

TCNOpen TRDP  
PrototypeV1.0

Generated by Doxygen 1.5.6

Fri Jul 12 15:37:32 2013



# Contents

<b>1</b>	<b>The TRDP Light Library API Specification</b>	<b>1</b>
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	Conventions of the API . . . . .	4
<b>2</b>	<b>Data Structure Index</b>	<b>5</b>
2.1	Data Structures . . . . .	5
<b>3</b>	<b>File Index</b>	<b>7</b>
3.1	File List . . . . .	7
<b>4</b>	<b>Data Structure Documentation</b>	<b>9</b>
4.1	GNU_PACKED Struct Reference . . . . .	9
4.1.1	Detailed Description . . . . .	10
4.1.2	Field Documentation . . . . .	10
4.1.2.1	protocolVersion . . . . .	10
4.1.2.2	msgType . . . . .	10
4.1.2.3	datasetLength . . . . .	11
4.2	PD_ELE Struct Reference . . . . .	12
4.2.1	Detailed Description . . . . .	13
4.2.2	Field Documentation . . . . .	13
4.2.2.1	pFrame . . . . .	13
4.3	TAU_MARSHALL_INFO_T Struct Reference . . . . .	15
4.3.1	Detailed Description . . . . .	15
4.4	TRDP_CAR_INFO_T Struct Reference . . . . .	16

4.4.1	Detailed Description	17
4.4.2	Field Documentation	17
4.4.2.1	orient	17
4.4.2.2	pDevInfo	17
4.5	TRDP_COMID_DSID_MAP_T Struct Reference	18
4.5.1	Detailed Description	18
4.6	TRDP_CST_INFO_T Struct Reference	19
4.6.1	Detailed Description	20
4.6.2	Field Documentation	20
4.6.2.1	owner	20
4.6.2.2	orient	20
4.6.2.3	pFctInfo	20
4.6.2.4	pCarInfo	20
4.7	TRDP_DATASET Struct Reference	21
4.7.1	Detailed Description	21
4.8	TRDP_DATASET_ELEMENT_T Struct Reference	22
4.8.1	Detailed Description	22
4.8.2	Field Documentation	22
4.8.2.1	type	22
4.9	TRDP_DBG_CONFIG_T Struct Reference	23
4.9.1	Detailed Description	23
4.10	TRDP_DEVICE_INFO_T Struct Reference	24
4.10.1	Detailed Description	25
4.10.2	Field Documentation	25
4.10.2.1	orient	25
4.11	TRDP_FCT_INFO_T Struct Reference	26
4.11.1	Detailed Description	26
4.12	TRDP_HANDLE Struct Reference	27
4.12.1	Detailed Description	27
4.13	TRDP_LIST_STATISTICS_T Struct Reference	28
4.13.1	Detailed Description	28
4.14	TRDP_MARSHALL_CONFIG_T Struct Reference	29
4.14.1	Detailed Description	29
4.15	TRDP_MD_CONFIG_T Struct Reference	30
4.15.1	Detailed Description	31
4.16	TRDP_MD_INFO_T Struct Reference	32

4.16.1 Detailed Description . . . . .	33
4.16.2 Field Documentation . . . . .	33
4.16.2.1 msgType . . . . .	33
4.17 TRDP_MD_STATISTICS_T Struct Reference . . . . .	34
4.17.1 Detailed Description . . . . .	35
4.18 TRDP_MEM_CONFIG_T Struct Reference . . . . .	36
4.18.1 Detailed Description . . . . .	36
4.19 TRDP_MEM_STATISTICS_T Struct Reference . . . . .	37
4.19.1 Detailed Description . . . . .	37
4.20 TRDP_PD_CONFIG_T Struct Reference . . . . .	38
4.20.1 Detailed Description . . . . .	38
4.21 TRDP_PD_INFO_T Struct Reference . . . . .	39
4.21.1 Detailed Description . . . . .	39
4.21.2 Field Documentation . . . . .	40
4.21.2.1 msgType . . . . .	40
4.22 TRDP_PD_STATISTICS_T Struct Reference . . . . .	41
4.22.1 Detailed Description . . . . .	42
4.23 TRDP_PROCESS_CONFIG_T Struct Reference . . . . .	43
4.23.1 Detailed Description . . . . .	43
4.24 TRDP_PROP_INFO_T Struct Reference . . . . .	44
4.24.1 Detailed Description . . . . .	44
4.25 TRDP_PUB_STATISTICS_T Struct Reference . . . . .	45
4.25.1 Detailed Description . . . . .	45
4.25.2 Field Documentation . . . . .	45
4.25.2.1 destAddr . . . . .	45
4.26 TRDP_RED_STATISTICS_T Struct Reference . . . . .	46
4.26.1 Detailed Description . . . . .	46
4.27 TRDP_SDT_PAR_T Struct Reference . . . . .	47
4.27.1 Detailed Description . . . . .	47
4.28 TRDP_SEND_PARAM_T Struct Reference . . . . .	48
4.28.1 Detailed Description . . . . .	48
4.29 TRDP_SESSION Struct Reference . . . . .	49
4.29.1 Detailed Description . . . . .	50
4.30 TRDP_SOCKET_TCP Struct Reference . . . . .	51
4.30.1 Detailed Description . . . . .	51
4.31 TRDP_SOCKETS Struct Reference . . . . .	52

4.31.1 Detailed Description . . . . .	52
4.31.2 Field Documentation . . . . .	53
4.31.2.1 usage . . . . .	53
4.32 TRDP_STATISTICS_T Struct Reference . . . . .	54
4.32.1 Detailed Description . . . . .	55
4.33 TRDP_SUBS_STATISTICS_T Struct Reference . . . . .	56
4.33.1 Detailed Description . . . . .	56
4.33.2 Field Documentation . . . . .	56
4.33.2.1 filterAddr . . . . .	56
4.33.2.2 timeout . . . . .	56
4.33.2.3 toBehav . . . . .	57
4.33.2.4 numRecv . . . . .	57
4.34 TRDP_TRAIN_INFO_T Struct Reference . . . . .	58
4.34.1 Detailed Description . . . . .	59
4.34.2 Field Documentation . . . . .	59
4.34.2.1 operator . . . . .	59
4.34.2.2 topoCnt . . . . .	59
4.34.2.3 pCstInfo . . . . .	59
4.35 TRDP_VERSION_T Struct Reference . . . . .	60
4.35.1 Detailed Description . . . . .	60
4.36 TRDP_XML_DOC_HANDLE_T Struct Reference . . . . .	61
4.36.1 Detailed Description . . . . .	61
4.37 VOS SOCK_OPT_T Struct Reference . . . . .	62
4.37.1 Detailed Description . . . . .	62
4.37.2 Field Documentation . . . . .	62
4.37.2.1 qos . . . . .	62
4.38 VOS_TIME_T Struct Reference . . . . .	63
4.38.1 Detailed Description . . . . .	63
4.38.2 Field Documentation . . . . .	63
4.38.2.1 tv_usec . . . . .	63
<b>5 File Documentation . . . . .</b>	<b>65</b>
5.1 tau_addr.h File Reference . . . . .	65
5.1.1 Detailed Description . . . . .	67
5.1.2 Function Documentation . . . . .	67
5.1.2.1 tau_addr2CarId . . . . .	67
5.1.2.2 tau_addr2CarNo . . . . .	68

5.1.2.3	<a href="#">tau_addr2CstId</a>	68
5.1.2.4	<a href="#">tau_addr2CstNo</a>	68
5.1.2.5	<a href="#">tau_addr2IecCarNo</a>	69
5.1.2.6	<a href="#">tau_addr2IecCstNo</a>	69
5.1.2.7	<a href="#">tau_addr2Uri</a>	69
5.1.2.8	<a href="#">tau_carNo2Ids</a>	70
5.1.2.9	<a href="#">tau_cstNo2CstId</a>	70
5.1.2.10	<a href="#">tau_getOwnAddr</a>	70
5.1.2.11	<a href="#">tau_getOwnIds</a>	71
5.1.2.12	<a href="#">tau_iecCarNo2Ids</a>	71
5.1.2.13	<a href="#">tau_iecCstNo2CstId</a>	71
5.1.2.14	<a href="#">tau_label2CarId</a>	72
5.1.2.15	<a href="#">tau_label2CarNo</a>	72
5.1.2.16	<a href="#">tau_label2CstId</a>	72
5.1.2.17	<a href="#">tau_label2CstNo</a>	73
5.1.2.18	<a href="#">tau_label2IecCarNo</a>	73
5.1.2.19	<a href="#">tau_label2IecCstNo</a>	73
5.1.2.20	<a href="#">tau_uri2Addr</a>	74
5.2	<a href="#">tau_marshall.c File Reference</a>	75
5.2.1	<a href="#">Detailed Description</a>	76
5.2.2	<a href="#">Function Documentation</a>	76
5.2.2.1	<a href="#">tau_calcDatasetSize</a>	76
5.2.2.2	<a href="#">tau_calcDatasetSizeByComId</a>	77
5.2.2.3	<a href="#">tau_initMarshall</a>	77
5.2.2.4	<a href="#">tau_marshall</a>	78
5.2.2.5	<a href="#">tau_marshallIDs</a>	78
5.2.2.6	<a href="#">tau_unmarshall</a>	79
5.2.2.7	<a href="#">tau_unmarshallIDs</a>	79
5.3	<a href="#">tau_marshall.h File Reference</a>	81
5.3.1	<a href="#">Detailed Description</a>	82
5.3.2	<a href="#">Function Documentation</a>	82
5.3.2.1	<a href="#">tau_calcDatasetSize</a>	82
5.3.2.2	<a href="#">tau_calcDatasetSizeByComId</a>	83
5.3.2.3	<a href="#">tau_initMarshall</a>	83
5.3.2.4	<a href="#">tau_marshall</a>	84
5.3.2.5	<a href="#">tau_marshallIDs</a>	85

5.3.2.6	<a href="#">tau_unmarshall</a>	85
5.3.2.7	<a href="#">tau_unmarshallDs</a>	86
5.4	<a href="#">tau_tti.h File Reference</a>	87
5.4.1	<a href="#">Detailed Description</a>	89
5.4.2	<a href="#">Enumeration Type Documentation</a>	89
5.4.2.1	<a href="#">TRDP_FCT_T</a>	89
5.4.2.2	<a href="#">TRDP_INAUG_STATE_T</a>	90
5.4.3	<a href="#">Function Documentation</a>	90
5.4.3.1	<a href="#">tau_getCarDevCnt</a>	90
5.4.3.2	<a href="#">tau_getCarInfo</a>	90
5.4.3.3	<a href="#">tau_getCarOrient</a>	91
5.4.3.4	<a href="#">tau_getCstCarCnt</a>	91
5.4.3.5	<a href="#">tau_getCstFctCnt</a>	92
5.4.3.6	<a href="#">tau_getCstFctInfo</a>	92
5.4.3.7	<a href="#">tau_getCstInfo</a>	92
5.4.3.8	<a href="#">tau_getDevInfo</a>	93
5.4.3.9	<a href="#">tau_getEtbState</a>	93
5.4.3.10	<a href="#">tau_getIecCarOrient</a>	93
5.4.3.11	<a href="#">tau_getTrnCarCnt</a>	94
5.4.3.12	<a href="#">tau_getTrnCstCnt</a>	94
5.4.3.13	<a href="#">tau_getTrnInfo</a>	94
5.5	<a href="#">tau_xml.c File Reference</a>	95
5.5.1	<a href="#">Detailed Description</a>	96
5.5.2	<a href="#">Define Documentation</a>	97
5.5.2.1	<a href="#">TRDP_SDT_DEFAULT_CMTHR</a>	97
5.5.3	<a href="#">Function Documentation</a>	97
5.5.3.1	<a href="#">tau_freeTelegrams</a>	97
5.5.3.2	<a href="#">tau_freeXmlDoc</a>	97
5.5.3.3	<a href="#">tau_prepareXmlDoc</a>	97
5.5.3.4	<a href="#">tau_readXmlDatasetConfig</a>	98
5.5.3.5	<a href="#">tau_readXmlDeviceConfig</a>	98
5.5.3.6	<a href="#">tau_readXmlInterfaceConfig</a>	98
5.6	<a href="#">tau_xml.h File Reference</a>	100
5.6.1	<a href="#">Detailed Description</a>	101
5.6.2	<a href="#">Enumeration Type Documentation</a>	102
5.6.2.1	<a href="#">TRDP_DBG_OPTION_T</a>	102



5.6.3	Function Documentation . . . . .	102
5.6.3.1	tau_freeTelegrams . . . . .	102
5.6.3.2	tau_freeXmlDoc . . . . .	103
5.6.3.3	tau_prepareXmlDoc . . . . .	103
5.6.3.4	tau_readXmlDatasetConfig . . . . .	103
5.6.3.5	tau_readXmlDeviceConfig . . . . .	104
5.6.3.6	tau_readXmlInterfaceConfig . . . . .	104
5.7	trdp_dllmain.c File Reference . . . . .	106
5.7.1	Detailed Description . . . . .	106
5.8	trdp_if.c File Reference . . . . .	107
5.8.1	Detailed Description . . . . .	109
5.8.2	Function Documentation . . . . .	110
5.8.2.1	tlc_closeSession . . . . .	110
5.8.2.2	tlc_getInterval . . . . .	110
5.8.2.3	tlc_getVersion . . . . .	111
5.8.2.4	tlc_getVersionString . . . . .	111
5.8.2.5	tlc_init . . . . .	111
5.8.2.6	tlc_openSession . . . . .	112
5.8.2.7	tlc_process . . . . .	114
5.8.2.8	tlc_reinitSession . . . . .	116
5.8.2.9	tlc_setTopoCount . . . . .	116
5.8.2.10	tlc_terminate . . . . .	117
5.8.2.11	tlp_get . . . . .	117
5.8.2.12	tlp_getRedundant . . . . .	119
5.8.2.13	tlp_publish . . . . .	119
5.8.2.14	tlp_put . . . . .	122
5.8.2.15	tlp_request . . . . .	122
5.8.2.16	tlp_setRedundant . . . . .	124
5.8.2.17	tlp_subscribe . . . . .	125
5.8.2.18	tlp_unpublish . . . . .	126
5.8.2.19	tlp_unsubscribe . . . . .	127
5.8.2.20	trdp_isValidSession . . . . .	128
5.8.2.21	trdp_sessionQueue . . . . .	128
5.9	trdp_if.h File Reference . . . . .	129
5.9.1	Detailed Description . . . . .	129
5.9.2	Function Documentation . . . . .	130

5.9.2.1	trdp_isValidSession	130
5.9.2.2	trdp_sessionQueue	130
5.10	trdp_if_light.h File Reference	131
5.10.1	Detailed Description	135
5.10.2	Function Documentation	135
5.10.2.1	tlc_closeSession	135
5.10.2.2	tlc_freeBuf	136
5.10.2.3	tlc_getInterval	136
5.10.2.4	tlc_getJoinStatistics	137
5.10.2.5	tlc_getListStatistics	138
5.10.2.6	tlc_getPubStatistics	139
5.10.2.7	tlc_getRedStatistics	140
5.10.2.8	tlc_getStatistics	141
5.10.2.9	tlc_getSubsStatistics	142
5.10.2.10	tlc_getVersion	142
5.10.2.11	tlc_getVersionString	143
5.10.2.12	tlc_init	143
5.10.2.13	tlc_openSession	144
5.10.2.14	tlc_process	147
5.10.2.15	tlc_reinitSession	149
5.10.2.16	tlc_resetStatistics	149
5.10.2.17	tlc_setTopoCount	150
5.10.2.18	tlc_terminate	151
5.10.2.19	t1m_abortSession	151
5.10.2.20	t1m_addListener	152
5.10.2.21	t1m_confirm	152
5.10.2.22	t1m_delListener	153
5.10.2.23	t1m_notify	153
5.10.2.24	t1m_reply	154
5.10.2.25	t1m_replyErr	155
5.10.2.26	t1m_replyQuery	155
5.10.2.27	t1m_request	156
5.10.2.28	t1p_get	157
5.10.2.29	t1p_getRedundant	159
5.10.2.30	t1p_publish	160
5.10.2.31	t1p_put	163

5.10.2.32	tlp_request	164
5.10.2.33	tlp_setRedundant	166
5.10.2.34	tlp_subscribe	167
5.10.2.35	tlp_unpublish	169
5.10.2.36	tlp_unsubscribe	170
5.11	trdp_mdcom.c File Reference	172
5.11.1	Detailed Description	173
5.11.2	Function Documentation	174
5.11.2.1	trdp_closeMDSessions	174
5.11.2.2	trdp_getTCPSocket	174
5.11.2.3	trdp_mdCheck	175
5.11.2.4	trdp_mdCheckListenSocks	175
5.11.2.5	trdp_mdCheckPending	176
5.11.2.6	trdp_mdCheckTimeouts	177
5.11.2.7	trdp_mdFreeSession	177
5.11.2.8	trdp_mdRecv	177
5.11.2.9	trdp_mdRecvPacket	178
5.11.2.10	trdp_mdSend	179
5.11.2.11	trdp_mdSendPacket	179
5.11.2.12	trdp_mdSetSessionTimeout	180
5.11.2.13	trdp_mdUpdatePacket	180
5.12	trdp_mdcom.h File Reference	181
5.12.1	Detailed Description	182
5.12.2	Function Documentation	182
5.12.2.1	trdp_closeMDSessions	182
5.12.2.2	trdp_getTCPSocket	183
5.12.2.3	trdp_mdCheckListenSocks	183
5.12.2.4	trdp_mdCheckPending	184
5.12.2.5	trdp_mdCheckTimeouts	185
5.12.2.6	trdp_mdFreeSession	185
5.12.2.7	trdp_mdRecv	185
5.12.2.8	trdp_mdSend	186
5.12.2.9	trdp_mdSendPacket	187
5.12.2.10	trdp_mdSetSessionTimeout	187
5.12.2.11	trdp_mdUpdatePacket	188
5.13	trdp_pdcom.c File Reference	189

5.13.1	Detailed Description	190
5.13.2	Function Documentation	191
5.13.2.1	trdp_pdCheck	191
5.13.2.2	trdp_pdCheckListenSocks	191
5.13.2.3	trdp_pdCheckPending	192
5.13.2.4	trdp_pdDataUpdate	192
5.13.2.5	trdp_pdDistribute	193
5.13.2.6	trdp_pdHandleTimeOuts	193
5.13.2.7	trdp_pdInit	194
5.13.2.8	trdp_pdReceive	194
5.13.2.9	trdp_pdSend	195
5.13.2.10	trdp_pdSendQueued	196
5.13.2.11	trdp_pdUpdate	196
5.14	trdp_pdcom.h File Reference	197
5.14.1	Detailed Description	198
5.14.2	Function Documentation	199
5.14.2.1	trdp_pdCheck	199
5.14.2.2	trdp_pdCheckListenSocks	199
5.14.2.3	trdp_pdCheckPending	200
5.14.2.4	trdp_pdDataUpdate	200
5.14.2.5	trdp_pdDistribute	201
5.14.2.6	trdp_pdHandleTimeOuts	201
5.14.2.7	trdp_pdInit	202
5.14.2.8	trdp_pdReceive	202
5.14.2.9	trdp_pdSend	203
5.14.2.10	trdp_pdSendQueued	204
5.14.2.11	trdp_pdUpdate	204
5.15	trdp_private.h File Reference	205
5.15.1	Detailed Description	208
5.15.2	Enumeration Type Documentation	208
5.15.2.1	TRDP_MD_ELE_ST_T	208
5.15.2.2	TRDP_PRIV_FLAGS_T	209
5.15.2.3	TRDP SOCK_TYPE_T	209
5.16	trdp_proto.h File Reference	210
5.16.1	Detailed Description	212
5.16.2	Define Documentation	212

5.16.2.1	TRDP_COMID_ECHO	212
5.16.2.2	TRDP_DEST_URI_SIZE	212
5.16.2.3	TRDP_MAX_FILE_NAME_LEN	213
5.16.2.4	TRDP_MAX_LABEL_LEN	213
5.16.2.5	TRDP_MAX_URI_HOST_LEN	213
5.16.2.6	TRDP_MAX_URI_LEN	213
5.16.2.7	TRDP_MAX_URI_USER_LEN	213
5.16.2.8	TRDP_STATISTICS_REQUEST_DSID	213
5.16.3	Enumeration Type Documentation	213
5.16.3.1	TRDP_MSG_T	213
5.17	trdp_stats.c File Reference	215
5.17.1	Detailed Description	216
5.17.2	Function Documentation	217
5.17.2.1	tlc_getJoinStatistics	217
5.17.2.2	tlc_getListStatistics	217
5.17.2.3	tlc_getPubStatistics	218
5.17.2.4	tlc_getRedStatistics	218
5.17.2.5	tlc_getStatistics	219
5.17.2.6	tlc_getSubsStatistics	219
5.17.2.7	tlc_resetStatistics	220
5.17.2.8	trdp_initStats	220
5.17.2.9	trdp_pdPrepareStats	221
5.17.2.10	trdp_UpdateStats	221
5.18	trdp_stats.h File Reference	222
5.18.1	Detailed Description	222
5.18.2	Function Documentation	223
5.18.2.1	trdp_initStats	223
5.18.2.2	trdp_pdPrepareStats	223
5.19	trdp_types.h File Reference	225
5.19.1	Detailed Description	230
5.19.2	Typedef Documentation	230
5.19.2.1	TRDP_IP_ADDR_T	230
5.19.2.2	TRDP_MARSHALL_T	230
5.19.2.3	TRDP_MD_CALLBACK_T	231
5.19.2.4	TRDP_PD_CALLBACK_T	231
5.19.2.5	TRDP_PRINT_DBG_T	231

5.19.2.6	TRDP_TIME_T	231
5.19.2.7	TRDP_UNMARSHALL_T	231
5.19.3	Enumeration Type Documentation	232
5.19.3.1	TRDP_DATA_TYPE_T	232
5.19.3.2	TRDP_ERR_T	232
5.19.3.3	TRDP_FLAGS_T	233
5.19.3.4	TRDP_OPTION_T	234
5.19.3.5	TRDP_RED_STATE_T	234
5.19.3.6	TRDP_REPLY_STATUS_T	234
5.19.3.7	TRDP_TO_BEHAVIOR_T	234
5.20	trdp_utils.c File Reference	235
5.20.1	Detailed Description	237
5.20.2	Function Documentation	237
5.20.2.1	am_big_endian	237
5.20.2.2	printSocketUsage	237
5.20.2.3	trdp_getSeqCnt	237
5.20.2.4	trdp_initSockets	238
5.20.2.5	trdp_isAddressed	238
5.20.2.6	trdp_isRcvSeqCnt	238
5.20.2.7	trdp_packetSizeMD	239
5.20.2.8	trdp_packetSizePD	239
5.20.2.9	trdp_queueAppLast	239
5.20.2.10	trdp_queueDelElement	240
5.20.2.11	trdp_queueFindComId	240
5.20.2.12	trdp_queueFindPubAddr	240
5.20.2.13	trdp_queueFindSubAddr	240
5.20.2.14	trdp_queueInsFirst	241
5.20.2.15	trdp_releaseSocket	241
5.20.2.16	trdp_requestSocket	241
5.20.2.17	trdp_SockAddJoin	242
5.20.2.18	trdp_SockDelJoin	243
5.20.2.19	trdp_SockIsJoined	243
5.21	trdp_utils.h File Reference	244
5.21.1	Detailed Description	245
5.21.2	Function Documentation	246
5.21.2.1	am_big_endian	246

5.21.2.2	<a href="#">trdp_getSeqCnt</a>	246
5.21.2.3	<a href="#">trdp_initSockets</a>	247
5.21.2.4	<a href="#">trdp_initUncompletedTCP</a>	247
5.21.2.5	<a href="#">trdp_isAddressed</a>	247
5.21.2.6	<a href="#">trdp_isRcvSeqCnt</a>	247
5.21.2.7	<a href="#">trdp_packetSizeMD</a>	248
5.21.2.8	<a href="#">trdp_packetSizePD</a>	248
5.21.2.9	<a href="#">trdp_queueAppLast</a>	248
5.21.2.10	<a href="#">trdp_queueDelElement</a>	248
5.21.2.11	<a href="#">trdp_queueFindComId</a>	249
5.21.2.12	<a href="#">trdp_queueFindPubAddr</a>	249
5.21.2.13	<a href="#">trdp_queueFindSubAddr</a>	249
5.21.2.14	<a href="#">trdp_queueInsFirst</a>	249
5.21.2.15	<a href="#">trdp_releaseSocket</a>	250
5.21.2.16	<a href="#">trdp_requestSocket</a>	250
5.22	<a href="#">vos_mem.c File Reference</a>	252
5.22.1	<a href="#">Detailed Description</a>	253
5.22.2	<a href="#">Function Documentation</a>	254
5.22.2.1	<a href="#">vos_bsearch</a>	254
5.22.2.2	<a href="#">vos_memAlloc</a>	254
5.22.2.3	<a href="#">vos_memCount</a>	255
5.22.2.4	<a href="#">vos_memDelete</a>	255
5.22.2.5	<a href="#">vos_memFree</a>	255
5.22.2.6	<a href="#">vos_memInit</a>	256
5.22.2.7	<a href="#">vos_mutexLocalCreate</a>	256
5.22.2.8	<a href="#">vos_mutexLocalDelete</a>	257
5.22.2.9	<a href="#">vos_qsort</a>	257
5.22.2.10	<a href="#">vos_queueCreate</a>	257
5.22.2.11	<a href="#">vos_queueDestroy</a>	258
5.22.2.12	<a href="#">vos_queueReceive</a>	258
5.22.2.13	<a href="#">vos_queueSend</a>	259
5.22.2.14	<a href="#">vos_strncpy</a>	260
5.22.2.15	<a href="#">vos_strnicmp</a>	260
5.23	<a href="#">vos_mem.h File Reference</a>	261
5.23.1	<a href="#">Detailed Description</a>	263
5.23.2	<a href="#">Define Documentation</a>	263

5.23.2.1	VOS_MEM_BLOCKSIZEs	263
5.23.2.2	VOS_MEM_PREALLOCATE	263
5.23.3	Function Documentation	263
5.23.3.1	vos_bsearch	263
5.23.3.2	vos_memAlloc	264
5.23.3.3	vos_memCount	264
5.23.3.4	vos_memDelete	265
5.23.3.5	vos_memFree	265
5.23.3.6	vos_memInit	265
5.23.3.7	vos_qsort	266
5.23.3.8	vos_queueCreate	267
5.23.3.9	vos_queueDestroy	267
5.23.3.10	vos_queueReceive	268
5.23.3.11	vos_queueSend	269
5.23.3.12	vos_strncpy	270
5.23.3.13	vos_strncmp	270
5.24	vos_private.h File Reference	271
5.24.1	Detailed Description	271
5.24.2	Function Documentation	272
5.24.2.1	vos_mutexLocalCreate	272
5.24.2.2	vos_mutexLocalDelete	272
5.25	vos_private.h File Reference	273
5.25.1	Detailed Description	273
5.25.2	Function Documentation	274
5.25.2.1	vos_mutexLocalCreate	274
5.25.2.2	vos_mutexLocalDelete	274
5.26	vos_shared_mem.c File Reference	275
5.26.1	Detailed Description	276
5.26.2	Function Documentation	276
5.26.2.1	vos_sharedClose	276
5.26.2.2	vos_sharedOpen	276
5.27	vos_shared_mem.c File Reference	278
5.27.1	Detailed Description	279
5.27.2	Function Documentation	279
5.27.2.1	vos_sharedClose	279
5.27.2.2	vos_sharedOpen	279



5.28	<a href="#">vos_shared_mem.h File Reference</a>	281
5.28.1	<a href="#">Detailed Description</a>	281
5.28.2	<a href="#">Function Documentation</a>	282
5.28.2.1	<a href="#">vos_sharedClose</a>	282
5.28.2.2	<a href="#">vos_sharedOpen</a>	283
5.29	<a href="#">vos_sock.c File Reference</a>	285
5.29.1	<a href="#">Detailed Description</a>	287
5.29.2	<a href="#">Function Documentation</a>	288
5.29.2.1	<a href="#">vos_dottedIP</a>	288
5.29.2.2	<a href="#">vos_getInterfaces</a>	288
5.29.2.3	<a href="#">vos_getMacAddress</a>	289
5.29.2.4	<a href="#">vos_htonl</a>	289
5.29.2.5	<a href="#">vos_htons</a>	289
5.29.2.6	<a href="#">vos_ipDotted</a>	289
5.29.2.7	<a href="#">vos_isMulticast</a>	290
5.29.2.8	<a href="#">vos_ntohl</a>	290
5.29.2.9	<a href="#">vos_ntohs</a>	290
5.29.2.10	<a href="#">vos_select</a>	290
5.29.2.11	<a href="#">vos_sockAccept</a>	291
5.29.2.12	<a href="#">vos_sockBind</a>	291
5.29.2.13	<a href="#">vos_sockClose</a>	292
5.29.2.14	<a href="#">vos_sockConnect</a>	292
5.29.2.15	<a href="#">vos_sockGetMAC</a>	292
5.29.2.16	<a href="#">vos_sockInit</a>	293
5.29.2.17	<a href="#">vos_sockJoinMC</a>	293
5.29.2.18	<a href="#">vos_sockLeaveMC</a>	293
5.29.2.19	<a href="#">vos_sockListen</a>	294
5.29.2.20	<a href="#">vos_sockOpenTCP</a>	294
5.29.2.21	<a href="#">vos_sockOpenUDP</a>	295
5.29.2.22	<a href="#">vos_sockReceiveTCP</a>	295
5.29.2.23	<a href="#">vos_sockReceiveUDP</a>	296
5.29.2.24	<a href="#">vos_sockSendTCP</a>	297
5.29.2.25	<a href="#">vos_sockSendUDP</a>	297
5.29.2.26	<a href="#">vos_sockSetBuffer</a>	298
5.29.2.27	<a href="#">vos_sockSetMulticastIf</a>	298
5.29.2.28	<a href="#">vos_sockSetOptions</a>	298

5.29.2.29	<code>vos_sockTerm</code>	299
5.30	<code>vos_sock.c</code> File Reference	300
5.30.1	Detailed Description	302
5.30.2	Function Documentation	303
5.30.2.1	<code>recvmsg</code>	303
5.30.2.2	<code>vos_dottedIP</code>	303
5.30.2.3	<code>vos_getInterfaces</code>	303
5.30.2.4	<code>vos_htonl</code>	304
5.30.2.5	<code>vos_htons</code>	304
5.30.2.6	<code>vos_ipDotted</code>	304
5.30.2.7	<code>vos_isMulticast</code>	305
5.30.2.8	<code>vos_ntohl</code>	305
5.30.2.9	<code>vos_ntohs</code>	305
5.30.2.10	<code>vos_select</code>	305
5.30.2.11	<code>vos_sockAccept</code>	306
5.30.2.12	<code>vos_sockBind</code>	306
5.30.2.13	<code>vos_sockClose</code>	307
5.30.2.14	<code>vos_sockConnect</code>	307
5.30.2.15	<code>vos_sockGetMAC</code>	308
5.30.2.16	<code>vos_sockInit</code>	308
5.30.2.17	<code>vos_sockJoinMC</code>	308
5.30.2.18	<code>vos_sockLeaveMC</code>	309
5.30.2.19	<code>vos_sockListen</code>	309
5.30.2.20	<code>vos_sockOpenTCP</code>	309
5.30.2.21	<code>vos_sockOpenUDP</code>	310
5.30.2.22	<code>vos_sockReceiveTCP</code>	310
5.30.2.23	<code>vos_sockReceiveUDP</code>	311
5.30.2.24	<code>vos_sockSendTCP</code>	312
5.30.2.25	<code>vos_sockSendUDP</code>	312
5.30.2.26	<code>vos_sockSetBuffer</code>	313
5.30.2.27	<code>vos_sockSetMulticastIf</code>	313
5.30.2.28	<code>vos_sockSetOptions</code>	313
5.30.2.29	<code>vos_sockTerm</code>	313
5.31	<code>vos_sock.h</code> File Reference	314
5.31.1	Detailed Description	317
5.31.2	Define Documentation	317

5.31.2.1	VOS_MAX_SOCKET_CNT	317
5.31.2.2	VOS_TTL_MULTICAST	317
5.31.3	Function Documentation	317
5.31.3.1	vos_dottedIP	317
5.31.3.2	vos_getInterfaces	318
5.31.3.3	vos_htonl	319
5.31.3.4	vos_htons	319
5.31.3.5	vos_ipDotted	319
5.31.3.6	vos_isMulticast	320
5.31.3.7	vos_ntohl	320
5.31.3.8	vos_ntohs	320
5.31.3.9	vos_select	321
5.31.3.10	vos_sockAccept	321
5.31.3.11	vos_sockBind	322
5.31.3.12	vos_sockClose	323
5.31.3.13	vos_sockConnect	324
5.31.3.14	vos_sockGetMAC	325
5.31.3.15	vos_sockInit	325
5.31.3.16	vos_sockJoinMC	325
5.31.3.17	vos_sockLeaveMC	326
5.31.3.18	vos_sockListen	327
5.31.3.19	vos_sockOpenTCP	328
5.31.3.20	vos_sockOpenUDP	329
5.31.3.21	vos_sockReceiveTCP	330
5.31.3.22	vos_sockReceiveUDP	331
5.31.3.23	vos_sockSendTCP	332
5.31.3.24	vos_sockSendUDP	333
5.31.3.25	vos_sockSetMulticastIf	334
5.31.3.26	vos_sockSetOptions	335
5.31.3.27	vos_sockTerm	336
5.32	vos_thread.c File Reference	337
5.32.1	Detailed Description	339
5.32.2	Function Documentation	339
5.32.2.1	cyclicThread	339
5.32.2.2	vos_addTime	340
5.32.2.3	vos_clearTime	340

5.32.2.4	<code>vos_cmpTime</code>	340
5.32.2.5	<code>vos_divTime</code>	341
5.32.2.6	<code>vos_getTime</code>	341
5.32.2.7	<code>vos_getTimeStamp</code>	341
5.32.2.8	<code>vos_getUuid</code>	341
5.32.2.9	<code>vos_mulTime</code>	341
5.32.2.10	<code>vos_mutexCreate</code>	342
5.32.2.11	<code>vos_mutexDelete</code>	342
5.32.2.12	<code>vos_mutexLocalCreate</code>	342
5.32.2.13	<code>vos_mutexLocalDelete</code>	343
5.32.2.14	<code>vos_mutexLock</code>	343
5.32.2.15	<code>vos_mutexTryLock</code>	343
5.32.2.16	<code>vos_mutexUnlock</code>	344
5.32.2.17	<code>vos_semaCreate</code>	344
5.32.2.18	<code>vos_semaDelete</code>	344
5.32.2.19	<code>vos_semaGive</code>	345
5.32.2.20	<code>vos_semaTake</code>	345
5.32.2.21	<code>vos_subTime</code>	345
5.32.2.22	<code>vos_threadCreate</code>	345
5.32.2.23	<code>vos_threadDelay</code>	346
5.32.2.24	<code>vos_threadInit</code>	346
5.32.2.25	<code>vos_threadIsActive</code>	347
5.32.2.26	<code>vos_threadTerm</code>	347
5.32.2.27	<code>vos_threadTerminate</code>	347
5.33	<code>vos_thread.c</code> File Reference	348
5.33.1	Detailed Description	350
5.33.2	Function Documentation	351
5.33.2.1	<code>cyclicThread</code>	351
5.33.2.2	<code>vos_addTime</code>	351
5.33.2.3	<code>vos_clearTime</code>	351
5.33.2.4	<code>vos_cmpTime</code>	351
5.33.2.5	<code>vos_divTime</code>	352
5.33.2.6	<code>vos_getFreeThreadHandle</code>	352
5.33.2.7	<code>vos_getTime</code>	352
5.33.2.8	<code>vos_getTimeStamp</code>	352
5.33.2.9	<code>vos_getUuid</code>	353

5.33.2.10	<a href="#">vos_mulTime</a>	353
5.33.2.11	<a href="#">vos_mutexCreate</a>	353
5.33.2.12	<a href="#">vos_mutexDelete</a>	354
5.33.2.13	<a href="#">vos_mutexLocalCreate</a>	354
5.33.2.14	<a href="#">vos_mutexLocalDelete</a>	354
5.33.2.15	<a href="#">vos_mutexLock</a>	354
5.33.2.16	<a href="#">vos_mutexTryLock</a>	355
5.33.2.17	<a href="#">vos_mutexUnlock</a>	355
5.33.2.18	<a href="#">vos_semaCreate</a>	355
5.33.2.19	<a href="#">vos_semaDelete</a>	356
5.33.2.20	<a href="#">vos_semaGive</a>	356
5.33.2.21	<a href="#">vos_semaTake</a>	356
5.33.2.22	<a href="#">vos_subTime</a>	357
5.33.2.23	<a href="#">vos_threadCreate</a>	357
5.33.2.24	<a href="#">vos_threadDelay</a>	358
5.33.2.25	<a href="#">vos_threadInit</a>	358
5.33.2.26	<a href="#">vos_threadIsActive</a>	358
5.33.2.27	<a href="#">vos_threadTerm</a>	358
5.33.2.28	<a href="#">vos_threadTerminate</a>	359
5.34	<a href="#">vos_thread.h File Reference</a>	360
5.34.1	<a href="#">Detailed Description</a>	363
5.34.2	<a href="#">Function Documentation</a>	363
5.34.2.1	<a href="#">vos_addTime</a>	363
5.34.2.2	<a href="#">vos_clearTime</a>	363
5.34.2.3	<a href="#">vos_cmpTime</a>	363
5.34.2.4	<a href="#">vos_divTime</a>	364
5.34.2.5	<a href="#">vos_getTime</a>	364
5.34.2.6	<a href="#">vos_getTimeStamp</a>	365
5.34.2.7	<a href="#">vos_getUuid</a>	365
5.34.2.8	<a href="#">vos_mulTime</a>	365
5.34.2.9	<a href="#">vos_mutexCreate</a>	366
5.34.2.10	<a href="#">vos_mutexDelete</a>	366
5.34.2.11	<a href="#">vos_mutexLock</a>	367
5.34.2.12	<a href="#">vos_mutexTryLock</a>	367
5.34.2.13	<a href="#">vos_mutexUnlock</a>	368
5.34.2.14	<a href="#">vos_semaCreate</a>	368

5.34.2.15	<a href="#">vos_semaDelete</a>	369
5.34.2.16	<a href="#">vos_semaGive</a>	369
5.34.2.17	<a href="#">vos_semaTake</a>	370
5.34.2.18	<a href="#">vos_subTime</a>	370
5.34.2.19	<a href="#">vos_threadCreate</a>	371
5.34.2.20	<a href="#">vos_threadDelay</a>	372
5.34.2.21	<a href="#">vos_threadInit</a>	373
5.34.2.22	<a href="#">vos_threadIsActive</a>	373
5.34.2.23	<a href="#">vos_threadTerm</a>	373
5.34.2.24	<a href="#">vos_threadTerminate</a>	374
5.35	<a href="#">vos_types.h File Reference</a>	375
5.35.1	<a href="#">Detailed Description</a>	376
5.35.2	<a href="#">Typedef Documentation</a>	377
5.35.2.1	<a href="#">VOS_PRINT_DBG_T</a>	377
5.35.3	<a href="#">Enumeration Type Documentation</a>	377
5.35.3.1	<a href="#">VOS_ERR_T</a>	377
5.35.3.2	<a href="#">VOS_LOG_T</a>	378
5.36	<a href="#">vos_utils.c File Reference</a>	379
5.36.1	<a href="#">Detailed Description</a>	379
5.36.2	<a href="#">Function Documentation</a>	380
5.36.2.1	<a href="#">vos_crc32</a>	380
5.36.2.2	<a href="#">vos_init</a>	380
5.36.2.3	<a href="#">vos_initRuntimeConsts</a>	381
5.36.2.4	<a href="#">vos_isBigEndian</a>	381
5.36.2.5	<a href="#">vos_terminate</a>	381
5.37	<a href="#">vos_utils.h File Reference</a>	382
5.37.1	<a href="#">Detailed Description</a>	383
5.37.2	<a href="#">Define Documentation</a>	383
5.37.2.1	<a href="#">VOS_MAX_ERR_STR_SIZE</a>	383
5.37.2.2	<a href="#">VOS_MAX_FRMT_SIZE</a>	383
5.37.2.3	<a href="#">VOS_MAX_PRNT_STR_SIZE</a>	383
5.37.3	<a href="#">Function Documentation</a>	384
5.37.3.1	<a href="#">vos_crc32</a>	384
5.37.3.2	<a href="#">vos_init</a>	384
5.37.3.3	<a href="#">vos_terminate</a>	385

# 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

#### 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

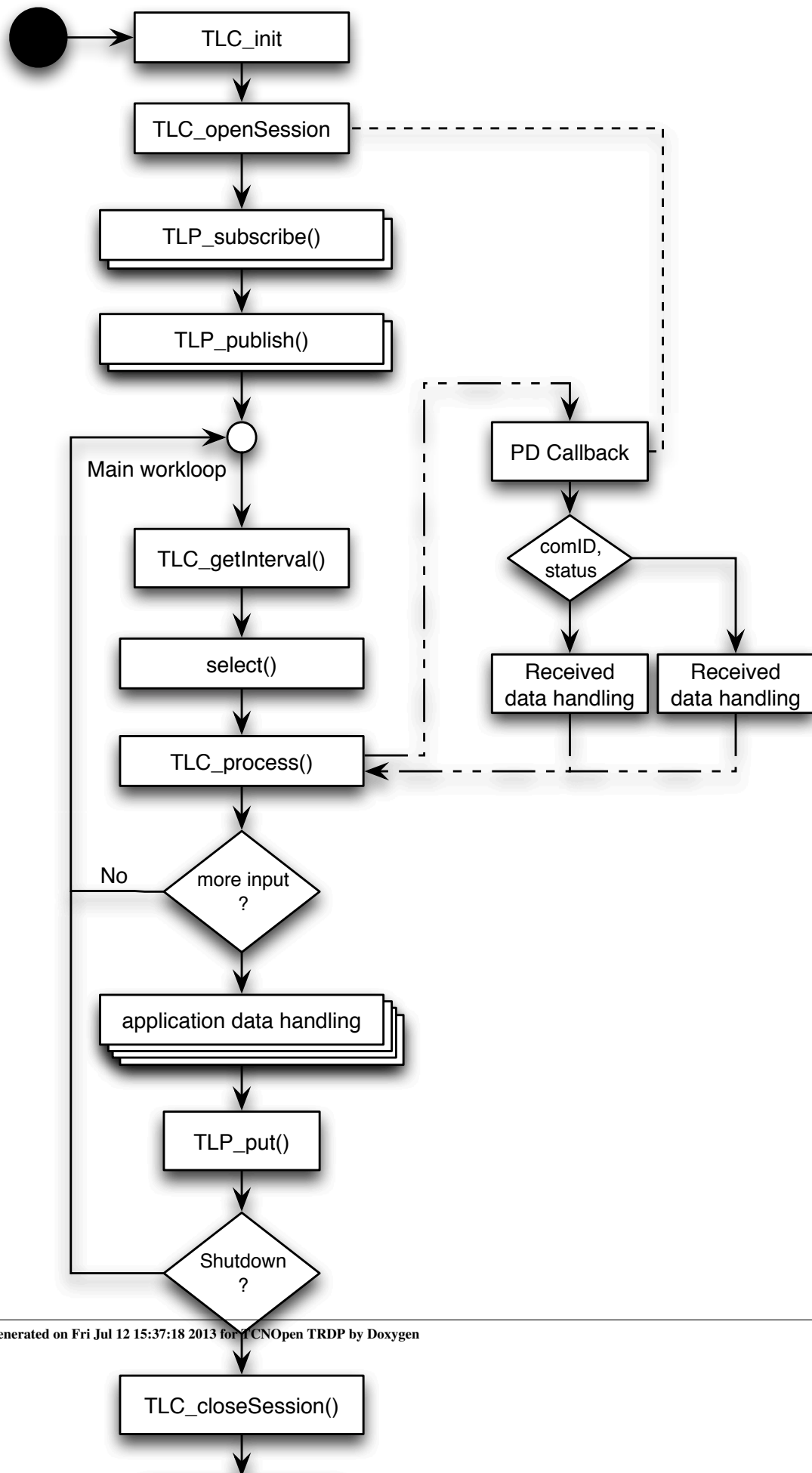
## 1.2 Terminology

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.

The following diagram shows how these pieces of software are interrelated.





## 1.3 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.

## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">GNU_PACKED</a> (TRDP process data header - network order and alignment ) . . . . .	9
<a href="#">PD_ELE</a> (Queue element for PD packets to send or receive ) . . . . .	12
<a href="#">TAU_MARSHALL_INFO_T</a> (Marshalling info, used to and from wire ) . . . . .	15
<a href="#">TRDP_CAR_INFO_T</a> (Car information structure ) . . . . .	16
<a href="#">TRDP_COMID_DSID_MAP_T</a> (ComId - data set mapping element definition ) . . . . .	18
<a href="#">TRDP_CST_INFO_T</a> (Consist information structure ) . . . . .	19
<a href="#">TRDP_DATASET</a> (Dataset definition ) . . . . .	21
<a href="#">TRDP_DATASET_ELEMENT_T</a> (Dataset element definition ) . . . . .	22
<a href="#">TRDP_DBG_CONFIG_T</a> (Control for debug output device/file on application level ) . . . . .	23
<a href="#">TRDP_DEVICE_INFO_T</a> (Device information structure ) . . . . .	24
<a href="#">TRDP_FCT_INFO_T</a> (Device information structure ) . . . . .	26
<a href="#">TRDP_HANDLE</a> (Hidden handle definition, used as unique addressing item ) . . . . .	27
<a href="#">TRDP_LIST_STATISTICS_T</a> (Information about a particular MD listener ) . . . . .	28
<a href="#">TRDP_MARSHALL_CONFIG_T</a> (Marshaling/unmarshalling configuration ) . . . . .	29
<a href="#">TRDP_MD_CONFIG_T</a> (Default MD configuration ) . . . . .	30
<a href="#">TRDP_MD_INFO_T</a> (Message data info from received telegram; allows the application to generate responses ) . . . . .	32
<a href="#">TRDP_MD_STATISTICS_T</a> (Structure containing all general MD statistics information ) . . . .	34
<a href="#">TRDP_MEM_CONFIG_T</a> (Enumeration type for memory pre-fragmentation, reuse of VOS definition ) . . . . .	36
<a href="#">TRDP_MEM_STATISTICS_T</a> (TRDP statistics type definitions ) . . . . .	37
<a href="#">TRDP_PD_CONFIG_T</a> (Default PD configuration ) . . . . .	38
<a href="#">TRDP_PD_INFO_T</a> (Process data info from received telegram; allows the application to generate responses ) . . . . .	39
<a href="#">TRDP_PD_STATISTICS_T</a> (Structure containing all general PD statistics information ) . . . .	41
<a href="#">TRDP_PROCESS_CONFIG_T</a> (Various flags/general TRDP options for library initialization ) .	43
<a href="#">TRDP_PROP_INFO_T</a> (Properties information structure ) . . . . .	44
<a href="#">TRDP_PUB_STATISTICS_T</a> (Table containing particular PD publishing information ) . . . . .	45
<a href="#">TRDP_RED_STATISTICS_T</a> (A table containing PD redundant group information ) . . . . .	46
<a href="#">TRDP_SDT_PAR_T</a> (Types to read out the XML configuration ) . . . . .	47
<a href="#">TRDP_SEND_PARAM_T</a> (Quality/type of service and time to live ) . . . . .	48
<a href="#">TRDP_SESSION</a> (Session/application variables store ) . . . . .	49
<a href="#">TRDP_SOCKET_TCP</a> (TCP parameters ) . . . . .	51

<a href="#">TRDP_SOCKETS</a> (Socket item) . . . . .	<a href="#">52</a>
<a href="#">TRDP_STATISTICS_T</a> (Structure containing all general memory, PD and MD statistics information) . . . . .	<a href="#">54</a>
<a href="#">TRDP_SUBS_STATISTICS_T</a> (Table containing particular PD subscription information) . . .	<a href="#">56</a>
<a href="#">TRDP_TRAIN_INFO_T</a> (Train information structure) . . . . .	<a href="#">58</a>
<a href="#">TRDP_VERSION_T</a> (Version information) . . . . .	<a href="#">60</a>
<a href="#">TRDP_XML_DOC_HANDLE_T</a> (Parsed XML document handle) . . . . .	<a href="#">61</a>
<a href="#">VOS SOCK_OPT_T</a> (Common socket options) . . . . .	<a href="#">62</a>
<a href="#">VOS_TIME_T</a> (Timer value compatible with timeval / select) . . . . .	<a href="#">63</a>

# Chapter 3

## File Index

### 3.1 File List

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

<a href="#">tau_addr.h</a> (TRDP utility interface definitions ) . . . . .	65
<a href="#">tau_marshall.c</a> (Marshalling functions for TRDP ) . . . . .	75
<a href="#">tau_marshall.h</a> (TRDP utility interface definitions ) . . . . .	81
<a href="#">tau_tti.h</a> (TRDP utility interface definitions ) . . . . .	87
<a href="#">tau_xml.c</a> (Functions for XML file parsing ) . . . . .	95
<a href="#">tau_xml.h</a> (TRDP utility interface definitions ) . . . . .	100
<a href="#">trdp_dllmain.c</a> (Windows DLL main function ) . . . . .	106
<a href="#">trdp_if.c</a> (Functions for ECN communication ) . . . . .	107
<a href="#">trdp_if.h</a> (Typedefs for TRDP communication ) . . . . .	129
<a href="#">trdp_if_light.h</a> (TRDP Light interface functions (API) ) . . . . .	131
<a href="#">trdp_mdcom.c</a> (Functions for MD communication ) . . . . .	172
<a href="#">trdp_mdcom.h</a> (Functions for MD communication ) . . . . .	181
<a href="#">trdp_pdcom.c</a> (Functions for PD communication ) . . . . .	189
<a href="#">trdp_pdcom.h</a> (Functions for PD communication ) . . . . .	197
<a href="#">trdp_private.h</a> (Typedefs for TRDP communication ) . . . . .	205
<a href="#">trdp_proto.h</a> (Definitions for the TRDP protocol ) . . . . .	210
<a href="#">trdp_stats.c</a> (Statistics functions for TRDP communication ) . . . . .	215
<a href="#">trdp_stats.h</a> (Statistics for TRDP communication ) . . . . .	222
<a href="#">trdp_types.h</a> (Typedefs for TRDP communication ) . . . . .	225
<a href="#">trdp_utils.c</a> (Helper functions for TRDP communication ) . . . . .	235
<a href="#">trdp_utils.h</a> (Common utilities for TRDP communication ) . . . . .	244
<a href="#">vos_mem.c</a> (Memory functions ) . . . . .	252
<a href="#">vos_mem.h</a> (Memory and queue functions for OS abstraction ) . . . . .	261
<a href="#">posix/vos_private.h</a> (Private definitions for the OS abstraction layer ) . . . . .	271
<a href="#">windows/vos_private.h</a> (Private definitions for the OS abstraction layer ) . . . . .	273
<a href="#">posix/vos_shared_mem.c</a> (Shared Memory functions ) . . . . .	275
<a href="#">windows/vos_shared_mem.c</a> (Shared Memory functions ) . . . . .	278
<a href="#">vos_shared_mem.h</a> (Shared Memory functions for OS abstraction ) . . . . .	281
<a href="#">posix/vos_sock.c</a> (Socket functions ) . . . . .	285
<a href="#">windows/vos_sock.c</a> (Socket functions ) . . . . .	300
<a href="#">vos_sock.h</a> (Typedefs for OS abstraction ) . . . . .	314
<a href="#">posix/vos_thread.c</a> (Multitasking functions ) . . . . .	337
<a href="#">windows/vos_thread.c</a> (Multitasking functions ) . . . . .	348

<a href="#">vos_thread.h</a> (Threading functions for OS abstraction) . . . . .	360
<a href="#">vos_types.h</a> (Typedefs for OS abstraction) . . . . .	375
<a href="#">vos_utils.c</a> (Common functions for VOS) . . . . .	379
<a href="#">vos_utils.h</a> (Typedefs for OS abstraction) . . . . .	382

## Chapter 4

# Data Structure Documentation

### 4.1 GNU\_PACKED Struct Reference

TRDP process data header - network order and alignment.

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

#### Data Fields

- UINT32 [sequenceCounter](#)  
*Unique counter (autom incremented).*
- UINT16 [protocolVersion](#)  
*fix value for compatibility (set by the API)*
- UINT16 [msgType](#)  
*of datagram: PD Request (0x5072) or PD\_MSG (0x5064)*
- UINT32 [comId](#)  
*set by user: unique id*
- UINT32 [topoCount](#)  
*set by user: ETB to use, '0' to deactivate*
- UINT32 [datasetLength](#)  
*length of the data to transmit 0.*
- UINT32 [reserved](#)  
*before used for ladder support*
- UINT32 [replyComId](#)  
*used in PD request*
- UINT32 [replyIpAddress](#)  
*used for PD request*

- UINT32 [frameChecksum](#)  
*CRC32 of header.*
- INT32 [replyStatus](#)  
*0 = OK*
- UINT8 [sessionID](#) [16]  
*UUID as a byte stream.*
- UINT32 [replyTimeout](#)  
*in us*
- UINT8 [sourceURI](#) [32]  
*User part of URI.*
- UINT8 [destinationURI](#) [32]  
*User part of URI.*
- PD\_HEADER\_T [frameHead](#)  
*Packet header in network byte order.*
- UINT8 [data](#) [TRDP\_MAX\_PD\_PACKET\_SIZE]  
*data ready to be sent or received (with CRCs)*

### 4.1.1 Detailed Description

TRDP process data header - network order and alignment.

TRDP PD packet.

TRDP message data header - network order and alignment.

