

5.29.3.3 vos_getInterfaces()

```
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>	pAddrCnt and/or ifAddrs == NULL
<i>VOS_MEM_ERR</i>	memory allocation error
<i>VOS SOCK_ERR</i>	GetAdaptersInfo() error

5.29.3.4 vos_htonl()

```
UINT32 vos_htonl (
    UINT32 val )
```

Byte swapping 4 Bytes.

Parameters

in	<i>val</i>	Initial value.
----	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.29.3.5 vos_htonll()

```
UINT64 vos_htonll (
    UINT64 val )
```

Byte swapping 8 Bytes.

Parameters

in	<i>val</i>	Initial value.
----	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.29.3.6 vos_htons()

```
UINT16 vos_htons (
    UINT16 val )
```

Byte swapping 2 Bytes.

Parameters

in	<i>val</i>	Initial value.
----	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.29.3.7 vos_ifnameFromVlanId()

```
VOS_ERR_T vos_ifnameFromVlanId (  
    UINT16 vlanId,  
    VOS_IP4_ADDR_T ipAddr )
```

Get the interface name for a given VLAN ID (and IP address, if given)

Parameters

in	<i>vlanId</i>	vlan ID to find
in	<i>ipAddr</i>	IP to find (0 = match any)

Return values

<i>VOS_NO_ERR</i>	if found
<i>VOS_INIT_ERR</i>	vos_sockInit needs to be called first
<i>VOS_PARAM_ERR</i>	vlan 1..4094 allowed (0=no VLAN, 4095=wildcard)

5.29.3.8 vos_ipDotted()

```
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.
-----------	----------------------------

5.29.3.9 vos_isMulticast()

```
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

5.29.3.10 vos_netIfUp()

```
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

5.29.3.11 vos_ntohl()

```
UINT32 vos_ntohl (
    UINT32 val )
```

Byte swapping 4 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.29.3.12 vos_ntohl()

```
UINT64 vos_ntohl (
    UINT64 val )
```

Byte swapping 8 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.29.3.13 vos_ntohs()

```
UINT16 vos_ntohs (
    UINT16 val )
```

Byte swapping 2 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.29.3.14 vos_select()

```
INT32 vos_select (
```

```

VOS_SOCKET_T 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

in	<i>highDesc</i>	max. socket descriptor
in, out	<i>pReadableFD</i>	pointer to readable socket set
in, out	<i>pWriteableFD</i>	pointer to writeable socket set
in, out	<i>pErrorFD</i>	pointer to error socket set
in	<i>pTimeout</i>	pointer to time out value

Return values

<i>number</i>	of ready file descriptors
---------------	---------------------------

5.29.3.15 vos_sockAccept()

```

VOS_ERR_T vos_sockAccept (
    VOS_SOCKET_T sock,
    VOS_SOCKET_T * 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

5.29.3.16 vos_sockBind()

```
VOS_ERR_T vos_sockBind (  
    VOS_SOCK_T 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

5.29.3.17 vos_sockClose()

```
VOS_ERR_T vos_sockClose (  
    VOS_SOCK_T sock )
```

Close a socket.

Release any resources aquired 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

5.29.3.18 vos_sockConnect()

```
VOS_ERR_T vos_sockConnect (
    VOS_SOCK_T 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

5.29.3.19 vos_sockGetMAC()

```
VOS_ERR_T vos_sockGetMAC (
    UINT8 pMAC[6] )
```

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
<i>VOS_SOCK_ERR</i>	socket not available or option not supported

5.29.3.20 vos_sockInit()

```
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

5.29.3.21 vos_sockJoinMC()

```
VOS_ERR_T vos_sockJoinMC (
    VOS_SOCK_T 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

5.29.3.22 vos_sockLeaveMC()

```
VOS_ERR_T vos_sockLeaveMC (
    VOS_SOCK_T 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

5.29.3.23 vos_sockListen()

```
VOS_ERR_T vos_sockListen (
    VOS_SOCK_T 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

5.29.3.24 vos_sockOpenTCP()

```
VOS_ERR_T vos_sockOpenTCP (
    VOS_SOCK_T * 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

5.29.3.25 vos_sockOpenTSN()

```
VOS_ERR_T vos_sockOpenTSN (
    VOS_SOCK_T * pSock,
    const VOS_SOCK_OPT_T * pOptions )
```

Create a TSN 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

5.29.3.26 vos_sockOpenUDP()

```
VOS_ERR_T vos_sockOpenUDP (
    VOS_SOCK_T * 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_SOCKET_ERR</i>	socket not available or option not supported

5.29.3.27 vos_sockPrintOptions()

```
void vos_sockPrintOptions (
    VOS_SOCKET_T sock )
```

Debug output main socket options.

Parameters

in	<i>sock</i>	socket
----	-------------	--------

5.29.3.28 vos_sockReceiveTCP()

```
VOS_ERR_T vos_sockReceiveTCP (
    VOS_SOCKET_T 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

5.29.3.29 vos_sockReceiveTSN()

```
VOS_ERR_T vos_sockReceiveTSN (
    VOS_SOCK_T sock,
    UINT8 * pBuffer,
    UINT32 * pSize,
    UINT32 * pSrcIPAddr,
    UINT16 * pSrcIPPort,
    UINT32 * pDstIPAddr,
    BOOL8 peek )
```

Receive TSN (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

5.29.3.30 vos_sockReceiveUDP()

```
VOS_ERR_T vos_sockReceiveUDP (
    VOS_SOCK_T sock,
    UINT8 * pBuffer,
    UINT32 * pSize,
    UINT32 * pSrcIPAddr,
    UINT16 * pSrcIPPort,
    UINT32 * pDstIPAddr,
```

```

    UINT32 * pSrcIFAddr,
    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
out	<i>pSrcIFAddr</i>	pointer to source network interface IP
in	<i>peek</i>	if true, leave data in queue

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	sock descriptor unknown, parameter error
VOS_IO_ERR	data could not be read
VOS_NODATA_ERR	no data
VOS_BLOCK_ERR	Call would have blocked in blocking mode

5.29.3.31 vos_sockSendTCP()

```

VOS_ERR_T vos_sockSendTCP (
    VOS_SOCK_T 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

5.29.3.32 vos_sockSendTSN()

```

VOS_ERR_T vos_sockSendTSN (
    VOS_SOCK_T sock,
    const UINT8 * pBuffer,
    UINT32 * pSize,
    VOS_IP4_ADDR_T srcIpAddress,
    VOS_IP4_ADDR_T dstIpAddress,
    UINT16 port,
    VOS_TIMEVAL_T * pTxTime )

```

Send TSN over UDP 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
in	<i>srcIpAddress</i>	source IP
in	<i>dstIpAddress</i>	destination IP
in	<i>port</i>	destination port
in	<i>pTxTime</i>	absolute time when to send this packet

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_BLOCK_ERR</i>	Call would have blocked in blocking mode

5.29.3.33 vos_sockSendUDP()

```

VOS_ERR_T vos_sockSendUDP (
    VOS_SOCK_T 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

5.29.3.34 vos_sockSetMulticastIf()

```

VOS_ERR_T vos_sockSetMulticastIf (
    VOS_SOCK_T 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

5.29.3.35 vos_sockSetOptions()

```

VOS_ERR_T vos_sockSetOptions (

```

```
VOS_SOCK_T sock,
const VOS_SOCK_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

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid

5.29.3.36 vos_sockTerm()

```
void vos_sockTerm (
    void )
```

De-Initialize the socket library.

Must be called after last socket call

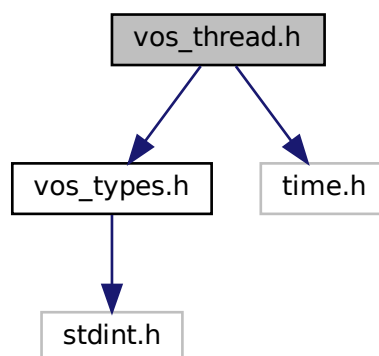
5.30 vos_thread.h File Reference

Threading functions for OS abstraction.

```
#include "vos_types.h"
```

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

Include dependency graph for vos_thread.h:



- #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.

- typedef UINT8 **VOS_THREAD_PRIORITY_T**
Thread priority range from 1 (lowest) to 255 (highest), 0 default of the target system
- typedef void(* **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

- enum `VOS_THREAD_POLICY_T`
Thread policy matching pthread/Posix defines
- enum `VOS_SEMA_STATE_T`
State of the semaphore

Functions

- [VOS_ERR_T vos_threadInit](#) (void)
Initialize the thread library.
- void [vos_threadTerm](#) (void)
De-Initialize the thread library.
- [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.
- [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.
- [VOS_ERR_T vos_threadTerminate](#) ([VOS_THREAD_T](#) thread)
Terminate a thread.
- [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.
- [VOS_ERR_T vos_threadDelay](#) ([UINT32](#) delay)
Delay the execution of the current thread by the given delay in us.
- [VOS_ERR_T vos_threadSelf](#) ([VOS_THREAD_T](#) *pThread)
Return thread handle of calling task.
- void [vos_getTime](#) ([VOS_TIMEVAL_T](#) *pTime)
Return the current monotonic time in sec and us.
- void [vos_getRealTime](#) ([VOS_TIMEVAL_T](#) *pTime)
Return the current real time in sec and us.
- const [CHAR8](#) * [vos_getTimeStamp](#) (void)
Get a time-stamp string.
- void [vos_clearTime](#) ([VOS_TIMEVAL_T](#) *pTime)
Clear the time stamp.
- void [vos_addTime](#) ([VOS_TIMEVAL_T](#) *pTime, const [VOS_TIMEVAL_T](#) *pAdd)
Add the second to the first time stamp, return sum in first.
- void [vos_subTime](#) ([VOS_TIMEVAL_T](#) *pTime, const [VOS_TIMEVAL_T](#) *pSub)
Subtract the second from the first time stamp, return diff in first.
- [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.
- void [vos_divTime](#) ([VOS_TIMEVAL_T](#) *pTime, [UINT32](#) divisor)
Divide the first time by the second, return quotient in first.
- void [vos_mulTime](#) ([VOS_TIMEVAL_T](#) *pTime, [UINT32](#) mul)
Multiply the first time by the second, return product in first.
- void [vos_getUuid](#) ([VOS_UUID_T](#) pUuid)
Get a universal unique identifier according to RFC 4122 time based version.
- [VOS_ERR_T vos_mutexCreate](#) ([VOS_MUTEX_T](#) *pMutex)
Create a mutex.
- void [vos_mutexDelete](#) ([VOS_MUTEX_T](#) pMutex)
Delete a mutex.
- [VOS_ERR_T vos_mutexLock](#) ([VOS_MUTEX_T](#) pMutex)
Take a mutex.
- [VOS_ERR_T vos_mutexTryLock](#) ([VOS_MUTEX_T](#) pMutex)
Try to take a mutex.
- [VOS_ERR_T vos_mutexUnlock](#) ([VOS_MUTEX_T](#) pMutex)

Release a mutex.