### 4.1.2 Field Documentation

#### 4.1.2.1 UINT16 GNU\_PACKED::protocolVersion

fix value for compatibility (set by the API)

fix value for compatibility

#### 4.1.2.2 UINT16 GNU\_PACKED::msgType

of datagram: PD Request (0x5072) or PD\_MSG (0x5064)

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



#### 4.1.2.3 UINT32 GNU\_PACKED::datasetLength

length of the data to transmit 0.

defined by user: length of data to transmit

..1436 without padding and FCS

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

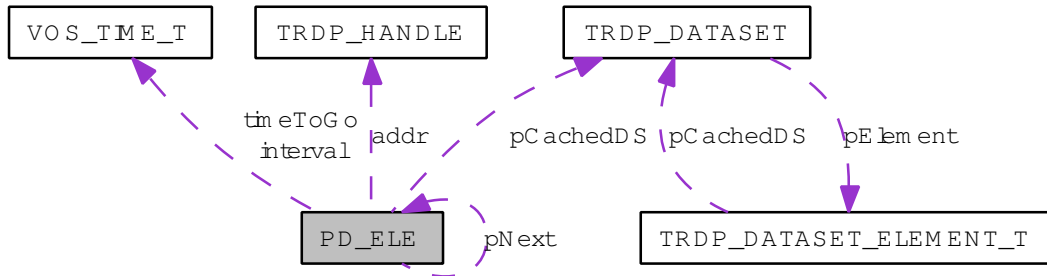
- [trdp\\_proto.h](#)
- [trdp\\_private.h](#)

## 4.2 PD\_ELE Struct Reference

Queue element for PD packets to send or receive.

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

Collaboration diagram for PD\_ELE:



### Data Fields

- struct [PD\\_ELE](#) \* [pNext](#)  
*pointer to next element or NULL*
- [UINT32](#) [magic](#)  
*prevent acces through dangeling pointer*
- [TRDP\\_ADDRESSES\\_T](#) [addr](#)  
*handle of publisher/subscriber*
- [TRDP\\_IP\\_ADDR\\_T](#) [pullIpAddress](#)  
*In case of pulling a PD this is the requested Ip.*
- [UINT32](#) [redId](#)  
*Redundancy group ID or zero.*
- [UINT32](#) [curSeqCnt](#)  
*the last sent or received sequence counter*
- [UINT32](#) [curSeqCnt4Pull](#)  
*the last sent sequence counter for PULL*
- [UINT32](#) [numRxTx](#)  
*Counter for received packets (statistics).*
- [UINT32](#) [updPkts](#)  
*Counter for updated packets (statistics).*
- [UINT32](#) [getPkts](#)  
*Counter for read packets (statistics).*

- [TRDP\\_ERR\\_T lastErr](#)  
*Last error (timeout).*
- [TRDP\\_PRIV\\_FLAGS\\_T privFlags](#)  
*private flags*
- [TRDP\\_FLAGS\\_T pktFlags](#)  
*flags*
- [TRDP\\_TIME\\_T interval](#)  
*time out value for received packets or interval for packets to send (set from ms)*
- [TRDP\\_TIME\\_T timeToGo](#)  
*next time this packet must be sent/rcv*
- [TRDP\\_TO\\_BEHAVIOR\\_T toBehavior](#)  
*timeout behavior for packets*
- [UINT32 dataSize](#)  
*net data size*
- [UINT32 grossSize](#)  
*complete packet size (header, data, padding, FCS)*
- [UINT32 sendSize](#)  
*data size sent out*
- [TRDP\\_DATASET\\_T \\* pCachedDS](#)  
*Pointer to dataset element if known.*
- [INT32 socketIdx](#)  
*index into the socket list*
- [const void \\* userRef](#)  
*from subscribe()*
- [PD\\_PACKET\\_T \\* pFrame](#)  
*header.*

### 4.2.1 Detailed Description

Queue element for PD packets to send or receive.

### 4.2.2 Field Documentation

#### 4.2.2.1 PD\_PACKET\_T\* PD\_ELE::pFrame

header .

.. data + FCS...

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

- [trdp\\_private.h](#)

## 4.3 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 \* [pDst](#)  
*destination pointer*
- UINT8 \* [pDstEnd](#)  
*last destination*

### 4.3.1 Detailed Description

Marshalling info, used to and from wire.

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

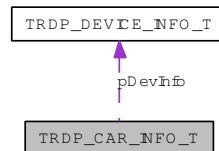
- [tau\\_marshall.c](#)

## 4.4 TRDP\_CAR\_INFO\_T Struct Reference

car information structure.

```
#include <tau_tti.h>
```

Collaboration diagram for TRDP\_CAR\_INFO\_T:



### Data Fields

- TRDP\_LABEL\_T [id](#)  
*Unique car identifier (Label) / IEC identification number.*
- TRDP\_LABEL\_T [type](#)  
*car type*
- UINT8 [orient](#)  
*0 == opposite, 1 == same orientation rel.*
- UINT8 [lead](#)  
*0 == car is not leading*
- UINT8 [leadDir](#)  
*0 == leading direction 1, 1 == leading direction 2*
- UINT8 [no](#)  
*sequence number of car in consist*
- UINT8 [iecNo](#)  
*IEC sequence number of car in train.*
- UINT8 [reachable](#)  
*0 == car not reachable, inserted manually*
- UINT16 [devCnt](#)  
*number of devices in the car*
- TRDP\_DEVICE\_INFO\_T \* [pDevInfo](#)  
*Pointer to device info list for application use and convenience.*
- UINT16 [propLen](#)  
*car property length*
- UINT8 \* [pProp](#)  
*Pointer to car properties for application use and convenience.*

### 4.4.1 Detailed Description

car information structure.

### 4.4.2 Field Documentation

#### 4.4.2.1 `UINT8 TRDP_CAR_INFO_T::orient`

0 == opposite, 1 == same orientation rel.

to consist

#### 4.4.2.2 `TRDP_DEVICE_INFO_T* TRDP_CAR_INFO_T::pDevInfo`

Pointer to device info list for application use and convenience.

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

- [tau\\_tti.h](#)

## 4.5 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*

### 4.5.1 Detailed Description

ComId - data set mapping element definition.

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

- [trdp\\_types.h](#)

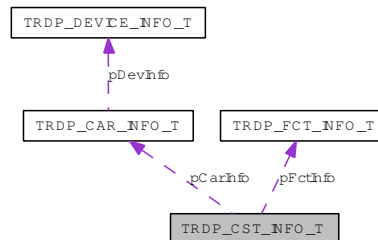


## 4.6 TRDP\_CST\_INFO\_T Struct Reference

consist information structure.

```
#include <tau_tti.h>
```

Collaboration diagram for TRDP\_CST\_INFO\_T:



### Data Fields

- [TRDP\\_LABEL\\_T id](#)  
*Unique consist identifier (Label) / IEC identification number taken from 1st car in consist.*
- [TRDP\\_LABEL\\_T owner](#)  
*consist owner, e.g.*
- [TRDP\\_UUID\\_T uuid](#)  
*consist UUID for inauguration purposes*
- [UINT8 orient](#)  
*opposite(0) or same(1) orientation rel.*
- [UINT8 lead](#)  
*0 == consist is not leading*
- [UINT8 leadDir](#)  
*0 == leading direction 1, 1 == leading direction 2*
- [UINT8 tcnNo](#)  
*sequence number of consist in train*
- [UINT8 iecNo](#)  
*IEC sequence number of consist in train.*
- [UINT8 reachable](#)  
*0 == consist not reachable, inserted manually*
- [UINT8 ecnCnt](#)  
*number of cars in the consist*
- [UINT8 etbCnt](#)

*number of cars in the consist*

- `UINT16 fctCnt`  
*number of public functions in the consist*
- `TRDP_FCT_INFO_T * pFctInfo`  
*Pointer to function info list for application use and convenience.*
- `UINT16 carCnt`  
*number of cars in the consist*
- `TRDP_CAR_INFO_T * pCarInfo`  
*Pointer to car info list for application use and convenience.*
- `UINT16 propLen`  
*consist property length*
- `UINT8 * pProp`  
*Pointer to consist properties for application use and convenience.*

### 4.6.1 Detailed Description

consist information structure.

### 4.6.2 Field Documentation

#### 4.6.2.1 `TRDP_LABEL_T TRDP_CST_INFO_T::owner`

consist owner, e.g.

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

#### 4.6.2.2 `UINT8 TRDP_CST_INFO_T::orient`

opposite(0) or same(1) orientation rel.

to train

#### 4.6.2.3 `TRDP_FCT_INFO_T* TRDP_CST_INFO_T::pFctInfo`

Pointer to function info list for application use and convenience.

#### 4.6.2.4 `TRDP_CAR_INFO_T* TRDP_CST_INFO_T::pCarInfo`

Pointer to car info list for application use and convenience.

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

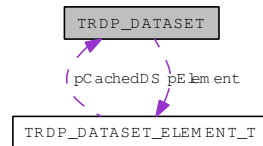
- [tau\\_tti.h](#)

## 4.7 TRDP\_DATASET Struct Reference

Dataset definition.

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

Collaboration diagram for TRDP\_DATASET:



### Data Fields

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

#### 4.7.1 Detailed Description

Dataset definition.

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

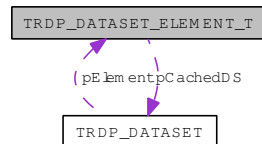
- [trdp\\_types.h](#)

## 4.8 TRDP\_DATASET\_ELEMENT\_T Struct Reference

Dataset element definition.

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

Collaboration diagram for TRDP\_DATASET\_ELEMENT\_T:



### Data Fields

- [UINT32 type](#)  
*Data type (TRDP\_DATA\_TYPE\_T 1.*
- [UINT32 size](#)  
*Number of items or TDRP\_VAR\_SIZE (0).*
- [struct TRDP\\_DATASET \\* pCachedDS](#)  
*Used internally for marshalling speed-up.*

### 4.8.1 Detailed Description

Dataset element definition.

### 4.8.2 Field Documentation

#### 4.8.2.1 [UINT32 TRDP\\_DATASET\\_ELEMENT\\_T::type](#)

Data type (TRDP\_DATA\_TYPE\_T 1.

..99) or dataset id > 1000

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

- [trdp\\_types.h](#)

## 4.9 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.*

### 4.9.1 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](#)

## 4.10 TRDP\_DEVICE\_INFO\_T Struct Reference

device information structure

```
#include <tau_tti.h>
```

### Data Fields

- TRDP\_IP\_ADDR [addr1](#)  
*First device IP address.*
- TRDP\_IP\_ADDR [addr2](#)  
*Second device IP address.*
- TRDP\_LABEL\_T [id](#)  
*consist unique device identifier (Label) / host name*
- TRDP\_LABEL\_T [type](#)  
*device type (reserved key words ETBN, ETBR, FCT)*
- UINT8 [orient](#)  
*device orientation 0=opposite, 1=same rel.*
- TRDP\_LABEL\_T [redId](#)  
*redundant device Id if available*
- UINT8 [ecnId1](#)  
*First consist network id the device is connected to.*
- UINT8 [ecnId2](#)  
*Second consist network id the device is connected to.*
- UINT8 [etbId1](#)  
*First Ethernet train backbone id.*
- UINT8 [etbId2](#)  
*Second Ethernet train backbone id.*
- UINT16 [fctCnt](#)  
*number of public functions on the device*
- UINT32 \* [pFctNo](#)  
*Pointer to function number list for application use and convenience.*
- UINT16 [propLen](#)  
*device property length*
- UINT8 \* [pProp](#)  
*Pointer to device properties for application use and convenience.*

### 4.10.1 Detailed Description

device information structure

### 4.10.2 Field Documentation

#### 4.10.2.1 UINT8 TRDP\_DEVICE\_INFO\_T::orient

device orientation 0=opposite, 1=same rel.

to car

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

- [tau\\_tti.h](#)

## 4.11 TRDP\_FCT\_INFO\_T Struct Reference

device information structure

```
#include <tau_tti.h>
```

### Data Fields

- [TRDP\\_LABEL\\_T id](#)  
*function identifier (name)*
- [TRDP\\_FCT\\_T type](#)  
*function type*
- [UINT32 no](#)  
*unique function number in consist, should be the list index number*
- [TRDP\\_IP\\_ADDR addr](#)  
*Device IP address/multicast address.*
- [UINT8 ecnId](#)  
*Consist network id the device is connected to.*
- [UINT8 etbId](#)  
*Ethernet train backbone id.*

### 4.11.1 Detailed Description

device information structure

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

- [tau\\_tti.h](#)



## 4.12 TRDP\_HANDLE Struct Reference

Hidden handle definition, used as unique addressing item.

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

### Data Fields

- [UINT32 comId](#)  
*comId for packets to send/receive*
- [TRDP\\_IP\\_ADDR\\_T srcIpAddr](#)  
*source IP for PD*
- [TRDP\\_IP\\_ADDR\\_T destIpAddr](#)  
*destination IP for PD*
- [TRDP\\_IP\\_ADDR\\_T mcGroup](#)  
*multicast group to join for PD*
- [UINT32 topoCount](#)  
*topocount belongs to addressing item*

### 4.12.1 Detailed Description

Hidden handle definition, used as unique addressing item.

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

- [trdp\\_private.h](#)

## 4.13 TRDP\_LIST\_STATISTICS\_T Struct Reference

Information about a particular MD listener.

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

### Data Fields

- [UINT32 comId](#)  
*ComId to listen to.*
- [TRDP\\_URI\\_USER\\_T uri](#)  
*URI user part to listen to.*
- [TRDP\\_IP\\_ADDR\\_T joinedAddr](#)  
*Joined IP address.*
- [UINT32 callBack](#)  
*Call back function reference if used.*
- [UINT32 queue](#)  
*Queue reference if used.*
- [UINT32 userRef](#)  
*User reference if used.*
- [UINT32 numRecv](#)  
*Number of received packets.*

### 4.13.1 Detailed Description

Information about a particular MD listener.

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

- [trdp\\_types.h](#)

## 4.14 TRDP\_MARSHALL\_CONFIG\_T Struct Reference

Marshaling/unmarshalling configuration.

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

### 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.*

### 4.14.1 Detailed Description

Marshaling/unmarshalling configuration.

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

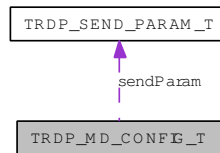
- [trdp\\_types.h](#)

## 4.15 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\\_SEND\\_PARAM\\_T sendParam](#)  
*Default send parameters.*
- [TRDP\\_FLAGS\\_T flags](#)  
*Default flags for MD packets.*
- [UINT32 replyTimeout](#)  
*Default reply timeout in us.*
- [UINT32 confirmTimeout](#)  
*Default confirmation timeout in us.*
- [UINT32 connectTimeout](#)  
*Default connection timeout in us.*
- [UINT32 sendingTimeout](#)  
*Default sending timeout in us.*
- [UINT16 udpPort](#)  
*Port to be used for UDP MD communication.*
- [UINT16 tcpPort](#)  
*Port to be used for TCP MD communication.*
- [UINT32 maxNumSessions](#)  
*Maximal number of replier sessions.*

### 4.15.1 Detailed Description

Default MD configuration.

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

- [trdp\\_types.h](#)

## 4.16 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 topoCount](#)  
*received topocount*
- [BOOL 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 destURI`  
*destination URI user part from MD header*
- `TRDP_URI_USER_T srcURI`  
*source URI user part from MD header*
- `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*

### 4.16.1 Detailed Description

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

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

### 4.16.2 Field Documentation

#### 4.16.2.1 `TRDP_MSG_T TRDP_MD_INFO_T::msgType`

Protocol ('PD', 'MD', .  
..)

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

- `trdp_types.h`

## 4.17 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*



### 4.17.1 Detailed Description

Structure containing all general MD statistics information.

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

- [trdp\\_types.h](#)

## 4.18 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_NBBLOCKSIZES]`  
*memory block structure*

### 4.18.1 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](#)

## 4.19 TRDP\_MEM\_STATISTICS\_T Struct Reference

TRDP statistics type definitions.

```
#include <trdp_types.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](#) [VOS\_MEM\_NBLOCKSIZES]  
*preallocated memory blocks*
- [UINT32 usedBlockSize](#) [VOS\_MEM\_NBLOCKSIZES]  
*used memory blocks*

### 4.19.1 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, toBehaviour, 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 all general memory statistics information.

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

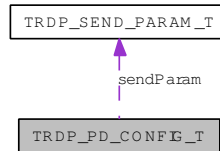
- [trdp\\_types.h](#)

## 4.20 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\\_SEND\\_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 behaviour.*
- UINT16 [port](#)  
*Port to be used for PD communication.*

### 4.20.1 Detailed Description

Default PD configuration.

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

- [trdp\\_types.h](#)

## 4.21 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 topoCount](#)  
*received 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*

### 4.21.1 Detailed Description

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

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

## 4.21.2 Field Documentation

### 4.21.2.1 TRDP\_MSG\_T TRDP\_PD\_INFO\_T::msgType

Protocol ('PD', 'MD', .

..)

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

- [trdp\\_types.h](#)

## 4.22 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*

### 4.22.1 Detailed Description

Structure containing all general PD statistics information.

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

- [trdp\\_types.h](#)



## 4.23 TRDP\_PROCESS\_CONFIG\_T Struct Reference

Various flags/general TRDP options for library initialization.

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

### Data Fields

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

### 4.23.1 Detailed Description

Various flags/general TRDP options for library initialization.

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

- [trdp\\_types.h](#)

## 4.24 TRDP\_PROP\_INFO\_T Struct Reference

properties information structure

```
#include <tau_tti.h>
```

### Data Fields

- [UINT32 crc](#)  
*property CRC*
- [UINT16 len](#)  
*function type*
- [UINT8 ver](#)  
*property version*
- [UINT8 rel](#)  
*property release*
- [UINT8 data](#) [1]  
*dummy field for data access*

### 4.24.1 Detailed Description

properties information structure

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

- [tau\\_tti.h](#)

## 4.25 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.*

### 4.25.1 Detailed Description

Table containing particular PD publishing information.

### 4.25.2 Field Documentation

#### 4.25.2.1 TRDP\_IP\_ADDR\_T TRDP\_PUB\_STATISTICS\_T::destAddr

IP address of destination for this publishing.

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

- [trdp\\_types.h](#)

## 4.26 TRDP\_RED\_STATISTICS\_T Struct Reference

A table containing PD redundant group information.

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

### Data Fields

- [UINT32 id](#)  
*Redundant Id.*
- [TRDP\\_RED\\_STATE\\_T state](#)  
*Redundant state.Leader or Follower.*

### 4.26.1 Detailed Description

A table containing PD redundant group information.

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

- [trdp\\_types.h](#)

## 4.27 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 reserved2](#)  
*Reserved for future use.*

### 4.27.1 Detailed Description

Types to read out the XML configuration.

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

- [tau\\_xml.h](#)

## 4.28 TRDP\_SEND\_PARAM\_T Struct Reference

Quality/type of service and time to live.

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

### Data Fields

- `UINT8 qos`  
*Quality of service (default should be 5 for PD and 3 for MD).*
- `UINT8 ttl`  
*Time to live (default should be 64).*

### 4.28.1 Detailed Description

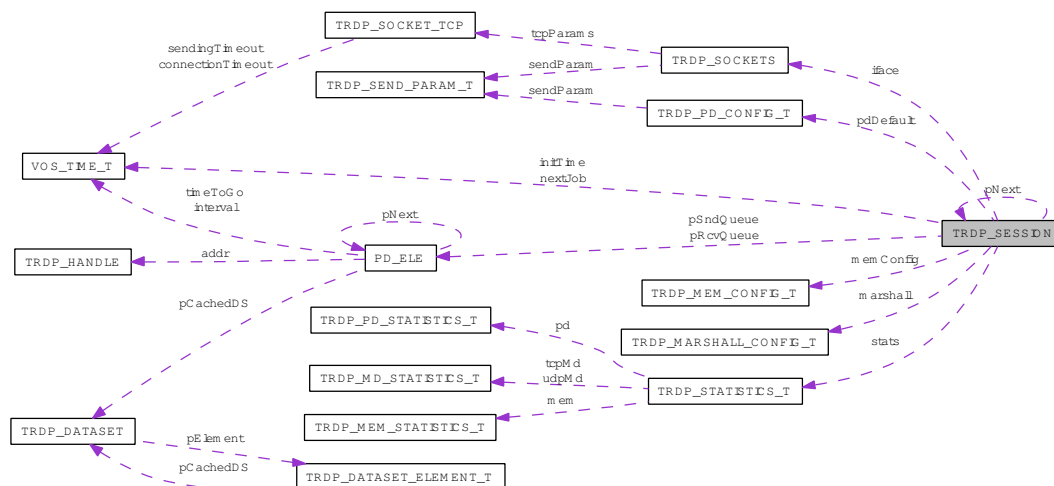
Quality/type of service and time to live.

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

- [trdp\\_types.h](#)

Session/application variables store.

Collaboration diagram for TRDP\_SESSION:



- struct TRDP\_SESSION \* pNext  
*Pointer to next session.*
- VOS\_MUTEX\_T mutex  
*protect this session*
- TRDP\_IP\_ADDR\_T realIP  
*Real IP address.*
- TRDP\_IP\_ADDR\_T virtualIP  
*Virtual IP address.*
- BOOL beQuiet  
*if set, only react on ownIP requests*
- UINT32 redID  
*redundant comId*
- UINT32 topoCount  
*current valid topocount or zero*
- TRDP\_TIME\_T nextJob  
*Store for next select interval.*

- [TRDP\\_PD\\_CONFIG\\_T pdDefault](#)  
*Default configuration for process data.*
- [TRDP\\_SOCKETS\\_T iface](#) [VOS\_MAX\_SOCKET\_CNT]  
*Collection of sockets to use.*
- [PD\\_ELE\\_T \\* pSndQueue](#)  
*pointer to first element of send queue*
- [PD\\_ELE\\_T \\* pRcvQueue](#)  
*pointer to first element of rcv queue*
- [TRDP\\_TIME\\_T initTime](#)  
*initialization time of session*
- [TRDP\\_STATISTICS\\_T stats](#)  
*statistics of this session*

#### 4.29.1 Detailed Description

Session/application variables store.

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

- [trdp\\_private.h](#)

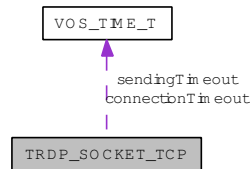


## 4.30 TRDP\_SOCKET\_TCP Struct Reference

TCP parameters.

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

Collaboration diagram for TRDP\_SOCKET\_TCP:



### Data Fields

- [TRDP\\_IP\\_ADDR\\_T cornerIp](#)  
*The other TCP corner Ip.*
- [BOOL notSend](#)  
*If the message has been sent uncompleted.*
- [TRDP\\_TIME\\_T connectionTimeout](#)  
*TCP socket connection Timeout.*
- [BOOL sendNotOk](#)  
*The sending timeout will be start.*
- [TRDP\\_TIME\\_T sendingTimeout](#)  
*The timeout sending the message.*
- [BOOL addFileDesc](#)  
*Ready to add the socket in the fd.*
- [BOOL morituri](#)  
*about to die*

#### 4.30.1 Detailed Description

TCP parameters.

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

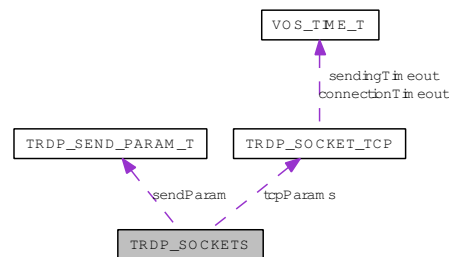
- [trdp\\_private.h](#)

## 4.31 TRDP\_SOCKETS Struct Reference

Socket item.

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

Collaboration diagram for TRDP\_SOCKETS:



### Data Fields

- INT32 [sock](#)  
*vos socket descriptor to use*
- [TRDP\\_IP\\_ADDR\\_T bindAddr](#)  
*Defines the interface to use.*
- [TRDP\\_SEND\\_PARAM\\_T sendParam](#)  
*Send parameters.*
- [TRDP\\_SOCKET\\_TYPE\\_T type](#)  
*Usage of this socket.*
- BOOL [rcvMostly](#)  
*Used for receiving.*
- INT16 [usage](#)  
*No.*
- [TRDP\\_SOCKET\\_TCP\\_T tcpParams](#)  
*Params used for TCP.*
- [TRDP\\_IP\\_ADDR\\_T mcGroups](#) [VOS\_MAX\_MULTICAST\_CNT]  
*List of multicast addresses for this socket.*

### 4.31.1 Detailed Description

Socket item.

## 4.31.2 Field Documentation

### 4.31.2.1 INT16 TRDP\_SOCKETS::usage

No.

of current users of this socket

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

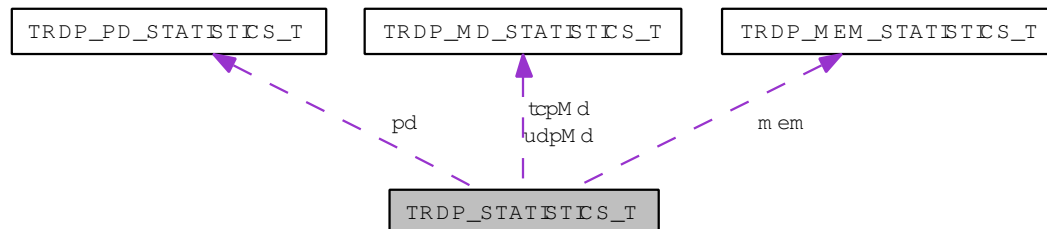
- [trdp\\_private.h](#)

## 4.32 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.*
- `TIMEDATE64` [timeStamp](#)  
*actual time stamp*
- `TIMEDATE32` [upTime](#)  
*time in sec since last initialisation*
- `TIMEDATE32` [statisticTime](#)  
*time in sec since last reset of statistics*
- `TRDP_LABEL_T` [hostName](#)  
*host name*
- `TRDP_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.*

#### 4.32.1 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](#)

### 4.33 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.*
- [void \\* callBack](#)  
*Reference for call back function if used.*
- [UINT32 timeout](#)  
*Time-out value in us.*
- [TRDP\\_ERR\\_T status](#)  
*Receive status information TRDP\_NO\_ERR, TRDP\_TIMEOUT\_ERR.*
- [TRDP\\_TO\\_BEHAVIOR\\_T toBehav](#)  
*Behaviour at time-out.*
- [UINT32 numRecv](#)  
*Number of packets received for this subscription.*

#### 4.33.1 Detailed Description

Table containing particular PD subscription information.

#### 4.33.2 Field Documentation

##### 4.33.2.1 TRDP\_IP\_ADDR\_T TRDP\_SUBS\_STATISTICS\_T::filterAddr

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

##### 4.33.2.2 UINT32 TRDP\_SUBS\_STATISTICS\_T::timeout

Time-out value in us.

0 = No time-out supervision

**4.33.2.3 TRDP\_TO\_BEHAVIOR\_T TRDP\_SUBS\_STATISTICS\_T::toBehav**

Behaviour at time-out.

Set data to zero / keep last value

**4.33.2.4 UINT32 TRDP\_SUBS\_STATISTICS\_T::numRecv**

Number of packets received for this subscription.

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

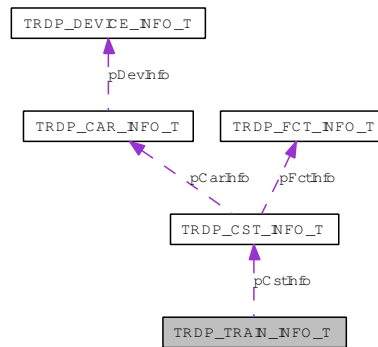
- [trdp\\_types.h](#)

## 4.34 TRDP\_TRAIN\_INFO\_T Struct Reference

train information structure.

```
#include <tau_tti.h>
```

Collaboration diagram for TRDP\_TRAIN\_INFO\_T:



### Data Fields

- `UINT32` [version](#)  
*Train info structure version.*
- `TRDP_LABEL_T` [id](#)  
*Train identifier.*
- `TRDP_LABEL_T` [operator](#)  
*Train operator e.g.*
- `TRDP_INAUG_STATE_T` [inaugState](#)  
*inauguration state*
- `UINT32` [topoCnt](#)  
*IEC (i.e.*
- `UINT8` [iecOrient](#)  
*0 == IEC reference orientation is opposite to TCN*
- `UINT16` [carCnt](#)  
*Total number of cars in train.*
- `UINT32` [cstCnt](#)  
*Total number of consists in train.*
- `TRDP_CST_INFO_T` \* [pCstInfo](#)  
*Pointer to consist info list for application use and convenience.*



### 4.34.1 Detailed Description

train information structure.

### 4.34.2 Field Documentation

#### 4.34.2.1 TRDP\_LABEL\_T TRDP\_TRAIN\_INFO\_T::operator

Train operator e.g.

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

#### 4.34.2.2 UINT32 TRDP\_TRAIN\_INFO\_T::topoCnt

IEC (i.e.

TCN) topography counter

#### 4.34.2.3 TRDP\_CST\_INFO\_T\* TRDP\_TRAIN\_INFO\_T::pCstInfo

Pointer to consist info list for application use and convenience.

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

- [tau\\_tti.h](#)

## 4.35 TRDP\_VERSION\_T Struct Reference

Version information.

```
#include <trdp_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.*

### 4.35.1 Detailed Description

Version information.

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

- [trdp\\_types.h](#)

## 4.36 TRDP\_XML\_DOC\_HANDLE\_T Struct Reference

Parsed XML document handle.

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

### Data Fields

- void \* [pXmlDocument](#)  
*Pointer to parsed XML document.*
- void \* [pRootElement](#)  
*Pointer to the document root element.*
- void \* [pXPathContext](#)  
*Pointer to prepared XPath context.*

### 4.36.1 Detailed Description

Parsed XML document handle.

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

- [tau\\_xml.h](#)

## 4.37 VOS\_SOCK\_OPT\_T Struct Reference

Common socket options.

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

### Data Fields

- **UINT8** [qos](#)  
*quality/type of service 0.*
- **UINT8** [ttl](#)  
*time to live for unicast (default 64)*
- **UINT8** [ttl\\_multicast](#)  
*time to live for multicast*
- **BOOL** [reuseAddrPort](#)  
*allow reuse of address and port*
- **BOOL** [nonBlocking](#)  
*use non blocking calls*

### 4.37.1 Detailed Description

Common socket options.

### 4.37.2 Field Documentation

#### 4.37.2.1 **UINT8** VOS\_SOCK\_OPT\_T::qos

quality/type of service 0.

..7

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

- [vos\\_sock.h](#)

## 4.38 VOS\_TIME\_T Struct Reference

Timer value compatible with timeval / select.

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

### Data Fields

- UINT32 [tv\\_sec](#)  
*full seconds*
- INT32 [tv\\_usec](#)  
*Micro seconds (max.*

### 4.38.1 Detailed Description

Timer value compatible with timeval / select.

Relative or absolute date, depending on usage

### 4.38.2 Field Documentation

#### 4.38.2.1 INT32 VOS\_TIME\_T::tv\_usec

Micro seconds (max.

value 999999)

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

- [vos\\_types.h](#)



# Chapter 5

## File Documentation

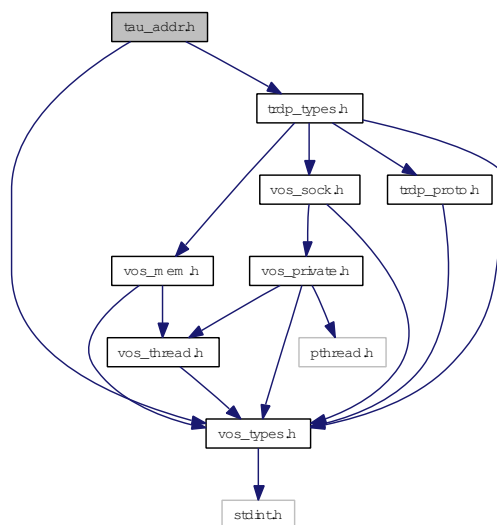
### 5.1 tau\_addr.h File Reference

TRDP utility interface definitions.

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

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

Include dependency graph for tau\_addr.h:



### Functions

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tau\\_getOwnIds](#) (TRDP\_LABEL\_T devId, TRDP\_LABEL\_T carId, TRDP\_LABEL\_T cstId)

*Who am I ?.*

- EXT\_DECL TRDP\_IP\_ADDR [tau\\_getOwnAddr](#) (void)

*Function to get the own IP address.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_uri2Addr](#) (TRDP\_IP\_ADDR \*pAddr, UINT32 \*pTopoCnt, const TRDP\_URI\_T uri)

*Function to convert a URI to an IP address.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_addr2Uri](#) (TRDP\_URI\_HOST\_T uri, UINT32 \*pTopoCnt, TRDP\_IP\_ADDR addr)

*Function to convert an IP address to a URI.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_label2CarId](#) (TRDP\_LABEL\_T carId, UINT32 \*pTopoCnt, const TRDP\_LABEL\_T carLabel, const TRDP\_LABEL\_T cstLabel)

*Function to retrieve the carId of the car with label carLabel in the consist with cstLabel.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_label2CarNo](#) (UINT8 \*pCarNo, UINT32 \*pTopoCnt, const TRDP\_LABEL\_T carLabel, const TRDP\_LABEL\_T cstLabel)

*Function The function delivers the car number to the given label.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_label2IecCarNo](#) (UINT8 \*pIecCarNo, UINT32 \*pTopoCnt, const TRDP\_LABEL\_T carLabel, const TRDP\_LABEL\_T cstLabel)

*Function The function delivers the IEC car number to the given label.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_carNo2Ids](#) (TRDP\_LABEL\_T carId, TRDP\_LABEL\_T cstId, UINT32 \*pTopoCnt, UINT8 carNo, UINT8 trnCstNo)

*Function to retrieve the car and consist id of the car given with carNo and trnCstNo.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_iecCarNo2Ids](#) (TRDP\_LABEL\_T carId, TRDP\_LABEL\_T cstId, UINT32 \*pTopoCnt, UINT8 iecCarNo)

*Function to retrieve the car and consist id from a given IEC car sequence number.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_addr2CarId](#) (TRDP\_LABEL\_T carId, UINT32 \*pTopoCnt, TRDP\_IP\_ADDR ipAddr)

*Function to retrieve the carId of the car hosting a device with the IPAddress ipAddr.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_addr2CarNo](#) (UINT8 \*pCarNo, UINT8 \*pTopoCnt, TRDP\_IP\_ADDR ipAddr)

*Function to retrieve the car number in consist of the car hosting the device with the IP address ipAddr.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_addr2IecCarNo](#) (UINT8 \*pIecCarNo, UINT8 \*pTopoCnt, TRDP\_IP\_ADDR ipAddr)

*Function to retrieve the IEC car sequence number of the car hosting the device with the IP address ipAddr.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_cstNo2CstId](#) (TRDP\_LABEL\_T cstId, UINT32 \*pTopoCnt, UINT8 cstNo)

*Function to retrieve the consist identifier of the consist with train consist sequence number cstNo.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_iecCstNo2CstId](#) (TRDP\_LABEL\_T cstId, UINT32 \*pTopoCnt, UINT8 iecCstNo)

*Function to retrieve the consist identifier of the consist with IEC sequence consist number iecCstNo.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_label2CstId](#) (TRDP\_LABEL\_T cstId, UINT32 \*pTopoCnt, const TRDP\_LABEL\_T carLabel, const TRDP\_LABEL\_T cstLabel)

*Function to retrieve the consist identifier of the consist hosting a car with label carLabel.*



- EXT\_DECL [TRDP\\_ERR\\_T tau\\_label2CstNo](#) (UINT8 \*pCstNo, UINT32 \*pTopoCnt, const TRDP\_LABEL\_T carLabel)  
*Function to retrieve the consist sequence number of the consist hosting a car with label carLabel.*
- EXT\_DECL [TRDP\\_ERR\\_T tau\\_label2IecCstNo](#) (UINT8 \*pIecCstNo, UINT32 \*pTopoCnt, const TRDP\_LABEL\_T carLabel)  
*Function to retrieve the leading car depending IEC consist sequence number of the consist hosting a car with label carLabel.*
- EXT\_DECL [TRDP\\_ERR\\_T tau\\_addr2CstId](#) (TRDP\_LABEL\_T cstId, UINT32 \*pTopoCnt, TRDP\_IP\_ADDR ipAddr)  
*Function to retrieve the consist identifier of the consist hosting the device with the IP-Address ipAddr.*
- EXT\_DECL [TRDP\\_ERR\\_T tau\\_addr2CstNo](#) (UINT8 \*pCstNo, UINT32 \*pTopoCnt, TRDP\_IP\_ADDR ipAddr)  
*Function to retrieve the consist sequence number of the consist hosting the device with the IP-Address ipAddr.*
- EXT\_DECL [TRDP\\_ERR\\_T tau\\_addr2IecCstNo](#) (UINT8 \*pIecCstNo, UINT32 \*pTopoCnt, TRDP\_IP\_ADDR ipAddr)  
*Function to retrieve the leading car depending iec consist number of the consist hosting the device with the IP-Address addr.*

### 5.1.1 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:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

#### Id

[tau\\_addr.h](#) 274 2013-01-10 11:00:43Z aweiss

### 5.1.2 Function Documentation

#### 5.1.2.1 EXT\_DECL TRDP\_ERR\_T tau\_addr2CarId (TRDP\_LABEL\_T carId, UINT32 \*pTopoCnt, TRDP\_IP\_ADDR ipAddr)

Function to retrieve the carId of the car hosting a device with the IPAddress ipAddr.

**Parameters:**

- *carId* Pointer to the car id to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *ipAddr* IP address. 0 means own address, so the own car id is returned.

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.1.2.2 EXT\_DECL TRDP\_ERR\_T tau\_addr2CarNo (UINT8 \* *pCarNo*, UINT8 \* *pTopoCnt*, TRDP\_IP\_ADDR *ipAddr*)

Function to retrieve the car number in consist of the car hosting the device with the IP address *ipAddr*.

**Parameters:**

- *pCarNo* Pointer to the car number in consist to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *ipAddr* IP address. 0 means own address, so the own car number is returned.

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.1.2.3 EXT\_DECL TRDP\_ERR\_T tau\_addr2CstId (TRDP\_LABEL\_T *cstId*, UINT32 \* *pTopoCnt*, TRDP\_IP\_ADDR *ipAddr*)

Function to retrieve the consist identifier of the consist hosting the device with the IP-Address *ipAddr*.

**Parameters:**

- *cstId* Pointer to the consist id to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *ipAddr* IP address. 0 means own device, so the own consist id is returned.

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.1.2.4 EXT\_DECL TRDP\_ERR\_T tau\_addr2CstNo (UINT8 \* *pCstNo*, UINT32 \* *pTopoCnt*, TRDP\_IP\_ADDR *ipAddr*)

Function to retrieve the consist sequence number of the consist hosting the device with the IP-Address *ipAddr*.

**Parameters:**

- *pCstNo* Pointer to the train consist number to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *ipAddr* IP address. 0 means own device, so the own consist number is returned.

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.1.2.5 EXT\_DECL TRDP\_ERR\_T tau\_addr2IecCarNo (UINT8 \* *pIecCarNo*, UINT8 \* *pTopoCnt*, TRDP\_IP\_ADDR *ipAddr*)

Function to retrieve the IEC car sequence number of the car hosting the device with the IP address *ipAddr*.

**Parameters:**

- *pIecCarNo* Pointer to the IEC car sequence number to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *ipAddr* IP address. 0 means own address, so the own IEC car number is returned.

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.1.2.6 EXT\_DECL TRDP\_ERR\_T tau\_addr2IecCstNo (UINT8 \* *pIecCstNo*, UINT32 \* *pTopoCnt*, TRDP\_IP\_ADDR *ipAddr*)

Function to retrieve the leading car depending iec consist number of the consist hosting the device with the IP-Address *addr*.

**Parameters:**

- *pIecCstNo* Pointer to the iec consist number to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *ipAddr* IP address. 0 means own device, so the own IEC consist number is returned.

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.1.2.7 EXT\_DECL TRDP\_ERR\_T tau\_addr2Uri (TRDP\_URI\_HOST\_T *uri*, UINT32 \* *pTopoCnt*, TRDP\_IP\_ADDR *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:**

- *uri* Pointer to a string to return the URI host part
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *addr* IP address, 0==own address

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.1.2.8 EXT\_DECL TRDP\_ERR\_T tau\_carNo2Ids (TRDP\_LABEL\_T *carId*, TRDP\_LABEL\_T *cstId*, UINT32 \**pTopoCnt*, UINT8 *carNo*, UINT8 *trnCstNo*)

Function to retrieve the car and consist id of the car given with *carNo* and *trnCstNo*.

**Parameters:**

- *carId* Pointer to the car id to be returned
- *cstId* Pointer to the consist id to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *carNo* Car number in consist. 0 means own car when *trnCstNo* == 0.
- ← *trnCstNo* Consist sequence number in train. 0 means own consist.

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.1.2.9 EXT\_DECL TRDP\_ERR\_T tau\_cstNo2CstId (TRDP\_LABEL\_T *cstId*, UINT32 \**pTopoCnt*, UINT8 *cstNo*)

Function to retrieve the consist identifier of the consist with train consist sequence number *cstNo*.

**Parameters:**

- *cstId* Pointer to the consist id to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *cstNo* Consist sequence number based on IP reference direction. 0 means own consist.

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.1.2.10 EXT\_DECL TRDP\_IP\_ADDR tau\_getOwnAddr (void)

Function to get the own IP address.

**Return values:**

- own* IP address

### 5.1.2.11 EXT\_DECL TRDP\_ERR\_T tau\_getOwnIds (TRDP\_LABEL\_T *devId*, TRDP\_LABEL\_T *carId*, TRDP\_LABEL\_T *cstId*)

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:

- *devId* Returns the device label (host name)
- *carId* Returns the car label
- *cstId* Returns the consist label

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

### 5.1.2.12 EXT\_DECL TRDP\_ERR\_T tau\_iecCarNo2Ids (TRDP\_LABEL\_T *carId*, TRDP\_LABEL\_T *cstId*, UINT32 \**pTopoCnt*, UINT8 *iecCarNo*)

Function to retrieve the car and consist id from a given IEC car sequence number.

#### Parameters:

- *carId* Pointer to the car id to be returned
- *cstId* Pointer to the consist id to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *iecCarNo* IEC car sequence number. 0 means own car.

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

### 5.1.2.13 EXT\_DECL TRDP\_ERR\_T tau\_iecCstNo2CstId (TRDP\_LABEL\_T *cstId*, UINT32 \**pTopoCnt*, UINT8 *iecCstNo*)

Function to retrieve the consist identifier of the consist with IEC sequence consist number *iecCstNo*.

#### Parameters:

- *cstId* Pointer to the consist id to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *iecCstNo* Consist sequence number based on the leading car depending iec reference direction. 0 means own consist.

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.1.2.14 EXT\_DECL TRDP\_ERR\_T tau\_label2CarId (TRDP\_LABEL\_T *carId*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *carLabel*, const TRDP\_LABEL\_T *cstLabel*)

Function to retrieve the carId of the car with label carLabel in the consist with cstLabel.

##### Parameters:

- *carId* Pointer to a label string to return the car id
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *carLabel* Pointer to the car label. NULL means own car if cstLabel == NULL.
- ← *cstLabel* Pointer to the consist label. NULL means own consist.

##### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.1.2.15 EXT\_DECL TRDP\_ERR\_T tau\_label2CarNo (UINT8 \* *pCarNo*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *carLabel*, const TRDP\_LABEL\_T *cstLabel*)

Function The function delivers the car number to the given label.

The first match of the table will be returned in case there is no unique label given.

##### Parameters:

- *pCarNo* Pointer to the car number to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *carLabel* Pointer to the car label. NULL means own car.
- ← *cstLabel* Pointer to the consist label. NULL means own consist.

##### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.1.2.16 EXT\_DECL TRDP\_ERR\_T tau\_label2CstId (TRDP\_LABEL\_T *cstId*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *carLabel*, const TRDP\_LABEL\_T *cstLabel*)

Function to retrieve the consist identifier of the consist hosting a car with label carLabel.

##### Parameters:

- *cstId* Pointer to the consist id to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *carLabel* Pointer to a car label. NULL means any car.
- ← *cstLabel* Pointer to a consist label. NULL means own consist.

##### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

### 5.1.2.17 EXT\_DECL TRDP\_ERR\_T tau\_label2CstNo (UINT8 \* *pCstNo*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *carLabel*)

Function to retrieve the consist sequence number of the consist hosting a car with label *carLabel*.

#### Parameters:

- *pCstNo* Pointer to the train consist number to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *carLabel* Pointer to a car label, NULL means own car, so the own consist number is returned.

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

### 5.1.2.18 EXT\_DECL TRDP\_ERR\_T tau\_label2IecCarNo (UINT8 \* *pIecCarNo*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *carLabel*, const TRDP\_LABEL\_T *cstLabel*)

Function The function delivers the IEC car number to the given label.

The first match of the table will be returned in case there is no unique label given.

#### Parameters:

- *pIecCarNo* Pointer to the IEC car sequence number to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *carLabel* Pointer to a car label. NULL means own car.
- ← *cstLabel* Pointer to a consist label. NULL means own consist.

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

### 5.1.2.19 EXT\_DECL TRDP\_ERR\_T tau\_label2IecCstNo (UINT8 \* *pIecCstNo*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *carLabel*)

Function to retrieve the leading car depending IEC consist sequence number of the consist hosting a car with label *carLabel*.

#### Parameters:

- *pIecCstNo* Pointer to the iec consist number to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *carLabel* Pointer to a car label. NULL means own car, so the own IEC consist number is returned.

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

### 5.1.2.20 EXT\_DECL TRDP\_ERR\_T tau\_uri2Addr (TRDP\_IP\_ADDR \* *pAddr*, UINT32 \* *pTopoCnt*, const TRDP\_URI\_T *uri*)

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:

→ *pAddr* Pointer to return the IP address

↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.

← *uri* Pointer to a URI or an IP Address string, NULL==own URI

#### Return values:

*TRDP\_NO\_ERR* no error

*TRDP\_PARAM\_ERR* Parameter error

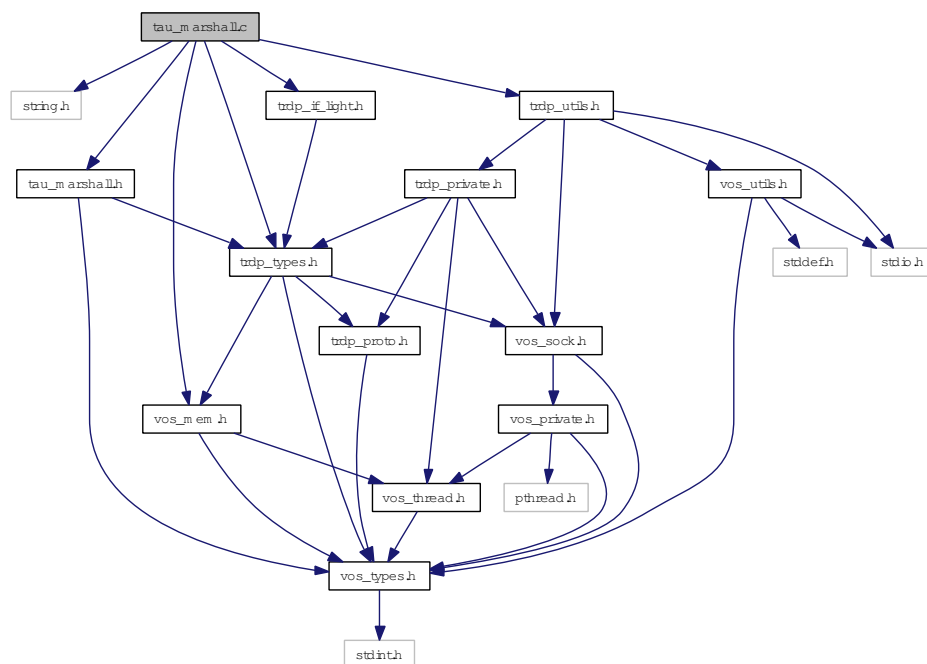


## 5.2 tau\_marshall.c File Reference

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, UINT8 \*pDest, UINT32 \*pDestSize, [TRDP\\_DATASET\\_T](#) \*\*ppDSPointer)

*marshall function.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_unmarshall](#) (void \*pRefCon, UINT32 comId, UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize, [TRDP\\_DATASET\\_T \\*\\*ppDSPointer](#))

*unmarshall function.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_marshallDs](#) (void \*pRefCon, UINT32 dsId, UINT8 \*pSrc, 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, 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 \*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 \*pDestSize, [TRDP\\_DATASET\\_T \\*\\*ppDSPointer](#))

*Calculate data set size by given ComId.*

## 5.2.1 Detailed Description

Marshalling functions for TRDP.

### Note:

Project: TCNOpen TRDP prototype stack

### Author:

Bernd Loehr, NewTec GmbH

### Remarks:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

### Id

[tau\\_marshall.c](#) 950 2013-06-13 13:51:41Z 97025

## 5.2.2 Function Documentation

### 5.2.2.1 EXT\_DECL TRDP\_ERR\_T tau\_calcDatasetSize (void \*pRefCon, UINT32 dsId, UINT8 \*pSrc, UINT32 \*pDestSize, TRDP\_DATASET\_T \*\*ppDSPointer)

Calculate data set size by given data set id.

### Parameters:

← *pRefCon* Pointer to user context

- ← *dsId* Dataset id to identify the structure out of a configuration
- ← *pSrc* Pointer to received original message
- *pDestSize* Pointer to the size of the data set
- ↔ *ppDSPointer* pointer to pointer to cached dataset, set NULL if not used, set content NULL if unknown

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_INIT\_ERR* marshalling not initialised
- TRDP\_PARAM\_ERR* data set id not existing

### 5.2.2.2 EXT\_DECL TRDP\_ERR\_T tau\_calcDatasetSizeByComId (void \* *pRefCon*, UINT32 *comId*, UINT8 \* *pSrc*, UINT32 \* *pDestSize*, TRDP\_DATASET\_T \*\* *ppDSPointer*)

Calculate data set size by given ComId.

**Parameters:**

- ← *pRefCon* Pointer to user context
- ← *comId* ComId id to identify the structure out of a configuration
- ← *pSrc* Pointer to received original message
- *pDestSize* Pointer to the size of the data set
- ↔ *ppDSPointer* pointer to pointer to cached dataset, set NULL if not used, set content NULL if unknown

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_INIT\_ERR* marshalling not initialised
- TRDP\_PARAM\_ERR* data set id not existing

### 5.2.2.3 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:**

- ↔ *ppRefCon* Returns a pointer to be used for the reference context of marshalling/unmarshalling
- ← *numComId* Number of datasets found in the configuration
- ← *pComIdDsIdMap* Pointer to an array of structures of type TRDP\_DATASET\_T
- ← *numDataSet* Number of datasets found in the configuration

← *pDataset* Pointer to an array of pointers to structures of type TRDP\_DATASET\_T

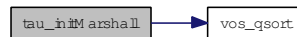
**Return values:**

**TRDP\_NO\_ERR** no error

**TRDP\_MEM\_ERR** provided buffer to small

**TRDP\_PARAM\_ERR** Parameter error

Here is the call graph for this function:



#### 5.2.2.4 EXT\_DECL TRDP\_ERR\_T tau\_marshall (void \*pRefCon, UINT32 comId, UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize, TRDP\_DATASET\_T \*\*ppDSPointer)

marshall function.

**Parameters:**

← *pRefCon* pointer to user context

← *comId* ComId to identify the structure out of a configuration

← *pSrc* pointer to received original message

← *pDest* pointer to a buffer for the treated message

↔ *pDestSize* size of the provide buffer / size of the treated message

↔ *ppDSPointer* pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

**Return values:**

**TRDP\_NO\_ERR** no error

**TRDP\_MEM\_ERR** provided buffer to small

**TRDP\_INIT\_ERR** marshalling not initialised

**TRDP\_COMID\_ERR** comid not existing

**TRDP\_PARAM\_ERR** Parameter error

#### 5.2.2.5 EXT\_DECL TRDP\_ERR\_T tau\_marshallDs (void \*pRefCon, UINT32 dsId, UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize, TRDP\_DATASET\_T \*\*ppDSPointer)

marshall data set function.

**Parameters:**

← *pRefCon* pointer to user context

← *dsId* Data set id to identify the structure out of a configuration

← *pSrc* pointer to received original message

← *pDest* pointer to a buffer for the treated message

- ↔ *pDestSize* size of the provide buffer / size of the treated message
- ↔ *ppDSPointer* pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer to small
- TRDP\_INIT\_ERR* marshalling not initialised
- TRDP\_COMID\_ERR* comid not existing
- TRDP\_PARAM\_ERR* Parameter error

### 5.2.2.6 EXT\_DECL TRDP\_ERR\_T tau\_unmarshall (void \* *pRefCon*, UINT32 *comId*, UINT8 \* *pSrc*, UINT8 \* *pDest*, UINT32 \* *pDestSize*, TRDP\_DATASET\_T \*\* *ppDSPointer*)

unmarshall function.

**Parameters:**

- ← *pRefCon* pointer to user context
- ← *comId* ComId to identify the structure out of a configuration
- ← *pSrc* pointer to received original message
- ← *pDest* pointer to a buffer for the treated message
- ↔ *pDestSize* size of the provide buffer / size of the treated message
- ↔ *ppDSPointer* pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer to small
- TRDP\_INIT\_ERR* marshalling not initialised
- TRDP\_COMID\_ERR* comid not existing

### 5.2.2.7 EXT\_DECL TRDP\_ERR\_T tau\_unmarshallDs (void \* *pRefCon*, UINT32 *dsId*, UINT8 \* *pSrc*, UINT8 \* *pDest*, UINT32 \* *pDestSize*, TRDP\_DATASET\_T \*\* *ppDSPointer*)

unmarshall data set function.

**Parameters:**

- ← *pRefCon* pointer to user context
- ← *dsId* Data set id to identify the structure out of a configuration
- ← *pSrc* pointer to received original message
- ← *pDest* pointer to a buffer for the treated message
- ↔ *pDestSize* size of the provide buffer / size of the treated message
- ↔ *ppDSPointer* pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

**Return values:**

*TRDP\_NO\_ERR* no error

*TRDP\_MEM\_ERR* provided buffer too small

*TRDP\_INIT\_ERR* marshalling not initialised

*TRDP\_COMID\_ERR* comid not existing

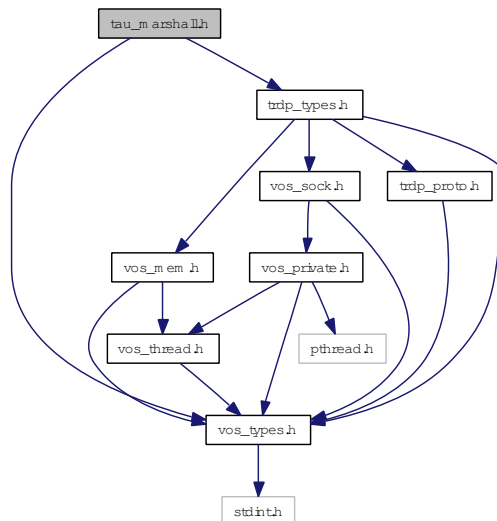
## 5.3 tau\_marshall.h File Reference

TRDP utility interface definitions.

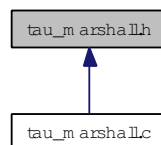
```
#include "vos_types.h"
```

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

Include dependency graph for tau\_marshall.h:



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



## Functions

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tau\\_initMarshall](#) (void \*\*ppRefCon, UINT32 numComId, [TRDP\\_COMID\\_DSID\\_MAP\\_T](#) \*pComIdDsIdMap, UINT32 numDataSet, [TRDP\\_DATASET\\_T](#) \*pDataset[])  
*Types for marshalling / unmarshalling.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tau\\_marshall](#) (void \*pRefCon, UINT32 comId, UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize, [TRDP\\_DATASET\\_T](#) \*\*ppDSPointer)  
*marshall function.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tau\\_marshallDs](#) (void \*pRefCon, UINT32 dsId, UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize, [TRDP\\_DATASET\\_T](#) \*\*ppDSPointer)  
*marshall data set function.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_unmarshall](#) (void \*pRefCon, UINT32 comId, UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize, [TRDP\\_DATASET\\_T](#) \*\*ppDSPointer)  
*unmarshall function.*
- EXT\_DECL [TRDP\\_ERR\\_T tau\\_unmarshallDs](#) (void \*pRefCon, UINT32 dsId, UINT8 \*pSrc, 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 \*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 \*pDestSize, [TRDP\\_DATASET\\_T](#) \*\*ppDSPointer)  
*Calculate data set size by given ComId.*

### 5.3.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- marshalling/unmarshalling

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Armin-H. Weiss (initial version)

#### Remarks:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

#### Id

[tau\\_marshall.h](#) 676 2013-04-18 15:27:42Z bloehr

### 5.3.2 Function Documentation

#### 5.3.2.1 EXT\_DECL [TRDP\\_ERR\\_T tau\\_calcDatasetSize](#) (void \*pRefCon, UINT32 dsId, UINT8 \*pSrc, UINT32 \*pDestSize, [TRDP\\_DATASET\\_T](#) \*\*ppDSPointer)

Calculate data set size by given data set id.

#### Parameters:

- ← *pRefCon* Pointer to user context
- ← *dsId* Dataset id to identify the structure out of a configuration



- ← *pSrc* Pointer to received original message
- *pDestSize* Pointer to the size of the data set
- ↔ *ppDSPointer* pointer to pointer to cached dataset, set NULL if not used, set content NULL if unknown

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_INIT\_ERR* marshalling not initialised
- TRDP\_PARAM\_ERR* data set id not existing

### 5.3.2.2 EXT\_DECL TRDP\_ERR\_T tau\_calcDatasetSizeByComId (void \* *pRefCon*, UINT32 *comId*, UINT8 \* *pSrc*, UINT32 \* *pDestSize*, TRDP\_DATASET\_T \*\* *ppDSPointer*)

Calculate data set size by given ComId.

**Parameters:**

- ← *pRefCon* Pointer to user context
- ← *comId* ComId id to identify the structure out of a configuration
- ← *pSrc* Pointer to received original message
- *pDestSize* Pointer to the size of the data set
- ↔ *ppDSPointer* pointer to pointer to cached dataset, set NULL if not used, set content NULL if unknown

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_INIT\_ERR* marshalling not initialised
- TRDP\_PARAM\_ERR* data set id not existing

### 5.3.2.3 EXT\_DECL TRDP\_ERR\_T tau\_initMarshall (void \*\* *ppRefCon*, UINT32 *numComId*, TRDP\_COMID\_DSID\_MAP\_T \* *pComIdsIdMap*, UINT32 *numDataSet*, TRDP\_DATASET\_T \* *pDataset*[ ])

Types for marshalling / unmarshalling.

Function to initialise the marshalling/unmarshalling.

**Parameters:**

- ↔ *ppRefCon* Returns a pointer to be used for the reference context of marshalling/unmarshalling
- ← *numComId* Number of datasets found in the configuration
- ← *pComIdsIdMap* Pointer to an array of structures of type TRDP\_DATASET\_T
- ← *numDataSet* Number of datasets found in the configuration
- ← *pDataset* Pointer to an array of pointers to structures of type TRDP\_DATASET\_T

**Return values:**

- TRDP\_NO\_ERR* no error

**TRDP\_MEM\_ERR** provided buffer to small  
**TRDP\_PARAM\_ERR** 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\(\)](#)).

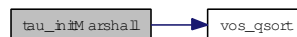
#### Parameters:

- ↔ **ppRefCon** Returns a pointer to be used for the reference context of marshalling/unmarshalling
- ← **numComId** Number of datasets found in the configuration
- ← **pComIdDsIdMap** Pointer to an array of structures of type TRDP\_DATASET\_T
- ← **numDataSet** Number of datasets found in the configuration
- ← **pDataset** Pointer to an array of pointers to structures of type TRDP\_DATASET\_T

#### Return values:

**TRDP\_NO\_ERR** no error  
**TRDP\_MEM\_ERR** provided buffer to small  
**TRDP\_PARAM\_ERR** Parameter error

Here is the call graph for this function:



#### 5.3.2.4 EXT\_DECL TRDP\_ERR\_T tau\_marshall (void \*pRefCon, UINT32 comId, UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize, TRDP\_DATASET\_T \*\*ppDSPointer)

marshall function.

#### Parameters:

- ← **pRefCon** pointer to user context
- ← **comId** ComId to identify the structure out of a configuration
- ← **pSrc** pointer to received original message
- ← **pDest** pointer to a buffer for the treated message
- ↔ **pDestSize** size of the provide buffer / size of the treated message
- ↔ **ppDSPointer** pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

#### Return values:

**TRDP\_NO\_ERR** no error  
**TRDP\_MEM\_ERR** provided buffer to small  
**TRDP\_INIT\_ERR** marshalling not initialised  
**TRDP\_COMID\_ERR** comid not existing  
**TRDP\_PARAM\_ERR** Parameter error

### 5.3.2.5 EXT\_DECL TRDP\_ERR\_T tau\_marshallDs (void \* *pRefCon*, UINT32 *dsId*, UINT8 \* *pSrc*, UINT8 \* *pDest*, UINT32 \* *pDestSize*, TRDP\_DATASET\_T \*\* *ppDSPointer*)

marshall data set function.

#### Parameters:

- ← *pRefCon* pointer to user context
- ← *dsId* Data set id to identify the structure out of a configuration
- ← *pSrc* pointer to received original message
- ← *pDest* pointer to a buffer for the treated message
- ↔ *pDestSize* size of the provide buffer / size of the treated message
- ↔ *ppDSPointer* pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer to small
- TRDP\_INIT\_ERR* marshalling not initialised
- TRDP\_COMID\_ERR* comid not existing
- TRDP\_PARAM\_ERR* Parameter error

### 5.3.2.6 EXT\_DECL TRDP\_ERR\_T tau\_unmarshall (void \* *pRefCon*, UINT32 *comId*, UINT8 \* *pSrc*, UINT8 \* *pDest*, UINT32 \* *pDestSize*, TRDP\_DATASET\_T \*\* *ppDSPointer*)

unmarshall function.

#### Parameters:

- ← *pRefCon* pointer to user context
- ← *comId* ComId to identify the structure out of a configuration
- ← *pSrc* pointer to received original message
- ← *pDest* pointer to a buffer for the treated message
- ↔ *pDestSize* size of the provide buffer / size of the treated message
- ↔ *ppDSPointer* pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer to small
- TRDP\_INIT\_ERR* marshalling not initialised
- TRDP\_COMID\_ERR* comid not existing

### 5.3.2.7 EXT\_DECL TRDP\_ERR\_T tau\_unmarshallDs (void \* *pRefCon*, UINT32 *dsId*, UINT8 \* *pSrc*, UINT8 \* *pDest*, UINT32 \* *pDestSize*, TRDP\_DATASET\_T \*\* *ppDSPointer*)

unmarshall data set function.

#### Parameters:

- ← *pRefCon* pointer to user context
- ← *dsId* Data set id to identify the structure out of a configuration
- ← *pSrc* pointer to received original message
- ← *pDest* pointer to a buffer for the treated message
- ↔ *pDestSize* size of the provide buffer / size of the treated message
- ↔ *ppDSPointer* pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer to small
- TRDP\_INIT\_ERR* marshalling not initialised
- TRDP\_COMID\_ERR* comid not existing

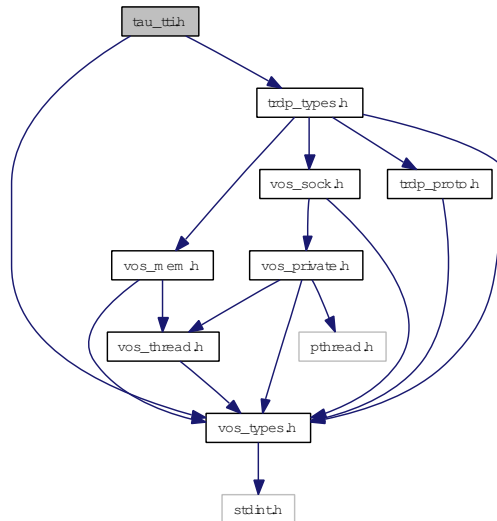
## 5.4 tau\_tti.h File Reference

TRDP utility interface definitions.

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

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

Include dependency graph for tau\_tti.h:



### Data Structures

- struct [TRDP\\_FCT\\_INFO\\_T](#)  
device information structure
- struct [TRDP\\_PROP\\_INFO\\_T](#)  
properties information structure
- struct [TRDP\\_DEVICE\\_INFO\\_T](#)  
device information structure
- struct [TRDP\\_CAR\\_INFO\\_T](#)  
car information structure.
- struct [TRDP\\_CST\\_INFO\\_T](#)  
consist information structure.
- struct [TRDP\\_TRAIN\\_INFO\\_T](#)  
train information structure.

### Enumerations

- enum [TRDP\\_INAUG\\_STATE\\_T](#) {

```

TRDP_INAUG_INVALID,
TRDP_INAUG_NOLEAD_UNCONF = 2,
TRDP_INAUG_LEAD_UNCONF = 3,
TRDP_INAUG_LEAD_CONF = 4 }

```

*Types for train configuration information.*

- enum `TRDP_FCT_T` {  
`TRDP_FCT_INVALID`,  
`TRDP_FCT_CAR` = 2,  
`TRDP_FCT_CST` = 3,  
`TRDP_FCT_TRAIN` = 4 }

*function types*

## Functions

- EXT\_DECL `TRDP_ERR_T tau_getEtbState` (`TRDP_INAUG_STATE_T` \*pInaugState, `UINT32` \*pTopoCnt)  
*Function to retrieve the inauguration state and the topography counter.*
- EXT\_DECL `TRDP_ERR_T tau_getTrnCstCnt` (`UINT16` \*pTrnCstCnt, `UINT32` \*pTopoCnt)  
*Function to retrieve the total number of consists in the train.*
- EXT\_DECL `TRDP_ERR_T tau_getTrnCarCnt` (`UINT16` \*pTrnCarCnt, `UINT32` \*pTopoCnt)  
*Function to retrieve the total number of consists in the train.*
- EXT\_DECL `TRDP_ERR_T tau_getCstCarCnt` (`UINT16` \*pCstCarCnt, `UINT32` \*pTopoCnt, `const TRDP_LABEL_T` cstLabel)  
*Function to retrieve the total number of cars in a consist.*
- EXT\_DECL `TRDP_ERR_T tau_getCstFctCnt` (`UINT16` \*pCstFctCnt, `UINT32` \*pTopoCnt, `const TRDP_LABEL_T` cstLabel)  
*Function to retrieve the total number of functions in a consist.*
- EXT\_DECL `TRDP_ERR_T tau_getCarDevCnt` (`UINT16` \*pDevCnt, `UINT32` \*pTopoCnt, `const TRDP_LABEL_T` carLabel, `const TRDP_LABEL_T` cstLabel)  
*Function to retrieve the total number of devices in a car.*
- EXT\_DECL `TRDP_ERR_T tau_getCstFctInfo` (`TRDP_FCT_INFO_T` \*pFctInfo, `UINT32` \*pTopoCnt, `const TRDP_LABEL_T` cstLabel, `UINT16` maxFctCnt)  
*Function to retrieve the function information of the consist.*
- EXT\_DECL `TRDP_ERR_T tau_getDevInfo` (`TRDP_DEV_INFO_T` \*pDevInfo, `UINT8` \*pDevProp, `UINT32` \*pDevFctNo, `UINT32` \*pTopoCnt, `const TRDP_LABEL_T` devLabel, `const TRDP_LABEL_T` carLabel, `const TRDP_LABEL_T` cstLabel, `UINT32` devPropLen, `UINT16` devFctCnt)  
*Function to retrieve the device information of a car's device.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_getCarInfo](#) ([TRDP\\_CAR\\_INFO\\_T](#) \*pCarInfo, [UINT8](#) \*pCarProp, [UINT32](#) \*pTopoCnt, const [TRDP\\_LABEL\\_T](#) carLabel, const [TRDP\\_LABEL\\_T](#) cstLabel, [UINT32](#) carPropLen)

*Function to retrieve the car information of a consist's car.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_getCstInfo](#) ([TRDP\\_CST\\_INFO\\_T](#) \*pCstInfo, [UINT8](#) \*pCstProp, [UINT32](#) \*pTopoCnt, const [TRDP\\_LABEL\\_T](#) cstLabel, [UINT32](#) cstPropLen)

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

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_getTrnInfo](#) ([TRDP\\_CST\\_INFO\\_T](#) \*pTrnInfo, [UINT32](#) \*pTopoCnt)

*Function to retrieve the train information.*

- \*\*\*\*\*  
DECL [TRDP\\_ERR\\_T tau\\_getCarOrient](#) ([UINT8](#) \*pCarOrient, [UINT8](#) \*pCstOrient, [UINT32](#) \*pTopoCnt, [TRDP\\_LABEL\\_T](#) carLabel, [TRDP\\_LABEL\\_T](#) cstLabel)

*Function to retrieve the orientation of the given car.*

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_getIecCarOrient](#) ([UINT8](#) \*plecCarOrient, [UINT8](#) \*plecCstOrient, [UINT32](#) \*pTopoCnt, [TRDP\\_LABEL\\_T](#) carLabel, [TRDP\\_LABEL\\_T](#) cstLabel)

*Function to retrieve the leading car depending IEC orientation of the given consist.*

### 5.4.1 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:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

#### Id

tau\_tci.h 274 2013-01-10 11:00:43Z aweiss

### 5.4.2 Enumeration Type Documentation

#### 5.4.2.1 enum TRDP\_FCT\_T

function types

**Enumerator:**

**TRDP\_FCT\_INVALID** Invalid type.  
 Device local function

**TRDP\_FCT\_CAR** Car control function.

**TRDP\_FCT\_CST** Consist control function.

**TRDP\_FCT\_TRAIN** Train control function.

**5.4.2.2 enum TRDP\_INAUG\_STATE\_T**

Types for train configuration information.

inauguration states

**Enumerator:**

**TRDP\_INAUG\_INVALID** Ongoing inauguration, DNS not yet available, no address transformation possible.  
 Error in train inauguration, DNS not available, trainwide communication not possible

**TRDP\_INAUG\_NOLEAD\_UNCONF** inauguration done, no leading vehicle set, inauguration unconfirmed

**TRDP\_INAUG\_LEAD\_UNCONF** inauguration done, leading vehicle set, inauguration unconfirmed

**TRDP\_INAUG\_LEAD\_CONF** inauguration done, leading vehicle set, inauguration confirmed

**5.4.3 Function Documentation****5.4.3.1 EXT\_DECL TRDP\_ERR\_T tau\_getCarDevCnt (UINT16 \* *pDevCnt*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *carLabel*, const TRDP\_LABEL\_T *cstLabel*)**

Function to retrieve the total number of devices in a car.

**Parameters:**

→ *pDevCnt* Pointer to the device count to be returned

↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.

← *carLabel* Pointer to a car label. NULL means own car if *cstLabel* == NULL.

← *cstLabel* Pointer to a consist label. NULL means own consist.

**Return values:**

**TRDP\_NO\_ERR** no error

**TRDP\_PARAM\_ERR** Parameter error

**5.4.3.2 EXT\_DECL TRDP\_ERR\_T tau\_getCarInfo (TRDP\_CAR\_INFO\_T \* *pCarInfo*, UINT8 \* *pCarProp*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *carLabel*, const TRDP\_LABEL\_T *cstLabel*, UINT32 *carPropLen*)**

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



**Parameters:**

- *pCarInfo* Pointer to the car info to be returned. Memory needs to be provided by application.
- *pCarProp* Pointer to application specific car properties to be returned. Memory needs to be provided by application. Set NULL if not used.
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *carLabel* Pointer to a car label. NULL means own car if cstLabel refers to own consist.
- ← *cstLabel* Pointer to a consist label. NULL means own consist.
- ← *carPropLen* Length of provided buffer for car properties.

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

**5.4.3.3 \*\*\*\*\***

**EXT\_DECL TRDP\_ERR\_T tau\_getCarOrient (UINT8 \* *pCarOrient*, UINT8 \* *pCstOrient*,  
UINT32 \* *pTopoCnt*, TRDP\_LABEL\_T *carLabel*, TRDP\_LABEL\_T *cstLabel*)**

Function to retrieve the orientation of the given car.

**Parameters:**

- *pCarOrient* Pointer to the car orientation to be returned
- *pCstOrient* Pointer to the consist orientation to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *carLabel* *carLabel* = NULL means own car if *cstLabel* == NULL
- ← *cstLabel* *cstLabel* = NULL means own consist

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

**5.4.3.4 EXT\_DECL TRDP\_ERR\_T tau\_getCstCarCnt (UINT16 \* *pCstCarCnt*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *cstLabel*)**

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

**Parameters:**

- *pCstCarCnt* Pointer to the number of cars to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *cstLabel* Pointer to a consist label. NULL means own consist.

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.4.3.5 EXT\_DECL TRDP\_ERR\_T tau\_getCstFctCnt (UINT16 \* *pCstFctCnt*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *cstLabel*)

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

##### Parameters:

- *pCstFctCnt* Pointer to the number of functions to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *cstLabel* Pointer to a consist label. NULL means own consist.

##### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.4.3.6 EXT\_DECL TRDP\_ERR\_T tau\_getCstFctInfo (TRDP\_FCT\_INFO\_T \* *pFctInfo*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *cstLabel*, UINT16 *maxFctCnt*)

Function to retrieve the function information of the consist.

##### Parameters:

- *pFctInfo* Pointer to function info list to be returned. Memory needs to be provided by application. Memory needs to be provided by application. Set NULL if not used.
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *cstLabel* Pointer to a consist label. NULL means own consist.
- ← *maxFctCnt* Maximal number of functions to be returned in provided buffer.

##### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.4.3.7 EXT\_DECL TRDP\_ERR\_T tau\_getCstInfo (TRDP\_CST\_INFO\_T \* *pCstInfo*, UINT8 \* *pCstProp*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *cstLabel*, UINT32 *cstPropLen*)

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

##### Parameters:

- *pCstInfo* Pointer to the consist info to be returned. Memory needs to be provided by application.
- *pCstProp* Pointer to application specific consist properties to be returned. Memory needs to be provided by application. Set NULL if not used.
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *cstLabel* Pointer to a consist label. NULL means own consist.
- ← *cstPropLen* Length of provided buffer for consist properties.

##### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.4.3.8 EXT\_DECL TRDP\_ERR\_T tau\_getDevInfo (TRDP\_DEV\_INFO\_T \* *pDevInfo*, UINT8 \* *pDevProp*, UINT32 \* *pDevFctNo*, UINT32 \* *pTopoCnt*, const TRDP\_LABEL\_T *devLabel*, const TRDP\_LABEL\_T *carLabel*, const TRDP\_LABEL\_T *cstLabel*, UINT32 *devPropLen*, UINT16 *devFctCnt*)

Function to retrieve the device information of a car's device.

##### Parameters:

- *pDevInfo* Pointer to device infos to be returned. Memory needs to be provided by application.
- *pDevProp* Pointer to application specific device properties to be returned. Memory needs to be provided by application. Set NULL if not used.
- *pDevFctNo* Pointer to device function number list to be returned. Memory needs to be provided by application. Set NULL if not used.
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *devLabel* Pointer to a device label. NULL means own device if carLabel ist referring to own car. "devxxx" possible, with xxx = 001...999
- ← *carLabel* Pointer to a car label. NULL means own car if cstLabel refers to the own consist.
- ← *cstLabel* Pointer to a consist label. NULL means own consist.
- ← *devPropLen* Length of provided buffer for device properties.
- ← *devFctCnt* Maximal number of functions to be returned in provided buffer pDevFctNo.

##### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.4.3.9 EXT\_DECL TRDP\_ERR\_T tau\_getEtbState (TRDP\_INAUG\_STATE\_T \* *pInaugState*, UINT32 \* *pTopoCnt*)

Function to retrieve the inauguration state and the topography counter.

##### Parameters:

- *pInaugState* Pointer to an inauguration state variable to be returned.
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.

##### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.4.3.10 EXT\_DECL TRDP\_ERR\_T tau\_getIecCarOrient (UINT8 \* *pIecCarOrient*, UINT8 \* *pIecCstOrient*, UINT32 \* *pTopoCnt*, TRDP\_LABEL\_T *carLabel*, TRDP\_LABEL\_T *cstLabel*)

Function to retrieve the leading car depending IEC orientation of the given consist.

##### Parameters:

- *pIecCarOrient* Pointer to the IEC car orientation to be returned

- *pIecCstOrient* Pointer to the IEC consist orientation to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
- ← *carLabel* carLabel = NULL means own car if cstLabel == NULL
- ← *cstLabel* cstLabel = NULL means own consist

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.4.3.11 EXT\_DECL TRDP\_ERR\_T tau\_getTrnCarCnt (UINT16 \* *pTrnCarCnt*, UINT32 \* *pTopoCnt*)

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

**Parameters:**

- *pTrnCarCnt* Pointer to the number of cars to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.4.3.12 EXT\_DECL TRDP\_ERR\_T tau\_getTrnCstCnt (UINT16 \* *pTrnCstCnt*, UINT32 \* *pTopoCnt*)

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

**Parameters:**

- *pTrnCstCnt* Pointer to the number of consists to be returned
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

#### 5.4.3.13 EXT\_DECL TRDP\_ERR\_T tau\_getTrnInfo (TRDP\_CST\_INFO\_T \* *pTrnInfo*, UINT32 \* *pTopoCnt*)

Function to retrieve the train information.

**Parameters:**

- *pTrnInfo* Pointer to the train info to be returned. Memory needs to be provided by application.
- ↔ *pTopoCnt* Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.

**Return values:**

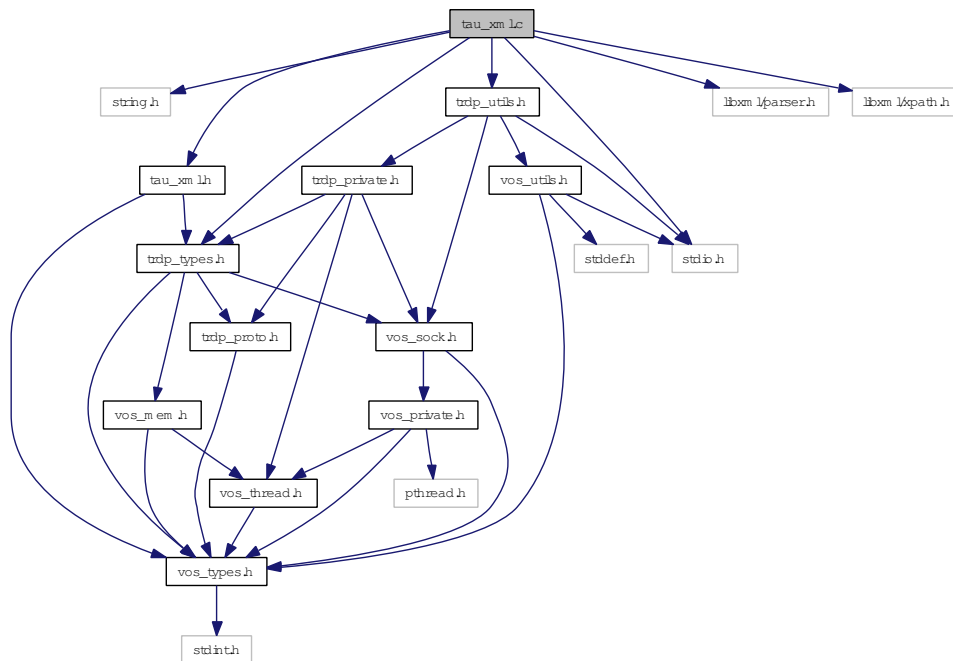
- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

## 5.5 tau\_xml.c File Reference

Functions for XML file parsing.

```
#include <string.h>
#include <stdio.h>
#include "trdp_types.h"
#include "trdp_utils.h"
#include "tau_xml.h"
#include "libxml/parser.h"
#include "libxml/xpath.h"
```

Include dependency graph for tau\_xml.c:



### Defines

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

## Functions

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tau\\_prepareXmlDoc](#) (const [CHAR8](#) \*pFileName, [TRDP\\_XML\\_DOC\\_HANDLE\\_T](#) \*pDocHnd)  
*Load XML file into DOM tree, prepare XPath context.*
- EXT\_DECL void [tau\\_freeXmlDoc](#) ([TRDP\\_XML\\_DOC\\_HANDLE\\_T](#) \*pDocHnd)  
*Free all the memory allocated by tau\_prepareXmlDoc.*
- EXT\_DECL [TRDP\\_ERR\\_T](#) [tau\\_readXmlDeviceConfig](#) (const [TRDP\\_XML\\_DOC\\_HANDLE\\_T](#) \*pDocHnd, [TRDP\\_MEM\\_CONFIG\\_T](#) \*pMemConfig, [TRDP\\_DBG\\_CONFIG\\_T](#) \*pDbgConfig, [UINT32](#) \*pNumComPar, [TRDP\\_COM\\_PAR\\_T](#) \*\*ppComPar, [UINT32](#) \*pNumIfConfig, [TRDP\\_IF\\_CONFIG\\_T](#) \*\*ppIfConfig)  
*Function to read the TRDP device configuration parameters out of the XML configuration file.*
- EXT\_DECL [TRDP\\_ERR\\_T](#) [tau\\_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.*
- 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.*

### 5.5.1 Detailed Description

Functions for XML file parsing.

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Tomas Svoboda, UniContorls a.s.

#### Remarks:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

#### Id

[tau\\_xml.c](#) 993 2013-06-25 13:07:28Z bloehr

## 5.5.2 Define Documentation

### 5.5.2.1 #define TRDP\_SDT\_DEFAULT\_CMTHR 10

Default SDT chan.

monitoring threshold

## 5.5.3 Function Documentation

### 5.5.3.1 EXT\_DECL void tau\_freeTelegrams (UINT32 *numExchgPar*, TRDP\_EXCHG\_PAR\_T \* *pExchgPar*)

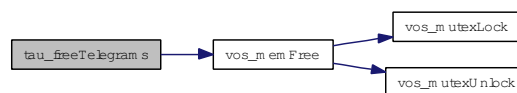
Free array of telegram configurations allocated by tau\_readXmlInterfaceConfig.

#### Parameters:

← *numExchgPar* Number of telegram configurations in the array

← *pExchgPar* Pointer to array of telegram configurations

Here is the call graph for this function:



### 5.5.3.2 EXT\_DECL void tau\_freeXmlDoc (TRDP\_XML\_DOC\_HANDLE\_T \* *pDocHnd*)

Free all the memory allocated by tau\_prepareXmlDoc.

#### Parameters:

← *pDocHnd* Handle of the parsed XML file

### 5.5.3.3 EXT\_DECL TRDP\_ERR\_T tau\_prepareXmlDoc (const CHAR8 \* *pFileName*, TRDP\_XML\_DOC\_HANDLE\_T \* *pDocHnd*)

Load XML file into DOM tree, prepare XPath context.

#### Parameters:

← *pFileName* Path and filename of the xml configuration file

→ *pDocHnd* Handle of the parsed XML file

#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_PARAM\_ERR** File does not exist

#### 5.5.3.4 EXT\_DECL TRDP\_ERR\_T tau\_readXmlDatasetConfig (const TRDP\_XML\_DOC\_HANDLE\_T \* *pDocHnd*, UINT32 \* *pNumComId*, TRDP\_COMID\_DSID\_MAP\_T \*\* *ppComIdDsIdMap*, UINT32 \* *pNumDataset*, papTRDP\_DATASET\_T *papDataset*)

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

##### Parameters:

- ← *pDocHnd* Handle of the XML document prepared by tau\_prepareXmlDoc
- *pNumComId* Pointer to the number of entries in the ComId DatasetId mapping list
- *ppComIdDsIdMap* Pointer to an array of a structures of type [TRDP\\_COMID\\_DSID\\_MAP\\_T](#)
- *pNumDataset* Pointer to the number of datasets found in the configuration
- *papDataset* Pointer to an array of pointers to a structures of type TRDP\_DATASET\_T

##### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer to small
- TRDP\_PARAM\_ERR* File not existing

#### 5.5.3.5 EXT\_DECL TRDP\_ERR\_T tau\_readXmlDeviceConfig (const TRDP\_XML\_DOC\_HANDLE\_T \* *pDocHnd*, TRDP\_MEM\_CONFIG\_T \* *pMemConfig*, TRDP\_DBG\_CONFIG\_T \* *pDbgConfig*, UINT32 \* *pNumComPar*, TRDP\_COM\_PAR\_T \*\* *ppComPar*, UINT32 \* *pNumIfConfig*, TRDP\_IF\_CONFIG\_T \*\* *ppIfConfig*)

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

##### Parameters:

- ← *pDocHnd* Handle of the XML document prepared by tau\_prepareXmlDoc
- *pMemConfig* Memory configuration
- *pDbgConfig* Debug printout configuration for application use
- *pNumComPar* Number of configured com parameters
- *ppComPar* Pointer to array of com parameters
- *pNumIfConfig* Number of configured interfaces
- *ppIfConfig* Pointer to an array of interface parameter sets

##### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer to small
- TRDP\_PARAM\_ERR* File not existing

#### 5.5.3.6 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



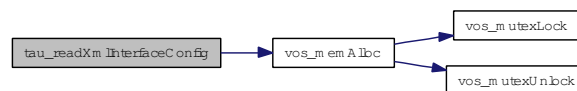
**Parameters:**

- ← *pDocHnd* Handle of the XML document prepared by tau\_prepareXmlDoc
- ← *pIfName* Interface name
- *pProcessConfig* TRDP process (session) configuration for the interface
- *pPdConfig* PD default configuration for the interface
- *pMdConfig* MD default configuration for the interface
- *pNumExchgPar* Number of configured telegrams
- *ppExchgPar* Pointer to array of telegram configurations

**Return values:**

- TRDP\_NO\_ERR** no error
- TRDP\_MEM\_ERR** provided buffer too small
- TRDP\_PARAM\_ERR** File not existing

Here is the call graph for this function:



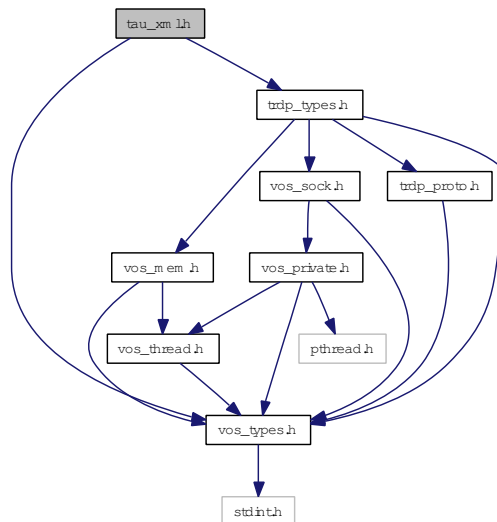
## 5.6 tau\_xml.h File Reference

TRDP utility interface definitions.

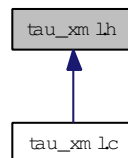
```
#include "vos_types.h"
```

```
#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.*

## Enumerations

- enum [TRDP\\_DBG\\_OPTION\\_T](#) {