- [VOS_ERR_T vos_semaCreate](#) ([VOS_SEMA_T](#) *pSema, [VOS_SEMA_STATE_T](#) initialState)

Create a semaphore.

- void [vos_semaDelete](#) ([VOS_SEMA_T](#) sema)

Delete a semaphore.

- [VOS_ERR_T vos_semaTake](#) ([VOS_SEMA_T](#) sema, UINT32 timeout)

Take a semaphore.

- void [vos_semaGive](#) ([VOS_SEMA_T](#) sema)

Give a semaphore.

5.30.1 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.

5.30.2 Function Documentation

5.30.2.1 vos_addTime()

```
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

5.30.2.2 vos_clearTime()

```
void vos_clearTime (
    VOS_TIMEVAL_T * pTime )
```

Clear the time stamp.

Parameters

out	<i>pTime</i>	Pointer to time value
-----	--------------	-----------------------

5.30.2.3 vos_cmpTime()

```
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

5.30.2.4 vos_divTime()

```
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

5.30.2.5 vos_getRealTime()

```
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
-----	--------------	-----------------------

5.30.2.6 vos_getTime()

```
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
-----	--------------	-----------------------

5.30.2.7 vos_getTimeStamp()

```
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"
------------------	------------------------

5.30.2.8 vos_getUuid()

```
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
-----	--------------	--

5.30.2.9 vos_mulTime()

```
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

5.30.2.10 vos_mutexCreate()

```
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

5.30.2.11 vos_mutexDelete()

```
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
-------------------	----------

5.30.2.12 vos_mutexLock()

```
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

5.30.2.13 vos_mutexTryLock()

```
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

5.30.2.14 vos_mutexUnlock()

```
VOS_ERR_T vos_mutexUnlock (
    VOS_MUTEX_T pMutex )
```

Release a mutex.

Unlock the mutex.

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

5.30.2.15 vos_semaCreate()

```
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 sempahore

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

5.30.2.16 vos_semaDelete()

```
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
----	-------------	------------------

5.30.2.17 vos_semaGive()

```
void vos_semaGive (
    VOS_SEMA_T sema )
```

Give a semaphore.

Release (increase) a semaphore.

Parameters

in	<i>sema</i>	semaphore handle
----	-------------	------------------

5.30.2.18 vos_semaTake()

```
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

Return values

<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

5.30.2.19 vos_subTime()

```
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

5.30.2.20 vos_threadCreate()

```
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

<code>VOS_NO_ERR</code>	no error
<code>VOS_INIT_ERR</code>	module not initialised
<code>VOS_NOINIT_ERR</code>	invalid handle
<code>VOS_PARAM_ERR</code>	parameter out of range/invalid

5.30.2.21 vos_threadCreateSync()

```

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)
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

<code>VOS_NO_ERR</code>	no error
<code>VOS_INIT_ERR</code>	module not initialised
<code>VOS_NOINIT_ERR</code>	invalid handle
<code>VOS_PARAM_ERR</code>	parameter out of range/invalid

5.30.2.22 vos_threadDelay()

```
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

5.30.2.23 vos_threadInit()

```
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

5.30.2.24 vos_threadIsActive()

```
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

<i>VOS_NO_ERR</i>	no error
-------------------	----------

Return values

<code>VOS_INIT_ERR</code>	module not initialised
<code>VOS_NOINIT_ERR</code>	invalid handle
<code>VOS_PARAM_ERR</code>	parameter out of range/invalid

5.30.2.25 vos_threadSelf()

```
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

<code>VOS_NO_ERR</code>	no error
<code>VOS_PARAM_ERR</code>	parameter out of range/invalid

5.30.2.26 vos_threadTerm()

```
void vos_threadTerm (
    void )
```

De-Initialize the thread library.

Must be called after last thread/timer call

5.30.2.27 vos_threadTerminate()

```
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

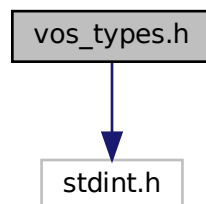
<code>VOS_NO_ERR</code>	no error
<code>VOS_INIT_ERR</code>	module not initialised
<code>VOS_NOINIT_ERR</code>	invalid handle
<code>VOS_PARAM_ERR</code>	parameter out of range/invalid

5.31 vos_types.h File Reference

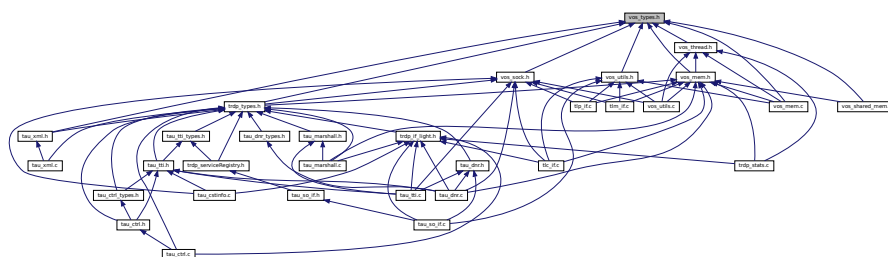
Typedefs for OS abstraction.

```
#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_SOCKET_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

5.31.1 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.

5.31.2 Typedef Documentation

5.31.2.1 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

5.31.2.2 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

5.31.3 Enumeration Type Documentation

5.31.3.1 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 SOCK_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

5.31.3.2 VOS_LOG_T

enum [VOS_LOG_T](#)

Categories for logging

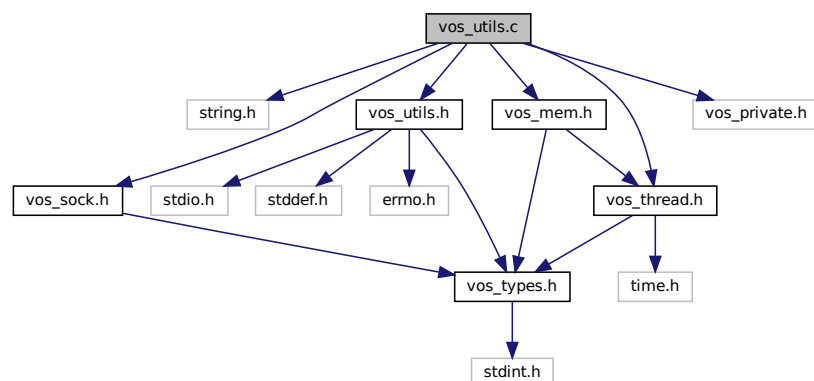
Enumerator

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

5.32 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.

5.32.1 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.

5.32.2 Function Documentation

5.32.2.1 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

<i>in</i>	<i>crc</i>	Initial value.
<i>in, out</i>	<i>pData</i>	Pointer to data.
<i>in</i>	<i>dataLen</i>	length in bytes of data.

Return values

<i>crc32</i>	according to IEEE802.3
--------------	------------------------

5.32.2.2 vos_getErrorString()

```
EXT_DECL const CHAR8* vos_getErrorString (
    VOS_ERR_T error )
```

Return a human readable error representation.

Parameters

<i>in</i>	<i>error</i>	The TRDP or VOS error code
-----------	--------------	----------------------------

Return values

<i>const</i>	string pointer to error string
--------------	--------------------------------

5.32.2.3 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>	
----------------------	--

5.32.2.4 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
--------------	--------

5.32.2.5 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

5.32.2.6 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
-------------------	--

5.32.2.7 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

<i>in</i>	<i>crc</i>	Initial value.
<i>in, out</i>	<i>pData</i>	Pointer to data.
<i>in</i>	<i>dataLen</i>	length in bytes of data.

Return values

<i>sc32</i>	according to IEC 61375-2-3
-------------	----------------------------

5.32.2.8 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.

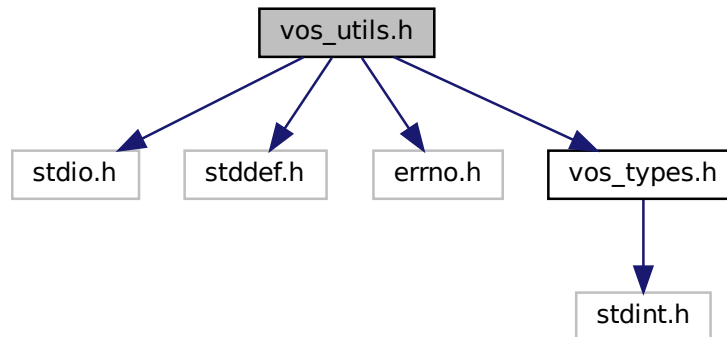
5.33 vos_utils.h File Reference

Typedefs for OS abstraction.