```

TRDP_DBG_DEFAULT = 0,
TRDP_DBG_OFF = 0x01,
TRDP_DBG_ERR = 0x02,
TRDP_DBG_WARN = 0x04,
TRDP_DBG_INFO = 0x08,
TRDP_DBG_DBG = 0x10,
TRDP_DBG_TIME = 0x20,
TRDP_DBG_LOC = 0x40,
TRDP_DBG_CAT = 0x80 }

```

*Control for debug output format on application level.*

## Functions

- EXT\_DECL [TRDP\\_ERR\\_T tau\\_prepareXmlDoc](#) (const [CHAR8](#) \*pFileName, [TRDP\\_XML\\_DOC\\_HANDLE\\_T](#) \*pDocHnd)  
*Load XML file into DOM tree, prepare XPath context.*
- EXT\_DECL void [tau\\_freeXmlDoc](#) ([TRDP\\_XML\\_DOC\\_HANDLE\\_T](#) \*pDocHnd)  
*Free all the memory allocated by tau\_prepareXmlDoc.*
- EXT\_DECL [TRDP\\_ERR\\_T tau\\_readXmlDeviceConfig](#) (const [TRDP\\_XML\\_DOC\\_HANDLE\\_T](#) \*pDocHnd, [TRDP\\_MEM\\_CONFIG\\_T](#) \*pMemConfig, [TRDP\\_DBG\\_CONFIG\\_T](#) \*pDbgConfig, [UINT32](#) \*pNumComPar, [TRDP\\_COM\\_PAR\\_T](#) \*\*ppComPar, [UINT32](#) \*pNumIfConfig, [TRDP\\_IF\\_CONFIG\\_T](#) \*\*ppIfConfig)  
*Function to read the TRDP device configuration parameters out of the XML configuration file.*
- EXT\_DECL [TRDP\\_ERR\\_T tau\\_readXmlInterfaceConfig](#) (const [TRDP\\_XML\\_DOC\\_HANDLE\\_T](#) \*pDocHnd, const [CHAR8](#) \*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 [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.*
- EXT\_DECL void [tau\\_freeTelegrams](#) ([UINT32](#) numExchgPar, [TRDP\\_EXCHG\\_PAR\\_T](#) \*pExchgPar)  
*Free array of telegram configurations allocated by tau\_readXmlInterfaceConfig.*

### 5.6.1 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:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

[tau\\_xml.h](#) 406 2013-01-25 16:28:16Z bloehr

## 5.6.2 Enumeration Type Documentation

### 5.6.2.1 enum TRDP\_DBG\_OPTION\_T

Control for debug output format on application level.

**Enumerator:**

**TRDP\_DBG\_DEFAULT** Printout default.  
**TRDP\_DBG\_OFF** Printout off.  
**TRDP\_DBG\_ERR** Printout error.  
**TRDP\_DBG\_WARN** Printout warning and error.  
**TRDP\_DBG\_INFO** Printout info, warning and error.  
**TRDP\_DBG\_DBG** Printout debug, info, warning and error.  
**TRDP\_DBG\_TIME** Printout timestamp.  
**TRDP\_DBG\_LOC** Printout file name and line.  
**TRDP\_DBG\_CAT** Printout category (DBG, INFO, WARN, ERR).

## 5.6.3 Function Documentation

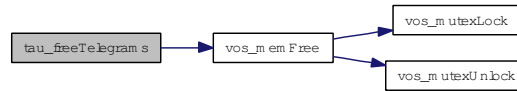
### 5.6.3.1 EXT\_DECL void tau\_freeTelegrams (UINT32 *numExchgPar*, TRDP\_EXCHG\_PAR\_T \**pExchgPar*)

Free array of telegram configurations allocated by tau\_readXmlInterfaceConfig.

**Parameters:**

- ← *numExchgPar* Number of telegram configurations in the array
- ← *pExchgPar* Pointer to array of telegram configurations

Here is the call graph for this function:



### 5.6.3.2 EXT\_DECL void tau\_freeXmlDoc (TRDP\_XML\_DOC\_HANDLE\_T \* *pDocHnd*)

Free all the memory allocated by tau\_prepareXmlDoc.

#### Parameters:

- ← *pDocHnd* Handle of the parsed XML file
- ← *pDocHnd* Handle of the parsed XML file

### 5.6.3.3 EXT\_DECL TRDP\_ERR\_T tau\_prepareXmlDoc (const CHAR8 \* *pFileName*, TRDP\_XML\_DOC\_HANDLE\_T \* *pDocHnd*)

Load XML file into DOM tree, prepare XPath context.

#### Parameters:

- ← *pFileName* Path and filename of the xml configuration file
- *pDocHnd* Handle of the parsed XML file

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* File does not exist

### 5.6.3.4 EXT\_DECL TRDP\_ERR\_T tau\_readXmlDatasetConfig (const TRDP\_XML\_DOC\_HANDLE\_T \* *pDocHnd*, UINT32 \* *pNumComId*, TRDP\_COMID\_DSID\_MAP\_T \*\* *ppComIdDsIdMap*, UINT32 \* *pNumDataset*, papTRDP\_DATASET\_T *papDataset*)

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

#### Parameters:

- ← *pDocHnd* Handle of the XML document prepared by tau\_prepareXmlDoc
- *pNumComId* Pointer to the number of entries in the ComId DatasetId mapping list
- *ppComIdDsIdMap* Pointer to an array of a structures of type [TRDP\\_COMID\\_DSID\\_MAP\\_T](#)
- *pNumDataset* Pointer to the number of datasets found in the configuration
- *papDataset* Pointer to an array of pointers to a structures of type TRDP\_DATASET\_T

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer too small
- TRDP\_PARAM\_ERR* File not existing

**5.6.3.5 EXT\_DECL TRDP\_ERR\_T tau\_readXmlDeviceConfig (const TRDP\_XML\_DOC\_HANDLE\_T \* *pDocHnd*, TRDP\_MEM\_CONFIG\_T \* *pMemConfig*, TRDP\_DBG\_CONFIG\_T \* *pDbgConfig*, UINT32 \* *pNumComPar*, TRDP\_COM\_PAR\_T \*\* *ppComPar*, UINT32 \* *pNumIfConfig*, TRDP\_IF\_CONFIG\_T \*\* *ppIfConfig*)**

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

**Parameters:**

- ← *pDocHnd* Handle of the XML document prepared by tau\_prepareXmlDoc
- *pMemConfig* Memory configuration
- *pDbgConfig* Debug printout configuration for application use
- *pNumComPar* Number of configured com parameters
- *ppComPar* Pointer to array of com parameters
- *pNumIfConfig* Number of configured interfaces
- *ppIfConfig* Pointer to an array of interface parameter sets

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer to small
- TRDP\_PARAM\_ERR* File not existing

**5.6.3.6 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:**

- ← *pDocHnd* Handle of the XML document prepared by tau\_prepareXmlDoc
- ← *pIfName* Interface name
- *pProcessConfig* TRDP process (session) configuration for the interface
- *pPdConfig* PD default configuration for the interface
- *pMdConfig* MD default configuration for the interface
- *pNumExchgPar* Number of configured telegrams
- *ppExchgPar* Pointer to array of telegram configurations

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer to small
- TRDP\_PARAM\_ERR* File not existing

Here is the call graph for this function:



## 5.7 trdp\_dllmain.c File Reference

Windows DLL main function.

### 5.7.1 Detailed Description

Windows DLL main function.

**Note:**

Project: TCNOpen TRDP prototype stack

**Author:**

Armin-H. Weiss, Bombardier

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

[trdp\\_dllmain.c](#) 950 2013-06-13 13:51:41Z 97025

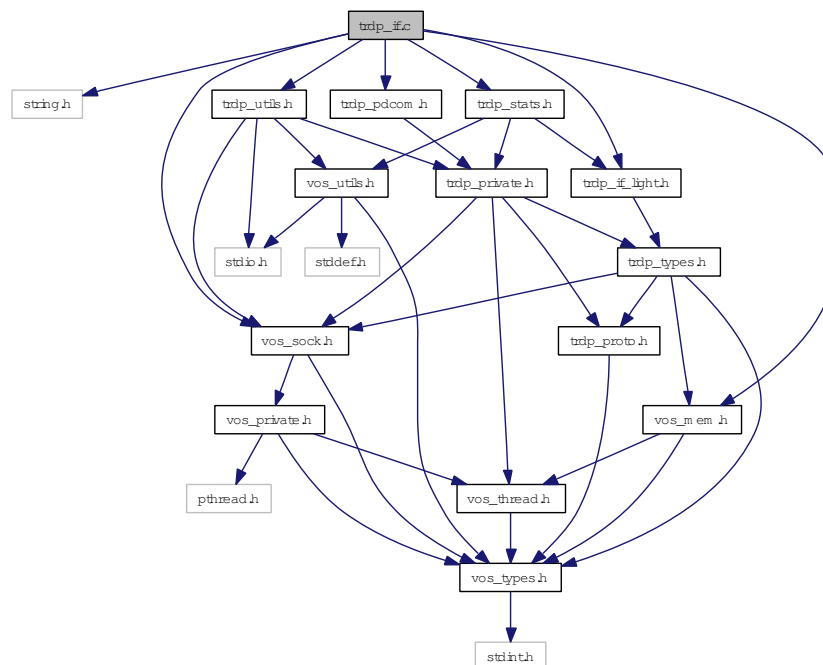


## 5.8 trdp\_if.c File Reference

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 dependency graph for trdp\_if.c:



### Functions

- **BOOL** `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.*
- **EXT\_DECL** **TRDP\_ERR\_T** `tlc_init` (const **TRDP\_PRINT\_DBG\_T** pPrintDebugString, 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\\_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.*
- const char \* [tlc\\_getVersionString](#) (void)  
*Return a human readable version representation.*
- EXT\_DECL const [TRDP\\_VERSION\\_T](#) \* [tlc\\_getVersion](#) (void)  
*Return version.*
- [TRDP\\_ERR\\_T](#) [tlp\\_setRedundant](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT32 redId, BOOL 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, BOOL \*pLeader)  
*Get status of redundant ComIds.*
- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_setTopoCount](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT32 topoCount)  
*Set new topocount for trainwide communication.*
- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlp\\_publish](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_PUB\\_T](#) \*pPubHandle, UINT32 comId, UINT32 topoCount, [TRDP\\_IP\\_ADDR\\_T](#) srcIpAddr, [TRDP\\_IP\\_ADDR\\_T](#) destIpAddr, UINT32 interval, UINT32 redId, [TRDP\\_FLAGS\\_T](#) pktFlags, const [TRDP\\_SEND\\_PARAM\\_T](#) \*pSendParam, const UINT8 \*pData, UINT32 dataSize)  
*Prepare for sending PD messages.*
- [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.*
- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getInterval](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_TIME\\_T](#) \*pInterval, [TRDP\\_FDS\\_T](#) \*pFileDesc, INT32 \*pNoDesc)  
*Get the lowest time interval for PDs.*

- EXT\_DECL [TRDP\\_ERR\\_T tlc\\_process](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_FDS\\_T](#) \*pRfds, [INT32](#) \*pCount)  
*Work loop of the TRDP handler.*
- EXT\_DECL [TRDP\\_ERR\\_T tlp\\_request](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_SUB\\_T](#) subHandle, [UINT32](#) comId, [UINT32](#) topoCount, [TRDP\\_IP\\_ADDR\\_T](#) srcIpAddr, [TRDP\\_IP\\_ADDR\\_T](#) destIpAddr, [UINT32](#) redId, [TRDP\\_FLAGS\\_T](#) pktFlags, const [TRDP\\_SEND\\_PARAM\\_T](#) \*pSendParam, const [UINT8](#) \*pData, [UINT32](#) dataSize, [UINT32](#) replyComId, [TRDP\\_IP\\_ADDR\\_T](#) replyIpAddr)  
*Initiate sending PD messages (PULL).*
- EXT\_DECL [TRDP\\_ERR\\_T tlp\\_subscribe](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_SUB\\_T](#) \*pSubHandle, const void \*pUserRef, [UINT32](#) comId, [UINT32](#) topoCount, [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, [UINT32](#) maxDataSize)  
*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\\_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.*

### 5.8.1 Detailed Description

Functions for ECN communication.

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Bernd Loehr, NewTec GmbH

#### Remarks:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

#### Id

[trdp\\_if.c](#) 1017 2013-07-10 08:56:49Z bloehr

BL 2013-06-24: ID 125: Time-out handling and ready descriptors fixed

BL 2013-02-01: ID 53: Zero dataset size fixed for PD

BL 2013-01-25: ID 20: Redundancy handling fixed

BL 2013-01-08: LADDER: Removed/Changed some ladder specific code in [tlp\\_subscribe\(\)](#)

BL 2012-12-03: ID 1: "using uninitialized PD\_ELE\_T.pullIpAddress variable" ID 2: "uninitialized PD\_ELE\_T newPD → pNext in tlp\_subscribe()"

## 5.8.2 Function Documentation

### 5.8.2.1 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:

← *appHandle* The handle returned by tlc\_openSession

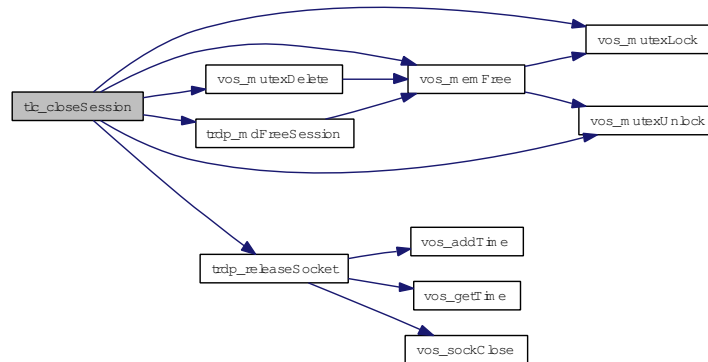
#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** handle NULL

Here is the call graph for this function:



### 5.8.2.2 EXT\_DECL TRDP\_ERR\_T tlc\_getInterval (TRDP\_APP\_SESSION\_T appHandle, TRDP\_TIME\_T \* pInterval, TRDP\_FDS\_T \* pFileDesc, INT32 \* pNoDesc)

Get the lowest time interval for PDs.

Return the maximum time interval suitable for 'select()' so that we can send due PD packets in time. If the PD send queue is empty, return zero time

#### Parameters:

← *appHandle* The handle returned by tlc\_openSession

→ *pInterval* pointer to needed interval

↔ *pFileDesc* pointer to file descriptor set

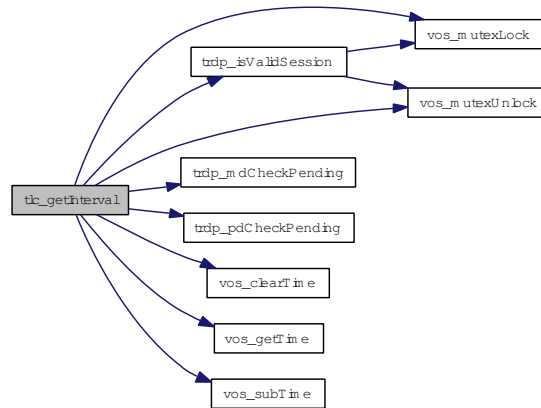
→ *pNoDesc* pointer to put no of highest used descriptors (for select())

#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



### 5.8.2.3 EXT\_DECL const TRDP\_VERSION\_T\* tlc\_getVersion (void)

Return version.

Return pointer to version structure

**Return values:**

**TRDP\_VERSION\_T**

### 5.8.2.4 const char\* tlc\_getVersionString (void)

Return a human readable version representation.

Return string in the form 'v.r.u.b'

**Return values:**

**const** string

### 5.8.2.5 EXT\_DECL TRDP\_ERR\_T tlc\_init (const TRDP\_PRINT\_DBG\_T pPrintDebugString, const TRDP\_MEM\_CONFIG\_T \* pMemConfig)

Initialize the TRDP stack.

`tlc_init` returns in `pAppHandle` a unique handle to be used in further calls to the stack.

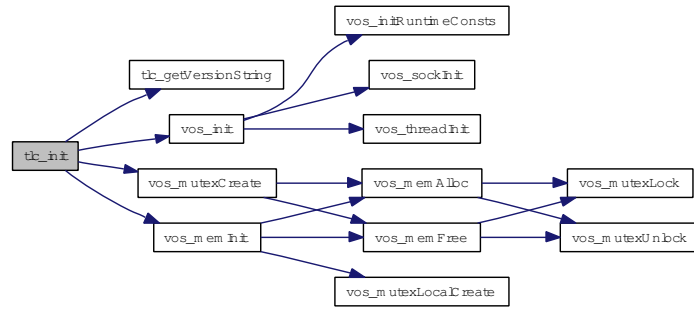
**Parameters:**

← **pPrintDebugString** Pointer to debug print function

← **pMemConfig** Pointer to memory configuration

**Return values:***TRDP\_NO\_ERR* no error*TRDP\_MEM\_ERR* memory allocation failed*TRDP\_PARAM\_ERR* initialization error

Here is the call graph for this function:



**5.8.2.6** `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.

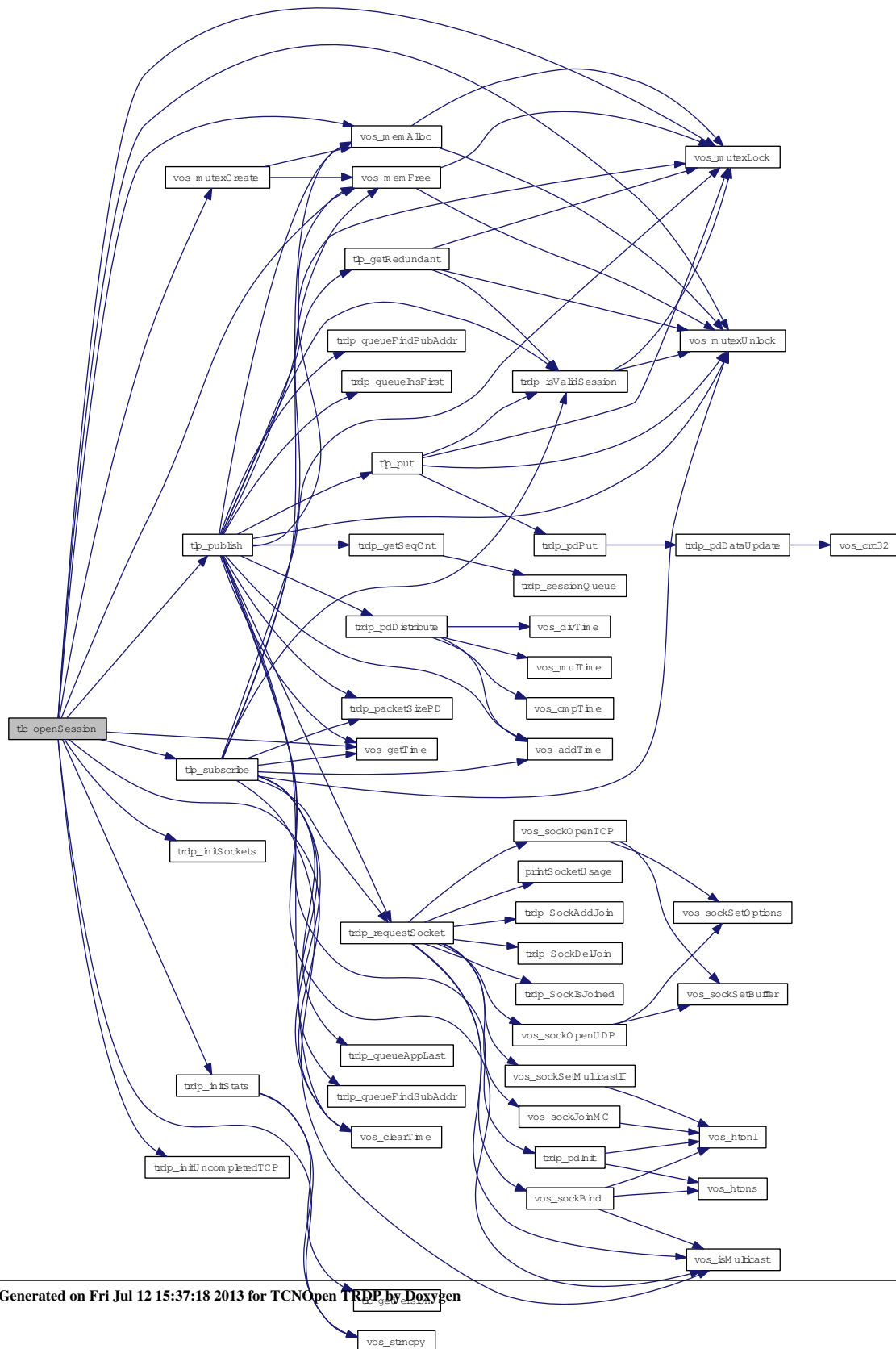
`tlc_openSession` returns in `pAppHandle` a unique handle to be used in further calls to the stack.

**Parameters:**

- *pAppHandle* A handle for further calls to the trdp stack
- ← *ownIpAddr* Own IP address, can be different for each process in multihoming systems, if zero, the default interface / IP will be used.
- ← *leaderIpAddr* IP address of redundancy leader
- ← *pMarshall* Pointer to marshalling configuration
- ← *pPdDefault* Pointer to default PD configuration
- ← *pMdDefault* Pointer to default MD configuration
- ← *pProcessConfig* 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:***TRDP\_NO\_ERR* no error*TRDP\_INIT\_ERR* not yet initied*TRDP\_PARAM\_ERR* parameter error*TRDP SOCK\_ERR* socket error

Here is the call graph for this function:



### 5.8.2.7 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 to be sent Search the receive queue for pending PDs (time out)

#### Parameters:

← *appHandle* The handle returned by tlc\_openSession

← *pRfds* pointer to set of ready descriptors

↔ *pCount* pointer to number of ready descriptors

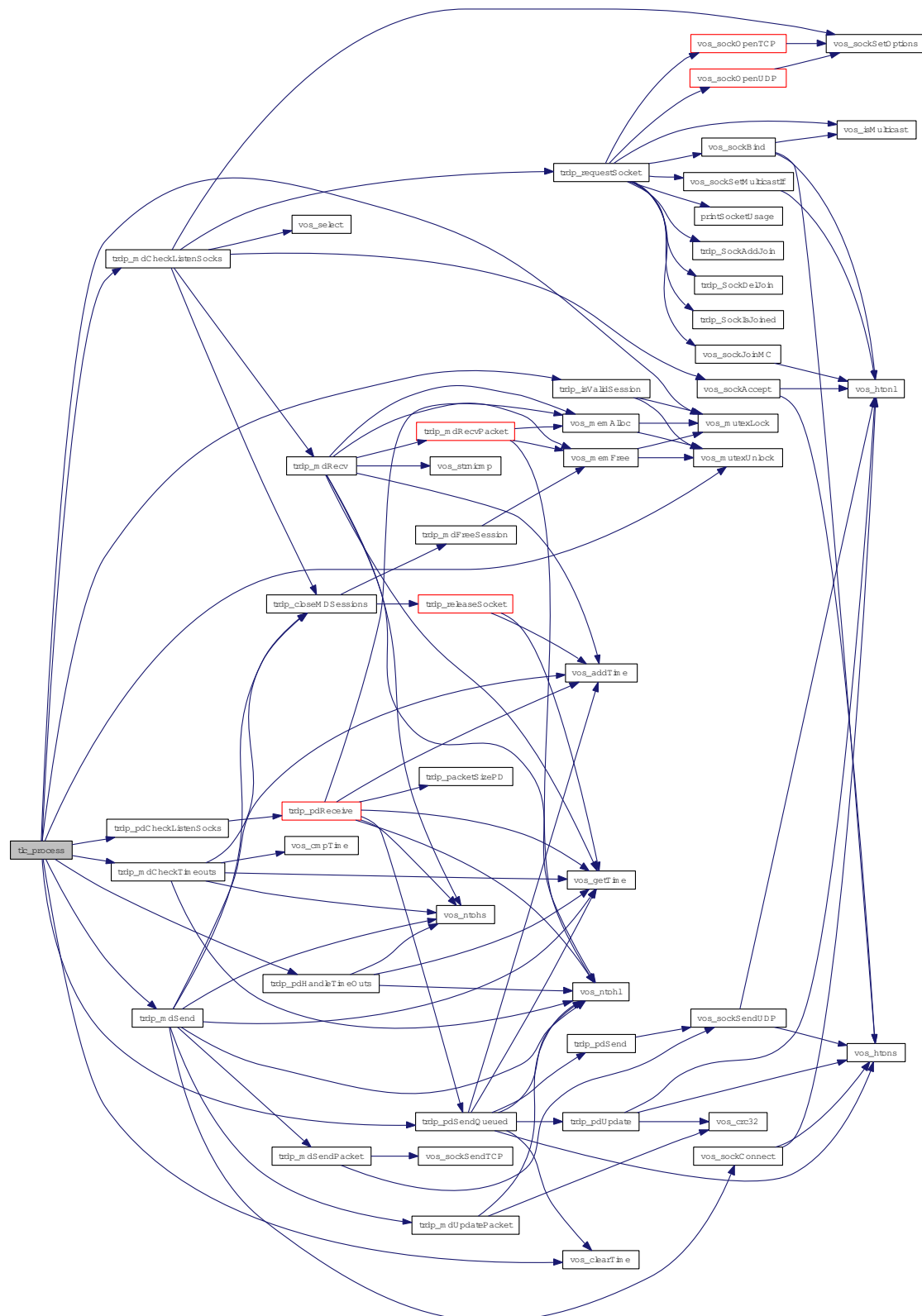
#### Return values:

*TRDP\_NO\_ERR* no error

*TRDP\_NOINIT\_ERR* handle invalid



Here is the call graph for this function:



### 5.8.2.8 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:

← *appHandle* The handle returned by tlc\_openSession

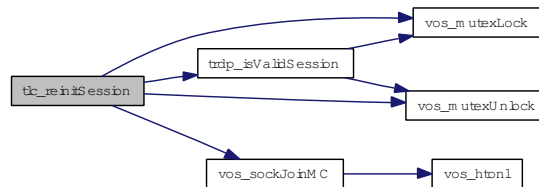
#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** handle NULL

Here is the call graph for this function:



### 5.8.2.9 EXT\_DECL TRDP\_ERR\_T tlc\_setTopoCount (TRDP\_APP\_SESSION\_T appHandle, UINT32 topoCount)

Set new topoCount for trainwide communication.

This value is used for validating outgoing and incoming packets only!

#### Parameters:

← *appHandle* the handle returned by tlc\_openSession

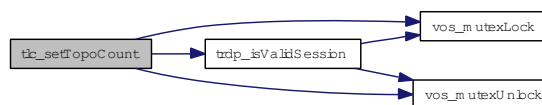
← *topoCount* New topoCount value

#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



### 5.8.2.10 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:**

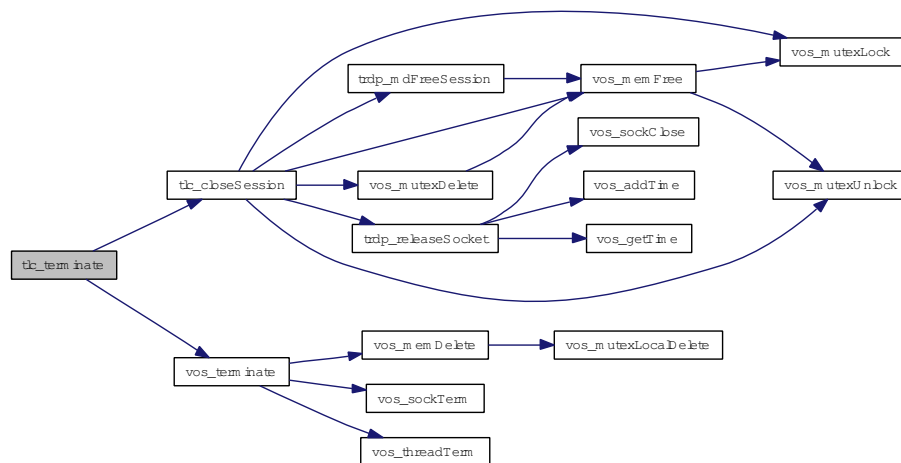
**TRDP\_NO\_ERR** no error

**TRDP\_INIT\_ERR** no error

**TRDP\_MEM\_ERR** TrafficStore nothing

**TRDP\_MUTEX\_ERR** TrafficStore mutex err

Here is the call graph for this function:



### 5.8.2.11 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:**

← **appHandle** the handle returned by tlc\_openSession

← **subHandle** the handle returned by subscription

↔ **pPdInfo** pointer to application's info buffer

↔ **pData** pointer to application's data buffer

↔ **pDataSize** in: size of buffer, out: size of data

**Return values:**

**TRDP\_NO\_ERR** no error

**TRDP\_PARAM\_ERR** parameter error

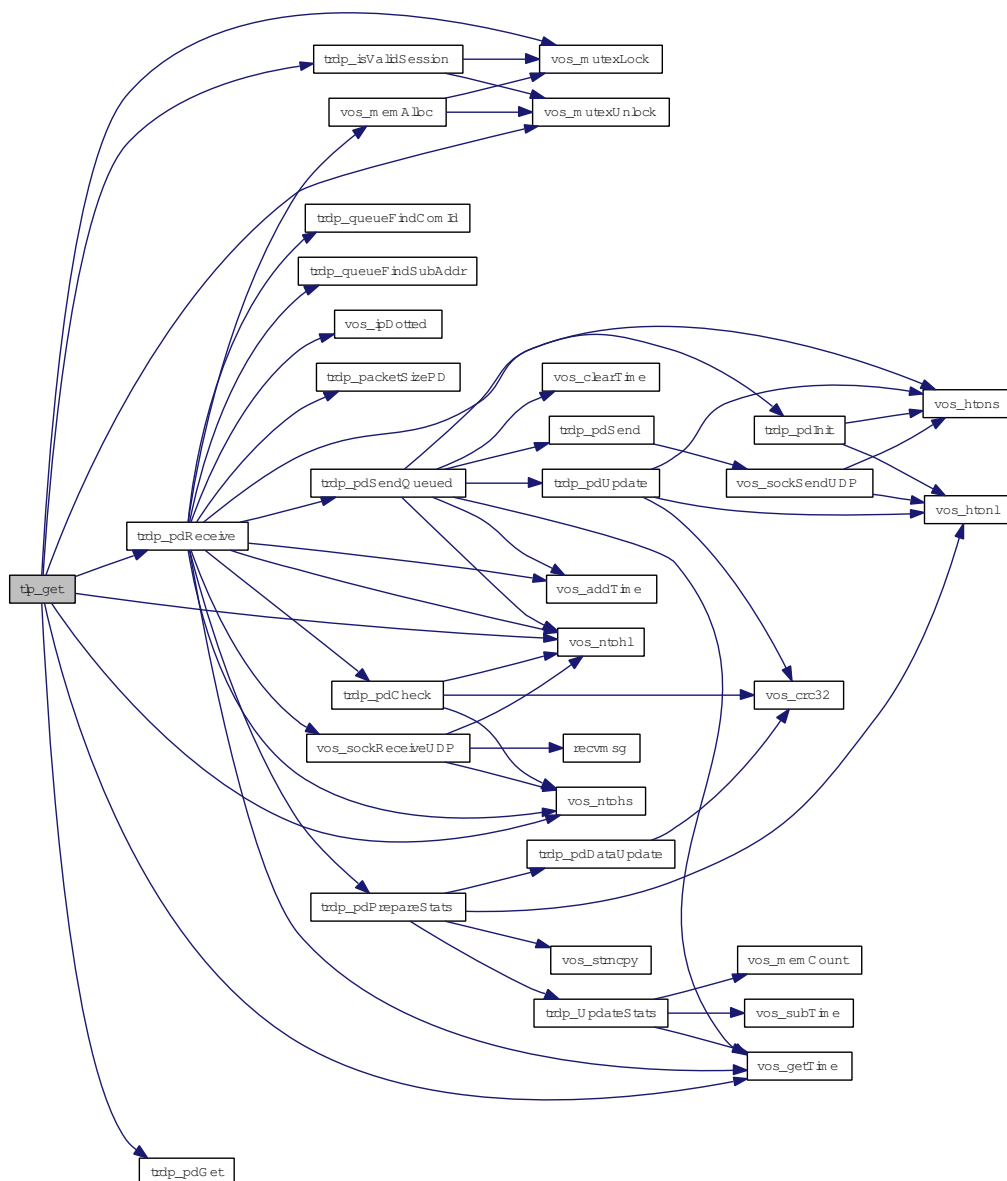
**TRDP\_SUB\_ERR** not subscribed

**TRDP\_TIMEOUT\_ERR** packet timed out

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_COMID\_ERR** ComID not found when marshalling

Here is the call graph for this function:



### 5.8.2.12 EXT\_DECL TRDP\_ERR\_T tlp\_getRedundant (TRDP\_APP\_SESSION\_T *appHandle*, UINT32 *redId*, BOOL \**pLeader*)

Get status of redundant ComIds.

Only the status of the first redundancy group entry is returned will be returned!

#### Parameters:

- ← *appHandle* the handle returned by tlc\_init
- ← *redId* will be returned for all ComID's with the given redId
- ↔ *pLeader* TRUE if we're sending this redundancy group (leader)

#### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error / redId not existing
- TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



### 5.8.2.13 EXT\_DECL TRDP\_ERR\_T tlp\_publish (TRDP\_APP\_SESSION\_T *appHandle*, TRDP\_PUB\_T \**pPubHandle*, UINT32 *comId*, UINT32 *topoCount*, TRDP\_IP\_ADDR\_T *srcIpAddr*, TRDP\_IP\_ADDR\_T *destIpAddr*, UINT32 *interval*, UINT32 *redId*, TRDP\_FLAGS\_T *pktFlags*, const TRDP\_SEND\_PARAM\_T \**pSendParam*, const UINT8 \**pData*, UINT32 *dataSize*)

Prepare for sending PD messages.

Queue a PD message, it will be send when trdp\_work has been called

#### Parameters:

- ← *appHandle* the handle returned by tlc\_openSession
- *pPubHandle* returned handle for related unprepare
- ← *comId* comId of packet to send
- ← *topoCount* valid topocount, 0 for local consist
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to
- ← *interval* frequency of PD packet (>= 10ms) in usec, 0 if PD PULL
- ← *redId* 0 - Non-redundant, > 0 valid redundancy group
- ← *pktFlags* OPTION: TRDP\_FLAGS\_DEFAULT, TRDP\_FLAGS\_NONE, TRDP\_FLAGS\_-  
MARSHALL, TRDP\_FLAGS\_CALLBACK

← *pSendParam* optional pointer to send parameter, NULL - default parameters are used

← *pData* pointer to packet data / dataset

← *dataSize* size of packet data <= 1436 without FCS

#### Return values:

*TRDP\_NO\_ERR* no error

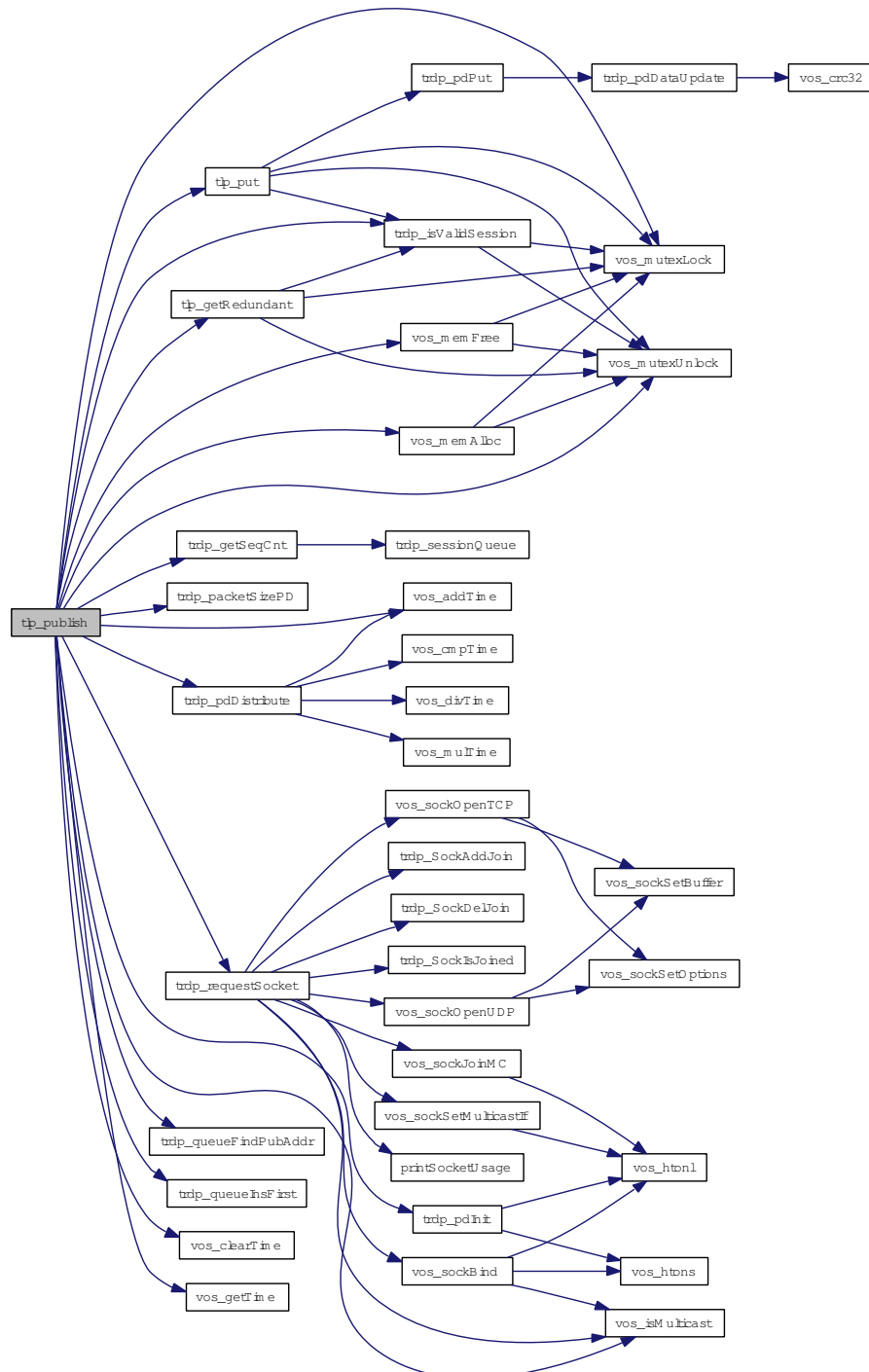
*TRDP\_PARAM\_ERR* parameter error

*TRDP\_MEM\_ERR* could not insert (out of memory)

*TRDP\_NOINIT\_ERR* handle invalid

*TRDP\_NOPUB\_ERR* Already published

Here is the call graph for this function:



#### 5.8.2.14 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.

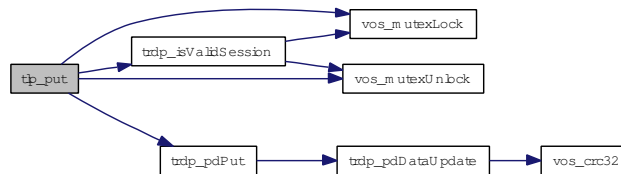
##### Parameters:

- ← *appHandle* the handle returned by tlc\_openSession
- ← *pubHandle* the handle returned by publish
- ↔ *pData* pointer to application's data buffer
- ↔ *dataSize* size of data

##### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error on uninitialized parameter or changed dataSize compared to published one
- TRDP\_NOPUB\_ERR** not published
- TRDP\_NOINIT\_ERR** handle invalid
- TRDP\_COMID\_ERR** ComID not found when marshalling

Here is the call graph for this function:



#### 5.8.2.15 EXT\_DECL TRDP\_ERR\_T tlp\_request (TRDP\_APP\_SESSION\_T *appHandle*, TRDP\_SUB\_T *subHandle*, UINT32 *comId*, UINT32 *topoCount*, TRDP\_IP\_ADDR\_T *srcIpAddr*, TRDP\_IP\_ADDR\_T *destIpAddr*, UINT32 *redId*, TRDP\_FLAGS\_T *pktFlags*, const TRDP\_SEND\_PARAM\_T \* *pSendParam*, const UINT8 \* *pData*, UINT32 *dataSize*, UINT32 *replyComId*, TRDP\_IP\_ADDR\_T *replyIpAddr*)

Initiate sending PD messages (PULL).

Send a PD request message

##### Parameters:

- ← *appHandle* the handle returned by tlc\_openSession
- ← *subHandle* handle from related subscribe
- ← *comId* comId of packet to be sent
- ← *topoCount* valid topocount, 0 for local consist
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack



← *destIpAddr* where to send the packet to

← *redId* 0 - Non-redundant, > 0 valid redundancy group

← *pktFlags* OPTION: TRDP\_FLAGS\_DEFAULT, TRDP\_FLAGS\_NONE, TRDP\_FLAGS\_MARSHALL, TRDP\_FLAGS\_CALLBACK

← *pSendParam* optional pointer to send parameter, NULL - default parameters are used

← *pData* pointer to packet data / dataset

← *dataSize* size of packet data

← *replyComId* comId of reply

← *replyIpAddr* IP for reply

#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_PARAM\_ERR** parameter error

**TRDP\_MEM\_ERR** could not insert (out of memory)

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_NOSUB\_ERR** no matching subscription found



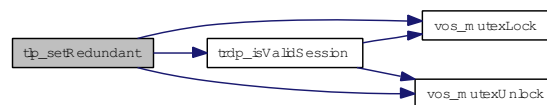
**Parameters:**

- ← *appHandle* the handle returned by tlc\_init
- ← *redId* will be set for all ComID's with the given redId, 0 to change for all redId
- ← *leader* TRUE if we send

**Return values:**

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error / redId not existing
- TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



**5.8.2.17** EXT\_DECL TRDP\_ERR\_T tlp\_subscribe (TRDP\_APP\_SESSION\_T *appHandle*, TRDP\_SUB\_T \**pSubHandle*, const void \**pUserRef*, UINT32 *comId*, UINT32 *topoCount*, 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*, UINT32 *maxDataSize*)

Prepare for receiving PD messages.

Subscribe to a specific PD ComID and source IP.

**Parameters:**

- ← *appHandle* the handle returned by tlc\_openSession
- *pSubHandle* return a handle for these messages
- ← *pUserRef* user supplied value returned within the info structure
- ← *comId* comId of packet to receive
- ← *topoCount* valid topocount, 0 for local consist
- ← *srcIpAddr1* IP for source filtering, set 0 if not used
- ← *srcIpAddr2* Second source IP address for source filtering, set to zero if not used. Used e.g. for source filtering of redundant devices.
- ← *pktFlags* OPTION: TRDP\_FLAGS\_DEFAULT, TRDP\_FLAGS\_NONE, TRDP\_FLAGS\_MARSHALL, TRDP\_FLAGS\_CALLBACK
- ← *destIpAddr* IP address to join
- ← *timeout* timeout (>= 10ms) in usec
- ← *toBehavior* timeout behavior
- ← *maxDataSize* expected max. size of packet data

**Return values:**

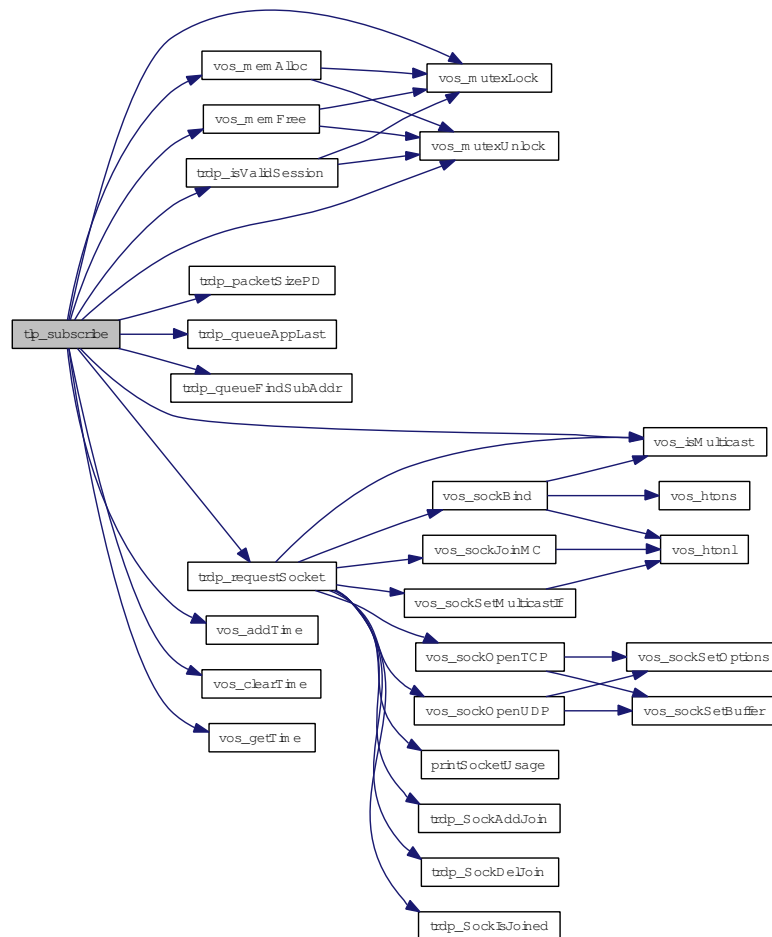
- TRDP\_NO\_ERR** no error

**TRDP\_PARAM\_ERR** parameter error

**TRDP\_MEM\_ERR** could not reserve memory (out of memory)

**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



#### 5.8.2.18 TRDP\_ERR\_T tlp\_unpublish (TRDP\_APP\_SESSION\_T appHandle, TRDP\_PUB\_T pubHandle)

Stop sending PD messages.

##### Parameters:

← **appHandle** the handle returned by tlc\_openSession

← **pubHandle** the handle returned by prepare

##### Return values:

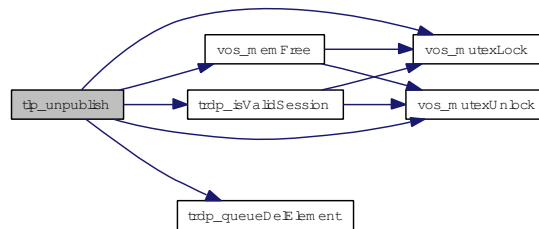
**TRDP\_NO\_ERR** no error

**TRDP\_PARAM\_ERR** parameter error

**TRDP\_NOPUB\_ERR** not published

**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



#### 5.8.2.19 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:

← **appHandle** the handle returned by tlc\_openSession

← **subHandle** the handle returned by subscription

##### Return values:

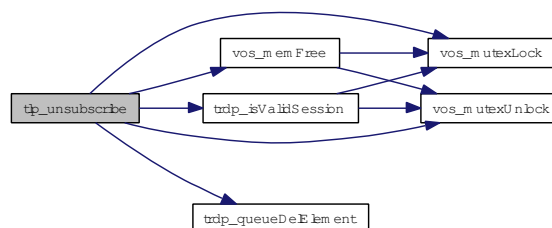
**TRDP\_NO\_ERR** no error

**TRDP\_PARAM\_ERR** parameter error

**TRDP\_NOSUB\_ERR** not subscribed

**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



### 5.8.2.20 BOOL trdp\_isValidSession (TRDP\_APP\_SESSION\_T *pSessionHandle*)

Check if the session handle is valid.

#### Parameters:

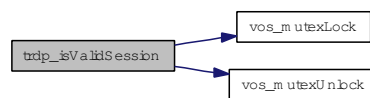
← *pSessionHandle* pointer to packet data (dataset)

#### Return values:

*TRUE* is valid

*FALSE* is invalid

Here is the call graph for this function:



### 5.8.2.21 TRDP\_APP\_SESSION\_T\* trdp\_sessionQueue (void)

Get the session queue head pointer.

#### Return values:

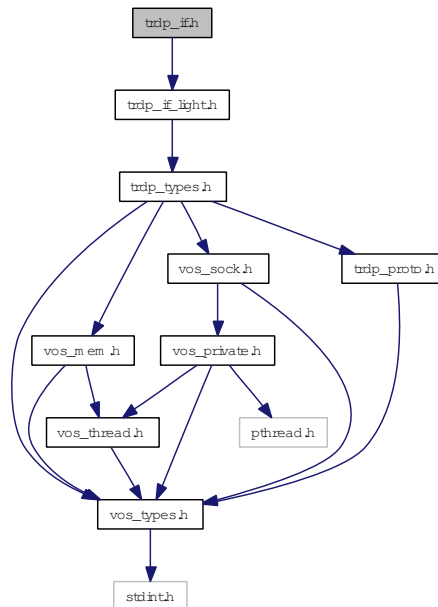
*&sSession*

## 5.9 trdp\_if.h File Reference

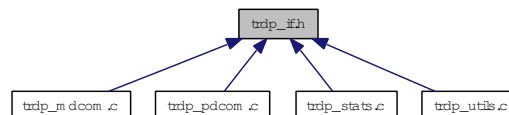
Typedefs for TRDP communication.

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

Include dependency graph for trdp\_if.h:



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



### Functions

- **BOOL** [trdp\\_isValidSession](#) (**TRDP\_APP\_SESSION\_T** pSessionHandle)  
*Check if the session handle is valid.*
- **TRDP\_APP\_SESSION\_T \*** [trdp\\_sessionQueue](#) (void)  
*Get the session queue head pointer.*

### 5.9.1 Detailed Description

Typedefs for TRDP communication.

#### Note:

Project: TCNOpen TRDP prototype stack

**Author:**

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

[trdp\\_if.h](#) 950 2013-06-13 13:51:41Z 97025

**5.9.2 Function Documentation****5.9.2.1 BOOL trdp\_isValidSession (TRDP\_APP\_SESSION\_T *pSessionHandle*)**

Check if the session handle is valid.

**Parameters:**

← *pSessionHandle* pointer to packet data (dataset)

**Return values:**

**TRUE** is valid

**FALSE** is invalid

**Parameters:**

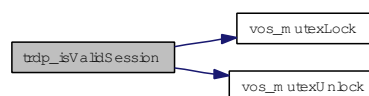
← *pSessionHandle* pointer to packet data (dataset)

**Return values:**

**TRUE** is valid

**FALSE** is invalid

Here is the call graph for this function:

**5.9.2.2 TRDP\_APP\_SESSION\_T\* trdp\_sessionQueue (void)**

Get the session queue head pointer.

**Return values:**

*&sSession*

*&sSession*

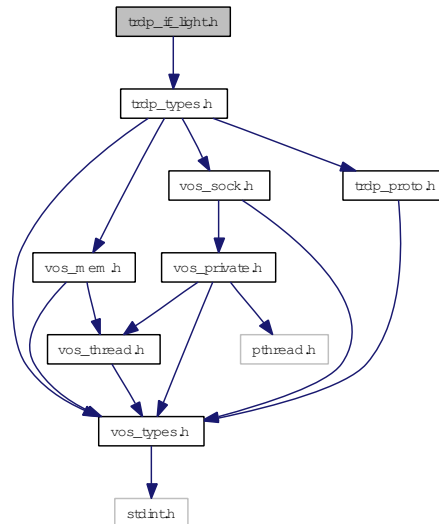


## 5.10 trdp\_if\_light.h File Reference

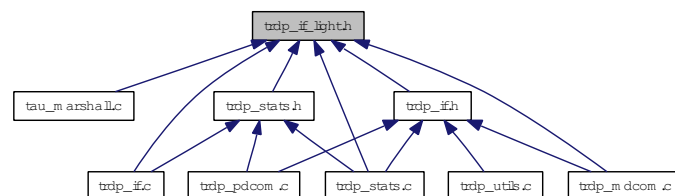
TRDP Light interface functions (API).

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

Include dependency graph for trdp\_if\_light.h:



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



### Defines

- `#define MD_SUPPORT 1`  
Support for message data can only be excluded during compile time!

### Functions

- `EXT_DECL TRDP_ERR_T tlc_init (const TRDP_PRINT_DBG_T pPrintDebugString, 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\\_reinitSession](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle)

*Re-Initialize.*

- 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\\_setTopoCount](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [UINT32](#) topoCount)

*Set new topocount for trainwide communication.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_freeBuf](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [char](#) \*pBuf)

*Frees the buffer reserved by the TRDP layer.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getInterval](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_TIME\\_T](#) \*pInterval, [TRDP\\_FDS\\_T](#) \*pFileDesc, [INT32](#) \*pNoDesc)

*Get the lowest time interval for PDs.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_process](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_FDS\\_T](#) \*pRfds, [INT32](#) \*pCount)

*Work loop of the TRDP handler.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlp\\_publish](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_PUB\\_T](#) \*pPubHandle, [UINT32](#) comId, [UINT32](#) topoCount, [TRDP\\_IP\\_ADDR\\_T](#) srcIpAddr, [TRDP\\_IP\\_ADDR\\_T](#) destIpAddr, [UINT32](#) interval, [UINT32](#) redId, [TRDP\\_FLAGS\\_T](#) pktFlags, const [TRDP\\_SEND\\_PARAM\\_T](#) \*pSendParam, const [UINT8](#) \*pData, [UINT32](#) dataSize)

*Prepare for sending PD messages.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlp\\_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\\_setRedundant](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [UINT32](#) redId, [BOOL](#) leader)

*Do not send redundant PD's when we are follower.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlp\\_getRedundant](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [UINT32](#) redId, [BOOL](#) \*pLeader)

*Get status of redundant ComIds.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlp\\_request](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_SUB\\_T](#) subHandle, [UINT32](#) comId, [UINT32](#) topoCount, [TRDP\\_IP\\_ADDR\\_T](#) srcIpAddr, [TRDP\\_IP\\_ADDR\\_T](#) destIpAddr, [UINT32](#) redId, [TRDP\\_FLAGS\\_T](#) pktFlags, const [TRDP\\_SEND\\_PARAM\\_T](#) \*pSendParam, const [UINT8](#) \*pData, [UINT32](#) dataSize, [UINT32](#) replyComId, [TRDP\\_IP\\_ADDR\\_T](#) replyIpAddr)

*Initiate sending PD messages (PULL).*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlp\\_subscribe](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_SUB\\_T](#) \*pSubHandle, const void \*pUserRef, [UINT32](#) comId, [UINT32](#) topoCount, [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, [UINT32](#) maxDataSize)

*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\\_get](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_SUB\\_T](#) subHandle, [TRDP\\_PD\\_INFO\\_T](#) \*pPdInfo, [UINT8](#) \*pData, [UINT32](#) \*pDataSize)

*Get the last valid PD message.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlm\\_notify](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, const void \*pUserRef, [UINT32](#) comId, [UINT32](#) topoCount, [TRDP\\_IP\\_ADDR\\_T](#) srcIpAddr, [TRDP\\_IP\\_ADDR\\_T](#) destIpAddr, [TRDP\\_FLAGS\\_T](#) pktFlags, const [TRDP\\_SEND\\_PARAM\\_T](#) \*pSendParam, const [UINT8](#) \*pData, [UINT32](#) dataSize, const [TRDP\\_URI\\_USER\\_T](#) sourceURI, const [TRDP\\_URI\\_USER\\_T](#) destURI)

*Initiate sending MD notification message.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlm\\_request](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, const void \*pUserRef, [TRDP\\_UUID\\_T](#) \*pSessionId, [UINT32](#) comId, [UINT32](#) topoCount, [TRDP\\_IP\\_ADDR\\_T](#) srcIpAddr, [TRDP\\_IP\\_ADDR\\_T](#) destIpAddr, [TRDP\\_FLAGS\\_T](#) pktFlags, [UINT32](#) numReplies, [UINT32](#) replyTimeout, const [TRDP\\_SEND\\_PARAM\\_T](#) \*pSendParam, const [UINT8](#) \*pData, [UINT32](#) dataSize, const [TRDP\\_URI\\_USER\\_T](#) sourceURI, const [TRDP\\_URI\\_USER\\_T](#) destURI)

*Initiate sending MD request message.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlm\\_confirm](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, const void \*pUserRef, const [TRDP\\_UUID\\_T](#) \*pSessionId, [UINT32](#) comId, [UINT32](#) topoCount, [TRDP\\_IP\\_ADDR\\_T](#) srcIpAddr, [TRDP\\_IP\\_ADDR\\_T](#) destIpAddr, [TRDP\\_FLAGS\\_T](#) pktFlags, [UINT16](#) userStatus, [TRDP\\_REPLY\\_STATUS\\_T](#) replyStatus, const [TRDP\\_SEND\\_PARAM\\_T](#) \*pSendParam, const [TRDP\\_URI\\_USER\\_T](#) sourceURI, const [TRDP\\_URI\\_USER\\_T](#) destURI)

*Initiate sending MD confirm message.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlm\\_abortSession](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, const [TRDP\\_UUID\\_T](#) \*pSessionId)

*Cancel an open session.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlm\\_addListener](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_LIS\\_T](#) \*pListenHandle, const void \*pUserRef, [UINT32](#) comId, [UINT32](#) topoCount, [TRDP\\_IP\\_ADDR\\_T](#) mcDestIpAddr, [TRDP\\_FLAGS\\_T](#) pktFlags, 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\\_reply](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, void \*pUserRef, const [TRDP\\_UUID\\_T](#) \*pSessionId, UINT32 topoCount, UINT32 comId, [TRDP\\_IP\\_ADDR\\_T](#) srcIpAddr, [TRDP\\_IP\\_ADDR\\_T](#) destIpAddr, [TRDP\\_FLAGS\\_T](#) pktFlags, UINT16 userStatus, const [TRDP\\_SEND\\_PARAM\\_T](#) \*pSendParam, const UINT8 \*pData, UINT32 dataSize, const [TRDP\\_URI\\_USER\\_T](#) sourceURI, const [TRDP\\_URI\\_USER\\_T](#) destURI)

*Send a MD reply message.*

- EXT\_DECL [TRDP\\_ERR\\_T tlm\\_replyQuery](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, void \*pUserRef, const [TRDP\\_UUID\\_T](#) \*pSessionId, UINT32 topoCount, UINT32 comId, [TRDP\\_IP\\_ADDR\\_T](#) srcIpAddr, [TRDP\\_IP\\_ADDR\\_T](#) destIpAddr, [TRDP\\_FLAGS\\_T](#) pktFlags, UINT16 userStatus, UINT32 confirmTimeout, const [TRDP\\_SEND\\_PARAM\\_T](#) \*pSendParam, const UINT8 \*pData, UINT32 dataSize, const [TRDP\\_URI\\_USER\\_T](#) sourceURI, const [TRDP\\_URI\\_USER\\_T](#) destURI)

*Send a MD reply message.*

- EXT\_DECL [TRDP\\_ERR\\_T tlm\\_replyErr](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, const [TRDP\\_UUID\\_T](#) \*pSessionId, UINT32 topoCount, UINT32 comId, [TRDP\\_IP\\_ADDR\\_T](#) srcIpAddr, [TRDP\\_IP\\_ADDR\\_T](#) destIpAddr, [TRDP\\_REPLY\\_STATUS\\_T](#) replyState, const [TRDP\\_SEND\\_PARAM\\_T](#) \*pSendParam, const [TRDP\\_URI\\_USER\\_T](#) sourceURI, const [TRDP\\_URI\\_USER\\_T](#) destURI)

*Send a MD error reply message.*

- EXT\_DECL const CHAR8 \* [tlc\\_getVersionString](#) (void)

*Return a human readable version representation.*

- EXT\_DECL const [TRDP\\_VERSION\\_T](#) \* [tlc\\_getVersion](#) (void)

*Return version.*

- EXT\_DECL [TRDP\\_ERR\\_T tlc\\_getStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_STATISTICS\\_T](#) \*pStatistics)

*Return statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T tlc\\_getSubsStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT16 \*pNumSubs, [TRDP\\_SUBS\\_STATISTICS\\_T](#) \*pStatistics)

*Return PD subscription statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T tlc\\_getPubStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT16 \*pNumPub, [TRDP\\_PUB\\_STATISTICS\\_T](#) \*pStatistics)

*Return PD publish statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T tlc\\_getListStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT16 \*pNumList, [TRDP\\_LIST\\_STATISTICS\\_T](#) \*pStatistics)

*Return MD listener statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T tlc\\_getRedStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT16 \*pNumRed, [TRDP\\_RED\\_STATISTICS\\_T](#) \*pStatistics)

*Return redundancy group statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T tlc\\_getJoinStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT16 \*pNumJoin, UINT32 \*pIpAddr)

*Return join statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T tlc\\_resetStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle)

*Reset statistics.*

### 5.10.1 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:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

#### Id

[trdp\\_if\\_light.h](#) 897 2013-06-05 15:03:51Z bloehr

### 5.10.2 Function Documentation

#### 5.10.2.1 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:

← *appHandle* The handle returned by tlc\_openSession

#### Return values:

*TRDP\_NO\_ERR* no error

*TRDP\_NOINIT\_ERR* handle invalid

*TRDP\_PARAM\_ERR* handle NULL

Clean up and release all resources of that session

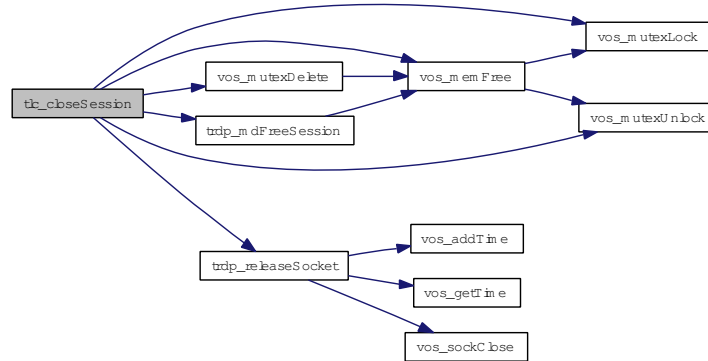
#### Parameters:

← *appHandle* The handle returned by tlc\_openSession

**Return values:**

**TRDP\_NO\_ERR** no error  
**TRDP\_NOINIT\_ERR** handle invalid  
**TRDP\_PARAM\_ERR** handle NULL

Here is the call graph for this function:



### 5.10.2.2 EXT\_DECL TRDP\_ERR\_T tlc\_freeBuf (TRDP\_APP\_SESSION\_T *appHandle*, char \* *pBuf*)

Frees the buffer reserved by the TRDP layer.

**Parameters:**

← **appHandle** The handle returned by tlc\_init  
 ← **pBuf** pointer to the buffer to be freed

**Return values:**

**TRDP\_NO\_ERR** no error  
**TRDP\_NOINIT\_ERR** handle invalid  
**TRDP\_PARAM\_ERR** buffer pointer invalid

### 5.10.2.3 EXT\_DECL TRDP\_ERR\_T tlc\_getInterval (TRDP\_APP\_SESSION\_T *appHandle*, TRDP\_TIME\_T \* *pInterval*, TRDP\_FDS\_T \* *pFileDesc*, INT32 \* *pNoDesc*)

Get the lowest time interval for PDs.

Return the maximum time interval suitable for 'select()' so that we can send due PD packets in time. If the PD send queue is empty, return zero time

**Parameters:**

← **appHandle** The handle returned by tlc\_init  
 → **pInterval** pointer to needed interval

↔ *pFileDesc* pointer to file descriptor set

→ *pNoDesc* pointer to put no of used descriptors (for select())

#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

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:

← *appHandle* The handle returned by tlc\_openSession

→ *pInterval* pointer to needed interval

↔ *pFileDesc* pointer to file descriptor set

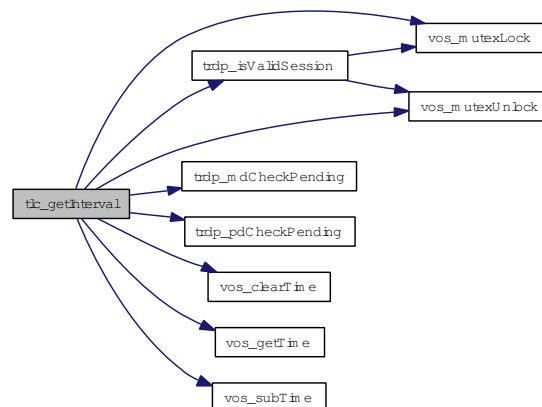
→ *pNoDesc* pointer to put no of highest used descriptors (for select())

#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



#### 5.10.2.4 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. must be provided by the user. The reserved length is given via pNumJoin implicitly.

#### Parameters:

← *appHandle* the handle returned by tlc\_openSession

- ↔ *pNumJoin* Pointer to the number of joined IP Addresses
- *pIpAddr* Pointer to a list with the joined IP addresses

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_NOINIT\_ERR* handle invalid
- TRDP\_PARAM\_ERR* parameter error
- TRDP\_MEM\_ERR* there are more items than requested

Memory for statistics information must be provided by the user.

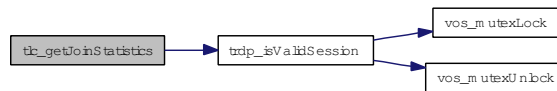
**Parameters:**

- ← *appHandle* the handle returned by `tlc_openSession`
- ↔ *pNumJoin* Pointer to the number of joined IP Addresses
- *pIpAddr* Pointer to a list with the joined IP addresses

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_NOINIT\_ERR* handle invalid
- TRDP\_PARAM\_ERR* parameter error
- TRDP\_MEM\_ERR* there are more items than requested

Here is the call graph for this function:



#### 5.10.2.5 `EXT_DECL TRDP_ERR_T tlc_getListStatistics (TRDP_APP_SESSION_T appHandle, UINT16 * pNumList, TRDP_LIST_STATISTICS_T * pStatistics)`

Return MD listener statistics.

Memory for statistics information must be provided by the user. The reserved length is given via `pNumList` implicitly.

**Parameters:**

- ← *appHandle* the handle returned by `tlc_openSession`
- ↔ *pNumList* Pointer to the number of listeners
- *pStatistics* Pointer to a list with the listener statistics information

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_NOINIT\_ERR* handle invalid



**TRDP\_PARAM\_ERR** parameter error

**TRDP\_MEM\_ERR** there are more subscriptions than requested

Memory for statistics information must be provided by the user.

#### Parameters:

← **appHandle** the handle returned by `tlc_openSession`

↔ **pNumList** Pointer to the number of listeners

→ **pStatistics** Pointer to a list with the listener statistics information

#### Return values:

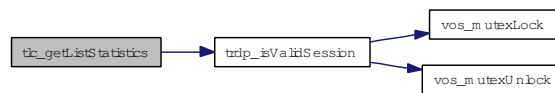
**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

**TRDP\_MEM\_ERR** there are more subscriptions than requested

Here is the call graph for this function:



#### 5.10.2.6 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. The reserved length is given via `pNumPub` implicitly.

#### Parameters:

← **appHandle** the handle returned by `tlc_openSession`

↔ **pNumPub** Pointer to the number of publishers

→ **pStatistics** pointer to a list with the publish statistics information

#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

**TRDP\_MEM\_ERR** there are more subscriptions than requested

Memory for statistics information must be provided by the user.

#### Parameters:

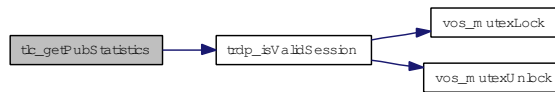
← **appHandle** the handle returned by `tlc_openSession`

- ↔ *pNumPub* Pointer to the number of publishers
- *pStatistics* Pointer to a list with the publish statistics information

**Return values:**

- TRDP\_NO\_ERR** no error
- TRDP\_NOINIT\_ERR** handle invalid
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_MEM\_ERR** there are more subscriptions than requested

Here is the call graph for this function:



#### 5.10.2.7 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. The reserved length is given via pNumRed implicitly.

**Parameters:**

- ← *appHandle* the handle returned by `tlc_openSession`
- ↔ *pNumRed* Pointer to the number of redundancy groups
- *pStatistics* Pointer to a list with the redundancy group information

**Return values:**

- TRDP\_NO\_ERR** no error
- TRDP\_NOINIT\_ERR** handle invalid
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_MEM\_ERR** there are more subscriptions than requested

Memory for statistics information must be provided by the user.

**Parameters:**

- ← *appHandle* the handle returned by `tlc_openSession`
- ↔ *pNumRed* Pointer to the number of redundancy groups
- *pStatistics* Pointer to a list with the redundancy group information

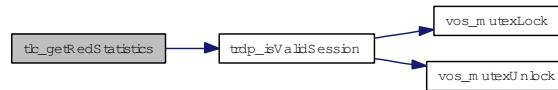
**Return values:**

- TRDP\_NO\_ERR** no error
- TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

**TRDP\_MEM\_ERR** there are more subscriptions than requested

Here is the call graph for this function:



#### 5.10.2.8 EXT\_DECL TRDP\_ERR\_T tlc\_getStatistics (TRDP\_APP\_SESSION\_T *appHandle*, TRDP\_STATISTICS\_T \**pStatistics*)

Return statistics.

Memory for statistics information must be preserved by the user.

##### Parameters:

← ***appHandle*** the handle returned by `tlc_init`

→ ***pStatistics*** Pointer to statistics for this application session

##### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

Memory for statistics information must be provided by the user.

##### Parameters:

← ***appHandle*** the handle returned by `tlc_openSession`

→ ***pStatistics*** Pointer to statistics for this application session

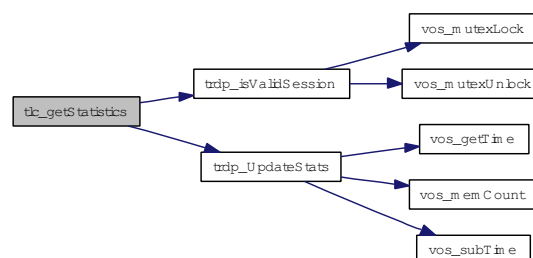
##### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

Here is the call graph for this function:



### 5.10.2.9 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. The reserved length is given via *pNumSub* implicitly.

#### Parameters:

- ← *appHandle* the handle returned by `tlc_openSession`
- ↔ *pNumSubs* In: The number of subscriptions requested Out: Number of subscriptions returned
- ↔ *pStatistics* Pointer to an array with the subscription statistics information

#### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_NOINIT\_ERR** handle invalid
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_MEM\_ERR** there are more subscriptions than requested

Memory for statistics information must be provided by the user.

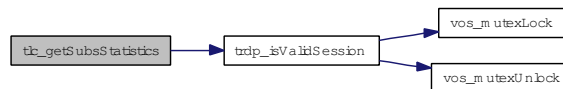
#### Parameters:

- ← *appHandle* the handle returned by `tlc_openSession`
- ↔ *pNumSubs* In: The number of subscriptions requested Out: Number of subscriptions returned
- ↔ *pStatistics* Pointer to an array with the subscription statistics information

#### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_NOINIT\_ERR** handle invalid
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_MEM\_ERR** there are more subscriptions than requested

Here is the call graph for this function:



### 5.10.2.10 EXT\_DECL const TRDP\_VERSION\_T\* tlc\_getVersion (void)

Return version.

Return pointer to version structure

#### Return values:

**const** `TRDP_VERSION_T`

Return pointer to version structure

**Return values:**

*TRDP\_VERSION\_T*

**5.10.2.11 EXT\_DECL const CHAR8\* tlc\_getVersionString (void)**

Return a human readable version representation.

Return string in the form 'v.r.u.b'

**Return values:**

*const* string

**5.10.2.12 EXT\_DECL TRDP\_ERR\_T tlc\_init (const TRDP\_PRINT\_DBG\_T *pPrintDebugString*,  
const TRDP\_MEM\_CONFIG\_T \* *pMemConfig*)**

Initialize the TRDP stack.

tlc\_init returns in pAppHandle a unique handle to be used in further calls to the stack.

**Parameters:**

← *pPrintDebugString* Pointer to debug print function

← *pMemConfig* Pointer to memory configuration

**Return values:**

*TRDP\_NO\_ERR* no error

*TRDP\_MEM\_ERR* memory allocation failed

*TRDP\_PARAM\_ERR* initialization error

tlc\_init returns in pAppHandle a unique handle to be used in further calls to the stack.

**Parameters:**

← *pPrintDebugString* Pointer to debug print function

← *pMemConfig* Pointer to memory configuration

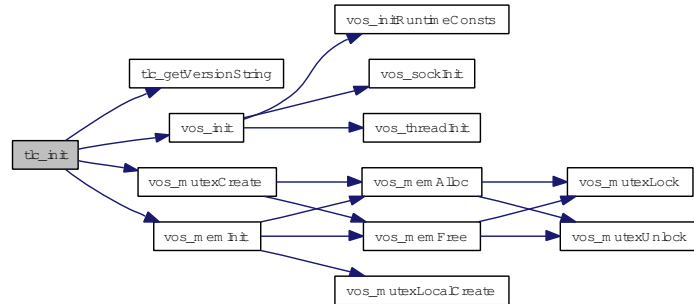
**Return values:**

*TRDP\_NO\_ERR* no error

*TRDP\_MEM\_ERR* memory allocation failed

*TRDP\_PARAM\_ERR* initialization error

Here is the call graph for this function:



**5.10.2.13** `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:

- **pAppHandle** A handle for further calls to the trdp stack
- ← **ownIpAddr** Own IP address, can be different for each process in multihoming systems, if zero, the default interface / IP will be used.
- ← **leaderIpAddr** IP address of redundancy leader
- ← **pMarshall** Pointer to marshalling configuration
- ← **pPdDefault** Pointer to default PD configuration
- ← **pMdDefault** Pointer to default MD configuration
- ← **pProcessConfig** 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:

- TRDP\_NO\_ERR** no error
- TRDP\_INIT\_ERR** not yet initied
- TRDP\_PARAM\_ERR** parameter error
- TRDP SOCK\_ERR** socket error

tlc\_openSession returns in pAppHandle a unique handle to be used in further calls to the stack.

#### Parameters:

- **pAppHandle** A handle for further calls to the trdp stack
- ← **ownIpAddr** Own IP address, can be different for each process in multihoming systems, if zero, the default interface / IP will be used.

← *leaderIpAddr* IP address of redundancy leader

← *pMarshall* Pointer to marshalling configuration

← *pPdDefault* Pointer to default PD configuration

← *pMdDefault* Pointer to default MD configuration

← *pProcessConfig* 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:

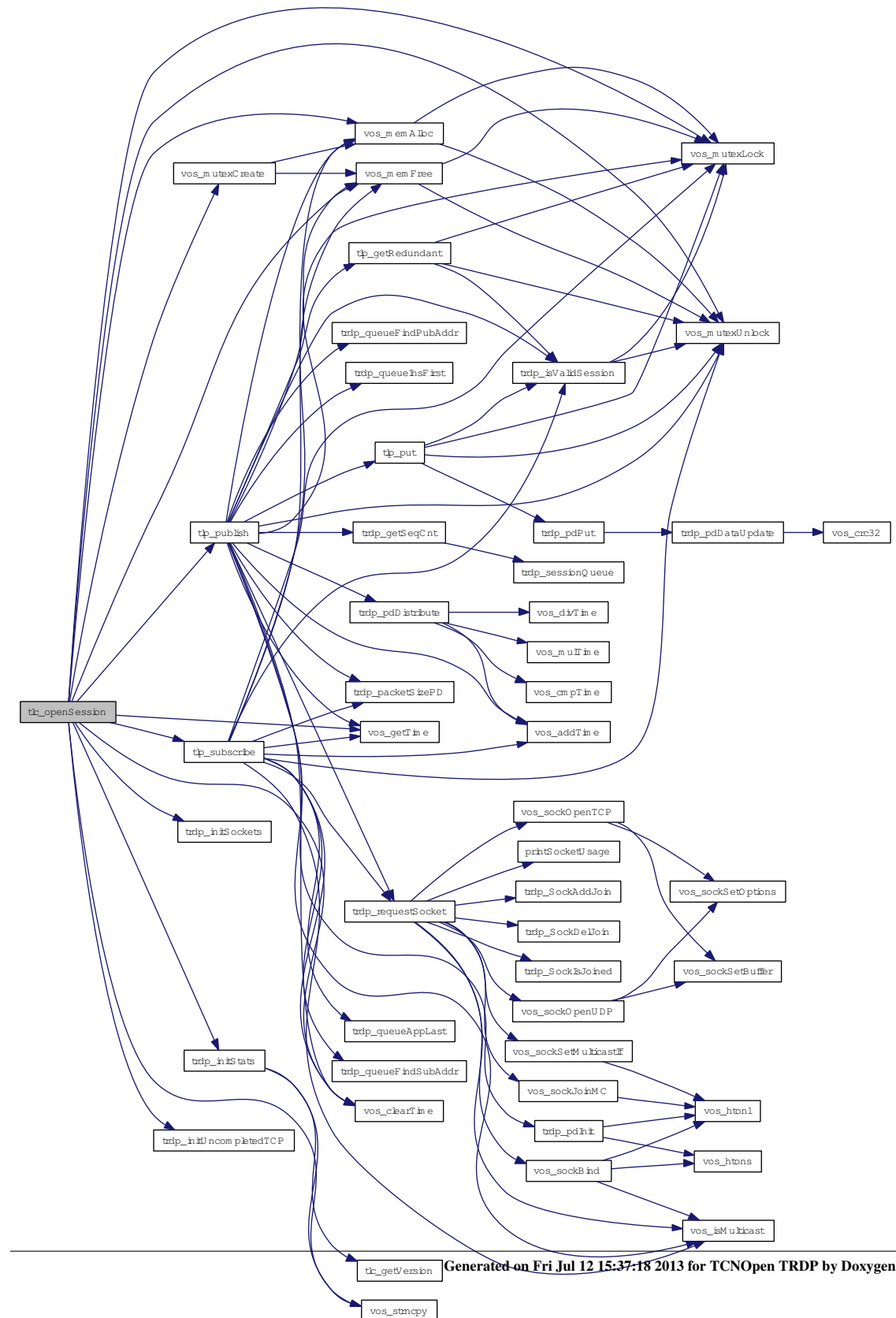
*TRDP\_NO\_ERR* no error

*TRDP\_INIT\_ERR* not yet initied

*TRDP\_PARAM\_ERR* parameter error

*TRDP SOCK\_ERR* socket error

Here is the call graph for this function:





### 5.10.2.14 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 to be sent Search the receive queue for pending PDs (time out)

#### Parameters:

← *appHandle* The handle returned by tlc\_init

← *pRfds* pointer to set of ready descriptors

↔ *pCount* pointer to number of ready descriptors

#### Return values:

*TRDP\_NO\_ERR* no error

*TRDP\_NOINIT\_ERR* handle invalid

Search the queue for pending PDs to be sent Search the receive queue for pending PDs (time out)

#### Parameters:

← *appHandle* The handle returned by tlc\_openSession

← *pRfds* pointer to set of ready descriptors

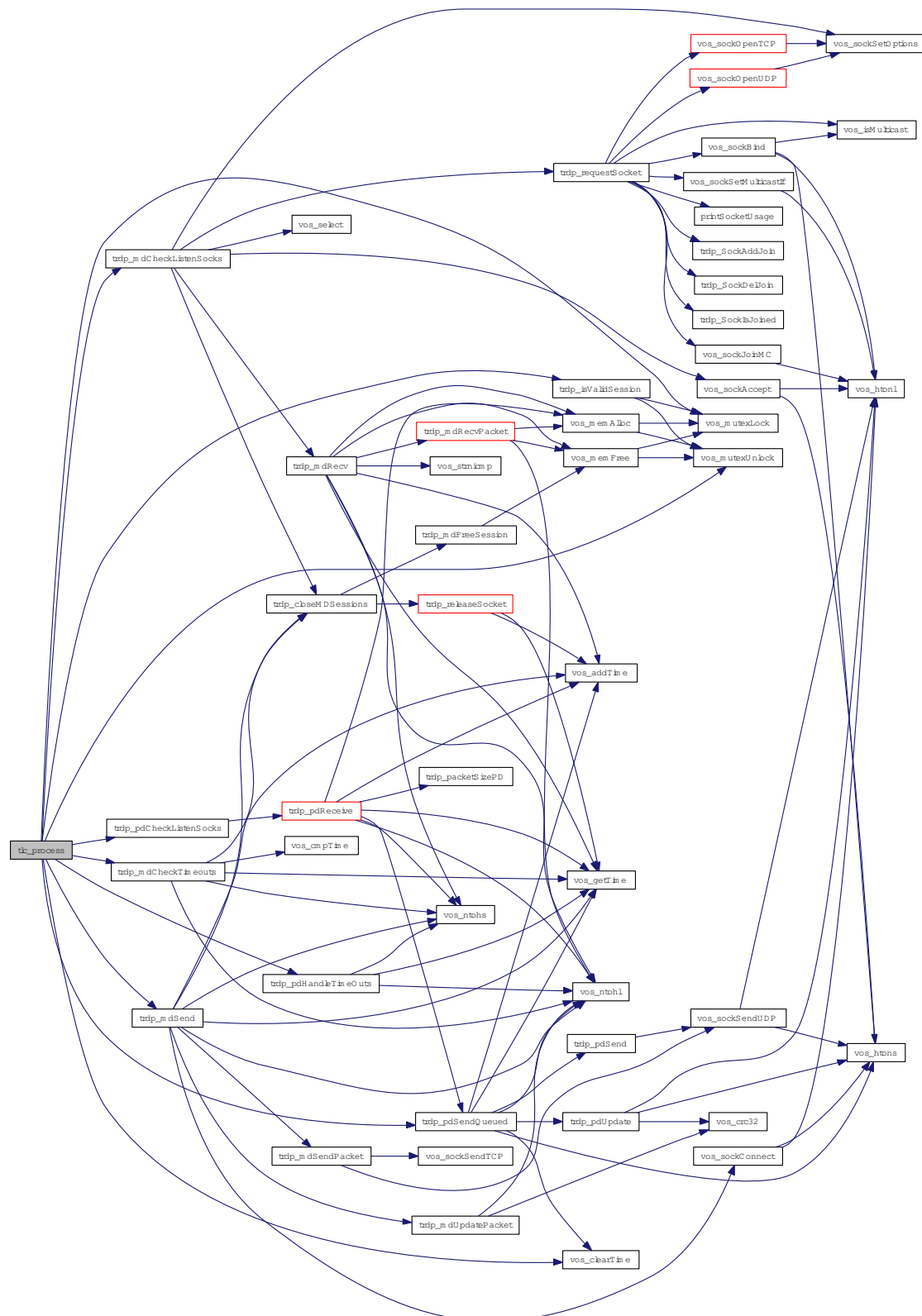
↔ *pCount* pointer to number of ready descriptors

#### Return values:

*TRDP\_NO\_ERR* no error

*TRDP\_NOINIT\_ERR* handle invalid

Here is the call graph for this function:



**5.10.2.15 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:**

← *appHandle* The handle returned by tlc\_openSession

**Return values:**

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** handle NULL

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:**

← *appHandle* The handle returned by tlc\_openSession

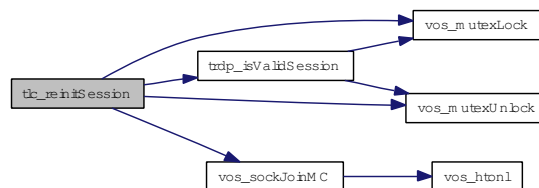
**Return values:**

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** handle NULL

Here is the call graph for this function:

**5.10.2.16 EXT\_DECL TRDP\_ERR\_T tlc\_resetStatistics (TRDP\_APP\_SESSION\_T *appHandle*)**

Reset statistics.

**Parameters:**

← *appHandle* the handle returned by tlc\_init

**Return values:**

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

**Parameters:**

← *appHandle* the handle returned by tlc\_openSession

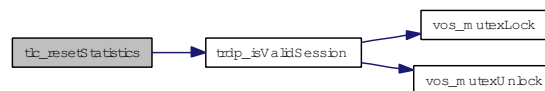
**Return values:**

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

Here is the call graph for this function:



#### 5.10.2.17 EXT\_DECL TRDP\_ERR\_T tlc\_setTopoCount (TRDP\_APP\_SESSION\_T appHandle, UINT32 topoCount)

Set new topocount for trainwide communication.

This value is used for validating outgoing and incoming packets only!

**Parameters:**

← *topoCount* New topocount value

This value is used for validating outgoing and incoming packets only!

**Parameters:**

← *appHandle* the handle returned by tlc\_openSession

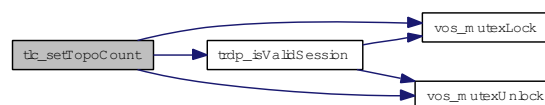
← *topoCount* New topoCount value

**Return values:**

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



### 5.10.2.18 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:

**TRDP\_NO\_ERR** no error

Clean up and close all sessions. Mainly used for debugging/test runs. No further calls to library allowed

#### Return values:

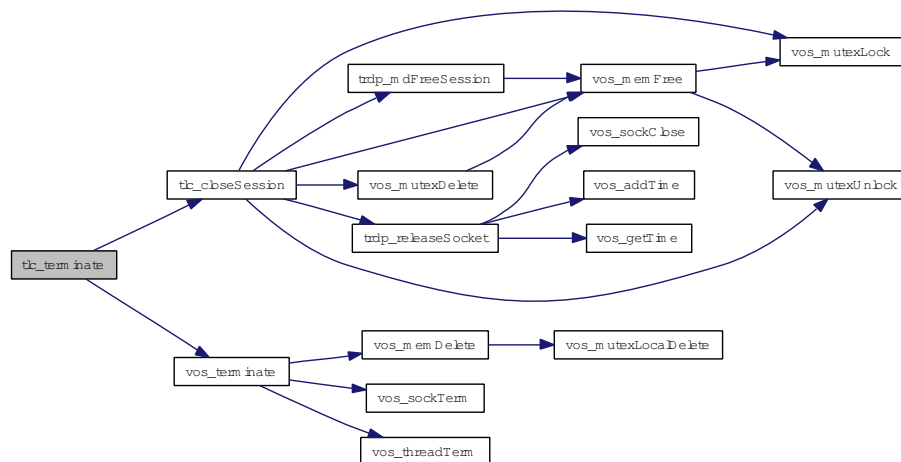
**TRDP\_NO\_ERR** no error

**TRDP\_INIT\_ERR** no error

**TRDP\_MEM\_ERR** TrafficStore nothing

**TRDP\_MUTEX\_ERR** TrafficStore mutex err

Here is the call graph for this function:



### 5.10.2.19 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:

← **appHandle** the handle returned by tlc\_init

← **pSessionId** Session ID returned by request

#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NO\_SESSION\_ERR** no such session

**TRDP\_NOINIT\_ERR** handle invalid

**5.10.2.20** **EXT\_DECL TRDP\_ERR\_T tlm\_addListener** (TRDP\_APP\_SESSION\_T *appHandle*, TRDP\_LIS\_T \* *pListenHandle*, const void \* *pUserRef*, UINT32 *comId*, UINT32 *topoCount*, TRDP\_IP\_ADDR\_T *mcDestIpAddr*, TRDP\_FLAGS\_T *pktFlags*, const TRDP\_URI\_USER\_T *destURI*)

Subscribe to MD messages.

Add a listener to TRDP to get notified when messages are received

**Parameters:**

- ← *appHandle* the handle returned by tlc\_init
- *pListenHandle* Listener ID returned
- ← *pUserRef* user supplied value returned with reply
- ← *comId* comId to be observed
- ← *topoCount* topocount to use
- ← *mcDestIpAddr* multicast group to listen on
- ← *pktFlags* OPTION: TRDP\_FLAGS\_DEFAULT, TRDP\_FLAGS\_MARSHALL, TRDP\_FLAGS\_TCP
- ← *destURI* only functional group of destination URI

**Return values:**

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_MEM\_ERR** out of memory
- TRDP\_NOINIT\_ERR** handle invalid

**5.10.2.21** **EXT\_DECL TRDP\_ERR\_T tlm\_confirm** (TRDP\_APP\_SESSION\_T *appHandle*, const void \* *pUserRef*, const TRDP\_UUID\_T \* *pSessionId*, UINT32 *comId*, UINT32 *topoCount*, TRDP\_IP\_ADDR\_T *srcIpAddr*, TRDP\_IP\_ADDR\_T *destIpAddr*, TRDP\_FLAGS\_T *pktFlags*, UINT16 *userStatus*, TRDP\_REPLY\_STATUS\_T *replyStatus*, const TRDP\_SEND\_PARAM\_T \* *pSendParam*, const TRDP\_URI\_USER\_T *sourceURI*, const TRDP\_URI\_USER\_T *destURI*)

Initiate sending MD confirm message.

Send a MD confirmation message

**Parameters:**

- ← *appHandle* the handle returned by tlc\_init
- ← *pUserRef* user supplied value returned with reply
- ← *pSessionId* Session ID returned by request
- ← *comId* comId of packet to be sent

- ← *topoCount* topocount to use
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to
- ← *pktFlags* OPTION: TRDP\_FLAGS\_DEFAULT
- ← *userStatus* Info for requester about application errors
- ← *replyStatus* Info for requester about stack errors
- ← *pSendParam* Pointer to send parameters, NULL to use default send parameters
- ← *sourceURI* only functional group of source URI
- ← *destURI* only functional group of destination URI

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* parameter error
- TRDP\_MEM\_ERR* out of memory
- TRDP\_NO\_SESSION\_ERR* no such session
- TRDP\_NOINIT\_ERR* handle invalid

#### 5.10.2.22 EXT\_DECL TRDP\_ERR\_T tlm\_delListener (TRDP\_APP\_SESSION\_T *appHandle*, TRDP\_LIS\_T *listenHandle*)

Remove Listener.

**Parameters:**

- ← *appHandle* the handle returned by tlc\_init
- *listenHandle* Listener ID returned

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* parameter error
- TRDP\_NOINIT\_ERR* handle invalid

#### 5.10.2.23 EXT\_DECL TRDP\_ERR\_T tlm\_notify (TRDP\_APP\_SESSION\_T *appHandle*, const void \* *pUserRef*, UINT32 *comId*, UINT32 *topoCount*, TRDP\_IP\_ADDR\_T *srcIpAddr*, TRDP\_IP\_ADDR\_T *destIpAddr*, TRDP\_FLAGS\_T *pktFlags*, const TRDP\_SEND\_PARAM\_T \* *pSendParam*, const UINT8 \* *pData*, UINT32 *dataSize*, const TRDP\_URI\_USER\_T *sourceURI*, const TRDP\_URI\_USER\_T *destURI*)

Initiate sending MD notification message.

Send a MD notification message

**Parameters:**

- ← *appHandle* the handle returned by tlc\_init
- ← *pUserRef* user supplied value returned with reply

- ← *comId* comId of packet to be sent
- ← *topoCount* topocount to use
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to
- ← *pktFlags* OPTIONS: TRDP\_FLAGS\_DEFAULT, TRDP\_FLAGS\_MARSHALL, TRDP\_FLAGS\_TCP
- ← *pSendParam* optional pointer to send parameter, NULL - default parameters are used
- ← *pData* pointer to packet data / dataset
- ← *dataSize* size of packet data
- ← *sourceURI* only functional group of source URI
- ← *destURI* only functional group of destination URI

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* parameter error
- TRDP\_MEM\_ERR* out of memory
- TRDP\_NOINIT\_ERR* handle invalid

**5.10.2.24** `EXT_DECL TRDP_ERR_T tlm_reply (TRDP_APP_SESSION_T appHandle, void * pUserRef, const TRDP_UUID_T * pSessionId, UINT32 topoCount, UINT32 comId, TRDP_IP_ADDR_T srcIpAddr, TRDP_IP_ADDR_T destIpAddr, TRDP_FLAGS_T pktFlags, UINT16 userStatus, const TRDP_SEND_PARAM_T * pSendParam, const UINT8 * pData, UINT32 dataSize, const TRDP_URI_USER_T sourceURI, const TRDP_URI_USER_T destURI)`

Send a MD reply message.

Send a MD reply message after receiving an request

**Parameters:**

- ← *appHandle* the handle returned by tlc\_init
- ← *pUserRef* user supplied value returned with reply
- ← *pSessionId* Session ID returned by indication
- ← *topoCount* topocount to use
- ← *comId* comId of packet to be sent
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to
- ← *pktFlags* OPTION: TRDP\_FLAGS\_DEFAULT, TRDP\_FLAGS\_MARSHALL
- ← *userStatus* Info for requester about application errors
- ← *pSendParam* pointer to send parameters, NULL to use default send parameters
- ← *pData* pointer to packet data / dataset
- ← *dataSize* size of packet data
- ← *sourceURI* only user part of source URI
- ← *destURI* only user part of destination URI



**Return values:**

**TRDP\_NO\_ERR** no error  
**TRDP\_PARAM\_ERR** parameter error  
**TRDP\_MEM\_ERR** out of memory  
**TRDP\_NO\_SESSION\_ERR** no such session  
**TRDP\_NOINIT\_ERR** handle invalid

**5.10.2.25** **EXT\_DECL TRDP\_ERR\_T tlm\_replyErr** (TRDP\_APP\_SESSION\_T *appHandle*, const TRDP\_UUID\_T \**pSessionId*, UINT32 *topoCount*, UINT32 *comId*, TRDP\_IP\_ADDR\_T *srcIpAddr*, TRDP\_IP\_ADDR\_T *destIpAddr*, TRDP\_REPLY\_STATUS\_T *replyState*, const TRDP\_SEND\_PARAM\_T \**pSendParam*, const TRDP\_URI\_USER\_T *sourceURI*, const TRDP\_URI\_USER\_T *destURI*)

Send a MD error reply message.

Send a MD error reply message after receiving an request

**Parameters:**

← **appHandle** the handle returned by tlc\_init  
 ← **pSessionId** Session ID returned by indication  
 ← **topoCount** topocount to use  
 ← **comId** comId of packet to be sent  
 ← **srcIpAddr** own IP address, 0 - srcIP will be set by the stack  
 ← **destIpAddr** where to send the packet to  
 ← **replyState** Info for requester about stack errors  
 ← **pSendParam** Pointer to send parameters, NULL to use default send parameters  
 ← **sourceURI** only user part of source URI  
 ← **destURI** only user part of destination URI

**Return values:**

**TRDP\_NO\_ERR** no error  
**TRDP\_PARAM\_ERR** parameter error  
**TRDP\_MEM\_ERR** out of memory  
**TRDP\_NO\_SESSION\_ERR** no such session  
**TRDP\_NOINIT\_ERR** handle invalid

**5.10.2.26** **EXT\_DECL TRDP\_ERR\_T tlm\_replyQuery** (TRDP\_APP\_SESSION\_T *appHandle*, void \**pUserRef*, const TRDP\_UUID\_T \**pSessionId*, UINT32 *topoCount*, UINT32 *comId*, TRDP\_IP\_ADDR\_T *srcIpAddr*, TRDP\_IP\_ADDR\_T *destIpAddr*, TRDP\_FLAGS\_T *pktFlags*, UINT16 *userStatus*, UINT32 *confirmTimeout*, const TRDP\_SEND\_PARAM\_T \**pSendParam*, const UINT8 \**pData*, UINT32 *dataSize*, const TRDP\_URI\_USER\_T *sourceURI*, const TRDP\_URI\_USER\_T *destURI*)

Send a MD reply message.

Send a MD reply message after receiving a request and ask for confirmation.

**Parameters:**

- ← *appHandle* the handle returned by `tlc_init`
- ← *pUserRef* user supplied value returned with reply
- ← *pSessionId* Session ID returned by indication
- ← *topoCount* topocount to use
- ← *comId* comId of packet to be sent
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to
- ← *pktFlags* OPTION: `TRDP_FLAGS_DEFAULT`, `TRDP_FLAGS_MARSHALL`
- ← *userStatus* Info for requester about application errors
- ← *confirmTimeout* timeout for confirmation
- ← *pSendParam* Pointer to send parameters, NULL to use default send parameters
- ← *pData* pointer to packet data / dataset
- ← *dataSize* size of packet data
- ← *sourceURI* only user part of source URI
- ← *destURI* only user part of destination URI

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* parameter error
- TRDP\_MEM\_ERR* out of memory
- TRDP\_NO\_SESSION\_ERR* no such session
- TRDP\_NOINIT\_ERR* handle invalid

**5.10.2.27** `EXT_DECL TRDP_ERR_T tlm_request (TRDP_APP_SESSION_T appHandle, const void * pUserRef, TRDP_UUID_T * pSessionId, UINT32 comId, UINT32 topoCount, TRDP_IP_ADDR_T srcIpAddr, TRDP_IP_ADDR_T destIpAddr, TRDP_FLAGS_T pktFlags, UINT32 numReplies, UINT32 replyTimeout, const TRDP_SEND_PARAM_T * pSendParam, const UINT8 * pData, UINT32 dataSize, const TRDP_URI_USER_T sourceURI, const TRDP_URI_USER_T destURI)`

Initiate sending MD request message.

Send a MD request message

**Parameters:**

- ← *appHandle* the handle returned by `tlc_init`
- ← *pUserRef* user supplied value returned with reply
- *pSessionId* return session ID
- ← *comId* comId of packet to be sent
- ← *topoCount* topocount to use
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to

- ← *pktFlags* OPTIONS: TRDP\_FLAGS\_DEFAULT, TRDP\_FLAGS\_MARSHALL, TRDP\_FLAGS\_TCP
- ← *numReplies* number of expected replies, 0 if unknown
- ← *replyTimeout* timeout for reply
- ← *pSendParam* Pointer to send parameters, NULL to use default send parameters
- ← *pData* pointer to packet data / dataset
- ← *dataSize* size of packet data
- ← *sourceURI* only functional group of source URI
- ← *destURI* only functional group of destination URI

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* parameter error
- TRDP\_MEM\_ERR* out of memory
- TRDP\_NOINIT\_ERR* handle invalid

#### 5.10.2.28 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 callback

**Parameters:**

- ← *appHandle* the handle returned by tlc\_init
- ← *subHandle* the handle returned by subscription
- ↔ *pPdInfo* pointer to application's info buffer
- ↔ *pData* pointer to application's data buffer
- ↔ *pDataSize* in: size of buffer, out: size of data

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* parameter error
- TRDP\_SUB\_ERR* not subscribed
- TRDP\_TIMEOUT\_ERR* packet timed out
- TRDP\_NOINIT\_ERR* handle invalid
- TRDP\_COMID\_ERR* ComID not found when marshalling

This allows polling of PDs instead of event driven handling by callbacks

**Parameters:**

- ← *appHandle* the handle returned by tlc\_openSession
- ← *subHandle* the handle returned by subscription

↔ *pPdInfo* pointer to application's info buffer

↔ *pData* pointer to application's data buffer

↔ *pDataSize* in: size of buffer, out: size of data

**Return values:**

*TRDP\_NO\_ERR* no error

*TRDP\_PARAM\_ERR* parameter error

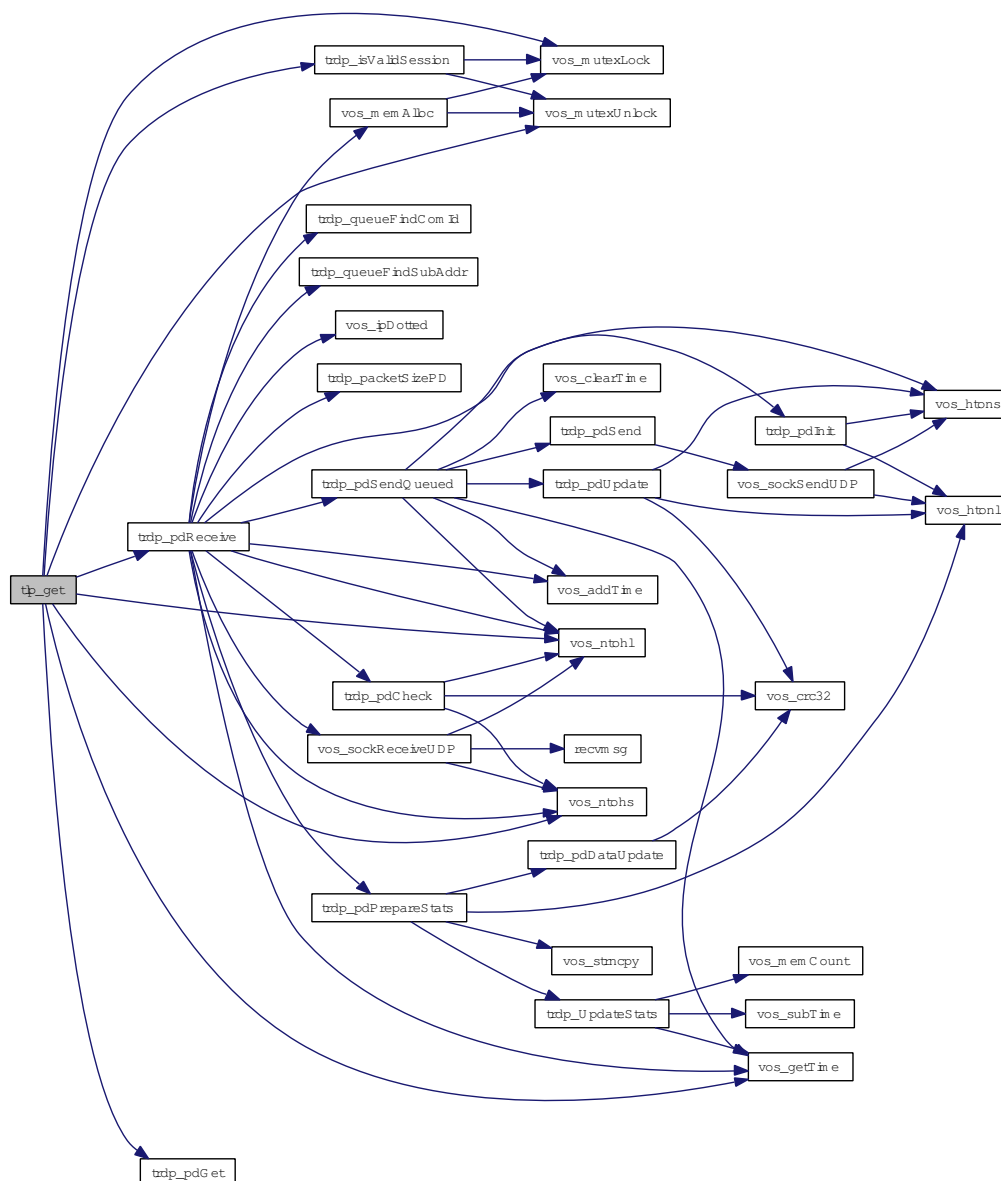
*TRDP\_SUB\_ERR* not subscribed

*TRDP\_TIMEOUT\_ERR* packet timed out

*TRDP\_NOINIT\_ERR* handle invalid

*TRDP\_COMID\_ERR* ComID not found when marshallng

Here is the call graph for this function:



#### 5.10.2.29 EXT\_DECL TRDP\_ERR\_T tlp\_getRedundant (TRDP\_APP\_SESSION\_T appHandle, UINT32 redId, BOOL \*pLeader)

Get status of redundant ComIds.

##### Parameters:

- ← **appHandle** the handle returned by tlc\_init
- ← **redId** will be set for all ComID's with the given redId, 0 for all redId
- ↔ **pLeader** TRUE if we send (leader)

**Return values:**

**TRDP\_NO\_ERR** no error  
**TRDP\_PARAM\_ERR** parameter error / redId not existing  
**TRDP\_NOINIT\_ERR** handle invalid

Only the status of the first redundancy group entry is returned will be returned!

**Parameters:**

← **appHandle** the handle returned by tlc\_init  
 ← **redId** will be returned for all ComID's with the given redId  
 ↔ **pLeader** TRUE if we're sending this redundancy group (leader)

**Return values:**

**TRDP\_NO\_ERR** no error  
**TRDP\_PARAM\_ERR** parameter error / redId not existing  
**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



**5.10.2.30** EXT\_DECL TRDP\_ERR\_T tlp\_publish (TRDP\_APP\_SESSION\_T *appHandle*, TRDP\_PUB\_T \* *pPubHandle*, UINT32 *comId*, UINT32 *topoCount*, TRDP\_IP\_ADDR\_T *srcIpAddr*, TRDP\_IP\_ADDR\_T *destIpAddr*, UINT32 *interval*, UINT32 *redId*, TRDP\_FLAGS\_T *pktFlags*, const TRDP\_SEND\_PARAM\_T \* *pSendParam*, const UINT8 \* *pData*, UINT32 *dataSize*)

Prepare for sending PD messages.

Queue a PD message, it will be send when trdp\_work has been called

**Parameters:**

← **appHandle** the handle returned by tlc\_init  
 → **pPubHandle** returned handle for related unprepare  
 ← **comId** comId of packet to send  
 ← **topoCount** valid topocount, 0 for local consist  
 ← **srcIpAddr** own IP address, 0 - srcIP will be set by the stack  
 ← **destIpAddr** where to send the packet to  
 ← **interval** frequency of PD packet (>= 10ms) in usec  
 ← **redId** 0 - Non-redundant, > 0 valid redundancy group

- ← *pktFlags* OPTION: TRDP\_FLAGS\_DEFAULT, TRDP\_FLAGS\_NONE, TRDP\_FLAGS\_MARSHALL, TRDP\_FLAGS\_CALLBACK
- ← *pSendParam* optional pointer to send parameter, NULL - default parameters are used
- ← *pData* pointer to packet data / dataset
- ← *dataSize* size of packet data

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* parameter error
- TRDP\_MEM\_ERR* could not insert (out of memory)
- TRDP\_NOINIT\_ERR* handle invalid

Queue a PD message, it will be send when trdp\_work has been called

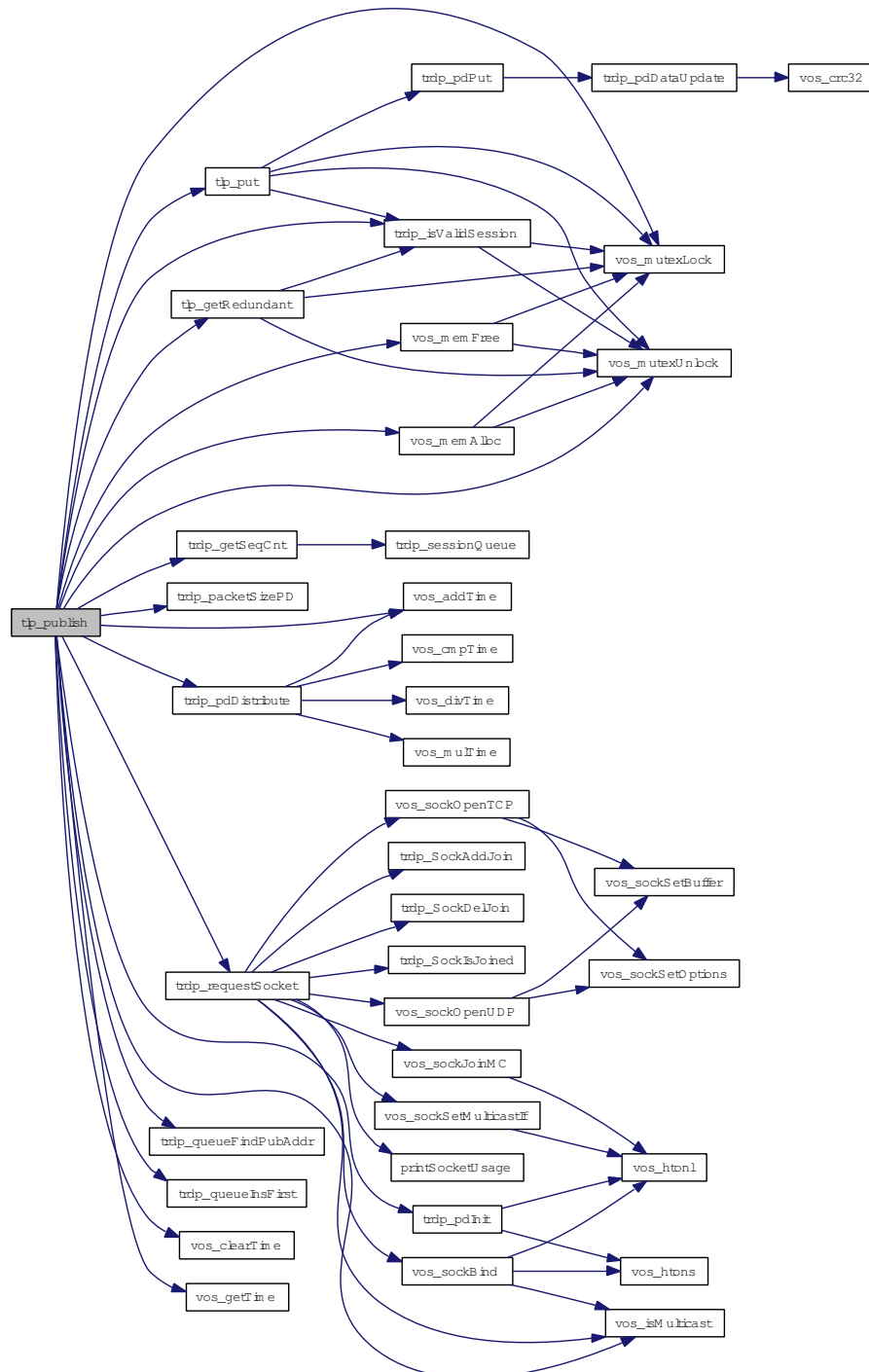
**Parameters:**

- ← *appHandle* the handle returned by tlc\_openSession
- *pPubHandle* returned handle for related unprepare
- ← *comId* comId of packet to send
- ← *topoCount* valid topocount, 0 for local consist
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to
- ← *interval* frequency of PD packet (>= 10ms) in usec, 0 if PD PULL
- ← *redId* 0 - Non-redundant, > 0 valid redundancy group
- ← *pktFlags* OPTION: TRDP\_FLAGS\_DEFAULT, TRDP\_FLAGS\_NONE, TRDP\_FLAGS\_MARSHALL, TRDP\_FLAGS\_CALLBACK
- ← *pSendParam* optional pointer to send parameter, NULL - default parameters are used
- ← *pData* pointer to packet data / dataset
- ← *dataSize* size of packet data <= 1436 without FCS

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* parameter error
- TRDP\_MEM\_ERR* could not insert (out of memory)
- TRDP\_NOINIT\_ERR* handle invalid
- TRDP\_NOPUB\_ERR* Already published

Here is the call graph for this function:





### 5.10.2.31 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:

- ← *appHandle* the handle returned by tlc\_init
- ← *pubHandle* the handle returned by publish
- ↔ *pData* pointer to application's data buffer
- ↔ *dataSize* size of data

#### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error on uninitialized parameter or changed dataSize compared to published one
- TRDP\_PUB\_ERR** not published
- TRDP\_NOINIT\_ERR** handle invalid
- TRDP\_COMID\_ERR** ComID not found when marshalling

Update previously published data. The new telegram will be sent earliest when tlc\_process is called.

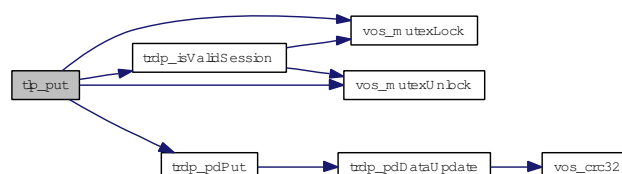
#### Parameters:

- ← *appHandle* the handle returned by tlc\_openSession
- ← *pubHandle* the handle returned by publish
- ↔ *pData* pointer to application's data buffer
- ↔ *dataSize* size of data

#### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error on uninitialized parameter or changed dataSize compared to published one
- TRDP\_NOPUB\_ERR** not published
- TRDP\_NOINIT\_ERR** handle invalid
- TRDP\_COMID\_ERR** ComID not found when marshalling

Here is the call graph for this function:



**5.10.2.32** **EXT\_DECL** **TRDP\_ERR\_T** **tlp\_request** (**TRDP\_APP\_SESSION\_T** *appHandle*, **TRDP\_SUB\_T** *subHandle*, **UINT32** *comId*, **UINT32** *topoCount*, **TRDP\_IP\_ADDR\_T** *srcIpAddr*, **TRDP\_IP\_ADDR\_T** *destIpAddr*, **UINT32** *redId*, **TRDP\_FLAGS\_T** *pktFlags*, **const** **TRDP\_SEND\_PARAM\_T** \* *pSendParam*, **const** **UINT8** \* *pData*, **UINT32** *dataSize*, **UINT32** *replyComId*, **TRDP\_IP\_ADDR\_T** *replyIpAddr*)

Initiate sending PD messages (PULL).

Send a PD request message

**Parameters:**

- ← *appHandle* the handle returned by `tlc_init`
- ← *subHandle* handle from related subscribe
- ← *comId* comId of packet to be sent
- ← *topoCount* valid topocount, 0 for local consist
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to
- ← *redId* 0 - Non-redundant, > 0 valid redundancy group
- ← *pktFlags* OPTIONS: `TTRDP_FLAGS_DEFAULT`, `TRDP_FLAGS_NONE`, `TRDP_FLAGS_-MARSHALL`, `TRDP_FLAGS_CALLBACK`
- ← *pSendParam* optional pointer to send parameter, NULL - default parameters are used
- ← *pData* pointer to packet data / dataset
- ← *dataSize* size of packet data
- ← *replyComId* comId of reply
- ← *replyIpAddr* IP for reply

**Return values:**

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_MEM\_ERR** could not insert (out of memory)
- TRDP\_NOINIT\_ERR** handle invalid

Send a PD request message

**Parameters:**

- ← *appHandle* the handle returned by `tlc_openSession`
- ← *subHandle* handle from related subscribe
- ← *comId* comId of packet to be sent
- ← *topoCount* valid topocount, 0 for local consist
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to
- ← *redId* 0 - Non-redundant, > 0 valid redundancy group
- ← *pktFlags* OPTION: `TRDP_FLAGS_DEFAULT`, `TRDP_FLAGS_NONE`, `TRDP_FLAGS_-MARSHALL`, `TRDP_FLAGS_CALLBACK`
- ← *pSendParam* optional pointer to send parameter, NULL - default parameters are used

← *pData* pointer to packet data / dataset

← *dataSize* size of packet data

← *replyComId* comId of reply

← *replyIpAddr* IP for reply

#### Return values:

*TRDP\_NO\_ERR* no error

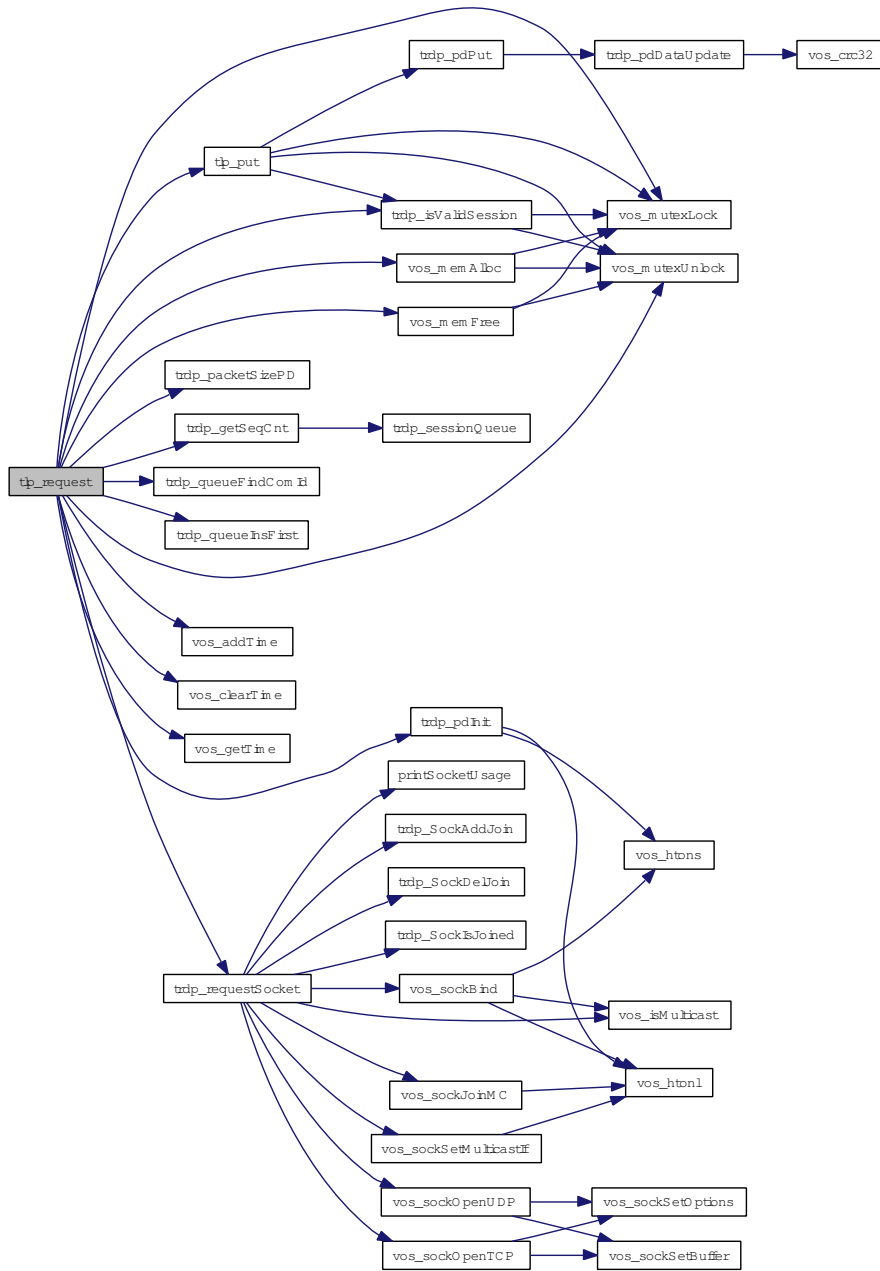
*TRDP\_PARAM\_ERR* parameter error

*TRDP\_MEM\_ERR* could not insert (out of memory)

*TRDP\_NOINIT\_ERR* handle invalid

*TRDP\_NOSUB\_ERR* no matching subscription found

Here is the call graph for this function:



### 5.10.2.33 EXT\_DECL TRDP\_ERR\_T tlp\_setRedundant (TRDP\_APP\_SESSION\_T *appHandle*, UINT32 *redId*, BOOL *leader*)

Do not send redundant PD's when we are follower.

#### Parameters:

← *appHandle* the handle returned by tlc\_init

← **redId** will be set for all ComID's with the given redId, 0 to change for all redId  
 ← **leader** TRUE if we send

**Return values:**

**TRDP\_NO\_ERR** no error  
**TRDP\_PARAM\_ERR** parameter error / redId not existing  
**TRDP\_NOINIT\_ERR** handle invalid

Do not send redundant PD's when we are follower.

**Parameters:**

← **appHandle** the handle returned by tlc\_init  
 ← **redId** will be set for all ComID's with the given redId, 0 to change for all redId  
 ← **leader** TRUE if we send

**Return values:**

**TRDP\_NO\_ERR** no error  
**TRDP\_PARAM\_ERR** parameter error / redId not existing  
**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



**5.10.234** EXT\_DECL TRDP\_ERR\_T tlp\_subscribe (TRDP\_APP\_SESSION\_T appHandle, TRDP\_SUB\_T \* pSubHandle, const void \* pUserRef, UINT32 comId, UINT32 topoCount, 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, UINT32 maxDataSize)

Prepare for receiving PD messages.

Subscribe to a specific PD ComID and source IP To unsubscribe, set maxDataSize to zero!

**Parameters:**

← **appHandle** the handle returned by tlc\_init  
 → **pSubHandle** return a handle for these messages  
 ← **pUserRef** user supplied value returned within the info structure  
 ← **comId** comId of packet to receive  
 ← **topoCount** valid topocount, 0 for local consist  
 ← **srcIpAddr1** IP for source filtering, set 0 if not used

- ← *srcIpAddr2* Second source IP address for source filtering, set to zero if not used. Used e.g. for source filtering of redundant devices.
- ← *destIpAddr* IP address to join
- ← *pktFlags* OPTION: TRDP\_FLAGS\_DEFAULT, TRDP\_FLAGS\_NONE, TRDP\_FLAGS\_MARSHALL, TRDP\_FLAGS\_CALLBACK
- ← *timeout* timeout (>= 10ms) in usec
- ← *toBehavior* OPTION: TRDP\_TO\_DEFAULT, TRDP\_TO\_SET\_TO\_ZERO, TRDP\_TO\_KEEP\_LAST\_VALUE
- ← *maxDataSize* expected max. size of packet data

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* parameter error
- TRDP\_MEM\_ERR* could not reserve memory (out of memory)
- TRDP\_NOINIT\_ERR* handle invalid

Subscribe to a specific PD ComID and source IP.

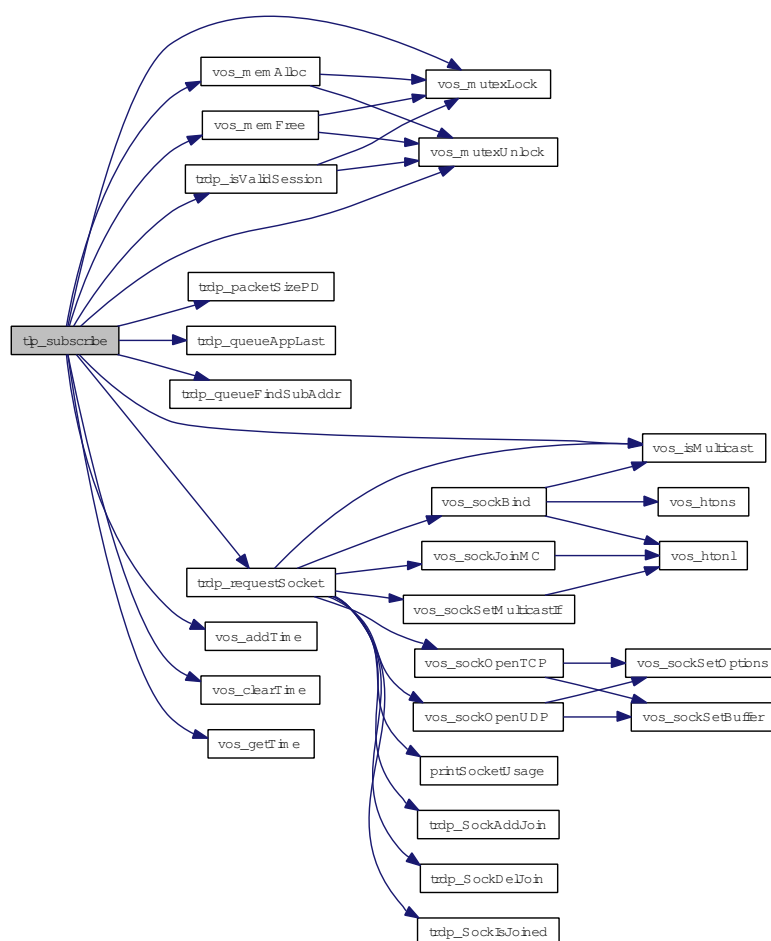
#### Parameters:

- ← *appHandle* the handle returned by tlc\_openSession
- *pSubHandle* return a handle for these messages
- ← *pUserRef* user supplied value returned within the info structure
- ← *comId* comId of packet to receive
- ← *topoCount* valid topocount, 0 for local consist
- ← *srcIpAddr1* IP for source filtering, set 0 if not used
- ← *srcIpAddr2* Second source IP address for source filtering, set to zero if not used. Used e.g. for source filtering of redundant devices.
- ← *pktFlags* OPTION: TRDP\_FLAGS\_DEFAULT, TRDP\_FLAGS\_NONE, TRDP\_FLAGS\_MARSHALL, TRDP\_FLAGS\_CALLBACK
- ← *destIpAddr* IP address to join
- ← *timeout* timeout (>= 10ms) in usec
- ← *toBehavior* timeout behavior
- ← *maxDataSize* expected max. size of packet data

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* parameter error
- TRDP\_MEM\_ERR* could not reserve memory (out of memory)
- TRDP\_NOINIT\_ERR* handle invalid

Here is the call graph for this function:



#### 5.10.2.35 EXT\_DECL TRDP\_ERR\_T tlp\_unpublish (TRDP\_APP\_SESSION\_T *appHandle*, TRDP\_PUB\_T *pubHandle*)

Stop sending PD messages.

##### Parameters:

- ← *appHandle* the handle returned by tlc\_init
- ← *pubHandle* the handle returned by prepare

##### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_NOPUB\_ERR** not published
- TRDP\_NOINIT\_ERR** handle invalid

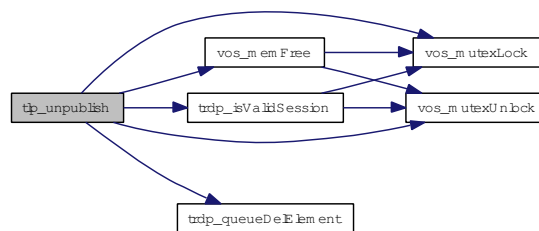
**Parameters:**

- ← *appHandle* the handle returned by `tlc_openSession`
- ← *pubHandle* the handle returned by `prepare`

**Return values:**

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_NOPUB\_ERR** not published
- TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



### 5.10.236 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:**

- ← *appHandle* the handle returned by `tlc_init`
- ← *subHandle* the handle returned by subscription

**Return values:**

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_SUB\_ERR** not subscribed
- TRDP\_NOINIT\_ERR** handle invalid

Unsubscribe to a specific PD ComID

**Parameters:**

- ← *appHandle* the handle returned by `tlc_openSession`
- ← *subHandle* the handle returned by subscription

**Return values:**

- TRDP\_NO\_ERR** no error

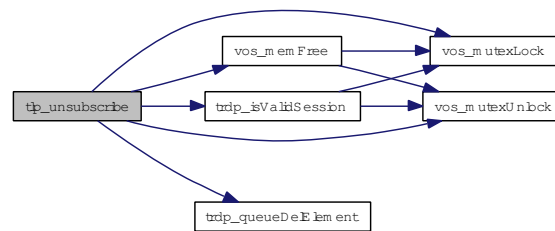


**TRDP\_PARAM\_ERR** parameter error

**TRDP\_NOSUB\_ERR** not subscribed

**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:

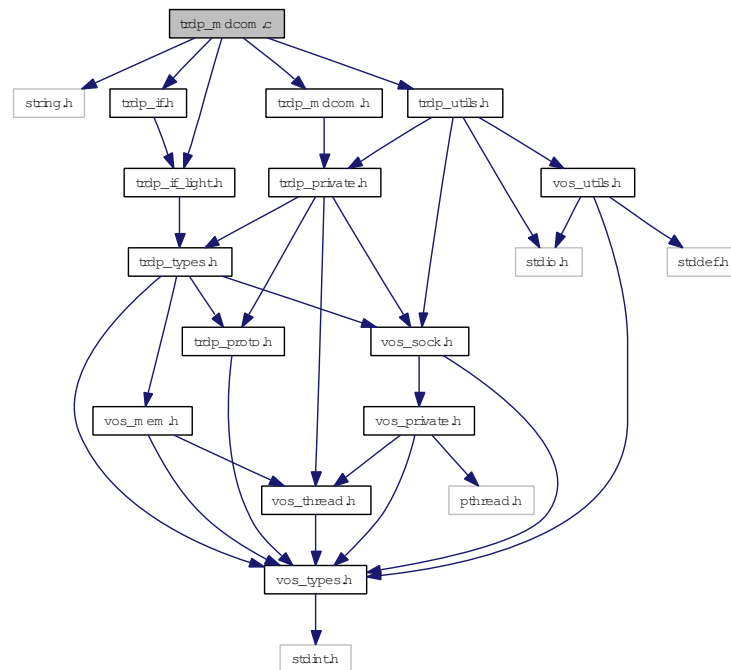


## 5.11 trdp\_mdcom.c File Reference

Functions for MD communication.

```
#include <string.h>
#include "trdp_if_light.h"
#include "trdp_if.h"
#include "trdp_utils.h"
#include "trdp_mdcom.h"
```

Include dependency graph for trdp\_mdcom.c:



### Functions

- [TRDP\\_ERR\\_T trdp\\_getTCPSocket](#) ([TRDP\\_SESSION\\_PT](#) pSession)  
*Initialize the specific parameters for message data Open a listening socket.*
- void [trdp\\_mdFreeSession](#) ([MD\\_ELE\\_T](#) \*pMDSession)  
*Free memory of session.*
- void [trdp\\_closeMDSessions](#) ([TRDP\\_SESSION\\_PT](#) appHandle, [INT32](#) socketIndex, [INT32](#) newSocket, [BOOL](#) checkAllSockets)  
*Close and free any session marked as dead.*
- void [trdp\\_mdSetSessionTimeout](#) ([MD\\_ELE\\_T](#) \*pMDSession, [UINT32](#) usTimeOut)  
*set time out*

- [TRDP\\_ERR\\_T trdp\\_mdCheck](#) ([TRDP\\_SESSION\\_PT](#) appHandle, [MD\\_HEADER\\_T](#) \*pPacket, [UINT32](#) packetSize, [BOOL](#) checkHeaderOnly)  
*Check for incoming md packet.*
- void [trdp\\_mdUpdatePacket](#) ([MD\\_ELE\\_T](#) \*pElement)  
*Update the header values.*
- [TRDP\\_ERR\\_T trdp\\_mdSendPacket](#) ([INT32](#) pdSock, [UINT32](#) port, [MD\\_ELE\\_T](#) \*pElement)  
*Send MD packet.*
- [TRDP\\_ERR\\_T trdp\\_mdRecvPacket](#) ([TRDP\\_SESSION\\_PT](#) appHandle, [INT32](#) mdSock, [MD\\_ELE\\_T](#) \*pElement)  
*Receive MD packet.*
- [TRDP\\_ERR\\_T trdp\\_mdRecv](#) ([TRDP\\_SESSION\\_PT](#) appHandle, [UINT32](#) sockIndex)  
*Receiving MD messages Read the receive socket for arriving MDs, copy the packet to a new MD\_ELE\_T Check for protocol errors and dispatch to proper receive queue.*
- [TRDP\\_ERR\\_T trdp\\_mdSend](#) ([TRDP\\_SESSION\\_PT](#) appHandle)  
*Sending MD messages Send the messages stored in the sendQueue Call user's callback if needed.*
- void [trdp\\_mdCheckPending](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_FDS\\_T](#) \*pFileDesc, [INT32](#) \*pNoDesc)  
*Check for pending packets, set FD if non blocking.*
- void [trdp\\_mdCheckListenSocks](#) ([TRDP\\_SESSION\\_PT](#) appHandle, [TRDP\\_FDS\\_T](#) \*pRfds, [INT32](#) \*pCount)  
*Checking receive connection requests and data Call user's callback if needed.*
- void [trdp\\_mdCheckTimeouts](#) ([TRDP\\_SESSION\\_PT](#) appHandle)  
*Checking message data timeouts Call user's callback if needed.*

### 5.11.1 Detailed Description

Functions for MD communication.

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Simone Pachera, FARsystems Gari Oiarbide, CAF Bernd Loehr, NewTec

#### Remarks:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2013.

#### Id

[trdp\\_mdcom.c](#) 950 2013-06-13 13:51:41Z 97025

## 5.11.2 Function Documentation

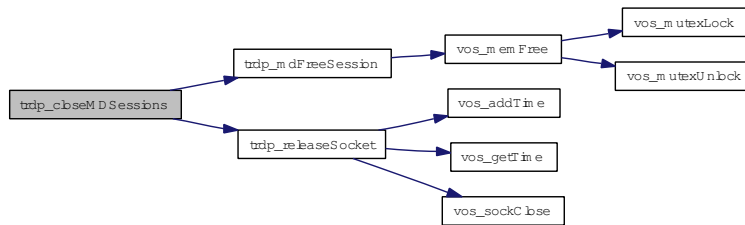
### 5.11.2.1 void trdp\_closeMDSessions (TRDP\_SESSION\_PT *appHandle*, INT32 *socketIndex*, INT32 *newSocket*, BOOL *checkAllSockets*)

Close and free any session marked as dead.

#### Parameters:

- ← *appHandle* session pointer
- ← *socketIndex* the old socket position in the iface[]
- ← *newSocket* the new socket
- ← *checkAllSockets* check all the sockets that are waiting to be closed

Here is the call graph for this function:



### 5.11.2.2 TRDP\_ERR\_T trdp\_getTCPSocket (TRDP\_SESSION\_PT *pSession*)

Initialize the specific parameters for message data Open a listening socket.

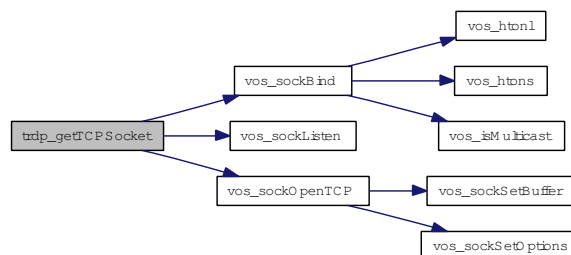
#### Parameters:

- ← *pSession* session parameters

#### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** initialization error

Here is the call graph for this function:



### 5.11.2.3 TRDP\_ERR\_T trdp\_mdCheck (TRDP\_SESSION\_PT *appHandle*, MD\_HEADER\_T \**pPacket*, UINT32 *packetSize*, BOOL *checkHeaderOnly*)

Check for incoming md packet.

#### Parameters:

- ← *appHandle* session pointer
- ← *pPacket* pointer to the packet to check
- ← *packetSize* size of the packet
- ← *checkHeaderOnly* TRUE if data crc should not be checked

#### Return values:

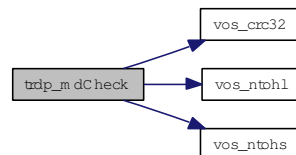
**TRDP\_NO\_ERR** no error

**TRDP\_TOPO\_ERR**

**TRDP\_WIRE\_ERR**

**TRDP\_CRC\_ERR**

Here is the call graph for this function:



### 5.11.2.4 void trdp\_mdCheckListenSocks (TRDP\_SESSION\_PT *appHandle*, TRDP\_FDS\_T \**pRfds*, INT32 \**pCount*)

Checking receive connection requests and data Call user's callback if needed.

#### Parameters:

- ← *appHandle* session pointer
- ← *pRfds* pointer to set of ready descriptors
- ↔ *pCount* pointer to number of ready descriptors



↔ *pNoDesc* pointer to number of ready descriptors

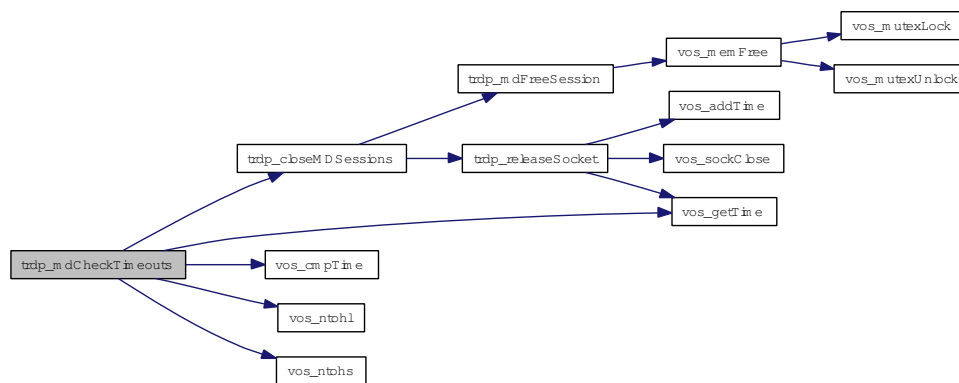
#### 5.11.2.6 void trdp\_mdCheckTimeouts (TRDP\_SESSION\_PT *appHandle*)

Checking message data timeouts Call user's callback if needed.

##### Parameters:

← *appHandle* session pointer

Here is the call graph for this function:



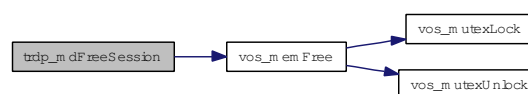
#### 5.11.2.7 void trdp\_mdFreeSession (MD\_ELEMENT \* *pMDSession*)

Free memory of session.

##### Parameters:

← *pMDSession* session pointer

Here is the call graph for this function:



#### 5.11.2.8 TRDP\_ERR\_T trdp\_mdRecv (TRDP\_SESSION\_PT *appHandle*, UINT32 *sockIndex*)

Receiving MD messages Read the receive socket for arriving MDs, copy the packet to a new MD\_ELEMENT  
Check for protocol errors and dispatch to proper receive queue.

Call user's callback if needed

##### Parameters:

← *appHandle* session pointer

← *sockIndex* index of the socket to read from

**Return values:**

**TRDP\_NO\_ERR** no error

**TRDP\_PARAM\_ERR** parameter error

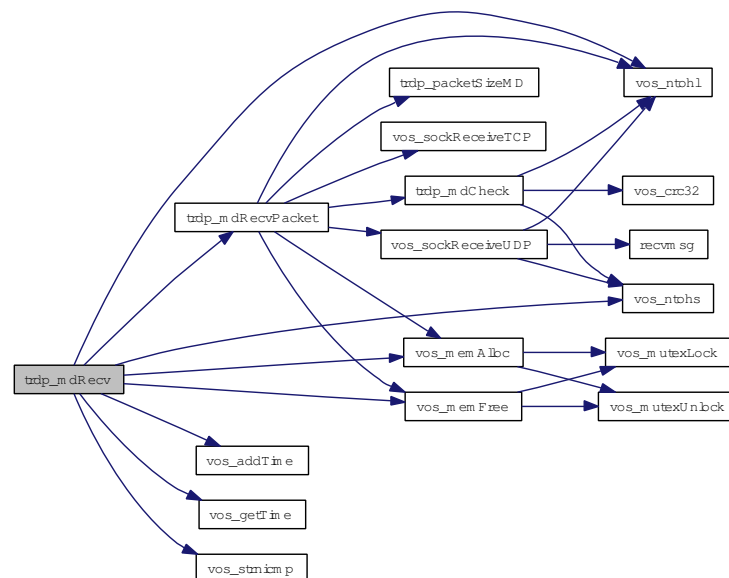
**TRDP\_WIRE\_ERR** protocol error (late packet, version mismatch)

**TRDP\_QUEUE\_ERR** not in queue

**TRDP\_CRC\_ERR** header checksum

**TRDP\_TOPOCOUNT\_ERR** invalid topocount

Here is the call graph for this function:



### 5.11.2.9 TRDP\_ERR\_T trdp\_mdRecvPacket (TRDP\_SESSION\_PT *appHandle*, INT32 *mdSock*, MD\_ELEMENT \* *pElement*)

Receive MD packet.

**Parameters:**

← *appHandle* session pointer

← *mdSock* socket descriptor

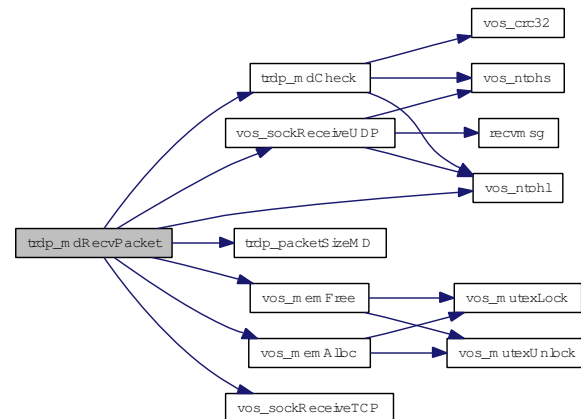
← *pElement* pointer to received packet

**Return values:**

!= TRDP\_NO\_ERR error



Here is the call graph for this function:



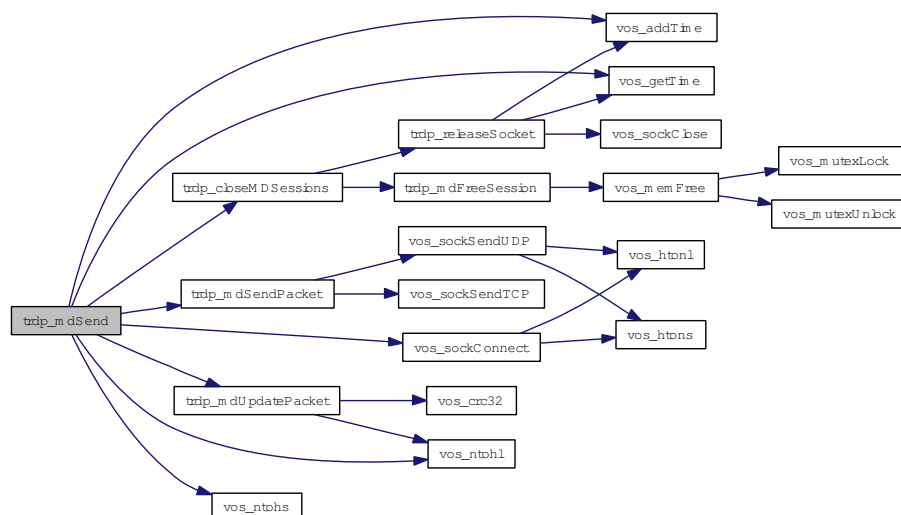
#### 5.11.2.10 TRDP\_ERR\_T trdp\_mdSend (TRDP\_SESSION\_PT *appHandle*)

Sending MD messages Send the messages stored in the sendQueue Call user's callback if needed.

##### Parameters:

← *appHandle* session pointer

Here is the call graph for this function:



#### 5.11.2.11 TRDP\_ERR\_T trdp\_mdSendPacket (INT32 *pdSock*, UINT32 *port*, MD\_ELEMENT \* *pElement*)

Send MD packet.

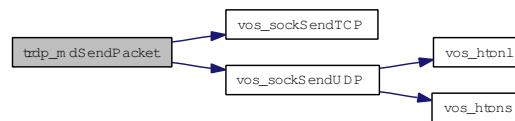
**Parameters:**

- ← *pdSock* socket descriptor
- ← *port* port on which to send
- ← *pElement* pointer to element to be sent

**Return values:**

- != NULL error

Here is the call graph for this function:



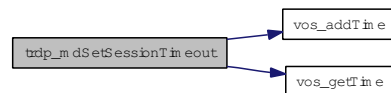
#### 5.11.2.12 void trdp\_mdSetSessionTimeout (MD\_ELE\_T \* *pMDSession*, UINT32 *usTimeOut*)

set time out

**Parameters:**

- ← *pMDSession* session pointer
- ← *usTimeOut* timeout in us

Here is the call graph for this function:



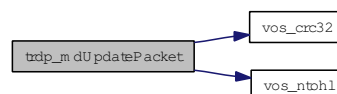
#### 5.11.2.13 void trdp\_mdUpdatePacket (MD\_ELE\_T \* *pElement*)

Update the header values.

**Parameters:**

- ← *pElement* pointer to the packet to update

Here is the call graph for this function:

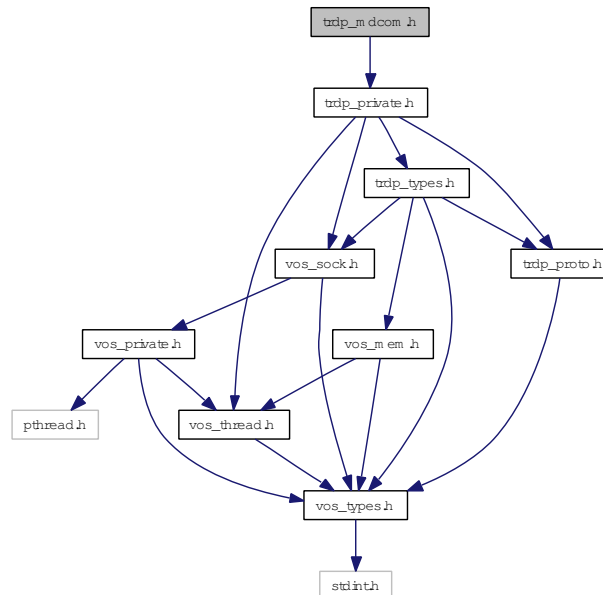


## 5.12 trdp\_mdcom.h File Reference

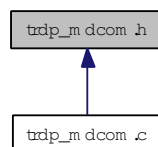
Functions for MD communication.