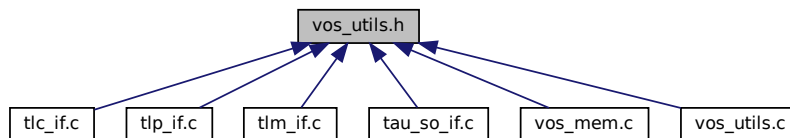
```
#include <stdio.h>
#include <stddef.h>
#include <errno.h>
```

```
#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_SET_ERRNO(VAL) errno = (VAL)`
This is the abstraction to set `errno`.
- `#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

- int `vos_hostIsBigEndian` (void)
Return 1 if this is a big endian machine.
- UINT32 `vos_crc32` (UINT32 crc, const UINT8 *pData, UINT32 dataLen)
Calculate CRC for the given buffer and length.
- 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.
- `VOS_ERR_T` `vos_init` (void *pRefCon, `VOS_PRINT_DBG_T` pDebugOutput)
Initialize the vos library.
- void `vos_terminate` (void)
DeInitialize the vos library.
- const CHAR8 * `vos_getVersionString` (void)
Return a human readable version representation.
- const `VOS_VERSION_T` * `vos_getVersion` (void)
Return version.
- const CHAR8 * `vos_getErrorString` (`VOS_ERR_T` error)
Return a human readable error representation.

5.33.1 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.

5.33.2 Macro Definition Documentation

5.33.2.1 INITFCS

```
#define INITFCS 0xffffffffu
```

CRC/FCS constants.

Initial FCS value

5.33.2.2 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

5.33.2.3 VOS_MAX_FRMT_SIZE

```
#define VOS_MAX_FRMT_SIZE 64u
```

Max.

size of the 'format' part

5.33.2.4 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

5.33.3 Function Documentation

5.33.3.1 vos_crc32()

```
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

<i>in</i>	<i>crc</i>	Initial value.
<i>in, out</i>	<i>pData</i>	Pointer to data.
<i>in</i>	<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

<i>in</i>	<i>crc</i>	Initial value.
<i>in, out</i>	<i>pData</i>	Pointer to data.
<i>in</i>	<i>dataLen</i>	length in bytes of data.

Return values

<i>crc32</i>	according to IEEE802.3
--------------	------------------------

5.33.3.2 vos_getErrorString()

```
const CHAR8* vos_getErrorString (
    VOS_ERR_T error )
```

Return a human readable error representation.

Parameters

<i>in</i>	<i>error</i>	The TRDP or VOS error code
-----------	--------------	----------------------------

Return values

<i>const</i>	string pointer to error string
--------------	--------------------------------

5.33.3.3 vos_getVersion()

```
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	
-------------------------------	--

5.33.3.4 vos_getVersionString()

```
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
--------------	--------

5.33.3.5 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

5.33.3.6 vos_init()

```
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 VOS_INTEGRATION_ERR if endianness/alignment mismatch VOS_SOCK_ERR sockets not supported VOS_UNKNOWN_ERR initialisation error
-------------------	--

5.33.3.7 vos_sc32()

```
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
-------------	----------------------------

5.33.3.8 vos_terminate()

```
void vos_terminate (  
    void )
```

Deinitialize the vos library.

Should be called last after TRDP stack/application does not use any VOS function anymore.

Index

- cnCnt
 - TRDP_ETB_INFO_T, [29](#)
- cnId
 - TRDP_FUNCTION_INFO_T, [30](#)
- cstId
 - TRDP_CONSIST_INFO_T, [21](#)
- cstInfoGetPropSize
 - tau_cstinfo.c, [76](#)
- cstList
 - TRDP_CSTINFOCTRL_T, [22](#)
- cstOwner
 - TRDP_CONSIST_INFO_T, [21](#)
- cstUUID
 - TRDP_OP_CONSIST_T, [40](#)
- cstVehNo
 - TRDP_FUNCTION_INFO_T, [30](#)
- ETB_CTRL_COMID
 - iec61375-2-3.h, [71](#)
- etbId
 - TRDP_FUNCTION_INFO_T, [30](#)
- fctDev
 - SRM_SERVICE_INFO_T, [14](#)
- fctId
 - TRDP_FUNCTION_INFO_T, [31](#)
- iec61375-2-3.h, [65](#)
 - ETB_CTRL_COMID, [71](#)
 - TRDP_ETBCTRL_DSID, [71](#)
 - TRDP_EXTRA_LABEL_LEN, [71](#)
 - TRDP_MAX_FILE_NAME_LEN, [71](#)
 - TRDP_MAX_LABEL_LEN, [71](#)
 - TRDP_MAX_MD_DATA_SIZE, [72](#)
 - TRDP_MAX_URI_HOST_LEN, [72](#)
 - TRDP_MAX_URI_LEN, [72](#)
 - TRDP_MAX_URI_USER_LEN, [72](#)
 - TRDP_MD_DEFAULT_REPLY_TIMEOUT, [72](#)
 - TRDP_MD_INFINITE_TIME, [72](#)
 - TRDP_MIN_PD_HEADER_SIZE, [73](#)
 - TRDP_MSG_PT, [73](#)
 - TRDP_PD_UDP_PORT, [73](#)
 - TRDP_PROCESS_DEFAULT_CYCLE_TIME, [73](#)
 - TRDP_PROTOCOL_VERSION_CHECK_MASK, [73](#)
 - TRDP_USR_URI_SIZE, [74](#)
 - TTDB_NET_DIR_REQ_COMID, [74](#)
 - TTDB_OP_DIR_INFO_COMID, [74](#)
 - TTDB_STAT_CST_REQ_COMID, [74](#)
 - TTDB_TRN_DIR_REQ_COMID, [74](#)
- INITFCS
 - vos_utils.h, [325](#)
- maxNoOfExtPublishers
 - TRDP_IDX_TABLE_T, [32](#)
- maxNoOfHighCatPublishers
 - TRDP_IDX_TABLE_T, [32](#)
- maxNoOfHighCatSubscriptions
 - TRDP_IDX_TABLE_T, [32](#)
- maxNoOfLowCatPublishers
 - TRDP_IDX_TABLE_T, [32](#)
- maxNoOfLowCatSubscriptions
 - TRDP_IDX_TABLE_T, [32](#)
- maxNoOfMidCatPublishers
 - TRDP_IDX_TABLE_T, [32](#)
- maxNoOfMidCatSubscriptions
 - TRDP_IDX_TABLE_T, [33](#)
- opCstList
 - TRDP_OP_TRAIN_DIR_T, [44](#)
- opVehList
 - TRDP_OP_TRAIN_DIR_T, [44](#)
- SOA_SAME_SERVICEID
 - trdp_serviceRegistry.h, [238](#)
- SRM_SERVICE_INFO_T, [13](#)
 - fctDev, [14](#)
- SRM_SERVICE_READ_REQ_COMID
 - trdp_serviceRegistry.h, [238](#)
- SRM_SRV_INFO_REQ_T, [15](#)
- SRM_SRVINFO_NOTIFY_COMID
 - trdp_serviceRegistry.h, [238](#)
- tau_addr2Uri
 - tau_dnr.c, [91](#)
 - tau_dnr.h, [97](#)
- tau_addService
 - tau_so_if.c, [120](#)
 - tau_so_if.h, [125](#)
- tau_calcDatasetSize
 - tau_marshall.c, [104](#)
 - tau_marshall.h, [111](#)
- tau_calcDatasetSizeByComId
 - tau_marshall.c, [105](#)
 - tau_marshall.h, [112](#)
- tau_copyCstInfo
 - tau_tti.c, [130](#)
- tau_cstinfo.c, [75](#)
 - cstInfoGetPropSize, [76](#)
- tau_ctrl.c, [77](#)

- tau_getEcspStat, 78
- tau_initEcspCtrl, 79
- tau_requestEcspConfirm, 79
- tau_requestEcspConfirmReply, 80
- tau_setEcspCtrl, 80
- tau_terminateEcspCtrl, 81
- tau_ctrl.h, 81
 - tau_getEcspStat, 83
 - tau_initEcspCtrl, 84
 - tau_requestEcspConfirm, 84
 - tau_requestEcspConfirmReply, 85
 - tau_setEcspCtrl, 86
 - tau_terminateEcspCtrl, 86
- tau_ctrl_types.h, 87
- tau_delnitDnr
 - tau_dnr.c, 92
 - tau_dnr.h, 97
- tau_delnitTTI
 - tau_tti.c, 130
 - tau_tti.h, 142
- tau_delService
 - tau_so_if.c, 121
 - tau_so_if.h, 125
- tau_dnr.c, 89
 - tau_addr2Uri, 91
 - tau_delnitDnr, 92
 - tau_DNRstatus, 92
 - tau_getOwnAddr, 92
 - tau_initDnr, 93
 - tau_uri2Addr, 94
- tau_dnr.h, 94
 - tau_addr2Uri, 97
 - tau_delnitDnr, 97
 - tau_DNRstatus, 98
 - tau_getOwnAddr, 98
 - tau_initDnr, 99
 - tau_uri2Addr, 100
 - TRDP_DNR_OPTS_T, 96
 - TRDP_DNR_OWN_THREAD, 97
- tau_dnr_types.h, 101
- tau_DNRstatus
 - tau_dnr.c, 92
 - tau_dnr.h, 98
- TAU_DNS_HEADER_T, 16
- tau_freeServicesList
 - tau_so_if.c, 121
 - tau_so_if.h, 126
- tau_freeTelegrams
 - tau_xml.c, 159
 - tau_xml.h, 169
- tau_freeXmlDatasetConfig
 - tau_xml.c, 159
 - tau_xml.h, 169
- tau_freeXmlDoc
 - tau_xml.c, 160
 - tau_xml.h, 170
- tau_getCstFctCnt
 - tau_tti.c, 130
- tau_tti.h, 143
- tau_getCstFctInfo
 - tau_tti.c, 131
 - tau_tti.h, 143
- tau_getCstInfo
 - tau_tti.c, 131
 - tau_tti.h, 144
- tau_getCstVehCnt
 - tau_tti.c, 133
 - tau_tti.h, 145
- tau_getEcspStat
 - tau_ctrl.c, 78
 - tau_ctrl.h, 83
- tau_getOpTrnDirectory
 - tau_tti.c, 133
 - tau_tti.h, 145
- tau_getOpTrnDirectoryStatusInfo
 - tau_tti.c, 134
 - tau_tti.h, 146
- tau_getOwnAddr
 - tau_dnr.c, 92
 - tau_dnr.h, 98
- tau_getOwnlds
 - tau_tti.c, 134
 - tau_tti.h, 147
- tau_getOwnOpCstNo
 - tau_tti.c, 135
 - tau_tti.h, 147
- tau_getOwnTrnCstNo
 - tau_tti.c, 135
 - tau_tti.h, 148
- tau_getServicesList
 - tau_so_if.c, 121
 - tau_so_if.h, 126
- tau_getStaticCstInfo
 - tau_tti.c, 136
 - tau_tti.h, 148
- tau_getTrnCstCnt
 - tau_tti.c, 136
 - tau_tti.h, 149
- tau_getTrnDirectory
 - tau_tti.c, 136
 - tau_tti.h, 149
- tau_getTrnVehCnt
 - tau_tti.c, 137
 - tau_tti.h, 150
- tau_getTTI
 - tau_tti.c, 137
 - tau_tti.h, 150
- tau_getVehInfo
 - tau_tti.c, 138
 - tau_tti.h, 151
- tau_getVehOrient
 - tau_tti.c, 138
 - tau_tti.h, 152
- tau_initDnr
 - tau_dnr.c, 93
 - tau_dnr.h, 99