```
#include "trdp_private.h"
```

Include dependency graph for trdp\_mdcom.h:



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



### Functions

- **TRDP\_ERR\_T** `trdp_getTCPSocket` (**TRDP\_SESSION\_PT** pSession)  
*Initialize the specific parameters for message data Open a listening socket.*
- void `trdp_closeMDSessions` (**TRDP\_SESSION\_PT** appHandle, INT32 socketIndex, INT32 newSocket, BOOL checkAllSockets)  
*Close and free any session marked as dead.*
- void `trdp_mdFreeSession` (MD\_ELE\_T \*pMDSession)  
*Free memory of session.*
- void `trdp_mdSetSessionTimeout` (MD\_ELE\_T \*pMDSession, UINT32 usTimeOut)  
*set time out*

- [TRDP\\_ERR\\_T trdp\\_mdSendPacket](#) (INT32 pdSock, UINT32 port, MD\_ELE\_T \*pPacket)  
*Send MD packet.*
- void [trdp\\_mdUpdatePacket](#) (MD\_ELE\_T \*pPacket)  
*Update the header values.*
- [TRDP\\_ERR\\_T trdp\\_mdRecv](#) (TRDP\_SESSION\_PT appHandle, UINT32 sock)  
*Receiving MD messages Read the receive socket for arriving MDs, copy the packet to a new MD\_ELE\_T  
Check for protocol errors and dispatch to proper receive queue.*
- [TRDP\\_ERR\\_T trdp\\_mdSend](#) (TRDP\_SESSION\_PT appHandle)  
*Sending MD messages Send the messages stored in the sendQueue Call user's callback if needed.*
- void [trdp\\_mdCheckPending](#) (TRDP\_APP\_SESSION\_T appHandle, TRDP\_FDS\_T \*pFileDesc, INT32 \*pNoDesc)  
*Check for pending packets, set FD if non blocking.*
- void [trdp\\_mdCheckListenSocks](#) (TRDP\_SESSION\_PT appHandle, TRDP\_FDS\_T \*pRfds, INT32 \*pCount)  
*Checking receive connection requests and data Call user's callback if needed.*
- void [trdp\\_mdCheckTimeouts](#) (TRDP\_SESSION\_PT appHandle)  
*Checking message data timeouts Call user's callback if needed.*

### 5.12.1 Detailed Description

Functions for MD communication.

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Bernd Loehr, NewTec GmbH

#### Remarks:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2013.

#### Id

[trdp\\_mdcom.h](#) 950 2013-06-13 13:51:41Z 97025

### 5.12.2 Function Documentation

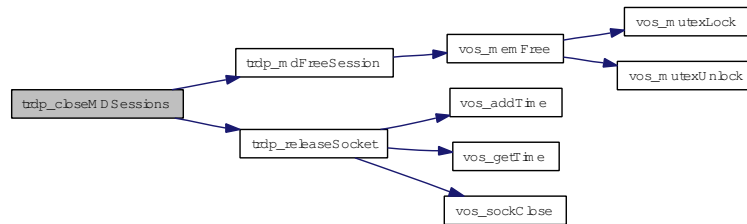
#### 5.12.2.1 void trdp\_closeMDSessions (TRDP\_SESSION\_PT appHandle, INT32 socketIndex, INT32 newSocket, BOOL checkAllSockets)

Close and free any session marked as dead.

**Parameters:**

- ← *appHandle* session pointer
- ← *socketIndex* the old socket position in the iface[]
- ← *newSocket* the new socket
- ← *checkAllSockets* check all the sockets that are waiting to be closed

Here is the call graph for this function:



### 5.12.2.2 TRDP\_ERR\_T trdp\_getTCPSocket (TRDP\_SESSION\_PT pSession)

Initialize the specific parameters for message data Open a listening socket.

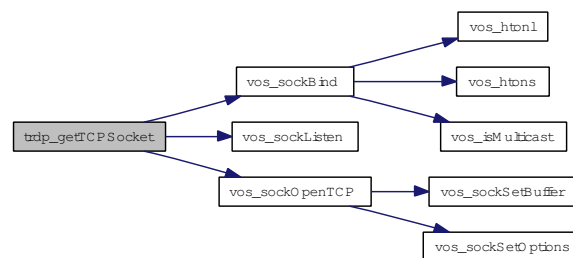
**Parameters:**

- ← *pSession* session parameters

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* initialization error

Here is the call graph for this function:



### 5.12.2.3 void trdp\_mdCheckListenSocks (TRDP\_SESSION\_PT appHandle, TRDP\_FDS\_T \* pRfds, INT32 \* pCount)

Checking receive connection requests and data Call user's callback if needed.

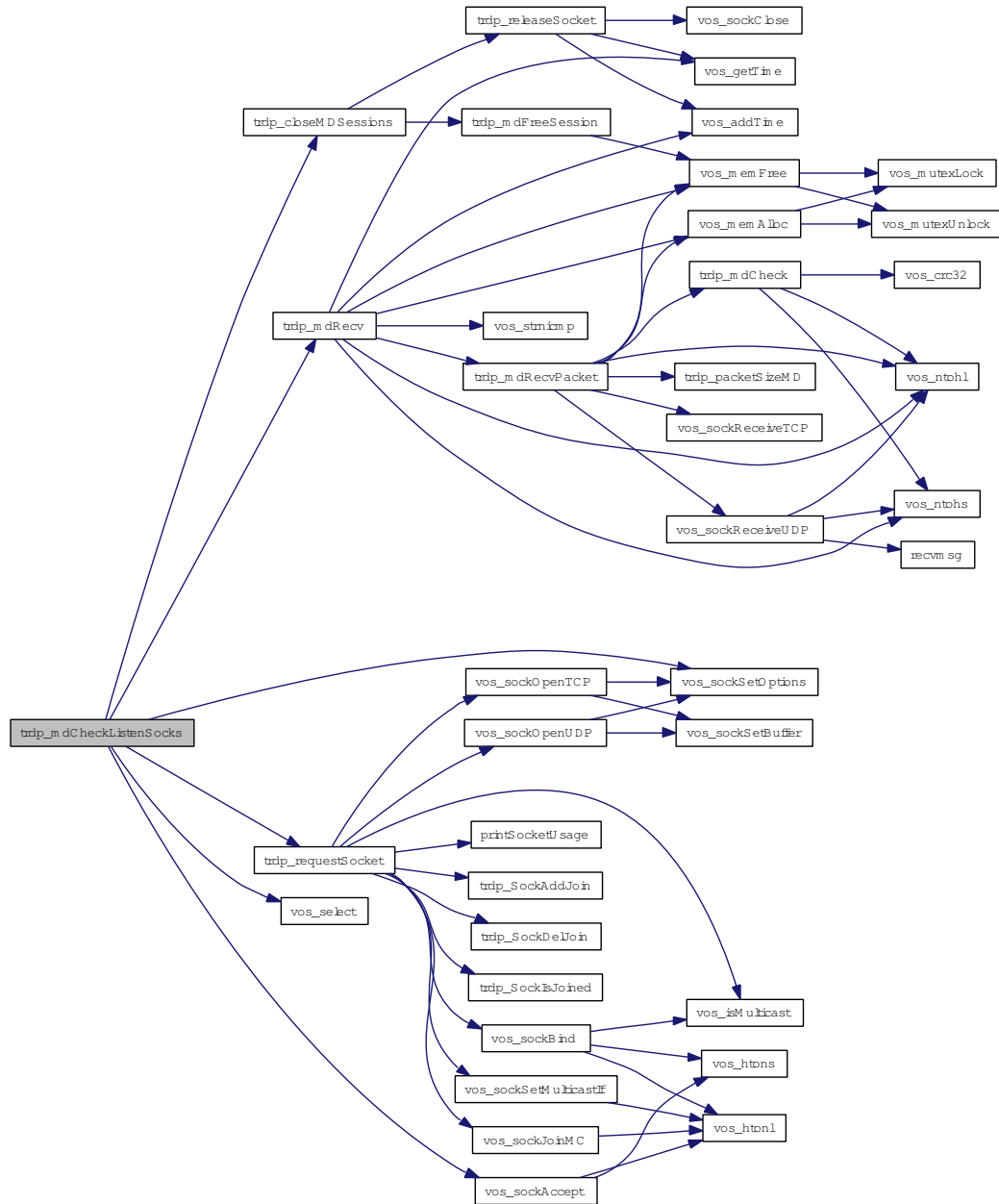
**Parameters:**

- ← *appHandle* session pointer

← *pRfds* pointer to set of ready descriptors

↔ *pCount* pointer to number of ready descriptors

Here is the call graph for this function:



#### 5.12.2.4 void trdp\_mdCheckPending (TRDP\_APP\_SESSION\_T appHandle, TRDP\_FDS\_T \* pFileDesc, INT32 \* pNoDesc)

Check for pending packets, set FD if non blocking.

**Parameters:**

- ← *appHandle* session pointer
- ↔ *pFileDesc* pointer to set of ready descriptors
- ↔ *pNoDesc* pointer to number of ready descriptors

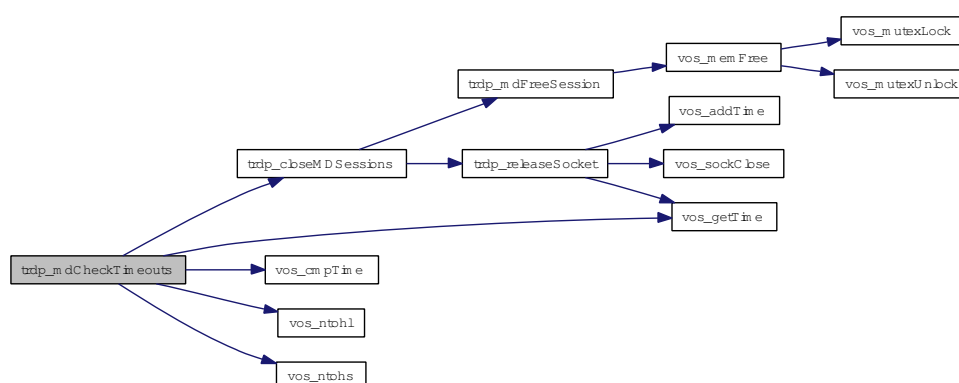
**5.12.2.5 void trdp\_mdCheckTimeouts (TRDP\_SESSION\_PT appHandle)**

Checking message data timeouts Call user's callback if needed.

**Parameters:**

- ← *appHandle* session pointer

Here is the call graph for this function:

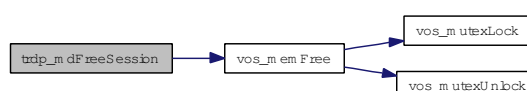
**5.12.2.6 void trdp\_mdFreeSession (MD\_ELEMENT \* pMDSession)**

Free memory of session.

**Parameters:**

- ← *pMDSession* session pointer

Here is the call graph for this function:

**5.12.2.7 TRDP\_ERR\_T trdp\_mdRecv (TRDP\_SESSION\_PT appHandle, UINT32 sockIndex)**

Receiving MD messages Read the receive socket for arriving MDs, copy the packet to a new MD\_ELEMENT Check for protocol errors and dispatch to proper receive queue.

Call user's callback if needed

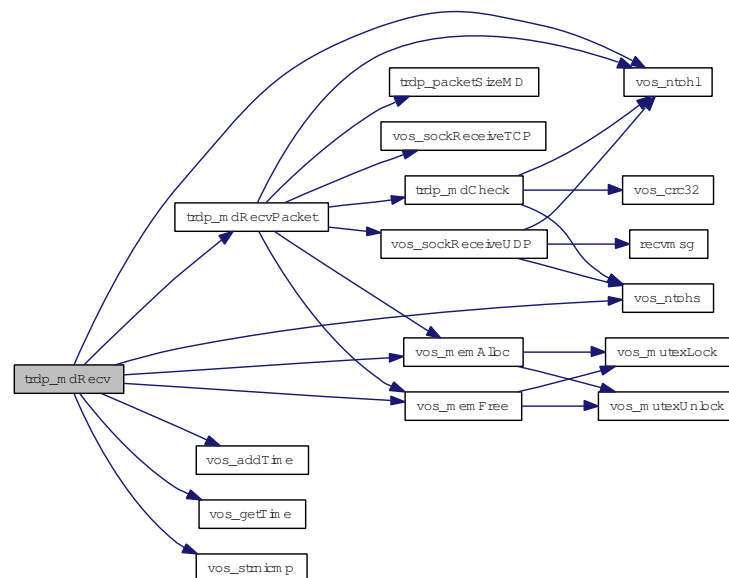
**Parameters:**

- ← *appHandle* session pointer
- ← *sockIndex* index of the socket to read from

**Return values:**

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_WIRE\_ERR** protocol error (late packet, version mismatch)
- TRDP\_QUEUE\_ERR** not in queue
- TRDP\_CRC\_ERR** header checksum
- TRDP\_TOPOCOUNT\_ERR** invalid topocount

Here is the call graph for this function:



### 5.12.2.8 TRDP\_ERR\_T trdp\_mdSend (TRDP\_SESSION\_PT appHandle)

Sending MD messages Send the messages stored in the sendQueue Call user's callback if needed.

**Parameters:**

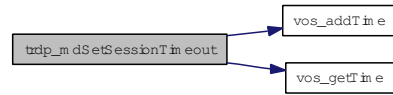
- ← *appHandle* session pointer





← *usTimeOut* timeout in us

Here is the call graph for this function:



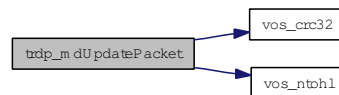
#### 5.12.2.11 void trdp\_mdUpdatePacket (MD\_ELE\_T \* *pElement*)

Update the header values.

##### Parameters:

← *pElement* pointer to the packet to update

Here is the call graph for this function:

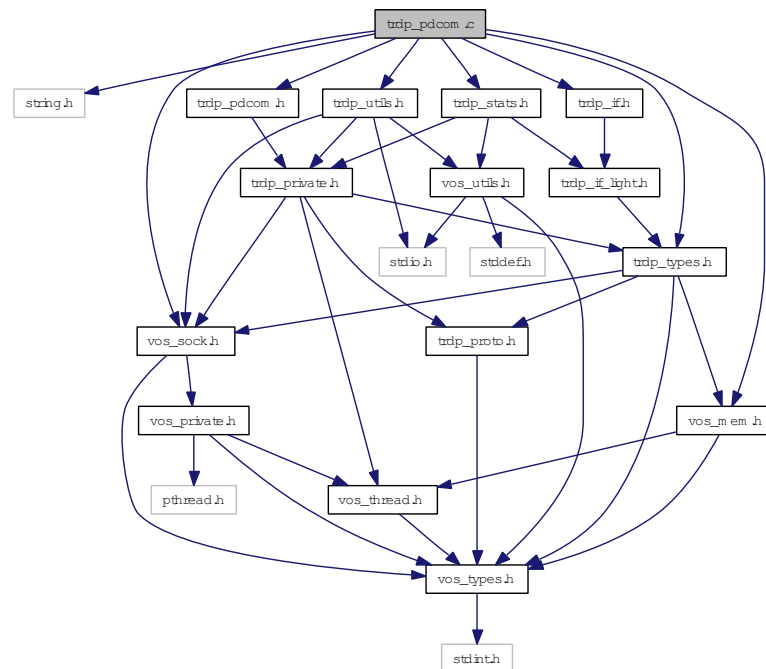


## 5.13 trdp\_pdcom.c File Reference

Functions for PD communication.

```
#include <string.h>
#include "trdp_types.h"
#include "trdp_utils.h"
#include "trdp_pdcom.h"
#include "trdp_if.h"
#include "trdp_stats.h"
#include "vos_sock.h"
#include "vos_mem.h"
```

Include dependency graph for trdp\_pdcom.c:



### Functions

- void [trdp\\_pdInit](#) ([PD\\_ELE\\_T](#) \*pPacket, [TRDP\\_MSG\\_T](#) type, UINT32 topoCount, UINT32 reply-ComId, UINT32 replyIpAddress)  
*Initialize/construct the packet Set the header infos.*
- [TRDP\\_ERR\\_T](#) [trdp\\_pdPut](#) ([PD\\_ELE\\_T](#) \*pPacket, [TRDP\\_MARSHALL\\_T](#) marshall, void \*refCon, const UINT8 \*pData, UINT32 dataSize)  
*Copy data Set the header infos.*
- void [trdp\\_pdDataUpdate](#) ([PD\\_ELE\\_T](#) \*pPacket)  
*Add padding and update data CRC.*

- `TRDP_ERR_T trdp_pdGet (PD_ELE_T *pPacket, TRDP_UNMARSHALL_T unmarshall, void *refCon, const UINT8 *pData, UINT32 *pDataSize)`  
*Copy data Set the header infos.*
- `TRDP_ERR_T trdp_pdSendQueued (TRDP_SESSION_PT appHandle)`  
*Send all due PD messages.*
- `TRDP_ERR_T trdp_pdReceive (TRDP_SESSION_PT appHandle, INT32 sock)`  
*Receiving PD messages Read the receive socket for arriving PDs, copy the packet to a new PD\_ELE\_T  
Check for protocol errors and compare the received data to the data in our receive queue.*
- `void trdp_pdCheckPending (TRDP_APP_SESSION_T appHandle, TRDP_FDS_T *pFileDesc, INT32 *pNoDesc)`  
*Check for pending packets, set FD if non blocking.*
- `void trdp_pdHandleTimeOuts (TRDP_SESSION_PT appHandle)`  
*Check for time outs.*
- `TRDP_ERR_T trdp_pdCheckListenSocks (TRDP_SESSION_PT appHandle, TRDP_FDS_T *pRfds, INT32 *pCount)`  
*Checking receive connection requests and data Call user's callback if needed.*
- `void trdp_pdUpdate (PD_ELE_T *pPacket)`  
*Update the header values.*
- `TRDP_ERR_T trdp_pdCheck (PD_HEADER_T *pPacket, UINT32 packetSize)`  
*Check if the PD header values and the CRCs are sane.*
- `TRDP_ERR_T trdp_pdSend (INT32 pdSock, PD_ELE_T *pPacket, UINT16 port)`  
*Send one PD packet.*
- `TRDP_ERR_T trdp_pdDistribute (PD_ELE_T *pSndQueue)`  
*Distribute send time of PD packets over time.*

### 5.13.1 Detailed Description

Functions for PD communication.

**Note:**

Project: TCNOpen TRDP prototype stack

**Author:**

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

[trdp\\_pdcom.c](#) 1010 2013-07-03 12:12:16Z bloehr

BL 2013-06-24: ID 125: Time-out handling and ready descriptors fixed BL 2013-04-09: ID 92: Pull request led to reset of push message type BL 2013-01-25: ID 20: Redundancy handling fixed

**5.13.2 Function Documentation****5.13.2.1 TRDP\_ERR\_T trdp\_pdCheck (PD\_HEADER\_T \* *pPacket*, UINT32 *packetSize*)**

Check if the PD header values and the CRCs are sane.

**Parameters:**

← *pPacket* pointer to the packet to check

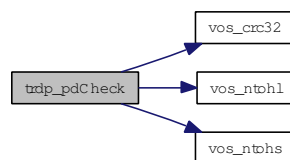
← *packetSize* max size to check

**Return values:**

*TRDP\_NO\_ERR*

*TRDP\_CRC\_ERR*

Here is the call graph for this function:

**5.13.2.2 TRDP\_ERR\_T trdp\_pdCheckListenSocks (TRDP\_SESSION\_PT *appHandle*, TRDP\_FDS\_T \* *pRfds*, INT32 \* *pCount*)**

Checking receive connection requests and data Call user's callback if needed.

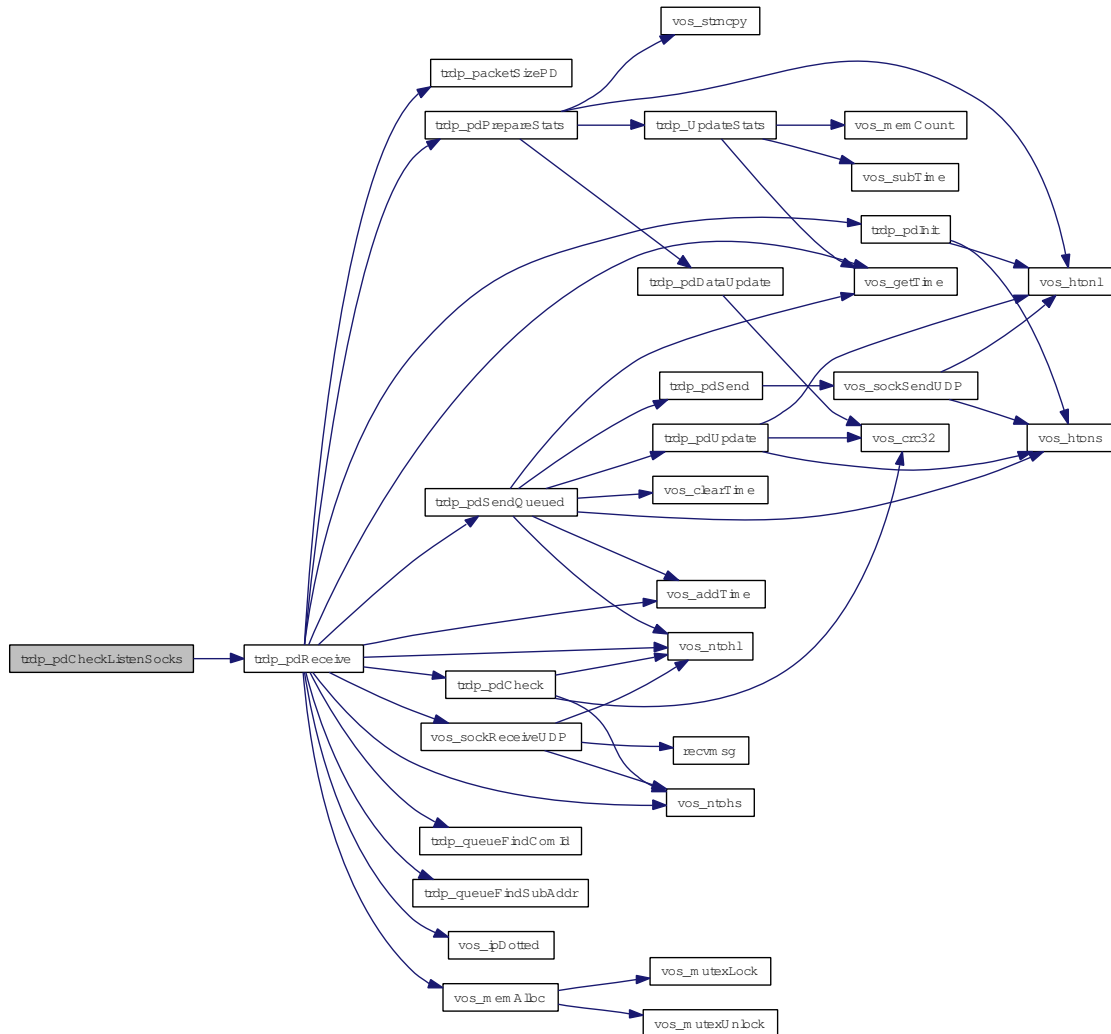
**Parameters:**

← *appHandle* session pointer

← *pRfds* pointer to set of ready descriptors

↔ *pCount* pointer to number of ready descriptors

Here is the call graph for this function:



### 5.13.2.3 void trdp\_pdCheckPending (TRDP\_APP\_SESSION\_T *appHandle*, TRDP\_FDS\_T \* *pFileDesc*, INT32 \* *pNoDesc*)

Check for pending packets, set FD if non blocking.

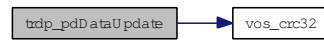
#### Parameters:

- ← *appHandle* session pointer
- ↔ *pFileDesc* pointer to set of ready descriptors
- ↔ *pNoDesc* pointer to number of ready descriptors

### 5.13.2.4 void trdp\_pdDataUpdate (PD\_ELE\_T \* *pPacket*)

Add padding and update data CRC.

Here is the call graph for this function:



#### 5.13.2.5 TRDP\_ERR\_T trdp\_pdDistribute (PD\_ELE\_T \* *pSndQueue*)

Distribute send time of PD packets over time.

The duration of PD packets on a 100MBit/s network ranges from 3us to 150us max. Because a cyclic thread scheduling below 5ms would put a too heavy load on the system, and PD packets cannot get larger than 1436 (+ UDP header), we will not account for differences in packet size. Another factor is the differences in intervals for different packets: We should only change the starting times of the packets within 1/2 the interval time. Otherwise a late addition of packets could lead to timeouts of already queued packets. Scheduling will be computed based on the smallest interval time.

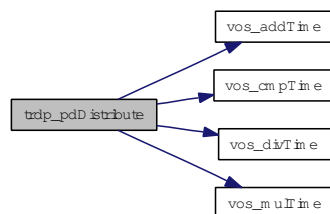
##### Parameters:

← *pSndQueue* pointer to send queue

##### Return values:

**TRDP\_NO\_ERR**

Here is the call graph for this function:



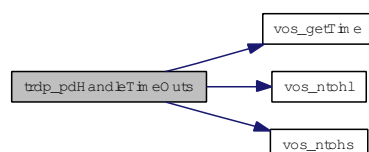
#### 5.13.2.6 void trdp\_pdHandleTimeOuts (TRDP\_SESSION\_PT *appHandle*)

Check for time outs.

##### Parameters:

← *appHandle* application handle

Here is the call graph for this function:



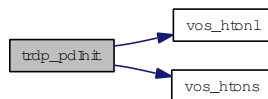
### 5.13.2.7 void trdp\_pdInit (PD\_ELE\_T \* *pPacket*, TRDP\_MSG\_T *type*, UINT32 *topoCount*, UINT32 *replyComId*, UINT32 *replyIpAddress*)

Initialize/construct the packet Set the header infos.

#### Parameters:

- ← *pPacket* pointer to the packet element to init
- ← *type* type the packet
- ← *topoCount* topocount to use for PD frame
- ← *replyComId* Pull request comId
- ← *replyIpAddress* Pull request Ip

Here is the call graph for this function:



### 5.13.2.8 TRDP\_ERR\_T trdp\_pdReceive (TRDP\_SESSION\_PT *appHandle*, INT32 *sock*)

Receiving PD messages Read the receive socket for arriving PDs, copy the packet to a new PD\_ELE\_T Check for protocol errors and compare the received data to the data in our receive queue.

If it is a new packet, check if it is a PD Request (PULL). If it is an update, exchange the existing entry with the new one Call user's callback if needed

#### Parameters:

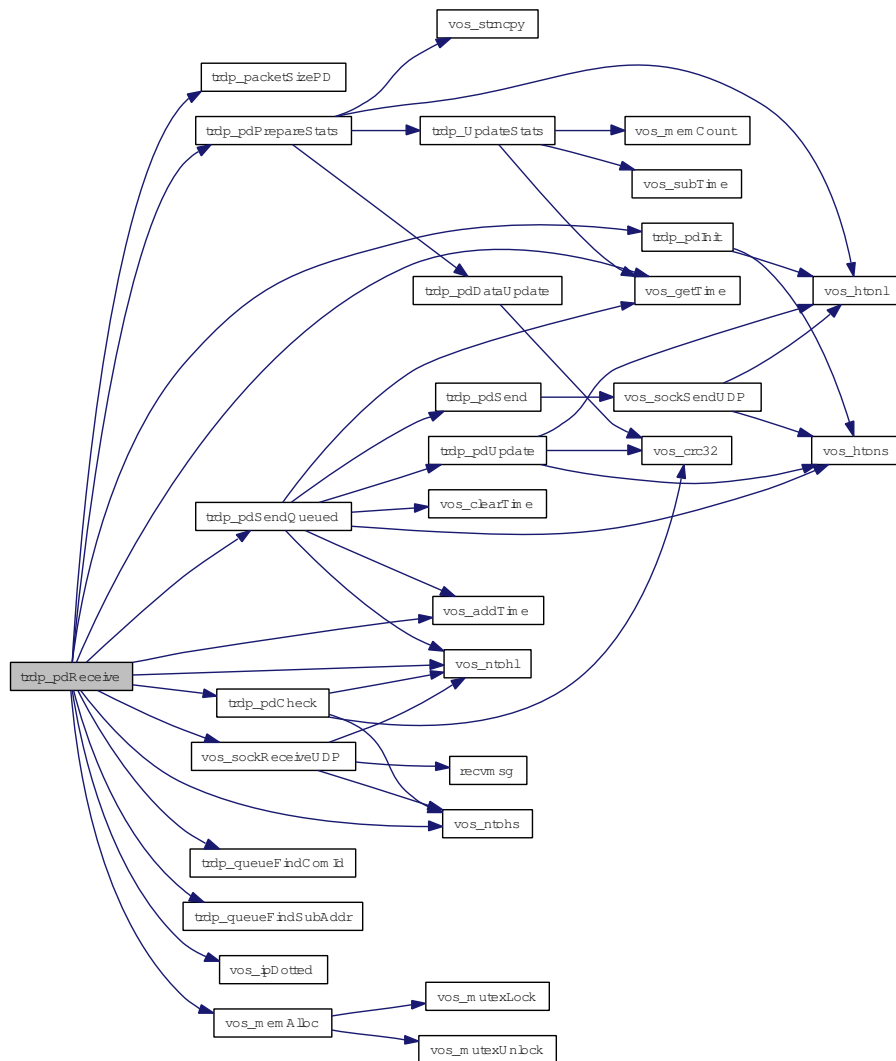
- ← *appHandle* session pointer
- ← *sock* the socket to read from

#### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_WIRE\_ERR** protocol error (late packet, version mismatch)
- TRDP\_QUEUE\_ERR** not in queue
- TRDP\_CRC\_ERR** header checksum
- TRDP\_TOPOCOUNT\_ERR** invalid topocount



Here is the call graph for this function:



### 5.13.2.9 TRDP\_ERR\_T trdp\_pdSend (INT32 *pdSock*, PD\_ELEM\_T \* *pPacket*, UINT16 *port*)

Send one PD packet.

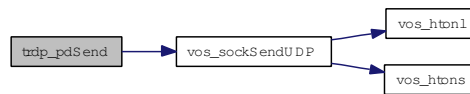
#### Parameters:

- ← *pdSock* socket descriptor
- ← *pPacket* pointer to packet to be sent
- ← *port* port on which to send

#### Return values:

- TRDP\_NO\_ERR**
- TRDP\_IO\_ERR**

Here is the call graph for this function:



#### 5.13.2.10 TRDP\_ERR\_T trdp\_pdSendQueued (TRDP\_SESSION\_PT *appHandle*)

Send all due PD messages.

##### Parameters:

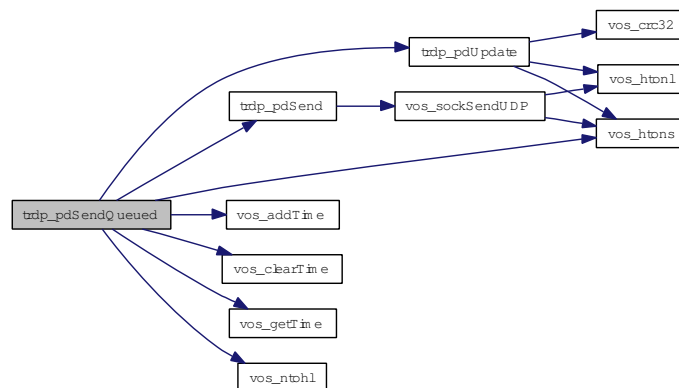
← *appHandle* session pointer

##### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_IO\_ERR** socket I/O error

Here is the call graph for this function:



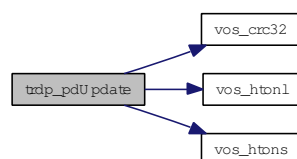
#### 5.13.2.11 void trdp\_pdUpdate (PD\_ELE\_T \**pPacket*)

Update the header values.

##### Parameters:

← *pPacket* pointer to the packet to update

Here is the call graph for this function:

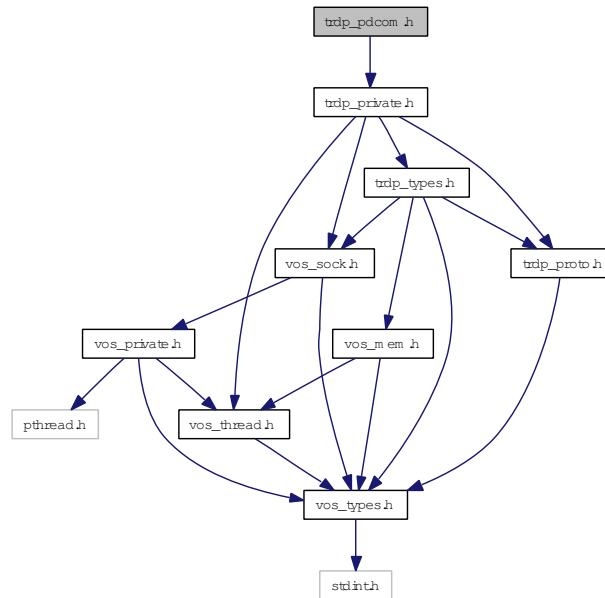


## 5.14 trdp\_pdcom.h File Reference

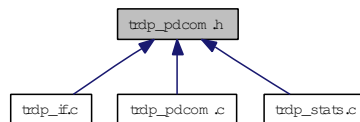
Functions for PD communication.

```
#include "trdp_private.h"
```

Include dependency graph for trdp\_pdcom.h:



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



### Functions

- void [trdp\\_pdInit](#) ([PD\\_ELE\\_T](#) \*, [TRDP\\_MSG\\_T](#), UINT32 topCount, UINT32 replyComId, UINT32 replyIpAddress)  
*Initialize/construct the packet Set the header infos.*
- void [trdp\\_pdUpdate](#) ([PD\\_ELE\\_T](#) \*)  
*Update the header values.*
- [TRDP\\_ERR\\_T](#) [trdp\\_pdPut](#) ([PD\\_ELE\\_T](#) \*, [TRDP\\_MARSHALL\\_T](#) func, void \*refCon, const UINT8 \*pData, UINT32 dataSize)  
*Copy data Set the header infos.*
- void [trdp\\_pdDataUpdate](#) ([PD\\_ELE\\_T](#) \*pPacket)  
*Add padding and update data CRC.*

- [TRDP\\_ERR\\_T trdp\\_pdCheck](#) ([PD\\_HEADER\\_T](#) \*pPacket, [UINT32](#) packetSize)  
*Check if the PD header values and the CRCs are sane.*
- [TRDP\\_ERR\\_T trdp\\_pdSend](#) ([INT32](#) pdSock, [PD\\_ELE\\_T](#) \*pPacket, [UINT16](#) port)  
*Send one PD packet.*
- [TRDP\\_ERR\\_T trdp\\_pdGet](#) ([PD\\_ELE\\_T](#) \*pPacket, [TRDP\\_UNMARSHALL\\_T](#) unmarshall, void \*refCon, const [UINT8](#) \*pData, [UINT32](#) \*pDataSize)  
*Copy data Set the header infos.*
- [TRDP\\_ERR\\_T trdp\\_pdSendQueued](#) ([TRDP\\_SESSION\\_PT](#) appHandle)  
*Send all due PD messages.*
- [TRDP\\_ERR\\_T trdp\\_pdReceive](#) ([TRDP\\_SESSION\\_PT](#) pSessionHandle, [INT32](#) sock)  
*Receiving PD messages Read the receive socket for arriving PDs, copy the packet to a new PD\_ELE\_T  
Check for protocol errors and compare the received data to the data in our receive queue.*
- void [trdp\\_pdCheckPending](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_FDS\\_T](#) \*pFileDesc, [INT32](#) \*pNoDesc)  
*Check for pending packets, set FD if non blocking.*
- void [trdp\\_pdHandleTimeOuts](#) ([TRDP\\_SESSION\\_PT](#) appHandle)  
*Check for time outs.*
- [TRDP\\_ERR\\_T trdp\\_pdCheckListenSocks](#) ([TRDP\\_SESSION\\_PT](#) appHandle, [TRDP\\_FDS\\_T](#) \*pRfds, [INT32](#) \*pCount)  
*Checking receive connection requests and data Call user's callback if needed.*
- [TRDP\\_ERR\\_T trdp\\_pdDistribute](#) ([PD\\_ELE\\_T](#) \*pSndQueue)  
*Distribute send time of PD packets over time.*

### 5.14.1 Detailed Description

Functions for PD communication.

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Bernd Loehr, NewTec GmbH

#### Remarks:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

#### Id

[trdp\\_pdcom.h](#) 950 2013-06-13 13:51:41Z 97025

## 5.14.2 Function Documentation

### 5.14.2.1 TRDP\_ERR\_T trdp\_pdCheck (PD\_HEADER\_T \* *pPacket*, UINT32 *packetSize*)

Check if the PD header values and the CRCs are sane.

#### Parameters:

← *pPacket* pointer to the packet to check

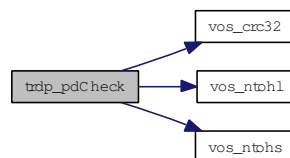
← *packetSize* max size to check

#### Return values:

**TRDP\_NO\_ERR**

**TRDP\_CRC\_ERR**

Here is the call graph for this function:



### 5.14.2.2 TRDP\_ERR\_T trdp\_pdCheckListenSocks (TRDP\_SESSION\_PT *appHandle*, TRDP\_FDS\_T \* *pRfds*, INT32 \* *pCount*)

Checking receive connection requests and data Call user's callback if needed.

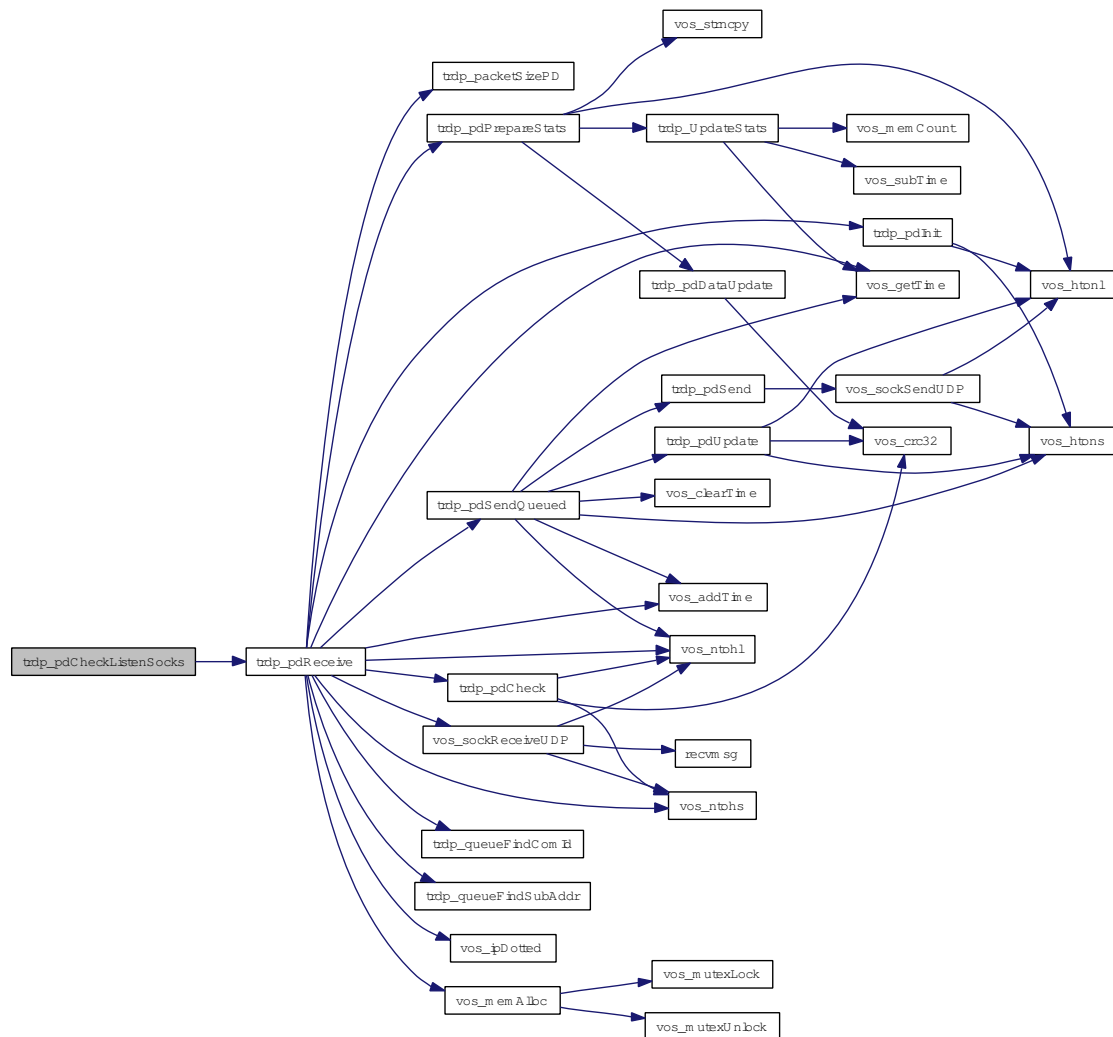
#### Parameters:

← *appHandle* session pointer

← *pRfds* pointer to set of ready descriptors

↔ *pCount* pointer to number of ready descriptors

Here is the call graph for this function:



#### 5.14.2.3 void trdp\_pdCheckPending (TRDP\_APP\_SESSION\_T *appHandle*, TRDP\_FDS\_T \* *pFileDesc*, INT32 \* *pNoDesc*)

Check for pending packets, set FD if non blocking.

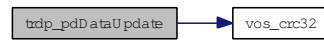
##### Parameters:

- ← *appHandle* session pointer
- ↔ *pFileDesc* pointer to set of ready descriptors
- ↔ *pNoDesc* pointer to number of ready descriptors

#### 5.14.2.4 void trdp\_pdDataUpdate (PD\_ELE\_T \* *pPacket*)

Add padding and update data CRC.

Here is the call graph for this function:



#### 5.14.2.5 TRDP\_ERR\_T trdp\_pdDistribute (PD\_ELE\_T \* *pSndQueue*)

Distribute send time of PD packets over time.

The duration of PD packets on a 100MBit/s network ranges from 3us to 150us max. Because a cyclic thread scheduling below 5ms would put a too heavy load on the system, and PD packets cannot get larger than 1436 (+ UDP header), we will not account for differences in packet size. Another factor is the differences in intervals for different packets: We should only change the starting times of the packets within 1/2 the interval time. Otherwise a late addition of packets could lead to timeouts of already queued packets. Scheduling will be computed based on the smallest interval time.

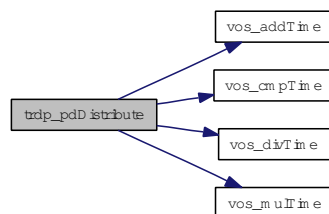
##### Parameters:

← *pSndQueue* pointer to send queue

##### Return values:

**TRDP\_NO\_ERR**

Here is the call graph for this function:



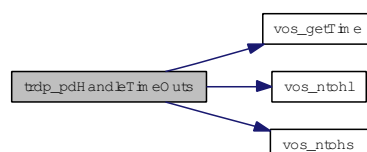
#### 5.14.2.6 void trdp\_pdHandleTimeOuts (TRDP\_SESSION\_PT *appHandle*)

Check for time outs.

##### Parameters:

← *appHandle* application handle

Here is the call graph for this function:



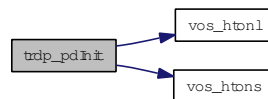
#### 5.14.2.7 void trdp\_pdInit (PD\_ELE\_T \* *pPacket*, TRDP\_MSG\_T *type*, UINT32 *topoCount*, UINT32 *replyComId*, UINT32 *replyIpAddress*)

Initialize/construct the packet Set the header infos.

##### Parameters:

- ← *pPacket* pointer to the packet element to init
- ← *type* type the packet
- ← *topoCount* topocount to use for PD frame
- ← *replyComId* Pull request comId
- ← *replyIpAddress* Pull request Ip

Here is the call graph for this function:



#### 5.14.2.8 TRDP\_ERR\_T trdp\_pdReceive (TRDP\_SESSION\_PT *appHandle*, INT32 *sock*)

Receiving PD messages Read the receive socket for arriving PDs, copy the packet to a new PD\_ELE\_T Check for protocol errors and compare the received data to the data in our receive queue.

If it is a new packet, check if it is a PD Request (PULL). If it is an update, exchange the existing entry with the new one Call user's callback if needed

##### Parameters:

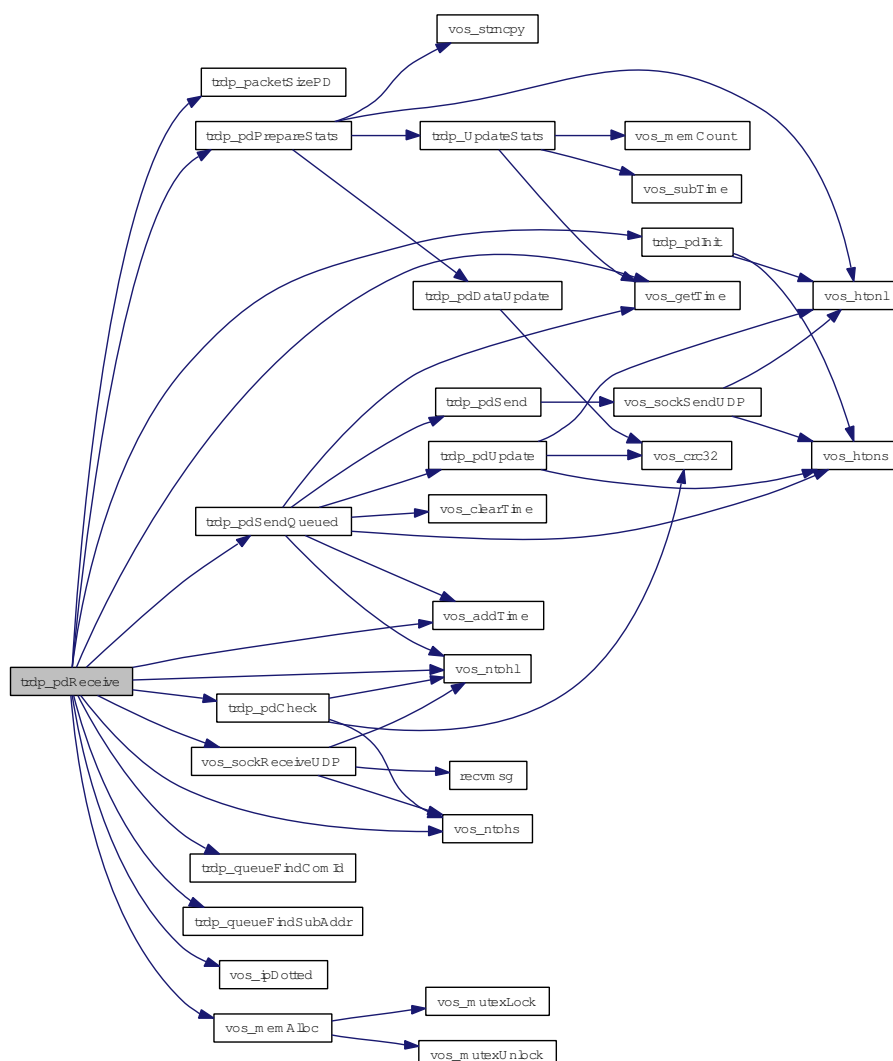
- ← *appHandle* session pointer
- ← *sock* the socket to read from

##### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_WIRE\_ERR** protocol error (late packet, version mismatch)
- TRDP\_QUEUE\_ERR** not in queue
- TRDP\_CRC\_ERR** header checksum
- TRDP\_TOPOCOUNT\_ERR** invalid topocount



Here is the call graph for this function:



#### 5.14.2.9 TRDP\_ERR\_T trdp\_pdSend (INT32 *pdSock*, PD\_ELEM\_T \* *pPacket*, UINT16 *port*)

Send one PD packet.

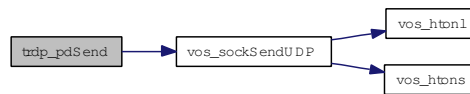
##### Parameters:

- ← *pdSock* socket descriptor
- ← *pPacket* pointer to packet to be sent
- ← *port* port on which to send

##### Return values:

- TRDP\_NO\_ERR**
- TRDP\_IO\_ERR**

Here is the call graph for this function:



#### 5.14.2.10 TRDP\_ERR\_T trdp\_pdSendQueued (TRDP\_SESSION\_PT *appHandle*)

Send all due PD messages.

##### Parameters:

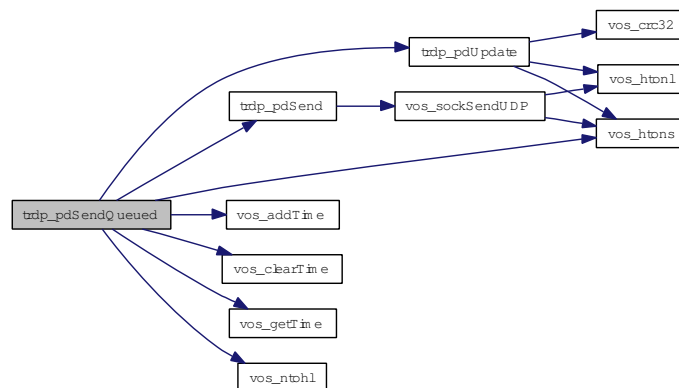
← *appHandle* session pointer

##### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_IO\_ERR** socket I/O error

Here is the call graph for this function:



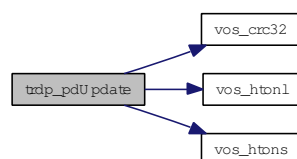
#### 5.14.2.11 void trdp\_pdUpdate (PD\_ELE\_T \**pPacket*)

Update the header values.

##### Parameters:

← *pPacket* pointer to the packet to update

Here is the call graph for this function:

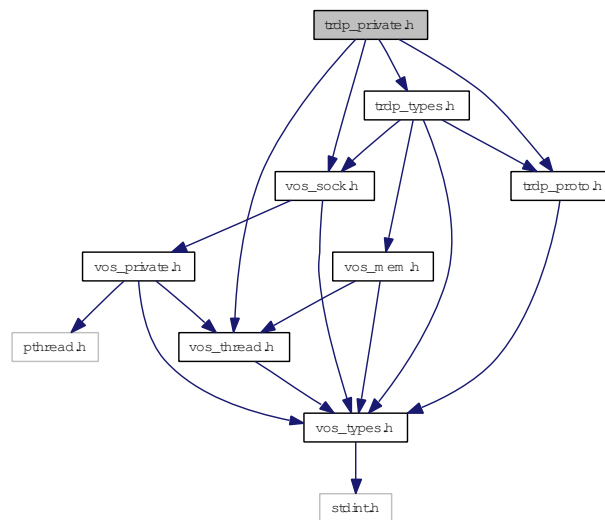


## 5.15 trdp\_private.h File Reference

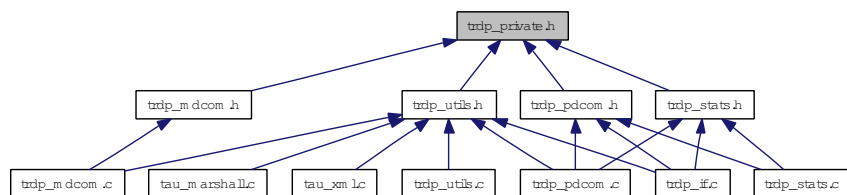
Typedefs for TRDP communication.

```
#include "trdp_types.h"
#include "trdp_proto.h"
#include "vos_thread.h"
#include "vos_sock.h"
```

Include dependency graph for trdp\_private.h:



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



### Data Structures

- struct [TRDP\\_HANDLE](#)  
*Hidden handle definition, used as unique addressing item.*
- struct [TRDP\\_SOCKET\\_TCP](#)  
*TCP parameters.*
- struct [TRDP\\_SOCKETS](#)  
*Socket item.*

- struct [GNU\\_PACKED](#)  
*TRDP process data header - network order and alignment.*
- struct [PD\\_ELE](#)  
*Queue element for PD packets to send or receive.*
- struct [TRDP\\_SESSION](#)  
*Session/application variables store.*

## Defines

- #define [TRDP\\_TIMER\\_GRANULARITY](#) 10000  
*granularity in us*
- #define [TRDP\\_TIMER\\_FOREVER](#) 0xffffffff  
*granularity in us*
- #define [TRDP\\_MD\\_DEFAULT\\_REPLY\\_TIMEOUT](#) 5000000  
*default reply time out 5s*
- #define [TRDP\\_MD\\_DEFAULT\\_CONFIRM\\_TIMEOUT](#) 1000000  
*default confirm time out 1s*
- #define [TRDP\\_MD\\_DEFAULT\\_CONNECTION\\_TIMEOUT](#) 60000000  
*Socket connection time out 1 minute.*
- #define [TRDP\\_MD\\_DEFAULT\\_SENDING\\_TIMEOUT](#) 5000000  
*Socket sending time out 5s.*
- #define [TRDP\\_PROCESS\\_DEFAULT\\_CYCLE\\_TIME](#) 10000  
*Default cycle time for TRDP process.*
- #define [TRDP\\_PROCESS\\_DEFAULT\\_PRIORITY](#) 64  
*Default priority of TRDP process.*
- #define [TRDP\\_PROCESS\\_DEFAULT\\_OPTIONS](#) TRDP\_OPTION\_TRAFFIC\_SHAPING  
*Default options for TRDP process.*
- #define [TRDP\\_DEBUG\\_DEFAULT\\_FILE\\_SIZE](#) 65536  
*Default maximum size of log file.*

## Typedefs

- typedef struct [TRDP\\_HANDLE](#) [TRDP\\_ADDRESSES\\_T](#)  
*Hidden handle definition, used as unique addressing item.*
- typedef struct [TRDP\\_SOCKET\\_TCP](#) [TRDP\\_SOCKET\\_TCP\\_T](#)

*TCP parameters.*

- typedef struct [TRDP\\_SOCKETS](#) [TRDP\\_SOCKETS\\_T](#)  
*Socket item.*
- typedef struct [PD\\_ELE](#) [PD\\_ELE\\_T](#)  
*Queue element for PD packets to send or receive.*
- typedef struct [TRDP\\_SESSION](#) [TRDP\\_SESSION\\_T](#)  
*Session/application variables store.*

## Enumerations

- enum [TRDP\\_MD\\_ELE\\_ST\\_T](#) {  
[TRDP\\_ST\\_NONE](#) = 0,  
[TRDP\\_ST\\_TX\\_NOTIFY\\_ARM](#) = 1,  
[TRDP\\_ST\\_TX\\_REQUEST\\_ARM](#) = 2,  
[TRDP\\_ST\\_TX\\_REPLY\\_ARM](#) = 3,  
[TRDP\\_ST\\_TX\\_REPLYQUERY\\_ARM](#) = 4,  
[TRDP\\_ST\\_TX\\_CONFIRM\\_ARM](#) = 5,  
[TRDP\\_ST\\_RX\\_READY](#) = 6,  
[TRDP\\_ST\\_TX\\_REQUEST\\_W4REPLY](#) = 7,  
[TRDP\\_ST\\_RX\\_REPLYQUERY\\_W4C](#) = 8,  
[TRDP\\_ST\\_RX\\_REQ\\_W4AP\\_REPLY](#) = 9,  
[TRDP\\_ST\\_TX\\_REQ\\_W4AP\\_CONFIRM](#) = 10,  
[TRDP\\_ST\\_RX\\_REPLY\\_SENT](#) = 11,  
[TRDP\\_ST\\_RX\\_NOTIFY\\_RECEIVED](#) = 12,  
[TRDP\\_ST\\_TX\\_REPLY\\_RECEIVED](#) = 13,  
[TRDP\\_ST\\_RX\\_CONF\\_RECEIVED](#) = 14 }  
*Internal MD state.*

- enum [TRDP\\_PRIV\\_FLAGS\\_T](#) { ,  
[TRDP\\_TIMED\\_OUT](#) = 0x2,  
[TRDP\\_INVALID\\_DATA](#) = 0x4,  
[TRDP\\_REQ\\_2B\\_SENT](#) = 0x8,  
[TRDP\\_PULL\\_SUB](#) = 0x10,  
[TRDP\\_REDUNDANT](#) = 0x20 }  
*Internal flags for packets.*

- enum [TRDP SOCK\\_TYPE\\_T](#) {  
[TRDP SOCK\\_PD](#) = 0,  
[TRDP SOCK\\_MD\\_UDP](#) = 1,  
[TRDP SOCK\\_MD\\_TCP](#) = 2 }  
*Socket usage.*

### 5.15.1 Detailed Description

Typedefs for TRDP communication.

TRDP internal type definitions

**Note:**

Project: TCNOpen TRDP prototype stack

**Author:**

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

[trdp\\_private.h](#) 995 2013-06-27 08:01:06Z bloehr

### 5.15.2 Enumeration Type Documentation

#### 5.15.2.1 enum TRDP\_MD\_ELE\_ST\_T

Internal MD state.

**Enumerator:**

*TRDP\_ST\_NONE* neutral value

*TRDP\_ST\_TX\_NOTIFY\_ARM* ready to send notify MD

*TRDP\_ST\_TX\_REQUEST\_ARM* ready to send request MD

*TRDP\_ST\_TX\_REPLY\_ARM* ready to send reply MD

*TRDP\_ST\_TX\_REPLYQUERY\_ARM* ready to send reply with confirm request MD

*TRDP\_ST\_TX\_CONFIRM\_ARM* ready to send confirm MD

*TRDP\_ST\_RX\_READY* armed listener

*TRDP\_ST\_TX\_REQUEST\_W4REPLY* request sent, wait for reply

*TRDP\_ST\_RX\_REPLYQUERY\_W4C* reply send, with confirm request MD

*TRDP\_ST\_RX\_REQ\_W4AP\_REPLY* request received, wait for application reply send

*TRDP\_ST\_TX\_REQ\_W4AP\_CONFIRM* reply conf.  
rq. tx, wait for application conf send

*TRDP\_ST\_RX\_REPLY\_SENT* reply sent

*TRDP\_ST\_RX\_NOTIFY\_RECEIVED* notification received, wait for application to accept

*TRDP\_ST\_TX\_REPLY\_RECEIVED* reply received

*TRDP\_ST\_RX\_CONF\_RECEIVED* confirmation received

### 5.15.2.2 enum TRDP\_PRIV\_FLAGS\_T

Internal flags for packets.

**Enumerator:**

- TRDP\_TIMED\_OUT* if set, inform the user
- TRDP\_INVALID\_DATA* if set, inform the user
- TRDP\_REQ\_2B\_SENT* if set, the request needs to be sent
- TRDP\_PULL\_SUB* if set, its a PULL subscription
- TRDP\_REDUNDANT* if set, packet should not be sent (redundant)

### 5.15.2.3 enum TRDP SOCK\_TYPE\_T

Socket usage.

**Enumerator:**

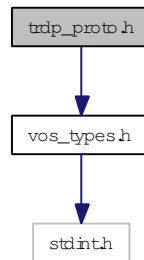
- TRDP SOCK\_PD* Socket is used for UDP process data.
- TRDP SOCK\_MD\_UDP* Socket is used for UDP message data.
- TRDP SOCK\_MD\_TCP* Socket is used for TCP message data.

## 5.16 trdp\_proto.h File Reference

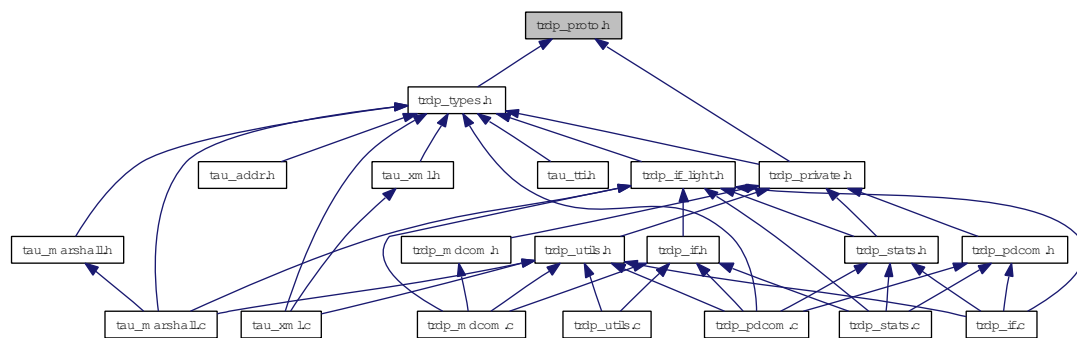
Definitions for the TRDP protocol.

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

Include dependency graph for trdp\_proto.h:



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



## Data Structures

- struct [GNU\\_PACKED](#)  
*TRDP process data header - network order and alignment.*
- struct [GNU\\_PACKED](#)  
*TRDP process data header - network order and alignment.*

## Defines

- #define [TRDP\\_PD\\_UDP\\_PORT](#) 20548  
*process data UDP port*
- #define [TRDP\\_MD\\_UDP\\_PORT](#) 20550  
*message data UDP port*
- #define [TRDP\\_MD\\_TCP\\_PORT](#) 20550



*message data TCP port*

- #define [TRDP\\_PROTO\\_VER](#) 0x0100  
*Protocol version.*
- #define [TRDP\\_PROTOCOL\\_VERSION\\_CHECK\\_MASK](#) 0xFF00  
*Version check, two digits are relevant.*
- #define [TRDP\\_SESS\\_ID\\_SIZE](#) 16  
*Session ID (UUID) size in MD header.*
- #define [TRDP\\_DEST\\_URI\\_SIZE](#) 32  
*max.*
- #define [TRDP\\_MIN\\_PD\\_HEADER\\_SIZE](#) sizeof(PD\_HEADER\_T)  
*PD header size with FCS.*
- #define [TRDP\\_MAX\\_PD\\_DATA\\_SIZE](#) 1432  
*PD data size without FCS.*
- #define [TRDP\\_MAX\\_LABEL\\_LEN](#) 16  
*Maximum values.*
- #define [TRDP\\_MAX\\_URI\\_USER\\_LEN](#) (2 \* TRDP\_MAX\_LABEL\_LEN)  
*URI user part incl.*
- #define [TRDP\\_MAX\\_URI\\_HOST\\_LEN](#) (4 \* TRDP\_MAX\_LABEL\_LEN)  
*URI host part length incl.*
- #define [TRDP\\_MAX\\_URI\\_LEN](#) ((6 \* TRDP\_MAX\_LABEL\_LEN) + 8)  
*URI length incl.*
- #define [TRDP\\_MAX\\_FILE\\_NAME\\_LEN](#) 128  
*path and file name length incl.*
- #define [TDRP\\_VAR\\_SIZE](#) 0  
*Variable size dataset.*
- #define [TRDP\\_COMID\\_ECHO](#) 10  
*TRDP reserved COMIDs in the range 1 .*
- #define [TRDP\\_STATISTICS\\_REQUEST\\_DSID](#) 31  
*TRDP reserved data set ids in the range 1 .*

## Enumerations

- enum [TRDP\\_MSG\\_T](#) {  
[TRDP\\_MSG\\_PD](#) = 0x5064,  
[TRDP\\_MSG\\_PP](#) = 0x5070,  
[TRDP\\_MSG\\_PR](#) = 0x5072,  
[TRDP\\_MSG\\_PE](#) = 0x5065,  
[TRDP\\_MSG\\_MN](#) = 0x4D6E,  
[TRDP\\_MSG\\_MR](#) = 0x4D72,  
[TRDP\\_MSG\\_MP](#) = 0x4D70,  
[TRDP\\_MSG\\_MQ](#) = 0x4D71,  
[TRDP\\_MSG\\_MC](#) = 0x4D63,  
[TRDP\\_MSG\\_ME](#) = 0x4D65 }

*Message Types.*

### 5.16.1 Detailed Description

Definitions for the TRDP protocol.

TRDP internal type definitions

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Bernd Loehr, NewTec GmbH

#### Remarks:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2013.

#### Id

[trdp\\_proto.h](#) 1017 2013-07-10 08:56:49Z bloehr

### 5.16.2 Define Documentation

#### 5.16.2.1 #define TRDP\_COMID\_ECHO 10

TRDP reserved COMIDs in the range 1 .

.. 1000

#### 5.16.2.2 #define TRDP\_DEST\_URI\_SIZE 32

max.

Dest URI size in MD header

### 5.16.2.3 #define TRDP\_MAX\_FILE\_NAME\_LEN 128

path and file name length incl.

terminating '0'

### 5.16.2.4 #define TRDP\_MAX\_LABEL\_LEN 16

Maximum values.

A uri is a string of the following form: trdp://[user part]@[host part]trdp://instLabel.funcLabel@devLabel.carLabel.cstLabel.trainLabel Hence the exact max. uri length is: 7 + (6 \* 15) + 5 \* (sizeof (separator)) + 1(terminating 0) to facilitate alignment the size will be increased by 1 byte label length incl. terminating '0'

### 5.16.2.5 #define TRDP\_MAX\_URI\_HOST\_LEN (4 \* TRDP\_MAX\_LABEL\_LEN)

URI host part length incl.

terminating '0'

### 5.16.2.6 #define TRDP\_MAX\_URI\_LEN ((6 \* TRDP\_MAX\_LABEL\_LEN) + 8)

URI length incl.

terminating '0' and 1 padding byte

### 5.16.2.7 #define TRDP\_MAX\_URI\_USER\_LEN (2 \* TRDP\_MAX\_LABEL\_LEN)

URI user part incl.

terminating '0'

### 5.16.2.8 #define TRDP\_STATISTICS\_REQUEST\_DSID 31

TRDP reserved data set ids in the range 1 .

.. 1000

## 5.16.3 Enumeration Type Documentation

### 5.16.3.1 enum TRDP\_MSG\_T

Message Types.

Enumerator:

*TRDP\_MSG\_PD* 'Pd' PD Data

*TRDP\_MSG\_PP* 'Pp' PD Data (Pull Reply)

*TRDP\_MSG\_PR* 'Pr' PD Request

*TRDP\_MSG\_PE* 'Pe' PD Error

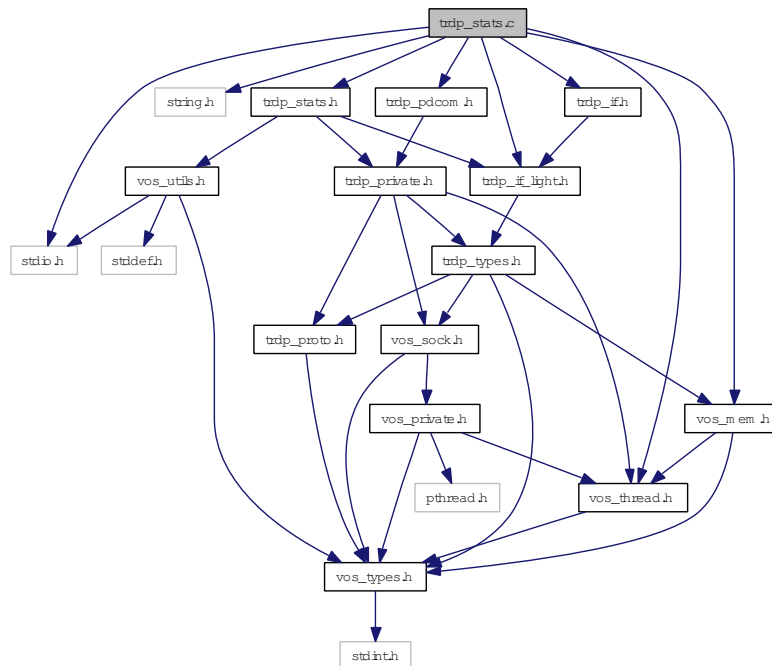
*TRDP\_MSG\_MN* 'Mn' MD Notification (Request without reply)  
*TRDP\_MSG\_MR* 'Mr' MD Request with reply  
*TRDP\_MSG\_MP* 'Mp' MD Reply without confirmation  
*TRDP\_MSG\_MQ* 'Mq' MD Reply with confirmation  
*TRDP\_MSG\_MC* 'Mc' MD Confirm  
*TRDP\_MSG\_ME* 'Me' MD Error

## 5.17 trdp\_stats.c File Reference

Statistics functions for TRDP communication.

```
#include <stdio.h>
#include <string.h>
#include "trdp_stats.h"
#include "trdp_if_light.h"
#include "trdp_if.h"
#include "trdp_pdcom.h"
#include "vos_mem.h"
#include "vos_thread.h"
```

Include dependency graph for trdp\_stats.c:



### Functions

- void [trdp\\_UpdateStats](#) (TRDP\_APP\_SESSION\_T appHandle)  
*Update the statistics.*
- void [trdp\\_initStats](#) (TRDP\_APP\_SESSION\_T appHandle)  
*Init statistics.*
- EXT\_DECL [TRDP\\_ERR\\_T tlc\\_resetStatistics](#) (TRDP\_APP\_SESSION\_T appHandle)  
*Reset statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_STATISTICS\\_T](#) \*pStatistics)

*Return statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getSubsStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT16 \*pNumSubs, [TRDP\\_SUBS\\_STATISTICS\\_T](#) \*pStatistics)

*Return PD subscription statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getPubStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT16 \*pNumPub, [TRDP\\_PUB\\_STATISTICS\\_T](#) \*pStatistics)

*Return PD publish statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getListStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT16 \*pNumList, [TRDP\\_LIST\\_STATISTICS\\_T](#) \*pStatistics)

*Return MD listener statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getRedStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT16 \*pNumRed, [TRDP\\_RED\\_STATISTICS\\_T](#) \*pStatistics)

*Return redundancy group statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getJoinStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT16 \*pNumJoin, UINT32 \*pIpAddr)

*Return join statistics.*

- void [trdp\\_pdPrepareStats](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [PD\\_ELE\\_T](#) \*pPacket)

*Fill the statistics packet.*

### 5.17.1 Detailed Description

Statistics functions for TRDP communication.

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Bernd Loehr, NewTec GmbH

#### Remarks:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

#### Id

[trdp\\_stats.c](#) 1005 2013-07-02 09:13:31Z bloehr

## 5.17.2 Function Documentation

### 5.17.2.1 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.

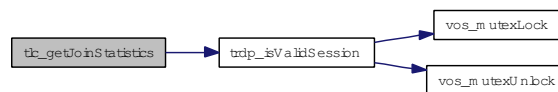
#### Parameters:

- ← *appHandle* the handle returned by tlc\_openSession
- ↔ *pNumJoin* Pointer to the number of joined IP Adresses
- *pIpAddr* Pointer to a list with the joined IP adresses

#### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_NOINIT\_ERR** handle invalid
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_MEM\_ERR** there are more items than requested

Here is the call graph for this function:



### 5.17.2.2 EXT\_DECL TRDP\_ERR\_T tlc\_getListStatistics (TRDP\_APP\_SESSION\_T *appHandle*, UINT16 \**pNumList*, TRDP\_LIST\_STATISTICS\_T \**pStatistics*)

Return MD listener statistics.

Memory for statistics information must be provided by the user.

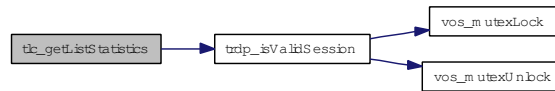
#### Parameters:

- ← *appHandle* the handle returned by tlc\_openSession
- ↔ *pNumList* Pointer to the number of listeners
- *pStatistics* Pointer to a list with the listener statistics information

#### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_NOINIT\_ERR** handle invalid
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_MEM\_ERR** there are more subscriptions than requested

Here is the call graph for this function:



### 5.17.2.3 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:

- ← *appHandle* the handle returned by tlc\_openSession
- ↔ *pNumPub* Pointer to the number of publishers
- *pStatistics* Pointer to a list with the publish statistics information

#### Return values:

- TRDP\_NO\_ERR** no error
- TRDP\_NOINIT\_ERR** handle invalid
- TRDP\_PARAM\_ERR** parameter error
- TRDP\_MEM\_ERR** there are more subscriptions than requested

Here is the call graph for this function:



### 5.17.2.4 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:

- ← *appHandle* the handle returned by tlc\_openSession
- ↔ *pNumRed* Pointer to the number of redundancy groups
- *pStatistics* Pointer to a list with the redundancy group information

#### Return values:

- TRDP\_NO\_ERR** no error

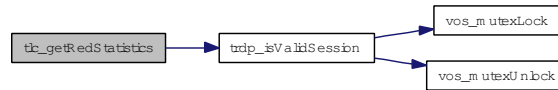


**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

**TRDP\_MEM\_ERR** there are more subscriptions than requested

Here is the call graph for this function:



#### 5.17.2.5 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:

← ***appHandle*** the handle returned by tlc\_openSession

→ ***pStatistics*** Pointer to statistics for this application session

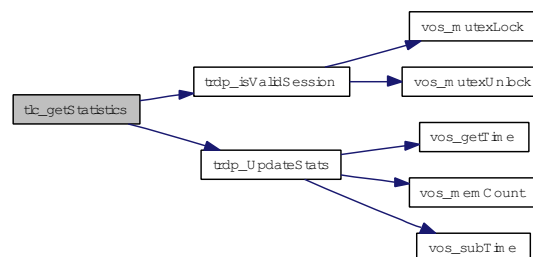
##### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

Here is the call graph for this function:



#### 5.17.2.6 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:

← ***appHandle*** the handle returned by tlc\_openSession

↔ *pNumSubs* In: The number of subscriptions requested Out: Number of subscriptions returned

↔ *pStatistics* Pointer to an array with the subscription statistics information

**Return values:**

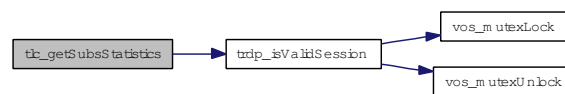
**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

**TRDP\_MEM\_ERR** there are more subscriptions than requested

Here is the call graph for this function:



### 5.17.2.7 EXT\_DECL TRDP\_ERR\_T tlc\_resetStatistics (TRDP\_APP\_SESSION\_T *appHandle*)

Reset statistics.

**Parameters:**

← *appHandle* the handle returned by tlc\_openSession

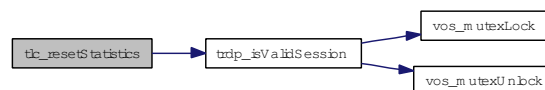
**Return values:**

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

Here is the call graph for this function:



### 5.17.2.8 void trdp\_initStats (TRDP\_APP\_SESSION\_T *appHandle*)

Init statistics.

Clear the stats structure for a session.

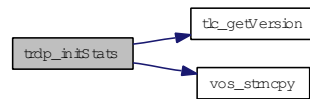
**Parameters:**

← *appHandle* the handle returned by tlc\_openSession

< host name

< leader host name

Here is the call graph for this function:



### 5.17.2.9 void trdp\_pdPrepareStats (TRDP\_APP\_SESSION\_T *appHandle*, PD\_ELE\_T \* *pPacket*)

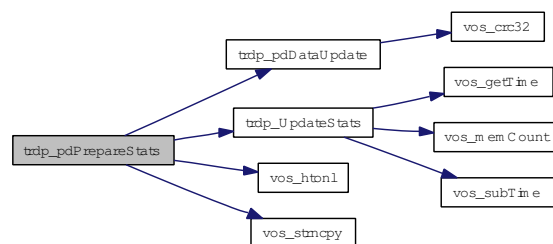
Fill the statistics packet.

#### Parameters:

← *appHandle* the handle returned by tlc\_openSession

↔ *pPacket* pointer to the packet to fill

Here is the call graph for this function:



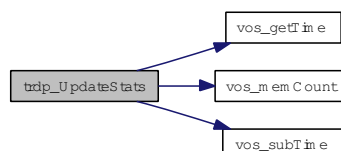
### 5.17.2.10 void trdp\_UpdateStats (TRDP\_APP\_SESSION\_T *appHandle*)

Update the statistics.

#### Parameters:

← *appHandle* the handle returned by tlc\_openSession

Here is the call graph for this function:

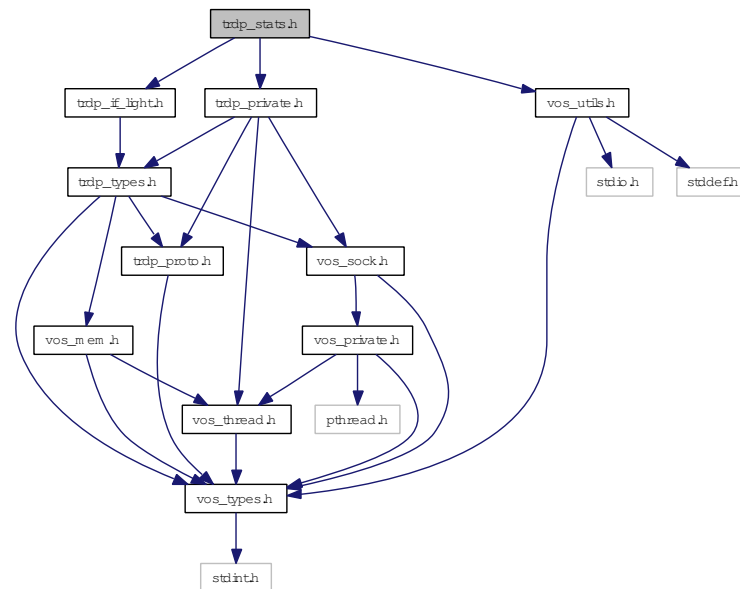


## 5.18 trdp\_stats.h File Reference

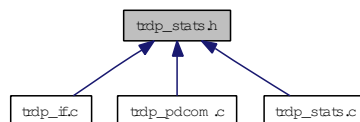
Statistics for TRDP communication.

```
#include "trdp_if_light.h"
#include "trdp_private.h"
#include "vos_utils.h"
```

Include dependency graph for trdp\_stats.h:



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



### Functions

- void [trdp\\_initStats](#) (TRDP\_APP\_SESSION\_T appHandle)  
*Init statistics.*
- void [trdp\\_pdPrepareStats](#) (TRDP\_APP\_SESSION\_T appHandle, PD\_ELE\_T \*pPacket)  
*Fill the statistics packet.*

### 5.18.1 Detailed Description

Statistics for TRDP communication.

**Note:**

Project: TCNOpen TRDP prototype stack

**Author:**

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

[trdp\\_stats.h](#) 950 2013-06-13 13:51:41Z 97025

## 5.18.2 Function Documentation

### 5.18.2.1 void trdp\_initStats (TRDP\_APP\_SESSION\_T *appHandle*)

Init statistics.

Clear the stats structure for a session.

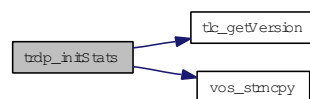
**Parameters:**

← *appHandle* the handle returned by `tlc_openSession`

< host name

< leader host name

Here is the call graph for this function:



### 5.18.2.2 void trdp\_pdPrepareStats (TRDP\_APP\_SESSION\_T *appHandle*, PD\_ELE\_T \* *pPacket*)

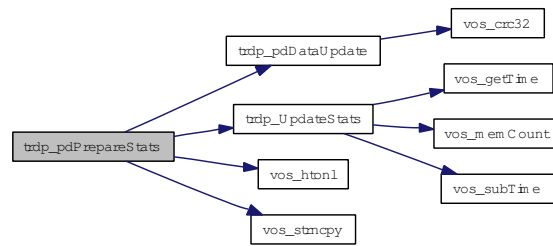
Fill the statistics packet.

**Parameters:**

← *appHandle* the handle returned by `tlc_openSession`

↔ *pPacket* pointer to the packet to fill

Here is the call graph for this function:

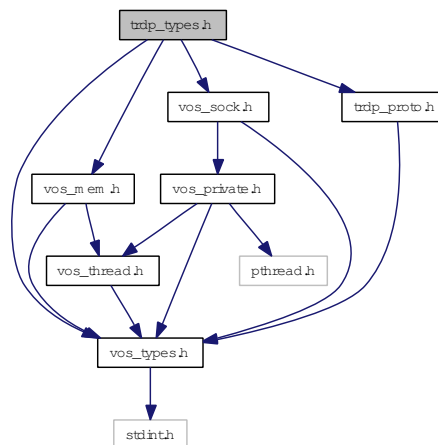


## 5.19 trdp\_types.h File Reference

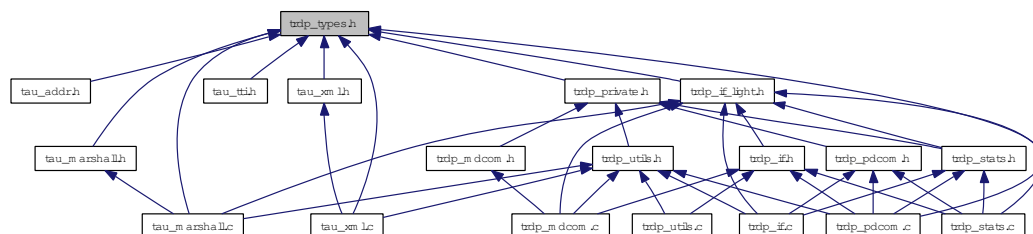
Typedefs for TRDP communication.

```
#include "vos_types.h"
#include "vos_mem.h"
#include "vos_sock.h"
#include "trdp_proto.h"
```

Include dependency graph for trdp\_types.h:



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



## Data Structures

- struct [TRDP\\_VERSION\\_T](#)  
Version information.
- 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\\_SEND\\_PARAM\\_T](#)

*Quality/type of service and time to live.*

- struct [TRDP\\_DATASET\\_ELEMENT\\_T](#)  
*Dataset element definition.*
- struct [TRDP\\_DATASET](#)  
*Dataset definition.*
- struct [TRDP\\_COMID\\_DSID\\_MAP\\_T](#)  
*ComId - data set mapping element definition.*
- struct [TRDP\\_MEM\\_STATISTICS\\_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.*



## Defines

- `#define USE_HEAP 0`  
*If this is set, we can allocate dynamically memory.*

## Typedefs

- `typedef VOS_IP4_ADDR_T TRDP_IP_ADDR_T`  
*TRDP general type definitions.*
- `typedef VOS_TIME_T TRDP_TIME_T`  
*Timer value compatible with timeval / select.*
- `typedef VOS_FDS_T TRDP_FDS_T`  
*File descriptor set compatible with fd\_set / select.*
- `typedef VOS_UUID_T TRDP_UUID_T`  
*UUID definition reuses the VOS definition.*
- `typedef struct TRDP_DATASET TRDP_DATASET_T`  
*Dataset definition.*
- `typedef TRDP_DATASET_T * pTRDP_DATASET_T`  
*Array of pointers to dataset.*
- `typedef VOS_PRINT_DBG_T TRDP_PRINT_DBG_T`  
*TRDP configuration type definitions.*
- `typedef VOS_LOG_T TRDP_LOG_T`  
*Categories for logging, reuse of the VOS definition.*
- `typedef TRDP_ERR_T(* TRDP_MARSHALL_T)(void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT8 *pDst, UINT32 *pDstSize, TRDP_DATASET_T **ppCachedDS)`  
*Function type for marshalling .*
- `typedef TRDP_ERR_T(* TRDP_UNMARSHALL_T)(void *pRefCon, UINT32 comId, UINT8 *pSrc, 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\\_UNKNOWN\\_ERR](#) = -99 }  
    Return codes for all API functions, -1.
- enum [TRDP\\_REPLY\\_STATUS\\_T](#)

*TRDP data transfer type definitions.*

- enum `TRDP_FLAGS_T` {  
    `TRDP_FLAGS_DEFAULT` = 0,  
    `TRDP_FLAGS_NONE` = 0x01,  
    `TRDP_FLAGS_MARSHALL` = 0x02,  
    `TRDP_FLAGS_CALLBACK` = 0x04,  
    `TRDP_FLAGS_TCP` = 0x08 }  
    *Various flags for PD and MD packets.*
- enum `TRDP_RED_STATE_T` {  
    `TRDP_RED_FOLLOWER` = 0,  
    `TRDP_RED_LEADER` = 1 }  
    *Redundancy states.*
- enum `TRDP_TO_BEHAVIOR_T` {  
    `TRDP_TO_DEFAULT` = 0,  
    `TRDP_TO_SET_TO_ZERO` = 1,  
    `TRDP_TO_KEEP_LAST_VALUE` = 2 }  
    *How invalid PD shall be handled.*
- enum `TRDP_DATA_TYPE_T` {  
    `TRDP_BOOLEAN` = 1,  
    `TRDP_CHAR8` = 2,  
    `TRDP_UTF16` = 3,  
    `TRDP_INT8` = 4,  
    `TRDP_INT16` = 5,  
    `TRDP_INT32` = 6,  
    `TRDP_INT64` = 7,  
    `TRDP_UINT8` = 8,  
    `TRDP_UINT16` = 9,  
    `TRDP_UINT32` = 10,  
    `TRDP_UINT64` = 11,  
    `TRDP_REAL32` = 12,  
    `TRDP_REAL64` = 13,  
    `TRDP_TIMEDATE32` = 14,  
    `TRDP_TIMEDATE48` = 15,  
    `TRDP_TIMEDATE64` = 16,  
    `TRDP_TYPE_MAX` = 30 }  
    *TRDP dataset description definitions.*
- enum `TRDP_OPTION_T` { ,  
    `TRDP_OPTION_BLOCK` = 0x01,  
    `TRDP_OPTION_TRAFFIC_SHAPING` = 0x02,  
    `TRDP_OPTION_NO_REUSE_ADDR` = 0x04 }  
    *Various flags/general TRDP options for library initialization.*

### 5.19.1 Detailed Description

Typedefs for TRDP communication.

F

**Note:**

Project: TCNOpen TRDP prototype stack

**Author:**

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

[trdp\\_types.h](#) 993 2013-06-25 13:07:28Z bloehr

### 5.19.2 Typedef Documentation

#### 5.19.2.1 typedef VOS\_IP4\_ADDR\_T TRDP\_IP\_ADDR\_T

TRDP general type definitions.

#### 5.19.2.2 typedef TRDP\_ERR\_T(\* TRDP\_MARSHALL\_T)(void \*pRefCon, UINT32 comId, UINT8 \*pSrc, UINT8 \*pDst, UINT32 \*pDstSize, TRDP\_DATASET\_T \*\*ppCachedDS)

Function type for marshalling .

The function must know about the dataset's alignment etc.

**Parameters:**

- ← *\*pRefCon* pointer to user context
- ← *comId* ComId to identify the structure out of a configuration
- ← *\*pSrc* pointer to received original message
- ← *\*pDst* pointer to a buffer for the treated message
- ↔ *\*pDstSize* size of the provide buffer / size of the treated message
- ↔ *\*ppCachedDS* pointer to pointer of cached dataset

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer to small
- TRDP\_COMID\_ERR* comid not existing

### 5.19.2.3 `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:

- ← *appHandle* handle returned also by `tlc_init`
- ← *pRefCon* pointer to user context
- ← *pMsg* pointer to received message information
- ← *pData* pointer to received data
- ← *dataSize* size of received data pointer to received data excl. padding and FCS !!!!

### 5.19.2.4 `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:

- ← *pRefCon* pointer to user context
- ← *appHandle* application handle returned by `tlc_openSession`
- ← *pMsg* pointer to received message information
- ← *pData* pointer to received data
- ← *dataSize* size of received data pointer to received data excl. padding and FCS !!!!

### 5.19.2.5 `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.

### 5.19.2.6 `typedef VOS_TIME_T TRDP_TIME_T`

Timer value compatible with `timeval` / `select`.

Relative or absolute date, depending on usage

### 5.19.2.7 `typedef TRDP_ERR_T(* TRDP_UNMARSHALL_T)(void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT8 *pDst, UINT32 *pDstSize, TRDP_DATASET_T **ppCachedDS)`

Function type for unmarshalling.

The function must know about the dataset's alignment etc.

#### Parameters:

- ← *pRefCon* pointer to user context
- ← *comId* ComId to identify the structure out of a configuration

← *\*pSrc* pointer to received original message  
 ← *\*pDst* pointer to a buffer for the treated message  
 ↔ *\*pDstSize* size of the provide buffer / size of the treated message  
 ↔ *\*ppCachedDS* pointer to pointer of cached dataset

#### Return values:

*TRDP\_NO\_ERR* no error  
*TRDP\_MEM\_ERR* provide buffer to small  
*TRDP\_COMID\_ERR* comid not existing

### 5.19.3 Enumeration Type Documentation

#### 5.19.3.1 enum TRDP\_DATA\_TYPE\_T

TRDP dataset description definitions.

Dataset element definition

#### Enumerator:

*TRDP\_BOOLEAN* =UINT8, 1 bit relevant (equal to zero = false, not equal to zero = true)  
*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 (== struct timeval)  
*TRDP\_TYPE\_MAX* Values greater are considered nested datasets.

#### 5.19.3.2 enum TRDP\_ERR\_T

Return codes for all API functions, -1.

.-29 taken over from vos

#### Enumerator:

*TRDP\_NO\_ERR* No error.

**TRDP\_PARAM\_ERR** Parameter missing or out of range.  
**TRDP\_INIT\_ERR** Call without valid initialization.  
**TRDP\_NOINIT\_ERR** Call with invalid handle.  
**TRDP\_TIMEOUT\_ERR** Timeout.  
**TRDP\_NODATA\_ERR** Non blocking mode: no data received.  
**TRDP SOCK\_ERR** Socket error / option not supported.  
**TRDP\_IO\_ERR** Socket IO error, data can't be received/sent.  
**TRDP\_MEM\_ERR** No more memory available.  
**TRDP\_SEMA\_ERR** Semaphore not available.  
**TRDP\_QUEUE\_ERR** Queue empty.  
**TRDP\_QUEUE\_FULL\_ERR** Queue full.  
**TRDP\_MUTEX\_ERR** Mutex not available.  
**TRDP\_THREAD\_ERR** Thread error.  
**TRDP\_BLOCK\_ERR** System call would have blocked in blocking mode.  
**TRDP\_INTEGRATION\_ERR** Alignment or endianness for selected target wrong.  
**TRDP\_NOCONN\_ERR** No TCP connection.  
**TRDP\_NOSESSION\_ERR** No such session.  
**TRDP\_SESSION\_ABORT\_ERR** Session aborted.  
**TRDP\_NOSUB\_ERR** No subscriber.  
**TRDP\_NOPUB\_ERR** No publisher.  
**TRDP\_NOLIST\_ERR** No listener.  
**TRDP\_CRC\_ERR** Wrong CRC.  
**TRDP\_WIRE\_ERR** Wire.  
**TRDP\_TOPO\_ERR** Invalid topo count.  
**TRDP\_COMID\_ERR** Unknown ComId.  
**TRDP\_STATE\_ERR** Call in wrong state.  
**TRDP\_APP\_TIMEOUT\_ERR** Application Timeout.  
**TRDP\_APP\_REPLYTO\_ERR** Application Reply Sent Timeout.  
**TRDP\_APP\_CONFIRMTO\_ERR** Application Confirm Sent Timeout.  
**TRDP\_REPLYTO\_ERR** Protocol Reply Timeout.  
**TRDP\_CONFIRMTO\_ERR** Protocol Confirm Timeout.  
**TRDP\_REQCONFIRMTO\_ERR** Protocol Confirm Timeout (Request sender).  
**TRDP\_PACKET\_ERR** Incomplete message data packet.  
**TRDP\_UNKNOWN\_ERR** Unspecified error.

### 5.19.3.3 enum TRDP\_FLAGS\_T

Various flags for PD and MD packets.

#### Enumerator:

**TRDP\_FLAGS\_DEFAULT** Default value defined in tlc\_openDession will be taken.  
**TRDP\_FLAGS\_NONE** No flags set.  
**TRDP\_FLAGS\_MARSHALL** Optional marshalling/unmarshalling in TRDP stack.  
**TRDP\_FLAGS\_CALLBACK** Use of callback function.  
**TRDP\_FLAGS\_TCP** Use TCP for message data.

#### 5.19.3.4 enum TRDP\_OPTION\_T

Various flags/general TRDP options for library initialization.

**Enumerator:**

**TRDP\_OPTION\_BLOCK** Default: Use nonblocking I/O calls, polling necessary Set: Read calls will block, use select().

**TRDP\_OPTION\_TRAFFIC\_SHAPING** Use traffic shaping - distribute packet sending Default: OFF.

**TRDP\_OPTION\_NO\_REUSE\_ADDR** Do not allow re-use of address/port (-> no multihoming) Default: Allow.

#### 5.19.3.5 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.

#### 5.19.3.6 enum TRDP\_REPLY\_STATUS\_T

TRDP data transfer type definitions.

Reply status messages

#### 5.19.3.7 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.

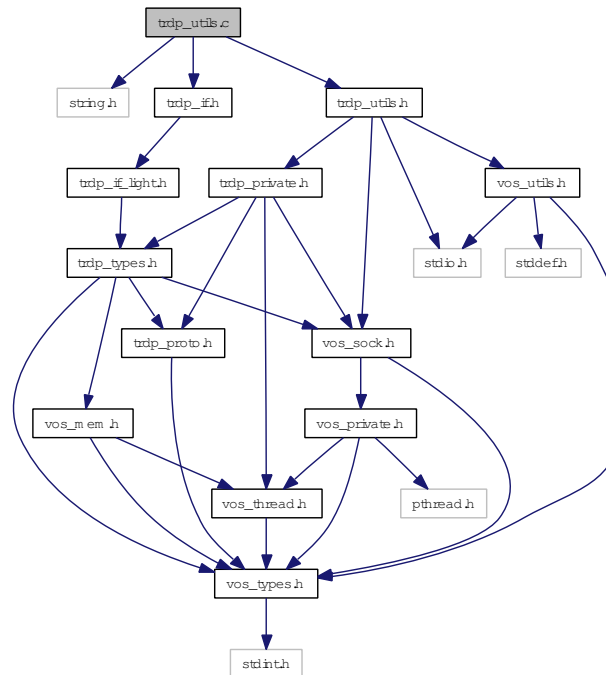


## 5.20 trdp\_utils.c File Reference

Helper functions for TRDP communication.