- tau_initEcspCtrl
 - tau_ctrl.c, [79](#)
 - tau_ctrl.h, [84](#)
- tau_initMarshall
 - tau_marshall.c, [106](#)
 - tau_marshall.h, [113](#)
- tau_initTTIaccess
 - tau_tti.c, [139](#)
 - tau_tti.h, [153](#)
- tau_marshall
 - tau_marshall.c, [107](#)
 - tau_marshall.h, [114](#)
- tau_marshall.c, [103](#)
 - tau_calcDatasetSize, [104](#)
 - tau_calcDatasetSizeByComId, [105](#)
 - tau_initMarshall, [106](#)
 - tau_marshall, [107](#)
 - tau_marshallDs, [107](#)
 - tau_unmarshall, [108](#)
 - tau_unmarshallDs, [109](#)
- tau_marshall.h, [109](#)
 - tau_calcDatasetSize, [111](#)
 - tau_calcDatasetSizeByComId, [112](#)
 - tau_initMarshall, [113](#)
 - tau_marshall, [114](#)
 - tau_marshallDs, [115](#)
 - tau_unmarshall, [116](#)
 - tau_unmarshallDs, [117](#)
- TAU_MARSHALL_INFO_T, [16](#)
- tau_marshallDs
 - tau_marshall.c, [107](#)
 - tau_marshall.h, [115](#)
- tau_prepareXmlDoc
 - tau_xml.c, [160](#)
 - tau_xml.h, [170](#)
- tau_prepareXmlMem
 - tau_xml.c, [160](#)
 - tau_xml.h, [171](#)
- tau_readXmlDatasetConfig
 - tau_xml.c, [161](#)
 - tau_xml.h, [171](#)
- tau_readXmlDeviceConfig
 - tau_xml.c, [161](#)
 - tau_xml.h, [172](#)
- tau_readXmlInterfaceConfig
 - tau_xml.c, [162](#)
 - tau_xml.h, [173](#)
- tau_readXmlMappedDeviceConfig
 - tau_xml.c, [163](#)
 - tau_xml.h, [173](#)
- tau_readXmlMappedDevices
 - tau_xml.c, [163](#)
 - tau_xml.h, [174](#)
- tau_readXmlMappedInterfaceConfig
 - tau_xml.c, [164](#)
 - tau_xml.h, [174](#)
- tau_readXmlServiceConfig
 - tau_xml.c, [164](#)
 - tau_xml.h, [175](#)
- tau_requestEcspConfirm
 - tau_ctrl.c, [79](#)
 - tau_ctrl.h, [84](#)
- tau_requestEcspConfirmReply
 - tau_ctrl.c, [80](#)
 - tau_ctrl.h, [85](#)
- tau_setEcspCtrl
 - tau_ctrl.c, [80](#)
 - tau_ctrl.h, [86](#)
- tau_so_if.c, [119](#)
 - tau_addService, [120](#)
 - tau_delService, [121](#)
 - tau_freeServicesList, [121](#)
 - tau_getServicesList, [121](#)
 - tau_updService, [122](#)
- tau_so_if.h, [123](#)
 - tau_addService, [125](#)
 - tau_delService, [125](#)
 - tau_freeServicesList, [126](#)
 - tau_getServicesList, [126](#)
 - tau_updService, [127](#)
- tau_terminateEcspCtrl
 - tau_ctrl.c, [81](#)
 - tau_ctrl.h, [86](#)
- tau_tti.c, [127](#)
 - tau_copyCstInfo, [130](#)
 - tau_delnitTTI, [130](#)
 - tau_getCstFctCnt, [130](#)
 - tau_getCstFctInfo, [131](#)
 - tau_getCstInfo, [131](#)
 - tau_getCstVehCnt, [133](#)
 - tau_getOpTrnDirectory, [133](#)
 - tau_getOpTrnDirectoryStatusInfo, [134](#)
 - tau_getOwnIds, [134](#)
 - tau_getOwnOpCstNo, [135](#)
 - tau_getOwnTrnCstNo, [135](#)
 - tau_getStaticCstInfo, [136](#)
 - tau_getTrnCstCnt, [136](#)
 - tau_getTrnDirectory, [136](#)
 - tau_getTrnVehCnt, [137](#)
 - tau_getTTI, [137](#)
 - tau_getVehInfo, [138](#)
 - tau_getVehOrient, [138](#)
 - tau_initTTIaccess, [139](#)
- tau_tti.h, [140](#)
 - tau_delnitTTI, [142](#)
 - tau_getCstFctCnt, [143](#)
 - tau_getCstFctInfo, [143](#)
 - tau_getCstInfo, [144](#)
 - tau_getCstVehCnt, [145](#)
 - tau_getOpTrnDirectory, [145](#)
 - tau_getOpTrnDirectoryStatusInfo, [146](#)
 - tau_getOwnIds, [147](#)
 - tau_getOwnOpCstNo, [147](#)
 - tau_getOwnTrnCstNo, [148](#)
 - tau_getStaticCstInfo, [148](#)
 - tau_getTrnCstCnt, [149](#)

- tau_getTrnDirectory, 149
- tau_getTrnVehCnt, 150
- tau_getTTI, 150
- tau_getVehInfo, 151
- tau_getVehOrient, 152
- tau_initTTIaccess, 153
- tau_tti_types.h, 153
- tau_unmarshall
 - tau_marshall.c, 108
 - tau_marshall.h, 116
- tau_unmarshallDs
 - tau_marshall.c, 109
 - tau_marshall.h, 117
- tau_updService
 - tau_so_if.c, 122
 - tau_so_if.h, 127
- tau_uri2Addr
 - tau_dnr.c, 94
 - tau_dnr.h, 100
- tau_xml.c, 156
 - tau_freeTelegrams, 159
 - tau_freeXmlDatasetConfig, 159
 - tau_freeXmlDoc, 160
 - tau_prepareXmlDoc, 160
 - tau_prepareXmlMem, 160
 - tau_readXmlDatasetConfig, 161
 - tau_readXmlDeviceConfig, 161
 - tau_readXmlInterfaceConfig, 162
 - tau_readXmlMappedDeviceConfig, 163
 - tau_readXmlMappedDevices, 163
 - tau_readXmlMappedInterfaceConfig, 164
 - tau_readXmlServiceConfig, 164
 - TRDP_SDT_DEFAULT_CMTHR, 158
 - TRDP_SDT_DEFAULT_LMIMAX, 159
- tau_xml.h, 165
 - tau_freeTelegrams, 169
 - tau_freeXmlDatasetConfig, 169
 - tau_freeXmlDoc, 170
 - tau_prepareXmlDoc, 170
 - tau_prepareXmlMem, 171
 - tau_readXmlDatasetConfig, 171
 - tau_readXmlDeviceConfig, 172
 - tau_readXmlInterfaceConfig, 173
 - tau_readXmlMappedDeviceConfig, 173
 - tau_readXmlMappedDevices, 174
 - tau_readXmlMappedInterfaceConfig, 174
 - tau_readXmlServiceConfig, 175
 - TRDP_DBG_DEFAULT, 168
 - TRDP_EXCHG_OPTION_T, 168
 - TRDP_EXCHG_SINK, 169
 - TRDP_EXCHG_SOURCE, 169
 - TRDP_EXCHG_SOURCESINK, 169
 - TRDP_EXCHG_UNSET, 169
- TCN_URI_T, 16
- tcnUriCnt
 - TRDP_DNS_REPLY_T, 26
 - TRDP_DNS_REQUEST_T, 28
- timeout
- TRDP_SUBS_STATISTICS_T, 56
- tlc_closeSession
 - tlc_if.c, 178
 - trdp_if_light.h, 213
- tlc_configSession
 - tlc_if.c, 178
 - trdp_if_light.h, 213
- tlc_getETBTopoCount
 - tlc_if.c, 179
 - trdp_if_light.h, 214
- tlc_getInterval
 - tlc_if.c, 179
 - trdp_if_light.h, 214
- tlc_getJoinStatistics
 - trdp_if_light.h, 215
 - trdp_stats.c, 240
- tlc_getOpTrainTopoCount
 - tlc_if.c, 180
 - trdp_if_light.h, 215
- tlc_getOwnIpAddress
 - tlc_if.c, 180
 - trdp_if_light.h, 216
- tlc_getPubStatistics
 - trdp_if_light.h, 216
 - trdp_stats.c, 240
- tlc_getRedStatistics
 - trdp_if_light.h, 217
 - trdp_stats.c, 241
- tlc_getStatistics
 - trdp_if_light.h, 217
 - trdp_stats.c, 242
- tlc_getSubsStatistics
 - trdp_if_light.h, 218
 - trdp_stats.c, 242
- tlc_getVersion
 - tlc_if.c, 180
 - trdp_if_light.h, 218
- tlc_getVersionString
 - tlc_if.c, 181
 - trdp_if_light.h, 219
- tlc_if.c, 176
 - tlc_closeSession, 178
 - tlc_configSession, 178
 - tlc_getETBTopoCount, 179
 - tlc_getInterval, 179
 - tlc_getOpTrainTopoCount, 180
 - tlc_getOwnIpAddress, 180
 - tlc_getVersion, 180
 - tlc_getVersionString, 181
 - tlc_init, 181
 - tlc_openSession, 182
 - tlc_presetIndexSession, 182
 - tlc_process, 183
 - tlc_reinitSession, 183
 - tlc_setETBTopoCount, 184
 - tlc_setOpTrainTopoCount, 184
 - tlc_terminate, 185
 - tlc_updateSession, 185

- trdp_getAccess, 186
- trdp_isValidSession, 186
- trdp_releaseAccess, 186
- trdp_sessionQueue, 187
- tlc_init
 - tlc_if.c, 181
 - trdp_if_light.h, 219
- tlc_openSession
 - tlc_if.c, 182
 - trdp_if_light.h, 219
- tlc_presetIndexSession
 - tlc_if.c, 182
 - trdp_if_light.h, 220
- tlc_process
 - tlc_if.c, 183
 - trdp_if_light.h, 221
- tlc_reinitSession
 - tlc_if.c, 183
 - trdp_if_light.h, 221
- tlc_resetStatistics
 - trdp_if_light.h, 222
 - trdp_stats.c, 243
- tlc_setETBTopoCount
 - tlc_if.c, 184
 - trdp_if_light.h, 222
- tlc_setOpTrainTopoCount
 - tlc_if.c, 184
 - trdp_if_light.h, 223
- tlc_terminate
 - tlc_if.c, 185
 - trdp_if_light.h, 223
- tlc_updateSession
 - tlc_if.c, 185
 - trdp_if_light.h, 223
- tlm_abortSession
 - tlm_if.c, 189
- tlm_addListener
 - tlm_if.c, 190
- tlm_confirm
 - tlm_if.c, 191
- tlm_delListener
 - tlm_if.c, 191
- tlm_getInterval
 - tlm_if.c, 192
- tlm_if.c, 187
 - tlm_abortSession, 189
 - tlm_addListener, 190
 - tlm_confirm, 191
 - tlm_delListener, 191
 - tlm_getInterval, 192
 - tlm_notify, 192
 - tlm_process, 193
 - tlm_readdListener, 194
 - tlm_reply, 194
 - tlm_replyQuery, 195
 - tlm_request, 196
- tlm_notify
 - tlm_if.c, 192
- tlm_process
 - tlm_if.c, 193
- tlm_readdListener
 - tlm_if.c, 194
- tlm_reply
 - tlm_if.c, 194
- tlm_replyQuery
 - tlm_if.c, 195
- tlm_request
 - tlm_if.c, 196
- tlp_get
 - tlp_if.c, 199
 - trdp_if_light.h, 224
- tlp_getInterval
 - tlp_if.c, 200
 - trdp_if_light.h, 225
- tlp_getRedundant
 - tlp_if.c, 200
 - trdp_if_light.h, 225
- tlp_if.c, 197
 - tlp_get, 199
 - tlp_getInterval, 200
 - tlp_getRedundant, 200
 - tlp_processReceive, 201
 - tlp_processSend, 201
 - tlp_publish, 202
 - tlp_put, 203
 - tlp_putImmediate, 203
 - tlp_republish, 204
 - tlp_request, 205
 - tlp_resubscribe, 206
 - tlp_setRedundant, 207
 - tlp_subscribe, 207
 - tlp_unpublish, 208
 - tlp_unsubscribe, 209
- tlp_processReceive
 - tlp_if.c, 201
 - trdp_if_light.h, 226
- tlp_processSend
 - tlp_if.c, 201
 - trdp_if_light.h, 226
- tlp_publish
 - tlp_if.c, 202
 - trdp_if_light.h, 227
- tlp_put
 - tlp_if.c, 203
 - trdp_if_light.h, 228
- tlp_putImmediate
 - tlp_if.c, 203
 - trdp_if_light.h, 228
- tlp_republish
 - tlp_if.c, 204
 - trdp_if_light.h, 229
- tlp_request
 - tlp_if.c, 205
 - trdp_if_light.h, 230
- tlp_resubscribe
 - tlp_if.c, 206

- trdp_if_light.h, 231
- tlp_setRedundant
 - tlp_if.c, 207
 - trdp_if_light.h, 232
- tlp_subscribe
 - tlp_if.c, 207
 - trdp_if_light.h, 232
- tlp_unpublish
 - tlp_if.c, 208
 - trdp_if_light.h, 233
- tlp_unsubscribe
 - tlp_if.c, 209
 - trdp_if_light.h, 234
- toBehav
 - TRDP_SUBS_STATISTICS_T, 57
- TRDP_APP_CONFIRMTO_ERR
 - trdp_types.h, 256
- TRDP_APP_REPLYTO_ERR
 - trdp_types.h, 256
- TRDP_APP_TIMEOUT_ERR
 - trdp_types.h, 256
- TRDP_BITSET8
 - trdp_types.h, 255
- TRDP_BLOCK_ERR
 - trdp_types.h, 256
- TRDP_CHAR8
 - trdp_types.h, 255
- TRDP_CLTR_CST_INFO_T, 17
- TRDP_COM_PARAM_T, 18
- TRDP_COMID_DSID_MAP_T, 18
- TRDP_COMID_ERR
 - trdp_types.h, 256
- TRDP_CONF_VEHICLE_T, 19
- TRDP_CONFIRMTO_ERR
 - trdp_types.h, 257
- TRDP_CONSIST_INFO_T, 19
 - cstId, 21
 - cstOwner, 21
- TRDP_CRC_ERR
 - trdp_types.h, 256
- TRDP_CSTINFOCTRL_T, 21
 - cstList, 22
 - version, 22
- TRDP_DATA_TYPE_T
 - trdp_types.h, 254
- TRDP_DATASET_ELEMENT_T, 23
- TRDP_DATASET_T, 24
- TRDP_DBG_CONFIG_T, 25
- TRDP_DBG_DEFAULT
 - tau_xml.h, 168
- TRDP_DNR_OPTS_T
 - tau_dnr.h, 96
- TRDP_DNR_OWN_THREAD
 - tau_dnr.h, 97
- TRDP_DNS_REPLY_T, 25
 - tcnUriCnt, 26
- TRDP_DNS_REQUEST_T, 27
 - tcnUriCnt, 28
- TRDP_ERR_T
 - trdp_types.h, 255
- TRDP_ETB_CTRL_VDP_T, 28
- TRDP_ETB_INFO_T, 29
 - cnCnt, 29
- TRDP_ETBCTRL_DSID
 - iec61375-2-3.h, 71
- TRDP_EXCHG_OPTION_T
 - tau_xml.h, 168
- TRDP_EXCHG_SINK
 - tau_xml.h, 169
- TRDP_EXCHG_SOURCE
 - tau_xml.h, 169
- TRDP_EXCHG_SOURCESINK
 - tau_xml.h, 169
- TRDP_EXCHG_UNSET
 - tau_xml.h, 169
- TRDP_EXTRA_LABEL_LEN
 - iec61375-2-3.h, 71
- TRDP_FLAGS_DEFAULT
 - trdp_types.h, 251
- TRDP_FUNCTION_INFO_T, 30
 - cnId, 30
 - cstVehNo, 30
 - etbId, 30
 - fctId, 31
- trdp_getAccess
 - tlc_if.c, 186
- TRDP_IDX_TABLE_T, 31
 - maxNoOfExtPublishers, 32
 - maxNoOfHighCatPublishers, 32
 - maxNoOfHighCatSubscriptions, 32
 - maxNoOfLowCatPublishers, 32
 - maxNoOfLowCatSubscriptions, 32
 - maxNoOfMidCatPublishers, 32
 - maxNoOfMidCatSubscriptions, 33
- trdp_if_light.h, 209
 - tlc_closeSession, 213
 - tlc_configSession, 213
 - tlc_getETBTopoCount, 214
 - tlc_getInterval, 214
 - tlc_getJoinStatistics, 215
 - tlc_getOpTrainTopoCount, 215
 - tlc_getOwnIpAddress, 216
 - tlc_getPubStatistics, 216
 - tlc_getRedStatistics, 217
 - tlc_getStatistics, 217
 - tlc_getSubsStatistics, 218
 - tlc_getVersion, 218
 - tlc_getVersionString, 219
 - tlc_init, 219
 - tlc_openSession, 219
 - tlc_presetIndexSession, 220
 - tlc_process, 221
 - tlc_reinitSession, 221
 - tlc_resetStatistics, 222
 - tlc_setETBTopoCount, 222
 - tlc_setOpTrainTopoCount, 223

- tlc_terminate, [223](#)
- tlc_updateSession, [223](#)
- tlp_get, [224](#)
- tlp_getInterval, [225](#)
- tlp_getRedundant, [225](#)
- tlp_processReceive, [226](#)
- tlp_processSend, [226](#)
- tlp_publish, [227](#)
- tlp_put, [228](#)
- tlp_putImmediate, [228](#)
- tlp_republish, [229](#)
- tlp_request, [230](#)
- tlp_resubscribe, [231](#)
- tlp_setRedundant, [232](#)
- tlp_subscribe, [232](#)
- tlp_unpublish, [233](#)
- tlp_unsubscribe, [234](#)
- TRDP_INIT_ERR
 - trdp_types.h, [255](#)
- trdp_initStats
 - trdp_stats.c, [243](#)
- TRDP_INT16
 - trdp_types.h, [255](#)
- TRDP_INT32
 - trdp_types.h, [255](#)
- TRDP_INT64
 - trdp_types.h, [255](#)
- TRDP_INT8
 - trdp_types.h, [255](#)
- TRDP_INTEGRATION_ERR
 - trdp_types.h, [256](#)
- TRDP_INUSE_ERR
 - trdp_types.h, [257](#)
- TRDP_INVALID
 - trdp_types.h, [255](#)
- TRDP_IO_ERR
 - trdp_types.h, [256](#)
- TRDP_IP_ADDR_T
 - trdp_types.h, [252](#)
- trdp_isValidSession
 - tlc_if.c, [186](#)
- TRDP_LIST_STATISTICS_T, [33](#)
- TRDP_MARSHALL_CONFIG_T, [34](#)
- TRDP_MARSHALL_T
 - trdp_types.h, [252](#)
- TRDP_MARSHALLING_ERR
 - trdp_types.h, [257](#)
- TRDP_MAX_FILE_NAME_LEN
 - iec61375-2-3.h, [71](#)
- TRDP_MAX_LABEL_LEN
 - iec61375-2-3.h, [71](#)
- TRDP_MAX_MD_DATA_SIZE
 - iec61375-2-3.h, [72](#)
- TRDP_MAX_URI_HOST_LEN
 - iec61375-2-3.h, [72](#)
- TRDP_MAX_URI_LEN
 - iec61375-2-3.h, [72](#)
- TRDP_MAX_URI_USER_LEN
 - iec61375-2-3.h, [72](#)
- TRDP_MD_CALLBACK_T
 - trdp_types.h, [253](#)
- TRDP_MD_CONFIG_T, [35](#)
- TRDP_MD_DEFAULT_REPLY_TIMEOUT
 - iec61375-2-3.h, [72](#)
- TRDP_MD_INFINITE_TIME
 - iec61375-2-3.h, [72](#)
- TRDP_MD_INFO_T, [36](#)
- TRDP_MD_STATISTICS_T, [38](#)
- TRDP_MEM_CONFIG_T, [39](#)
- TRDP_MEM_ERR
 - trdp_types.h, [256](#)
- TRDP_MIN_PD_HEADER_SIZE
 - iec61375-2-3.h, [73](#)
- TRDP_MSG_PT
 - iec61375-2-3.h, [73](#)
- TRDP_MUTEX_ERR
 - trdp_types.h, [256](#)
- TRDP_NO_ERR
 - trdp_types.h, [255](#)
- TRDP_NOCONN_ERR
 - trdp_types.h, [256](#)
- TRDP_NODATA_ERR
 - trdp_types.h, [256](#)
- TRDP_NOINIT_ERR
 - trdp_types.h, [256](#)
- TRDP_NOLIST_ERR
 - trdp_types.h, [256](#)
- TRDP_NOPUB_ERR
 - trdp_types.h, [256](#)
- TRDP_NOSESSION_ERR
 - trdp_types.h, [256](#)
- TRDP_NOSUB_ERR
 - trdp_types.h, [256](#)
- TRDP_OP_CONSIST_T, [39](#)
 - cstUUID, [40](#)
- TRDP_OP_TRAIN_DIR_STATE_T, [40](#)
 - trnId, [41](#)
 - trnOperator, [41](#)
- TRDP_OP_TRAIN_DIR_STATUS_INFO_T, [42](#)
- TRDP_OP_TRAIN_DIR_T, [43](#)
 - opCstList, [44](#)
 - opVehList, [44](#)
- TRDP_OP_VEHICLE_T, [44](#)
 - vehId, [45](#)
- TRDP_PACKET_ERR
 - trdp_types.h, [257](#)
- TRDP_PARAM_ERR
 - trdp_types.h, [255](#)
- TRDP_PD_CALLBACK_T
 - trdp_types.h, [253](#)
- TRDP_PD_CONFIG_T, [45](#)
- TRDP_PD_DEFAULT_TSN_PRIORITY
 - trdp_tsn_def.h, [246](#)
- TRDP_PD_INFO_T, [46](#)
- TRDP_PD_STATISTICS_T, [48](#)
- TRDP_PD_UDP_PORT

- iec61375-2-3.h, [73](#)
- trdp_pdPrepareStats
 - trdp_stats.c, [243](#)
- TRDP_PRINT_DBG_T
 - trdp_types.h, [253](#)
- TRDP_PROCESS_CONFIG_T, [49](#)
- TRDP_PROCESS_DEFAULT_CYCLE_TIME
 - iec61375-2-3.h, [73](#)
- TRDP_PROP_T, [49](#)
- TRDP_PROTOCOL_VERSION_CHECK_MASK
 - iec61375-2-3.h, [73](#)
- TRDP_PUB_STATISTICS_T, [50](#)
- TRDP_QUEUE_ERR
 - trdp_types.h, [256](#)
- TRDP_QUEUE_FULL_ERR
 - trdp_types.h, [256](#)
- TRDP_READ_COMPLETE_REPLY_T, [51](#)
- TRDP_REAL32
 - trdp_types.h, [255](#)
- TRDP_REAL64
 - trdp_types.h, [255](#)
- TRDP_RED_FOLLOWER
 - trdp_types.h, [257](#)
- TRDP_RED_LEADER
 - trdp_types.h, [257](#)
- TRDP_RED_STATE_T
 - trdp_types.h, [257](#)
- TRDP_RED_STATISTICS_T, [52](#)
- trdp_releaseAccess
 - tlc_if.c, [186](#)
- TRDP_REPLY_STATUS_T
 - trdp_types.h, [257](#)
- TRDP_REPLYTO_ERR
 - trdp_types.h, [256](#)
- TRDP_REQCONFIRMTO_ERR
 - trdp_types.h, [257](#)
- TRDP_SDT_DEFAULT_CMTHR
 - tau_xml.c, [158](#)
- TRDP_SDT_DEFAULT_LMIMAX
 - tau_xml.c, [159](#)
- TRDP_SDT_PAR_T, [53](#)
- TRDP_SEMA_ERR
 - trdp_types.h, [256](#)
- trdp_serviceRegistry.h, [234](#)
 - SOA_SAME_SERVICEID, [238](#)
 - SRM_SERVICE_READ_REQ_COMID, [238](#)
 - SRM_SRVINFO_NOTIFY_COMID, [238](#)
- TRDP_SESSION_ABORT_ERR
 - trdp_types.h, [256](#)
- trdp_sessionQueue
 - tlc_if.c, [187](#)
- TRDP_SHORT_VERSION_T, [53](#)
- TRDP_SOCKET_ERR
 - trdp_types.h, [256](#)
- TRDP_STATE_ERR
 - trdp_types.h, [256](#)
- TRDP_STATISTICS_REQUEST_T, [54](#)
- TRDP_STATISTICS_T, [55](#)
- trdp_stats.c, [239](#)
 - tlc_getJoinStatistics, [240](#)
 - tlc_getPubStatistics, [240](#)
 - tlc_getRedStatistics, [241](#)
 - tlc_getStatistics, [242](#)
 - tlc_getSubsStatistics, [242](#)
 - tlc_resetStatistics, [243](#)
 - trdp_initStats, [243](#)
 - trdp_pdPrepareStats, [243](#)
 - trdp_UpdateStats, [244](#)
- TRDP_SUBS_STATISTICS_T, [56](#)
 - timeout, [56](#)
 - toBehav, [57](#)
- TRDP_THREAD_ERR
 - trdp_types.h, [256](#)
- TRDP_TIME_T
 - trdp_types.h, [254](#)
- TRDP_TIMEDATE32
 - trdp_types.h, [255](#)
- TRDP_TIMEDATE48
 - trdp_types.h, [255](#)
- TRDP_TIMEDATE64
 - trdp_types.h, [255](#)
- TRDP_TIMEOUT_ERR
 - trdp_types.h, [256](#)
- TRDP_TO_BEHAVIOR_T
 - trdp_types.h, [257](#)
- TRDP_TO_DEFAULT
 - trdp_types.h, [258](#)
- TRDP_TO_KEEP_LAST_VALUE
 - trdp_types.h, [258](#)
- TRDP_TO_SET_TO_ZERO
 - trdp_types.h, [258](#)
- TRDP_TOPO_ERR
 - trdp_types.h, [256](#)
- TRDP_TRAIN_DIR_T, [57](#)
- TRDP_TRAIN_NET_DIR_ENTRY_T, [58](#)
- TRDP_TRAIN_NET_DIR_T, [59](#)
- trdp_tsn_def.h, [245](#)
 - TRDP_PD_DEFAULT_TSN_PRIORITY, [246](#)
- TRDP_TYPE_MAX
 - trdp_types.h, [255](#)
- trdp_types.h, [246](#)
 - TRDP_APP_CONFIRMTO_ERR, [256](#)
 - TRDP_APP_REPLYTO_ERR, [256](#)
 - TRDP_APP_TIMEOUT_ERR, [256](#)
 - TRDP_BITSET8, [255](#)
 - TRDP_BLOCK_ERR, [256](#)
 - TRDP_CHAR8, [255](#)
 - TRDP_COMID_ERR, [256](#)
 - TRDP_CONFIRMTO_ERR, [257](#)
 - TRDP_CRC_ERR, [256](#)
 - TRDP_DATA_TYPE_T, [254](#)
 - TRDP_ERR_T, [255](#)
 - TRDP_FLAGS_DEFAULT, [251](#)
 - TRDP_INIT_ERR, [255](#)
 - TRDP_INT16, [255](#)
 - TRDP_INT32, [255](#)

- TRDP_INT64, 255
- TRDP_INT8, 255
- TRDP_INTEGRATION_ERR, 256
- TRDP_INUSE_ERR, 257
- TRDP_INVALID, 255
- TRDP_IO_ERR, 256
- TRDP_IP_ADDR_T, 252
- TRDP_MARSHALL_T, 252
- TRDP_MARSHALLING_ERR, 257
- TRDP_MD_CALLBACK_T, 253
- TRDP_MEM_ERR, 256
- TRDP_MUTEX_ERR, 256
- TRDP_NO_ERR, 255
- TRDP_NOCONN_ERR, 256
- TRDP_NODATA_ERR, 256
- TRDP_NOINIT_ERR, 256
- TRDP_NOLIST_ERR, 256
- TRDP_NOPUB_ERR, 256
- TRDP_NOSESSION_ERR, 256
- TRDP_NOSUB_ERR, 256
- TRDP_PACKET_ERR, 257
- TRDP_PARAM_ERR, 255
- TRDP_PD_CALLBACK_T, 253
- TRDP_PRINT_DBG_T, 253
- TRDP_QUEUE_ERR, 256
- TRDP_QUEUE_FULL_ERR, 256
- TRDP_REAL32, 255
- TRDP_REAL64, 255
- TRDP_RED_FOLLOWER, 257
- TRDP_RED_LEADER, 257
- TRDP_RED_STATE_T, 257
- TRDP_REPLY_STATUS_T, 257
- TRDP_REPLYTO_ERR, 256
- TRDP_REQCONFIRMTO_ERR, 257
- TRDP_SEMA_ERR, 256
- TRDP_SESSION_ABORT_ERR, 256
- TRDP_SOCKET_ERR, 256
- TRDP_STATE_ERR, 256
- TRDP_THREAD_ERR, 256
- TRDP_TIME_T, 254
- TRDP_TIMEDATE32, 255
- TRDP_TIMEDATE48, 255
- TRDP_TIMEDATE64, 255
- TRDP_TIMEOUT_ERR, 256
- TRDP_TO_BEHAVIOR_T, 257
- TRDP_TO_DEFAULT, 258
- TRDP_TO_KEEP_LAST_VALUE, 258
- TRDP_TO_SET_TO_ZERO, 258
- TRDP_TOPO_ERR, 256
- TRDP_TYPE_MAX, 255
- TRDP_UINT16, 255
- TRDP_UINT32, 255
- TRDP_UINT64, 255
- TRDP_UINT8, 255
- TRDP_UNKNOWN_ERR, 257
- TRDP_UNMARSHALL_T, 254
- TRDP_UNRESOLVED_ERR, 257
- TRDP_UTF16, 255
- TRDP_WIRE_ERR, 256
- TRDP_XML_PARSER_ERR, 257
- USE_HEAP, 252
- TRDP_UINT16
 - trdp_types.h, 255
- TRDP_UINT32
 - trdp_types.h, 255
- TRDP_UINT64
 - trdp_types.h, 255
- TRDP_UINT8
 - trdp_types.h, 255
- TRDP_UNKNOWN_ERR
 - trdp_types.h, 257
- TRDP_UNMARSHALL_T
 - trdp_types.h, 254
- TRDP_UNRESOLVED_ERR
 - trdp_types.h, 257
- trdp_UpdateStats
 - trdp_stats.c, 244
- TRDP_USR_URI_SIZE
 - iec61375-2-3.h, 74
- TRDP_UTF16
 - trdp_types.h, 255
- TRDP_VEHICLE_INFO_T, 60
 - vehId, 61
- TRDP_WIRE_ERR
 - trdp_types.h, 256
- TRDP_XML_DOC_HANDLE_T, 61
- TRDP_XML_PARSER_ERR
 - trdp_types.h, 257
- trnId
 - TRDP_OP_TRAIN_DIR_STATE_T, 41
- trnOperator
 - TRDP_OP_TRAIN_DIR_STATE_T, 41
- TTDB_NET_DIR_REQ_COMID
 - iec61375-2-3.h, 74
- TTDB_OP_DIR_INFO_COMID
 - iec61375-2-3.h, 74
- TTDB_STAT_CST_REQ_COMID
 - iec61375-2-3.h, 74
- TTDB_TRN_DIR_REQ_COMID
 - iec61375-2-3.h, 74
- USE_HEAP
 - trdp_types.h, 252
- vehId
 - TRDP_OP_VEHICLE_T, 45
 - TRDP_VEHICLE_INFO_T, 61
- version
 - TRDP_CSTINFOCTRL_T, 22
- vos_addTime
 - vos_thread.h, 303
- VOS_BLOCK_ERR
 - vos_types.h, 317
- vos_bsearch
 - vos_mem.c, 260
 - vos_mem.h, 269
- vos_clearTime

- vos_thread.h, 303
- vos_cmpTime
 - vos_thread.h, 304
- vos_crc32
 - vos_utils.c, 319
 - vos_utils.h, 325
- vos_determineBindAddr
 - vos_sock.h, 283
- vos_divTime
 - vos_thread.h, 304
- vos_dottedIP
 - vos_sock.h, 284
- VOS_ERR_T
 - vos_types.h, 317
- vos_getErrorString
 - vos_utils.c, 320
 - vos_utils.h, 326
- vos_getInterfaces
 - vos_sock.h, 284
- vos_getRealTime
 - vos_thread.h, 304
- vos_getTime
 - vos_thread.h, 305
- vos_getTimeStamp
 - vos_thread.h, 305
- vos_getUuid
 - vos_thread.h, 305
- vos_getVersion
 - vos_utils.c, 320
 - vos_utils.h, 326
- vos_getVersionString
 - vos_utils.c, 320
 - vos_utils.h, 327
- vos_hostIsBigEndian
 - vos_utils.c, 321
 - vos_utils.h, 327
- vos_htonl
 - vos_sock.h, 285
- vos_htonll
 - vos_sock.h, 285
- vos_htons
 - vos_sock.h, 285
- vos_ifnameFromVlanId
 - vos_sock.h, 286
- vos_init
 - vos_utils.c, 321
 - vos_utils.h, 327
- VOS_INIT_ERR
 - vos_types.h, 317
- VOS_INTEGRATION_ERR
 - vos_types.h, 317
- VOS_INUSE_ERR
 - vos_types.h, 317
- VOS_IO_ERR
 - vos_types.h, 317
- vos_ipDotted
 - vos_sock.h, 286
- vos_isMulticast
 - vos_sock.h, 287
- VOS_LOG_DBG
 - vos_types.h, 318
- VOS_LOG_ERROR
 - vos_types.h, 318
- VOS_LOG_INFO
 - vos_types.h, 318
- VOS_LOG_T
 - vos_types.h, 317
- VOS_LOG_USR
 - vos_types.h, 318
- VOS_LOG_WARNING
 - vos_types.h, 318
- VOS_MAX_ERR_STR_SIZE
 - vos_utils.h, 325
- VOS_MAX_FRMT_SIZE
 - vos_utils.h, 325
- VOS_MAX_NUM_IF
 - vos_sock.h, 282
- VOS_MAX_PRNT_STR_SIZE
 - vos_utils.h, 325
- VOS_MAX_SOCKET_CNT
 - vos_sock.h, 282
- vos_mem.c, 258
 - vos_bsearch, 260
 - vos_memAlloc, 260
 - vos_memCount, 261
 - vos_memDelete, 261
 - vos_memFree, 261
 - vos_memInit, 262
 - vos_qsort, 262
 - vos_queueCreate, 263
 - vos_queueDestroy, 263
 - vos_queueReceive, 264
 - vos_queueSend, 264
 - vos_strncat, 265
 - vos_strncpy, 265
 - vos_strncmp, 266
- vos_mem.h, 266
 - vos_bsearch, 269
 - VOS_MEM_MAX_PREALLOCATE, 269
 - VOS_MEM_PREALLOCATE, 269
 - vos_memAlloc, 270
 - vos_memCount, 270
 - vos_memDelete, 271
 - vos_memFree, 271
 - vos_memInit, 272
 - vos_qsort, 272
 - vos_queueCreate, 273
 - vos_queueDestroy, 273
 - vos_queueReceive, 274
 - vos_queueSend, 274
 - vos_strncat, 275
 - vos_strncpy, 275
 - vos_strncmp, 276
- VOS_MEM_ERR
 - vos_types.h, 317
- VOS_MEM_MAX_PREALLOCATE

- vos_mem.h, [269](#)
- VOS_MEM_PREALLOCATE
 - vos_mem.h, [269](#)
- VOS_MEM_STATISTICS_T, [61](#)
- vos_memAlloc
 - vos_mem.c, [260](#)
 - vos_mem.h, [270](#)
- vos_memCount
 - vos_mem.c, [261](#)
 - vos_mem.h, [270](#)
- vos_memDelete
 - vos_mem.c, [261](#)
 - vos_mem.h, [271](#)
- vos_memFree
 - vos_mem.c, [261](#)
 - vos_mem.h, [271](#)
- vos_memInit
 - vos_mem.c, [262](#)
 - vos_mem.h, [272](#)
- vos_mulTime
 - vos_thread.h, [306](#)
- VOS_MUTEX_ERR
 - vos_types.h, [317](#)
- vos_mutexCreate
 - vos_thread.h, [306](#)
- vos_mutexDelete
 - vos_thread.h, [306](#)
- vos_mutexLock
 - vos_thread.h, [307](#)
- vos_mutexTryLock
 - vos_thread.h, [307](#)
- vos_mutexUnlock
 - vos_thread.h, [308](#)
- vos_netIfUp
 - vos_sock.h, [287](#)
- VOS_NO_ERR
 - vos_types.h, [317](#)
- VOS_NOCONN_ERR
 - vos_types.h, [317](#)
- VOS_NODATA_ERR
 - vos_types.h, [317](#)
- VOS_NOINIT_ERR
 - vos_types.h, [317](#)
- vos_ntohl
 - vos_sock.h, [287](#)
- vos_ntohll
 - vos_sock.h, [288](#)
- vos_ntohs
 - vos_sock.h, [288](#)
- VOS_PARAM_ERR
 - vos_types.h, [317](#)
- VOS_PRINT_DBG_T
 - vos_types.h, [316](#)
- vos_qsort
 - vos_mem.c, [262](#)
 - vos_mem.h, [272](#)
- VOS_QUEUE_ERR
 - vos_types.h, [317](#)
- VOS_QUEUE_FULL_ERR
 - vos_types.h, [317](#)
- VOS_QUEUE_T, [62](#)
- vos_queueCreate
 - vos_mem.c, [263](#)
 - vos_mem.h, [273](#)
- vos_queueDestroy
 - vos_mem.c, [263](#)
 - vos_mem.h, [273](#)
- vos_queueReceive
 - vos_mem.c, [264](#)
 - vos_mem.h, [274](#)
- vos_queueSend
 - vos_mem.c, [264](#)
 - vos_mem.h, [274](#)
- vos_sc32
 - vos_utils.c, [321](#)
 - vos_utils.h, [328](#)
- vos_select
 - vos_sock.h, [288](#)
- VOS_SEMA_ERR
 - vos_types.h, [317](#)
- vos_semaCreate
 - vos_thread.h, [308](#)
- vos_semaDelete
 - vos_thread.h, [308](#)
- vos_semaGive
 - vos_thread.h, [309](#)
- vos_semaTake
 - vos_thread.h, [309](#)
- vos_shared_mem.h, [276](#)
 - vos_sharedClose, [278](#)
 - vos_sharedOpen, [278](#)
- vos_sharedClose
 - vos_shared_mem.h, [278](#)
- vos_sharedOpen
 - vos_shared_mem.h, [278](#)
- vos_sock.h, [279](#)
 - vos_determineBindAddr, [283](#)
 - vos_dottedIP, [284](#)
 - vos_getInterfaces, [284](#)
 - vos_htonl, [285](#)
 - vos_htonll, [285](#)
 - vos_htons, [285](#)
 - vos_ifnameFromVlanId, [286](#)
 - vos_ipDotted, [286](#)
 - vos_isMulticast, [287](#)
 - VOS_MAX_NUM_IF, [282](#)
 - VOS_MAX_SOCKET_CNT, [282](#)
 - vos_netIfUp, [287](#)
 - vos_ntohl, [287](#)
 - vos_ntohll, [288](#)
 - vos_ntohs, [288](#)
 - vos_select, [288](#)
 - vos_sockAccept, [289](#)
 - vos_sockBind, [290](#)
 - vos_sockClose, [290](#)
 - vos_sockCmp, [282](#)

- vos_sockConnect, 290
- vos_sockGetMAC, 291
- vos_sockId, 283
- vos_sockInit, 291
- vos_sockJoinMC, 292
- vos_sockLeaveMC, 292
- vos_sockListen, 293
- vos_sockOpenTCP, 293
- vos_sockOpenTSN, 294
- vos_sockOpenUDP, 294
- vos_sockPrintOptions, 295
- vos_sockReceiveTCP, 295
- vos_sockReceiveTSN, 296
- vos_sockReceiveUDP, 296
- vos_sockSendTCP, 297
- vos_sockSendTSN, 298
- vos_sockSendUDP, 298
- vos_sockSetMulticastIf, 299
- vos_sockSetOptions, 299
- vos_sockTerm, 300
- VOS_TTL_MULTICAST, 283
- VOS_SOCKET_ERR
 - vos_types.h, 317
- VOS_SOCKET_OPT_T, 62
- vos_sockAccept
 - vos_sock.h, 289
- vos_sockBind
 - vos_sock.h, 290
- vos_sockClose
 - vos_sock.h, 290
- vos_sockCmp
 - vos_sock.h, 282
- vos_sockConnect
 - vos_sock.h, 290
- vos_sockGetMAC
 - vos_sock.h, 291
- vos_sockId
 - vos_sock.h, 283
- vos_sockInit
 - vos_sock.h, 291
- vos_sockJoinMC
 - vos_sock.h, 292
- vos_sockLeaveMC
 - vos_sock.h, 292
- vos_sockListen
 - vos_sock.h, 293
- vos_sockOpenTCP
 - vos_sock.h, 293
- vos_sockOpenTSN
 - vos_sock.h, 294
- vos_sockOpenUDP
 - vos_sock.h, 294
- vos_sockPrintOptions
 - vos_sock.h, 295
- vos_sockReceiveTCP
 - vos_sock.h, 295
- vos_sockReceiveTSN
 - vos_sock.h, 296
- vos_sockReceiveUDP
 - vos_sock.h, 296
- vos_sockSendTCP
 - vos_sock.h, 297
- vos_sockSendTSN
 - vos_sock.h, 298
- vos_sockSendUDP
 - vos_sock.h, 298
- vos_sockSetMulticastIf
 - vos_sock.h, 299
- vos_sockSetOptions
 - vos_sock.h, 299
- vos_sockTerm
 - vos_sock.h, 300
- vos_strncat
 - vos_mem.c, 265
 - vos_mem.h, 275
- vos_strncpy
 - vos_mem.c, 265
 - vos_mem.h, 275
- vos_strnicmp
 - vos_mem.c, 266
 - vos_mem.h, 276
- vos_subTime
 - vos_thread.h, 310
- vos_terminate
 - vos_utils.c, 322
 - vos_utils.h, 329
- vos_thread.h, 300
 - vos_addTime, 303
 - vos_clearTime, 303
 - vos_cmpTime, 304
 - vos_divTime, 304
 - vos_getRealTime, 304
 - vos_getTime, 305
 - vos_getTimeStamp, 305
 - vos_getUuid, 305
 - vos_mulTime, 306
 - vos_mutexCreate, 306
 - vos_mutexDelete, 306
 - vos_mutexLock, 307
 - vos_mutexTryLock, 307
 - vos_mutexUnlock, 308
 - vos_semaCreate, 308
 - vos_semaDelete, 308
 - vos_semaGive, 309
 - vos_semaTake, 309
 - vos_subTime, 310
 - vos_threadCreate, 310
 - vos_threadCreateSync, 311
 - vos_threadDelay, 311
 - vos_threadInit, 312
 - vos_threadIsActive, 312
 - vos_threadSelf, 313
 - vos_threadTerm, 313
 - vos_threadTerminate, 313
- VOS_THREAD_ERR
 - vos_types.h, 317

- vos_threadCreate
 - vos_thread.h, [310](#)
- vos_threadCreateSync
 - vos_thread.h, [311](#)
- vos_threadDelay
 - vos_thread.h, [311](#)
- vos_threadInit
 - vos_thread.h, [312](#)
- vos_threadIsActive
 - vos_thread.h, [312](#)
- vos_threadSelf
 - vos_thread.h, [313](#)
- vos_threadTerm
 - vos_thread.h, [313](#)
- vos_threadTerminate
 - vos_thread.h, [313](#)
- VOS_TIMEOUT_ERR
 - vos_types.h, [317](#)
- VOS_TIMEVAL_T
 - vos_types.h, [316](#)
- VOS_TTL_MULTICAST
 - vos_sock.h, [283](#)
- vos_types.h, [314](#)
 - VOS_BLOCK_ERR, [317](#)
 - VOS_ERR_T, [317](#)
 - VOS_INIT_ERR, [317](#)
 - VOS_INTEGRATION_ERR, [317](#)
 - VOS_INUSE_ERR, [317](#)
 - VOS_IO_ERR, [317](#)
 - VOS_LOG_DBG, [318](#)
 - VOS_LOG_ERROR, [318](#)
 - VOS_LOG_INFO, [318](#)
 - VOS_LOG_T, [317](#)
 - VOS_LOG_USR, [318](#)
 - VOS_LOG_WARNING, [318](#)
 - VOS_MEM_ERR, [317](#)
 - VOS_MUTEX_ERR, [317](#)
 - VOS_NO_ERR, [317](#)
 - VOS_NOCONN_ERR, [317](#)
 - VOS_NODATA_ERR, [317](#)
 - VOS_NOINIT_ERR, [317](#)
 - VOS_PARAM_ERR, [317](#)
 - VOS_PRINT_DBG_T, [316](#)
 - VOS_QUEUE_ERR, [317](#)
 - VOS_QUEUE_FULL_ERR, [317](#)
 - VOS_SEMA_ERR, [317](#)
 - VOS_SOCKET_ERR, [317](#)
 - VOS_THREAD_ERR, [317](#)
 - VOS_TIMEOUT_ERR, [317](#)
 - VOS_TIMEVAL_T, [316](#)
 - VOS_UNKNOWN_ERR, [317](#)
- VOS_UNKNOWN_ERR
 - vos_types.h, [317](#)
- vos_utils.c, [318](#)
 - vos_crc32, [319](#)
 - vos_getErrorString, [320](#)
 - vos_getVersion, [320](#)
 - vos_getVersionString, [320](#)
 - vos_hostIsBigEndian, [321](#)
 - vos_init, [321](#)
 - vos_sc32, [321](#)
 - vos_terminate, [322](#)
- vos_utils.h, [322](#)
 - INITFCS, [325](#)
 - vos_crc32, [325](#)
 - vos_getErrorString, [326](#)
 - vos_getVersion, [326](#)
 - vos_getVersionString, [327](#)
 - vos_hostIsBigEndian, [327](#)
 - vos_init, [327](#)
 - VOS_MAX_ERR_STR_SIZE, [325](#)
 - VOS_MAX_FRMT_SIZE, [325](#)
 - VOS_MAX_PRNT_STR_SIZE, [325](#)
 - vos_sc32, [328](#)
 - vos_terminate, [329](#)
- VOS_VERSION_T, [63](#)