```
#include <string.h>
#include "trdp_if.h"
#include "trdp_utils.h"
```

Include dependency graph for trdp\_utils.c:



## Functions

- void [printSocketUsage](#) (TRDP\_SOCKETS\_T iface[ ])
 

*Debug socket usage output.*
- BOOL [trdp\\_SockIsJoined](#) (const TRDP\_IP\_ADDR\_T mcList[VOS\_MAX\_MULTICAST\_CNT], TRDP\_IP\_ADDR\_T mcGroup)
 

*Check if a mc group is in the list.*
- BOOL [trdp\\_SockAddJoin](#) (TRDP\_IP\_ADDR\_T mcList[VOS\_MAX\_MULTICAST\_CNT], TRDP\_IP\_ADDR\_T mcGroup)
 

*Add mc group to the list.*
- BOOL [trdp\\_SockDelJoin](#) (TRDP\_IP\_ADDR\_T mcList[VOS\_MAX\_MULTICAST\_CNT], TRDP\_IP\_ADDR\_T mcGroup)
 

*remove mc group from the list*
- int [am\\_big\\_endian](#) ()

*Determine if we are Big or Little endian.*

- `UINT32 trdp_packetSizePD (UINT32 dataSize)`  
*Get the packet size from the raw data size.*
- `UINT32 trdp_packetSizeMD (UINT32 dataSize)`  
*Get the packet size from the raw data size.*
- `PD_ELE_T * trdp_queueFindComId (PD_ELE_T *pHead, UINT32 comId)`  
*Return the element with same comId.*
- `PD_ELE_T * trdp_queueFindPubAddr (PD_ELE_T *pHead, TRDP_ADDRESSES_T *addr)`  
*Return the element with same comId and IP addresses.*
- `PD_ELE_T * trdp_queueFindSubAddr (PD_ELE_T *pHead, TRDP_ADDRESSES_T *addr)`  
*Return the element with same comId and IP addresses.*
- `void trdp_queueDelElement (PD_ELE_T **ppHead, PD_ELE_T *pDelete)`  
*Delete an element.*
- `void trdp_queueAppLast (PD_ELE_T **ppHead, PD_ELE_T *pNew)`  
*Append an element at end of queue.*
- `void trdp_queueInsFirst (PD_ELE_T **ppHead, PD_ELE_T *pNew)`  
*Insert an element at front of queue.*
- `void trdp_initSockets (TRDP_SOCKETS_T iface[ ])`  
*Handle the socket pool: Initialize it.*
- `TRDP_ERR_T trdp_requestSocket (TRDP_SOCKETS_T iface[ ], UINT32 port, const TRDP_SEND_PARAM_T *params, TRDP_IP_ADDR_T srcIP, TRDP_IP_ADDR_T mcGroup, TRDP_SOCK_TYPE_T usage, TRDP_OPTION_T options, BOOL rcvMostly, INT32 useSocket, INT32 *pIndex, TRDP_IP_ADDR_T cornerIp)`  
*Handle the socket pool: Request a socket from our socket pool First we loop through the socket pool and check if there is already a socket which would suit us.*
- `void trdp_releaseSocket (TRDP_SOCKETS_T iface[ ], INT32 lIndex, UINT32 connectTimeout, BOOL checkAll)`  
*Handle the socket pool: if a received TCP socket is unused, the socket connection timeout is started.*
- `UINT32 trdp_getSeqCnt (UINT32 comId, TRDP_MSG_T msgType, TRDP_IP_ADDR_T srcIPAddr)`  
*Get the initial sequence counter for the comID/message type and subnet (source IP).*
- `BOOL trdp_isRcvSeqCnt (UINT32 seqCnt, UINT32 comId, TRDP_MSG_T msgType, TRDP_IP_ADDR_T srcIP)`  
*Check if the sequence counter for the comID/message type and subnet (source IP) has already been received.*
- `BOOL trdp_isAddressed (const TRDP_URI_USER_T listUri, const TRDP_URI_USER_T destUri)`  
*Check if listener URI is in addressing range of destination URI.*

### 5.20.1 Detailed Description

Helper functions for TRDP communication.

**Note:**

Project: TCNOpen TRDP prototype stack

**Author:**

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2013.

**Id**

[trdp\\_utils.c](#) 1021 2013-07-12 08:53:08Z cschneider

### 5.20.2 Function Documentation

#### 5.20.2.1 int am\_big\_endian ()

Determine if we are Big or Little endian.

**Return values:**

`!= 0` we are big endian  
`0` we are little endian

#### 5.20.2.2 void printSocketUsage (TRDP\_SOCKETS\_T *iface*[])

Debug socket usage output.

**Parameters:**

← *iface*[] List of sockets

#### 5.20.2.3 UINT32 trdp\_getSeqCnt (UINT32 *comId*, TRDP\_MSG\_T *msgType*, TRDP\_IP\_ADDR\_T *srcIpAddr*)

Get the initial sequence counter for the comID/message type and subnet (source IP).

If the comID/srcIP is not found elsewhere, return 0 - else return its current sequence number (the redundant packet needs the same seqNo)

Note: The standard demands that sequenceCounter is managed per comID/msgType at each publisher, but shall be the same for redundant telegrams (subnet/srcIP).

**Parameters:**

← *comId* comID to look for

← *msgType* PD/MD type  
 ← *srcIpAddr* Source IP address

**Return values:**

*return* the sequence number

Here is the call graph for this function:



#### 5.20.2.4 void trdp\_initSockets (TRDP\_SOCKETS\_T iface[])

Handle the socket pool: Initialize it.

**Parameters:**

← *iface* pointer to the socket pool

#### 5.20.2.5 BOOL trdp\_isAddressed (const TRDP\_URI\_USER\_T listUri, const TRDP\_URI\_USER\_T destUri)

Check if listener URI is in addressing range of destination URI.

**Parameters:**

← *listUri* Null terminated listener URI string to compare  
 ← *destUri* Null terminated destination URI string to compare

**Return values:**

*FALSE* - not in addressing range  
*TRUE* - listener URI is in addressing range of destination URI

Here is the call graph for this function:



#### 5.20.2.6 BOOL trdp\_isRcvSeqCnt (UINT32 seqCnt, UINT32 comId, TRDP\_MSG\_T msgType, TRDP\_IP\_ADDR\_T srcIP)

Check if the sequence counter for the comID/message type and subnet (source IP) has already been received.

Note: The standard demands that sequenceCounter is managed per comID/msgType at each publisher, but shall be the same for redundant telegrams (subnet/srcIP).

**Parameters:**

- ← *seqCnt* sequence counter received
- ← *comId* comID to look for
- ← *msgType* PD/MD type
- ← *srcIP* Source IP address

**Return values:**

- return* the sequence number

Here is the call graph for this function:

**5.20.2.7** `UINT32 trdp_packetSizeMD (UINT32 dataSize)`

Get the packet size from the raw data size.

**Parameters:**

- ← *dataSize* net data size (without padding or FCS)

**Return values:**

- packet* size the size of the complete packet to be sent or received

**5.20.2.8** `UINT32 trdp_packetSizePD (UINT32 dataSize)`

Get the packet size from the raw data size.

**Parameters:**

- ← *dataSize* net data size (without padding or FCS)

**Return values:**

- packet* size the size of the complete packet to be sent or received

**5.20.2.9** `void trdp_queueAppLast (PD_ELE_T **ppHead, PD_ELE_T *pNew)`

Append an element at end of queue.

**Parameters:**

- ← *ppHead* pointer to pointer to head of queue
- ← *pNew* pointer to element to append

### 5.20.2.10 void trdp\_queueDelElement (PD\_ELE\_T \*\* *ppHead*, PD\_ELE\_T \* *pDelete*)

Delete an element.

#### Parameters:

- ← *ppHead* pointer to pointer to head of queue
- ← *pDelete* pointer to element to delete

### 5.20.2.11 PD\_ELE\_T\* trdp\_queueFindComId (PD\_ELE\_T \* *pHead*, UINT32 *comId*)

Return the element with same comId.

#### Parameters:

- ← *pHead* pointer to head of queue
- ← *comId* ComID to search for

#### Return values:

- !=* NULL pointer to PD element
- NULL* No PD element found

### 5.20.2.12 PD\_ELE\_T\* trdp\_queueFindPubAddr (PD\_ELE\_T \* *pHead*, TRDP\_ADDRESSES\_T \* *addr*)

Return the element with same comId and IP addresses.

#### Parameters:

- ← *pHead* pointer to head of queue
- ← *addr* Pub/Sub handle (Address, ComID, srcIP & dest IP) to search for

#### Return values:

- !=* NULL pointer to PD element
- NULL* No PD element found

### 5.20.2.13 PD\_ELE\_T\* trdp\_queueFindSubAddr (PD\_ELE\_T \* *pHead*, TRDP\_ADDRESSES\_T \* *addr*)

Return the element with same comId and IP addresses.

#### Parameters:

- ← *pHead* pointer to head of queue
- ← *addr* Pub/Sub handle (Address, ComID, srcIP & dest IP) to search for

#### Return values:

- !=* NULL pointer to PD element
- NULL* No PD element found

#### 5.20.2.14 void trdp\_queueInsFirst (PD\_ELE\_T \*\* *ppHead*, PD\_ELE\_T \* *pNew*)

Insert an element at front of queue.

##### Parameters:

- ← *ppHead* pointer to pointer to head of queue
- ← *pNew* pointer to element to insert

#### 5.20.2.15 void trdp\_releaseSocket (TRDP\_SOCKETS\_T *iface*[], INT32 *lIndex*, UINT32 *connectTimeout*, BOOL *checkAll*)

Handle the socket pool: if a received TCP socket is unused, the socket connection timeout is started.

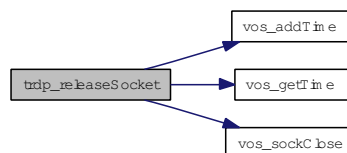
Handle the socket pool: Release a socket from our socket pool.

In Udp, Release a socket from our socket pool

##### Parameters:

- ↔ *iface* socket pool
- ← *lIndex* index of socket to release
- ← *connectTimeout* time out
- ← *checkAll* release all TCP pending sockets

Here is the call graph for this function:



#### 5.20.2.16 TRDP\_ERR\_T trdp\_requestSocket (TRDP\_SOCKETS\_T *iface*[], UINT32 *port*, const TRDP\_SEND\_PARAM\_T \* *params*, TRDP\_IP\_ADDR\_T *srcIP*, TRDP\_IP\_ADDR\_T *mcGroup*, TRDP\_SOCKET\_TYPE\_T *usage*, TRDP\_OPTION\_T *options*, BOOL *rcvMostly*, INT32 *useSocket*, INT32 \* *pIndex*, TRDP\_IP\_ADDR\_T *cornerIp*)

Handle the socket pool: Request a socket from our socket pool First we loop through the socket pool and check if there is already a socket which would suit us.

Handle the socket pool: Request a socket from our socket pool.

If a multicast group should be joined, we do that on an otherwise suitable socket - up to 20 multicast groups can be joined per socket. If a socket for multicast publishing is requested, we also use the source IP to determine the interface for outgoing multicast traffic.

##### Parameters:

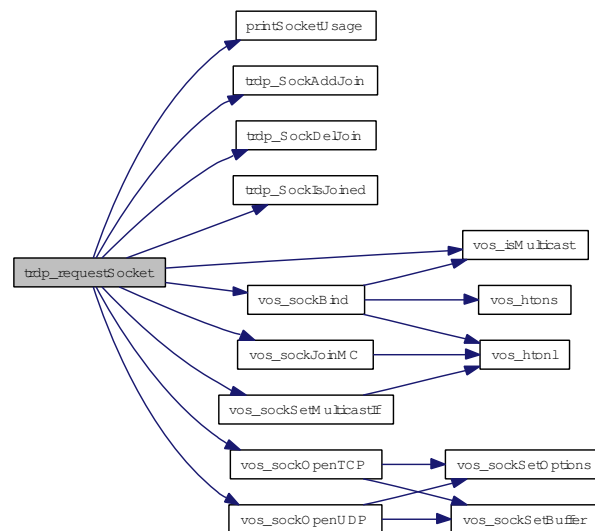
- ↔ *iface* socket pool
- ← *port* port to use

- ← *params* parameters to use
- ← *srcIP* IP to bind to (0 = any address)
- ← *mcGroup* MC group to join (0 = do not join)
- ← *usage* type and port to bind to (PD, MD/UDP, MD/TCP)
- ← *options* blocking/nonblocking
- ← *rcvMostly* primarily used for receiving (tbd: bind on sender, too?)
- *useSocket* socket to use, do not open a new one
- *pIndex* returned index of socket pool
- ← *cornerIp* only used for receiving

**Return values:**

- TRDP\_NO\_ERR**
- TRDP\_PARAM\_ERR**

Here is the call graph for this function:



### 5.20.2.17 **BOOL trdp\_SockAddJoin (TRDP\_IP\_ADDR\_T mcList[VOS\_MAX\_MULTICAST\_CNT], TRDP\_IP\_ADDR\_T mcGroup)**

Add mc group to the list.

**Parameters:**

- ← *mcList[]* List of multicast groups
- ← *mcGroup* multicast group

**Return values:**

- 1** if added 0 if list is full



**5.20.2.18** `BOOL trdp_SockDelJoin (TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT], TRDP_IP_ADDR_T mcGroup)`

remove mc group from the list

**Parameters:**

← *mcList*[ ] List of multicast groups

← *mcGroup* multicast group

**Return values:**

*1* if deleted 0 was not in list

**5.20.2.19** `BOOL trdp_SockIsJoined (const TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT], TRDP_IP_ADDR_T mcGroup)`

Check if a mc group is in the list.

**Parameters:**

← *mcList*[ ] List of multicast groups

← *mcGroup* multicast group

**Return values:**

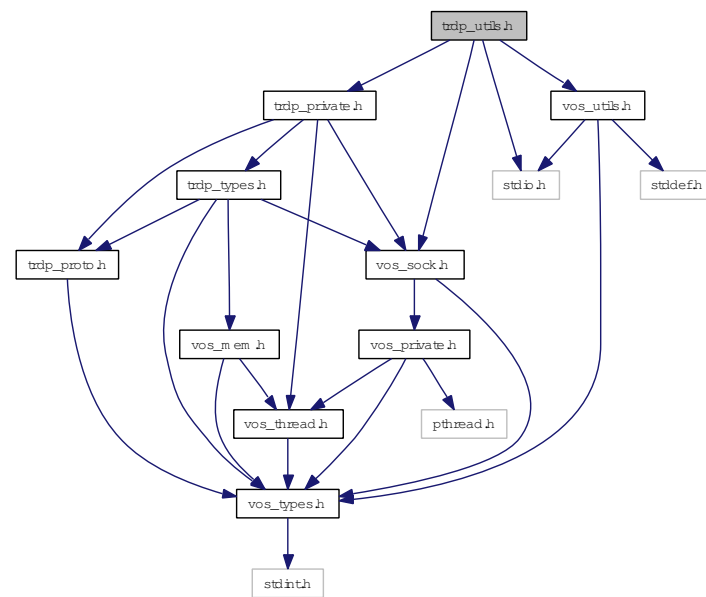
*1* if found 0 if not found

## 5.21 trdp\_utils.h File Reference

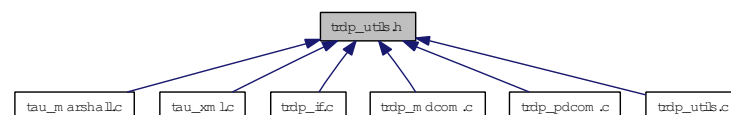
Common utilities for TRDP communication.

```
#include <stdio.h>
#include "trdp_private.h"
#include "vos_utils.h"
#include "vos_sock.h"
```

Include dependency graph for trdp\_utils.h:



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



## Functions

- `int am_big_endian()`  
*Determine if we are Big or Little endian.*
- `PD_ELEMENT * trdp_queueFindComId(PD_ELEMENT *pHead, UINT32 comId)`  
*Return the element with same comId.*
- `PD_ELEMENT * trdp_queueFindSubAddr(PD_ELEMENT *pHead, TRDP_ADDRESSES_T *pAddr)`  
*Return the element with same comId and IP addresses.*
- `PD_ELEMENT * trdp_queueFindPubAddr(PD_ELEMENT *pHead, TRDP_ADDRESSES_T *addr)`

*Return the element with same comId and IP addresses.*

- void [trdp\\_queueDelElement](#) (PD\_ELE\_T \*\*pHead, PD\_ELE\_T \*pDelete)  
*Delete an element.*
- void [trdp\\_queueAppLast](#) (PD\_ELE\_T \*\*pHead, PD\_ELE\_T \*pNew)  
*Append an element at end of queue.*
- void [trdp\\_queueInsFirst](#) (PD\_ELE\_T \*\*pHead, PD\_ELE\_T \*pNew)  
*Insert an element at front of queue.*
- void [trdp\\_initSockets](#) (TRDP\_SOCKETS\_T iface[ ])  
*Handle the socket pool: Initialize it.*
- void [trdp\\_initUncompletedTCP](#) (TRDP\_APP\_SESSION\_T appHandle)  
*???*
- TRDP\_ERR\_T [trdp\\_requestSocket](#) (TRDP\_SOCKETS\_T iface[ ], UINT32 port, const TRDP\_SEND\_PARAM\_T \*params, TRDP\_IP\_ADDR\_T srcIP, TRDP\_IP\_ADDR\_T mcGroup, TRDP SOCK\_TYPE\_T usage, TRDP\_OPTION\_T options, BOOL rcvMostly, INT32 useSocket, INT32 \*pIndex, TRDP\_IP\_ADDR\_T cornerIp)  
*Handle the socket pool: Request a socket from our socket pool.*
- void [trdp\\_releaseSocket](#) (TRDP\_SOCKETS\_T iface[ ], INT32 lIndex, UINT32 connectTimeout, BOOL checkAll)  
*Handle the socket pool: Release a socket from our socket pool.*
- UINT32 [trdp\\_packetSizePD](#) (UINT32 dataSize)  
*Get the packet size from the raw data size.*
- UINT32 [trdp\\_packetSizeMD](#) (UINT32 dataSize)  
*Get the packet size from the raw data size.*
- UINT32 [trdp\\_getSeqCnt](#) (UINT32 comID, TRDP\_MSG\_T msgType, TRDP\_IP\_ADDR\_T srcIP)  
*Get the initial sequence counter for the comID/message type and subnet (source IP).*
- BOOL [trdp\\_isRcvSeqCnt](#) (UINT32 seqCnt, UINT32 comId, TRDP\_MSG\_T msgType, TRDP\_IP\_ADDR\_T srcIP)  
*Check if the sequence counter for the comID/message type and subnet (source IP) has already been received.*
- BOOL [trdp\\_isAddressed](#) (const TRDP\_URI\_USER\_T listUri, const TRDP\_URI\_USER\_T destUri)  
*Check if listener URI is in addressing range of destination URI.*

### 5.21.1 Detailed Description

Common utilities for TRDP communication.

**Note:**

Project: TCNOpen TRDP prototype stack

**Author:**

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

[trdp\\_utils.h](#) 995 2013-06-27 08:01:06Z bloehr

## 5.21.2 Function Documentation

### 5.21.2.1 `int am_big_endian ()`

Determine if we are Big or Little endian.

**Return values:**

`!= 0` we are big endian

`0` we are little endian

### 5.21.2.2 `UINT32 trdp_getSeqCnt (UINT32 comId, TRDP_MSG_T msgType, TRDP_IP_ADDR_T srcIpAddr)`

Get the initial sequence counter for the comID/message type and subnet (source IP).

If the comID/srcIP is not found elsewhere, return 0 - else return its current sequence number (the redundant packet needs the same seqNo)

Note: The standard demands that sequenceCounter is managed per comID/msgType at each publisher, but shall be the same for redundant telegrams (subnet/srcIP).

**Parameters:**

← *comId* comID to look for

← *msgType* PD/MD type

← *srcIpAddr* Source IP address

**Return values:**

*return* the sequence number

Here is the call graph for this function:



**5.21.2.3 void trdp\_initSockets (TRDP\_SOCKETS\_T *iface*[])**

Handle the socket pool: Initialize it.

**Parameters:**

← *iface* pointer to the socket pool

**5.21.2.4 void trdp\_initUncompletedTCP (TRDP\_APP\_SESSION\_T *appHandle*)**

???

**Parameters:**

← *appHandle* session handle

**5.21.2.5 BOOL trdp\_isAddressed (const TRDP\_URI\_USER\_T *listUri*, const TRDP\_URI\_USER\_T *destUri*)**

Check if listener URI is in addressing range of destination URI.

**Parameters:**

← *listUri* Null terminated listener URI string to compare

← *destUri* Null terminated destination URI string to compare

**Return values:**

**FALSE** - not in addressing range

**TRUE** - listener URI is in addressing range of destination URI

Here is the call graph for this function:

**5.21.2.6 BOOL trdp\_isRcvSeqCnt (UINT32 *seqCnt*, UINT32 *comId*, TRDP\_MSG\_T *msgType*, TRDP\_IP\_ADDR\_T *srcIP*)**

Check if the sequence counter for the comID/message type and subnet (source IP) has already been received.

Note: The standard demands that sequenceCounter is managed per comID/msgType at each publisher, but shall be the same for redundant telegrams (subnet/srcIP).

**Parameters:**

← *seqCnt* sequence counter received

← *comId* comID to look for

← *msgType* PD/MD type

← *srcIP* Source IP address

**Return values:**

*return* the sequence number

Here is the call graph for this function:



### 5.21.2.7 UINT32 trdp\_packetSizeMD (UINT32 dataSize)

Get the packet size from the raw data size.

**Parameters:**

← *dataSize* net data size (without padding or FCS)

**Return values:**

*packet* size the size of the complete packet to be sent or received

### 5.21.2.8 UINT32 trdp\_packetSizePD (UINT32 dataSize)

Get the packet size from the raw data size.

**Parameters:**

← *dataSize* net data size (without padding or FCS)

**Return values:**

*packet* size the size of the complete packet to be sent or received

### 5.21.2.9 void trdp\_queueAppLast (PD\_ELE\_T \*\*ppHead, PD\_ELE\_T \*pNew)

Append an element at end of queue.

**Parameters:**

← *ppHead* pointer to pointer to head of queue

← *pNew* pointer to element to append

### 5.21.2.10 void trdp\_queueDelElement (PD\_ELE\_T \*\*ppHead, PD\_ELE\_T \*pDelete)

Delete an element.

**Parameters:**

← *ppHead* pointer to pointer to head of queue

← *pDelete* pointer to element to delete

**5.21.2.11 PD\_ELE\_T\* trdp\_queueFindComId (PD\_ELE\_T \* *pHead*, UINT32 *comId*)**

Return the element with same comId.

**Parameters:**

- ← *pHead* pointer to head of queue
- ← *comId* ComID to search for

**Return values:**

- != NULL pointer to PD element
- NULL No PD element found

**5.21.2.12 PD\_ELE\_T\* trdp\_queueFindPubAddr (PD\_ELE\_T \* *pHead*, TRDP\_ADDRESSES\_T \* *addr*)**

Return the element with same comId and IP addresses.

**Parameters:**

- ← *pHead* pointer to head of queue
- ← *addr* Pub/Sub handle (Address, ComID, srcIP & dest IP) to search for

**Return values:**

- != NULL pointer to PD element
- NULL No PD element found

**5.21.2.13 PD\_ELE\_T\* trdp\_queueFindSubAddr (PD\_ELE\_T \* *pHead*, TRDP\_ADDRESSES\_T \* *addr*)**

Return the element with same comId and IP addresses.

**Parameters:**

- ← *pHead* pointer to head of queue
- ← *addr* Pub/Sub handle (Address, ComID, srcIP & dest IP) to search for

**Return values:**

- != NULL pointer to PD element
- NULL No PD element found

**5.21.2.14 void trdp\_queueInsFirst (PD\_ELE\_T \*\* *ppHead*, PD\_ELE\_T \* *pNew*)**

Insert an element at front of queue.

**Parameters:**

- ← *ppHead* pointer to pointer to head of queue
- ← *pNew* pointer to element to insert

### 5.21.2.15 void trdp\_releaseSocket (TRDP\_SOCKETS\_T *iface*[], INT32 *lIndex*, UINT32 *connectTimeout*, BOOL *checkAll*)

Handle the socket pool: Release a socket from our socket pool.

#### Parameters:

- ↔ *iface* socket pool
- ← *lIndex* index of socket to release
- ← *connectTimeout* timeout value
- ← *checkAll* release all TCP pending sockets

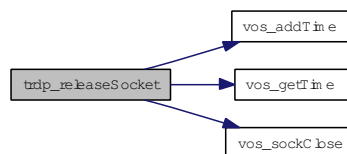
Handle the socket pool: Release a socket from our socket pool.

In Udp, Release a socket from our socket pool

#### Parameters:

- ↔ *iface* socket pool
- ← *lIndex* index of socket to release
- ← *connectTimeout* time out
- ← *checkAll* release all TCP pending sockets

Here is the call graph for this function:



### 5.21.2.16 TRDP\_ERR\_T trdp\_requestSocket (TRDP\_SOCKETS\_T *iface*[], UINT32 *port*, const TRDP\_SEND\_PARAM\_T \**params*, TRDP\_IP\_ADDR\_T *srcIP*, TRDP\_IP\_ADDR\_T *mcGroup*, TRDP SOCK\_TYPE\_T *usage*, TRDP\_OPTION\_T *options*, BOOL *rcvMostly*, INT32 *useSocket*, INT32 \**pIndex*, TRDP\_IP\_ADDR\_T *cornerIp*)

Handle the socket pool: Request a socket from our socket pool.

#### Parameters:

- ↔ *iface* socket pool
- ← *port* port to use
- ← *params* parameters to use
- ← *srcIP* IP to bind to (0 = any address)
- ← *mcGroup* MC group to join (0 = do not join)
- ← *usage* type and port to bind to
- ← *options* blocking/nonblocking
- ← *rcvMostly* only used for receiving



- *useSocket* socket to use, do not open a new one
- *pIndex* returned index of socket pool
- ← *cornerIp* only used for receiving

**Return values:****TRDP\_NO\_ERR****TRDP\_PARAM\_ERR** Handle the socket pool: Request a socket from our socket pool.

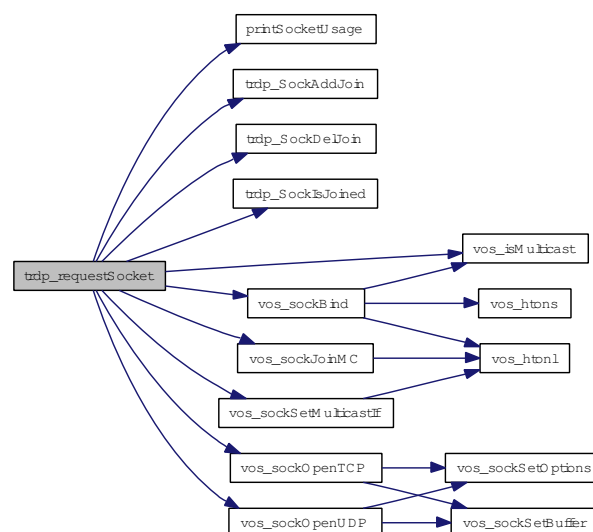
If a multicast group should be joined, we do that on an otherwise suitable socket - up to 20 multicast groups can be joined per socket. If a socket for multicast publishing is requested, we also use the source IP to determine the interface for outgoing multicast traffic.

**Parameters:**

- ↔ *iface* socket pool
- ← *port* port to use
- ← *params* parameters to use
- ← *srcIP* IP to bind to (0 = any address)
- ← *mcGroup* MC group to join (0 = do not join)
- ← *usage* type and port to bind to (PD, MD/UDP, MD/TCP)
- ← *options* blocking/nonblocking
- ← *rcvMostly* primarily used for receiving (tbd: bind on sender, too?)
- *useSocket* socket to use, do not open a new one
- *pIndex* returned index of socket pool
- ← *cornerIp* only used for receiving

**Return values:****TRDP\_NO\_ERR****TRDP\_PARAM\_ERR**

Here is the call graph for this function:

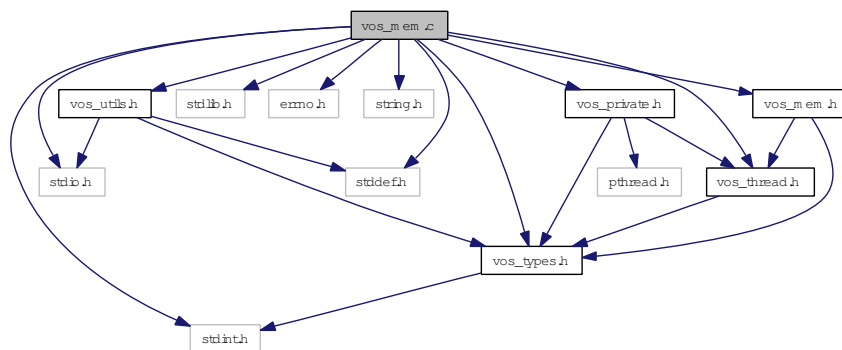


## 5.22 vos\_mem.c File Reference

Memory functions.

```
#include <stdio.h>
#include <stddef.h>
#include <stdint.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include "vos_types.h"
#include "vos_utils.h"
#include "vos_mem.h"
#include "vos_thread.h"
#include "vos_private.h"
```

Include dependency graph for vos\_mem.c:



## Functions

- **VOS\_ERR\_T vos\_mutexLocalCreate** (struct VOS\_MUTEX \*pMutex)  
*Create a recursive mutex.*
- void **vos\_mutexLocalDelete** (struct VOS\_MUTEX \*pMutex)  
*Delete a mutex.*
- EXT\_DECL **VOS\_ERR\_T vos\_memInit** (UINT8 \*pMemoryArea, UINT32 size, const UINT32 fragMem[VOS\_MEM\_NBLOCKSIZES])  
*Initialize the memory unit.*
- EXT\_DECL void **vos\_memDelete** (UINT8 \*pMemoryArea)  
*Delete the memory area.*
- EXT\_DECL UINT8 \* **vos\_memAlloc** (UINT32 size)  
*Allocate a block of memory (from memory area above).*

- EXT\_DECL void [vos\\_memFree](#) (void \*pMemBlock)  
*Deallocate a block of memory (from memory area above).*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_memCount](#) (UINT32 \*pAllocatedMemory, UINT32 \*pFreeMemory, UINT32 \*pMinFree, UINT32 \*pNumAllocBlocks, UINT32 \*pNumAllocErr, UINT32 \*pNumFreeErr, UINT32 blockSize[VOS\_MEM\_NBLOCKSIZES], UINT32 usedBlockSize[VOS\_MEM\_NBLOCKSIZES])  
*Return used and available memory (of memory area above).*
- EXT\_DECL void [vos\\_qsort](#) (void \*pBuf, UINT32 num, UINT32 size, int(\*compare)(const void \*, const void \*))  
*Sort an array.*
- EXT\_DECL void \* [vos\\_bsearch](#) (const void \*pKey, const void \*pBuf, UINT32 num, UINT32 size, int(\*compare)(const void \*, const void \*))  
*Binary search in a sorted array.*
- EXT\_DECL INT32 [vos\\_strncmp](#) (const CHAR8 \*pStr1, const CHAR8 \*pStr2, UINT32 count)  
*Case insensitive string compare.*
- EXT\_DECL void [vos\\_strncpy](#) (CHAR8 \*pStrDst, const CHAR8 \*pStrSrc, UINT32 count)  
*String copy with length limitation.*
- EXT\_DECL [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.*

### 5.22.1 Detailed Description

Memory functions.

OS abstraction of memory access and control

**Note:**

Project: TCNOpen TRDP prototype stack

**Author:**

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

[vos\\_mem.c](#) 997 2013-07-01 12:23:39Z bloehr

Changes: BL 2012-12-03: ID 1: "using uninitialized PD\_ELE\_T.pullIpAddress variable" ID 2: "uninitialized PD\_ELE\_T newPD → pNext in tlp\_subscribe()"

**5.22.2 Function Documentation****5.22.2.1 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:**

- ← *pKey* Key to search for
- ← *pBuf* Pointer to the array to sort
- ← *num* number of elements
- ← *size* size of one element
- ← *compare* 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:**

*Pointer* to found element or NULL

**5.22.2.2 EXT\_DECL UINT8\* vos\_memAlloc (UINT32 *size*)**

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

**Parameters:**

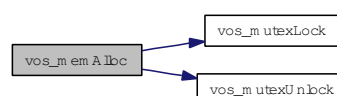
- ← *size* Size of requested block

**Return values:**

*Pointer* to memory area

*NULL* if no memory available

Here is the call graph for this function:



**5.22.2.3** EXT\_DECL VOS\_ERR\_T vos\_memCount (UINT32 \* *pAllocatedMemory*, UINT32 \* *pFreeMemory*, UINT32 \* *pMinFree*, UINT32 \* *pNumAllocBlocks*, UINT32 \* *pNumAllocErr*, UINT32 \* *pNumFreeErr*, UINT32 *blockSize*[VOS\_MEM\_NBBLOCKSIZES], UINT32 *usedBlockSize*[VOS\_MEM\_NBBLOCKSIZES])

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

**Parameters:**

- *pAllocatedMemory* Pointer to allocated memory size
- *pFreeMemory* Pointer to free memory size
- *pMinFree* Pointer to minimal free memory size in statistics interval
- *pNumAllocBlocks* Pointer to number of allocated memory blocks
- *pNumAllocErr* Pointer to number of allocation errors
- *pNumFreeErr* Pointer to number of free errors
- *blockSize* Pointer to list of memory block sizes
- *usedBlockSize* Pointer to list of used memory blocks

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_INIT\_ERR* module not initialised

**5.22.2.4** 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:**

- ← *pMemoryArea* Pointer to memory area used

Here is the call graph for this function:



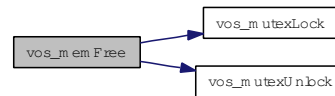
**5.22.2.5** EXT\_DECL void vos\_memFree (void \* *pMemBlock*)

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

**Parameters:**

- ← *pMemBlock* Pointer to memory block to be freed

Here is the call graph for this function:



#### 5.22.2.6 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.

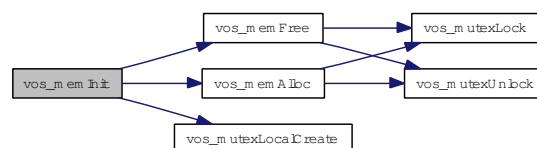
##### Parameters:

- ← *pMemoryArea* Pointer to memory area to use
- ← *size* Size of provided memory area
- ← *fragMem* Pointer to list of preallocated block sizes, used to fragment memory for large blocks

##### Return values:

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** parameter out of range/invalid
- VOS\_MEM\_ERR** no memory available
- VOS\_MUTEX\_ERR** no mutex available

Here is the call graph for this function:



#### 5.22.2.7 VOS\_ERR\_T vos\_mutexLocalCreate (struct VOS\_MUTEX \* *pMutex*)

Create a recursive mutex.

Fill in a mutex handle. The mutex storage must be already allocated.

##### Parameters:

- *pMutex* Pointer to mutex handle

##### Return values:

- VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised  
**VOS\_PARAM\_ERR** pMutex == NULL  
**VOS\_MUTEX\_ERR** no mutex available

#### 5.22.2.8 void vos\_mutexLocalDelete (struct VOS\_MUTEX \*pMutex)

Delete a mutex.

Release the resources taken by the mutex.

##### Parameters:

← **pMutex** Pointer to mutex struct

#### 5.22.2.9 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:

↔ **pBuf** Pointer to the array to sort  
 ← **num** number of elements  
 ← **size** size of one element  
 ← **compare** 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:

**none**

#### 5.22.2.10 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:

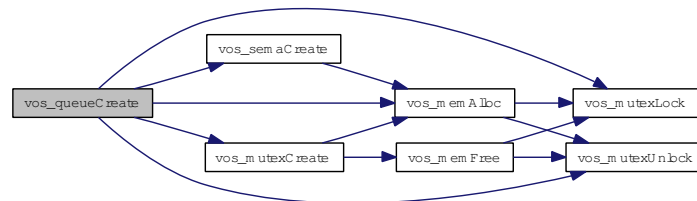
← **queueType** Define queue type (1 = FIFO, 2 = LIFO, 3 = PRIO)  
 ← **maxNoOfMsg** Maximum number of messages  
 → **pQueueHandle** Handle of created queue

##### Return values:

**VOS\_NO\_ERR** no error  
**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle  
**VOS\_PARAM\_ERR** parameter out of range/invalid  
**VOS\_INIT\_ERR** not supported  
**VOS\_QUEUE\_ERR** error creating queue

Here is the call graph for this function:



#### 5.22.2.11 EXT\_DECL VOS\_ERR\_T vos\_queueDestroy (VOS\_QUEUE\_T queueHandle)

Destroy a message queue.

Free all resources used by this queue

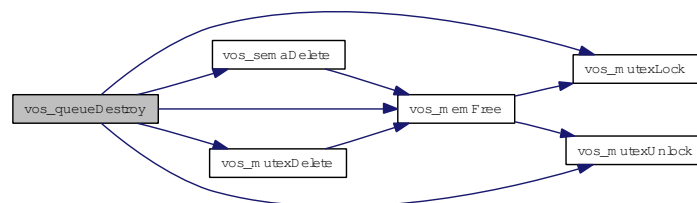
##### Parameters:

← *queueHandle* Queue handle

##### Return values:

**VOS\_NO\_ERR** no error  
**VOS\_INIT\_ERR** module not initialised  
**VOS\_NOINIT\_ERR** invalid handle  
**VOS\_PARAM\_ERR** parameter out of range/invalid

Here is the call graph for this function:



#### 5.22.2.12 EXT\_DECL VOS\_ERR\_T vos\_queueReceive (VOS\_QUEUE\_T queueHandle, UINT8 \*\*ppData, UINT32 \*pSize, UINT32 usTimeout)

Get a message.



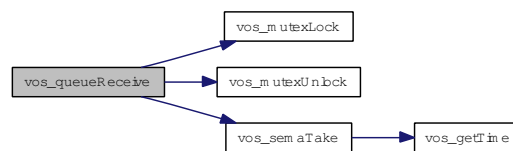
**Parameters:**

- ← *queueHandle* Queue handle
- *ppData* Pointer to data pointer to be received
- *pSize* Size of receive data
- ← *usTimeout* Maximum time to wait for a message (in usec)

**Return values:**

- VOSNO\_ERR** no error
- VOS\_INIT\_ERR** module not initialised
- VOS\_NOINIT\_ERR** invalid handle
- VOS\_PARAM\_ERR** parameter out of range/invalid
- VOS\_QUEUE\_ERR** queue is empty

Here is the call graph for this function:



### 5.22.2.13 EXT\_DECL VOS\_ERR\_T vos\_queueSend (VOS\_QUEUE\_T queueHandle, UINT8 \* pData, UINT32 size)

Send a message.

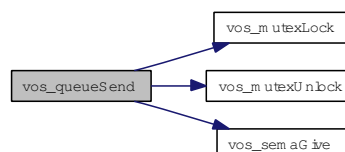
**Parameters:**

- ← *queueHandle* Queue handle
- ← *pData* Pointer to data to be sent
- ← *size* Size of data to be sent

**Return values:**

- VOS\_NO\_ERR** no error
- VOS\_INIT\_ERR** module not initialised
- VOS\_NOINIT\_ERR** invalid handle
- VOS\_PARAM\_ERR** parameter out of range/invalid
- VOS\_INIT\_ERR** not supported
- VOS\_QUEUE\_ERR** error creating queue

Here is the call graph for this function:



**5.22.2.14 EXT\_DECL void vos\_strncpy (CHAR8 \* *pStrDst*, const CHAR8 \* *pStrSrc*, UINT32 *count*)**

String copy with length limitation.

**Parameters:**

- ← *pStrDst* Destination string
- ← *pStrSrc* Null terminated string to copy
- ← *count* Maximum number of characters to copy

**Return values:**

*none*

**5.22.2.15 EXT\_DECL INT32 vos\_strnicmp (const CHAR8 \* *pStr1*, const CHAR8 \* *pStr2*, UINT32 *count*)**

Case insensitive string compare.

**Parameters:**

- ← *pStr1* Null terminated string to compare
- ← *pStr2* Null terminated string to compare
- ← *count* Maximum number of characters to compare

**Return values:**

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



## Functions

- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_memInit](#) (UINT8 \*pMemoryArea, UINT32 size, const UINT32 fragMem[VOS\_MEM\_NBLOCKSIZES])  
*Initialize the memory unit.*
- EXT\_DECL void [vos\\_memDelete](#) (UINT8 \*pMemoryArea)  
*Delete the memory area.*
- EXT\_DECL UINT8 \* [vos\\_memAlloc](#) (UINT32 size)  
*Allocate a block of memory (from memory area above).*
- EXT\_DECL void [vos\\_memFree](#) (void \*pMemBlock)  
*Deallocate a block of memory (from memory area above).*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_memCount](#) (UINT32 \*pAllocatedMemory, UINT32 \*pFreeMemory, UINT32 \*pMinFree, UINT32 \*pNumAllocBlocks, UINT32 \*pNumAllocErr, UINT32 \*pNumFreeErr, UINT32 blockSize[VOS\_MEM\_NBLOCKSIZES], UINT32 usedBlockSize[VOS\_MEM\_NBLOCKSIZES])  
*Return used and available memory (of memory area above).*
- EXT\_DECL void [vos\\_qsort](#) (void \*pBuf, UINT32 num, UINT32 size, int(\*compare)(const void \*, const void \*))  
*Sort an array.*
- EXT\_DECL void \* [vos\\_bsearch](#) (const void \*pKey, const void \*pBuf, UINT32 num, UINT32 size, int(\*compare)(const void \*, const void \*))  
*Binary search in a sorted array.*
- EXT\_DECL INT32 [vos\\_strnicmp](#) (const CHAR8 \*pStr1, const CHAR8 \*pStr2, UINT32 count)  
*Case insensitive string compare.*
- EXT\_DECL void [vos\\_strncpy](#) (CHAR8 \*pStr1, const CHAR8 \*pStr2, UINT32 count)  
*String copy 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.*

### 5.23.1 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:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

[vos\\_mem.h](#) 951 2013-06-13 13:56:42Z 97025

### 5.23.2 Define Documentation

#### 5.23.2.1 #define VOS\_MEM\_BLOCKSIZEs

**Value:**

```
{32, 48, 128, 180, 256, 512, 1024, 1480, 2048, \
    4096, 11520, 16384, 32768, 65536, 131072}
```

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.

#### 5.23.2.2 #define VOS\_MEM\_PREALLOCATE {0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 1, 0, 0, 0}

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.

### 5.23.3 Function Documentation

#### 5.23.3.1 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:**

← *pKey* Key to search for

- ← *pBuf* Pointer to the array to sort
- ← *num* number of elements
- ← *size* size of one element
- ← *compare* 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:**

*Pointer* to found element or NULL

### 5.23.3.2 EXT\_DECL UINT8\* vos\_memAlloc (UINT32 size)

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

**Parameters:**

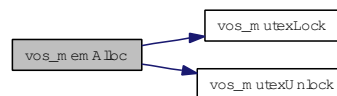
- ← *size* Size of requested block

**Return values:**

*Pointer* to memory area

NULL if no memory available

Here is the call graph for this function:



### 5.23.3.3 EXT\_DECL VOS\_ERR\_T vos\_memCount (UINT32 \*pAllocatedMemory, UINT32 \*pFreeMemory, UINT32 \*pMinFree, UINT32 \*pNumAllocBlocks, UINT32 \*pNumAllocErr, UINT32 \*pNumFreeErr, UINT32 blockSize[VOS\_MEM\_NBLOCKSIZES], UINT32 usedBlockSize[VOS\_MEM\_NBLOCKSIZES])

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

**Parameters:**

- *pAllocatedMemory* Pointer to allocated memory size
- *pFreeMemory* Pointer to free memory size
- *pMinFree* Pointer to minimal free memory size in statistics interval
- *pNumAllocBlocks* Pointer to number of allocated memory blocks
- *pNumAllocErr* Pointer to number of allocation errors
- *pNumFreeErr* Pointer to number of free errors
- *blockSize* Pointer to list of memory block sizes
- *usedBlockSize* Pointer to list of used memory blocks

**Return values:**

VOS\_NO\_ERR no error

VOS\_INIT\_ERR module not initialised

#### 5.23.3.4 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:

← *pMemoryArea* 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:

← *pMemoryArea* Pointer to memory area used

Here is the call graph for this function:



#### 5.23.3.5 EXT\_DECL void vos\_memFree (void \* *pMemBlock*)

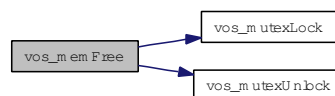
Deallocate a block of memory (from memory area above).

##### Parameters:

← *pMemBlock* Pointer to memory block to be freed

← *pMemBlock* Pointer to memory block to be freed

Here is the call graph for this function:



#### 5.23.3.6 EXT\_DECL VOS\_ERR\_T vos\_memInit (UINT8 \* *pMemoryArea*, UINT32 *size*, const UINT32 *fragMem*[VOS\_MEM\_NBLOCKSIZES])

Initialize the memory unit.

Init a supplied block of memory and prepare it for use with vos\_alloc and vos\_dealloc. The used block sizes can be supplied and will be preallocated.

##### Parameters:

← *pMemoryArea* Pointer to memory area to use

← *size* Size of provided memory area

← **fragMem** Pointer to list of preallocate block sizes, used to fragment memory for large blocks

#### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_MEM\_ERR** no memory available

Init a supplied block of memory and prepare it for use with `vos_memAlloc` and `vos_memFree`. The used block sizes can be supplied and will be preallocated. If half of the overall size of the requested memory area would be pre-allocated, either by the default pre-allocation table or a provided one, no pre-allocation takes place.

#### Parameters:

← **pMemoryArea** Pointer to memory area to use

← **size** Size of provided memory area

← **fragMem** Pointer to list of preallocated block sizes, used to fragment memory for large blocks

#### Return values:

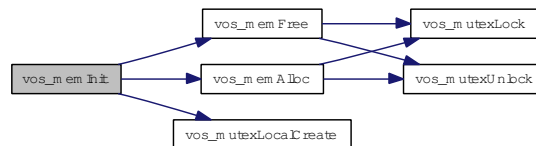
**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_MEM\_ERR** no memory available

**VOS\_MUTEX\_ERR** no mutex available

Here is the call graph for this function:



### 5.23.3.7 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:

↔ **pBuf** Pointer to the array to sort

← **num** number of elements

← **size** size of one element

← **compare** 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:

*none*



### 5.23.3.8 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

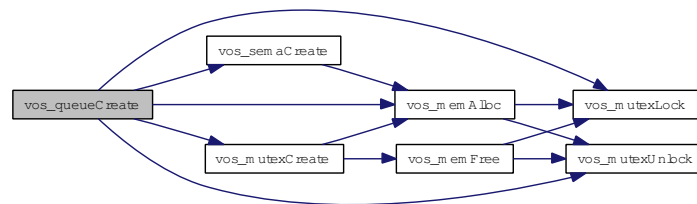
#### Parameters:

- ← *queueType* Define queue type (1 = FIFO, 2 = LIFO, 3 = PRIO)
- ← *maxNoOfMsg* Maximum number of messages
- *pQueueHandle* Handle of created queue

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_INIT\_ERR** module not initialised
- VOS\_NOINIT\_ERR** invalid handle
- VOS\_PARAM\_ERR** parameter out of range/invalid
- VOS\_INIT\_ERR** not supported
- VOS\_QUEUE\_ERR** error creating queue

Here is the call graph for this function:



### 5.23.3.9 EXT\_DECL VOS\_ERR\_T vos\_queueDestroy (VOS\_QUEUE\_T *queueHandle*)

Destroy a message queue.

Free all resources used by this queue

#### Parameters:

- ← *queueHandle* Queue handle

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_INIT\_ERR** module not initialised
- VOS\_NOINIT\_ERR** invalid handle
- VOS\_PARAM\_ERR** parameter out of range/invalid

Free all resources used by this queue

**Parameters:**

← *queueHandle* Queue handle

**Return values:**

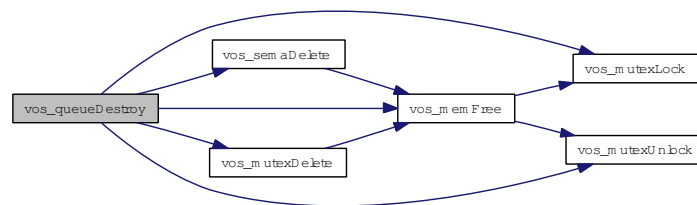
**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

**VOS\_PARAM\_ERR** parameter out of range/invalid

Here is the call graph for this function:



### 5.23.3.10 EXT\_DECL VOS\_ERR\_T vos\_queueReceive (VOS\_QUEUE\_T queueHandle, UINT8 \*\*ppData, UINT32 \*pSize, UINT32 usTimeout)

Get a message.

**Parameters:**

← *queueHandle* Queue handle

→ *ppData* Pointer to data pointer to be received

→ *pSize* Size of receive data

← *usTimeout* Maximum time to wait for a message (in usec)

**Return values:**

**VOSNO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

**VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_QUEUE\_ERR** queue is empty

**Parameters:**

← *queueHandle* Queue handle

→ *ppData* Pointer to data pointer to be received

→ *pSize* Size of receive data

← *usTimeout* Maximum time to wait for a message (in usec)

**Return values:**

**VOSNO\_ERR** no error

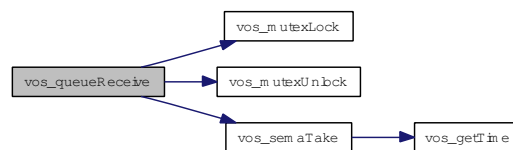
**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

**VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_QUEUE\_ERR** queue is empty

Here is the call graph for this function:



### 5.23.3.11 EXT\_DECL VOS\_ERR\_T vos\_queueSend (VOS\_QUEUE\_T *queueHandle*, UINT8 \**pData*, UINT32 *size*)

Send a message.

**Parameters:**

← *queueHandle* Queue handle

← *pData* Pointer to data to be sent

← *size* Size of data to be sent

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

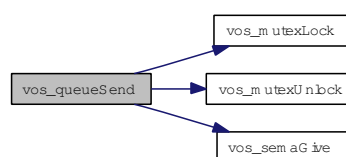
**VOS\_NOINIT\_ERR** invalid handle

**VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_INIT\_ERR** not supported

**VOS\_QUEUE\_ERR** error creating queue

Here is the call graph for this function:



**5.23.3.12 EXT\_DECL void vos\_strncpy (CHAR8 \* *pStrDst*, const CHAR8 \* *pStrSrc*, UINT32 *count*)**

String copy with length limitation.

**Parameters:**

- ← *pStrDst* Destination string
- ← *pStrSrc* Null terminated string to copy
- ← *count* Maximum number of characters to copy

**Return values:**

*none*

**5.23.3.13 EXT\_DECL INT32 vos\_strnicmp (const CHAR8 \* *pStr1*, const CHAR8 \* *pStr2*, UINT32 *count*)**

Case insensitive string compare.

**Parameters:**

- ← *pStr1* Null terminated string to compare
- ← *pStr2* Null terminated string to compare
- ← *count* Maximum number of characters to compare

**Return values:**

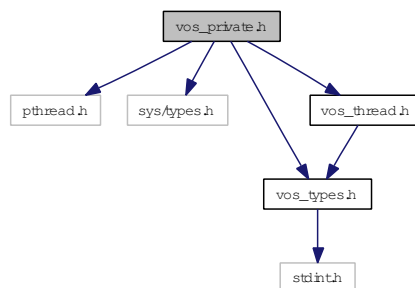
- 0* - equal
- <0* - string1 less than string 2
- >0* - string 1 greater than string 2

## 5.24 vos\_private.h File Reference

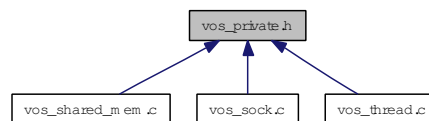
Private definitions for the OS abstraction layer.

```
#include <pthread.h>
#include <sys/types.h>
#include "vos_types.h"
#include "vos_thread.h"
```

Include dependency graph for posix/vos\_private.h:



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



### Functions

- [VOS\\_ERR\\_T vos\\_mutexLocalCreate](#) (struct VOS\_MUTEX \*pMutex)  
*Create a recursive mutex.*
- void [vos\\_mutexLocalDelete](#) (struct VOS\_MUTEX \*pMutex)  
*Delete a mutex.*

### 5.24.1 Detailed Description

Private definitions for the OS abstraction layer.

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

vos\_private.h 951 2013-06-13 13:56:42Z 97025

**5.24.2 Function Documentation****5.24.2.1 VOS\_ERR\_T vos\_mutexLocalCreate (struct VOS\_MUTEX \* *pMutex*)**

Create a recursive mutex.

Fill in a mutex handle. The mutex storage must be already allocated.

**Parameters:**

→ *pMutex* Pointer to mutex handle

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_PARAM\_ERR** *pMutex* == NULL

**VOS\_MUTEX\_ERR** no mutex available

**5.24.2.2 void vos\_mutexLocalDelete (struct VOS\_MUTEX \* *pMutex*)**

Delete a mutex.

Release the resources taken by the mutex.

**Parameters:**

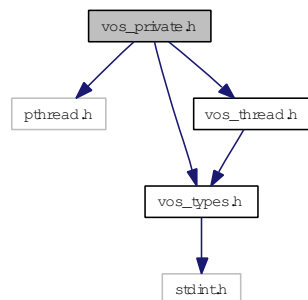
← *pMutex* Pointer to mutex struct

## 5.25 vos\_private.h File Reference

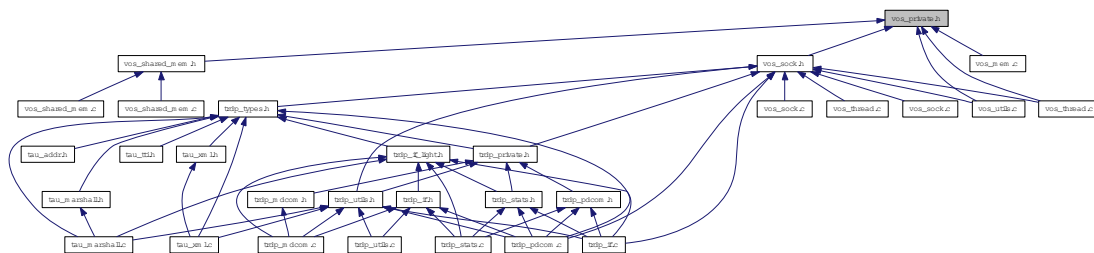
Private definitions for the OS abstraction layer.

```
#include <pthread.h>
#include "vos_types.h"
#include "vos_thread.h"
```

Include dependency graph for windows/vos\_private.h:



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



### Functions

- [VOS\\_ERR\\_T vos\\_mutexLocalCreate](#) (struct VOS\_MUTEX \*pMutex)  
*Create a recursive mutex.*
- void [vos\\_mutexLocalDelete](#) (struct VOS\_MUTEX \*pMutex)  
*Delete a mutex.*

### 5.25.1 Detailed Description

Private definitions for the OS abstraction layer.

#### Note:

Project: TCNOpen TRDP prototype stack

**Author:**

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

vos\_private.h 1012 2013-07-04 06:43:02Z cschneider

## 5.25.2 Function Documentation

### 5.25.2.1 VOS\_ERR\_T vos\_mutexLocalCreate (struct VOS\_MUTEX \* *pMutex*)

Create a recursive mutex.

Fill in a mutex handle. The mutex storage must be already allocated.

**Parameters:**

→ *pMutex* Pointer to mutex handle

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_PARAM\_ERR** *pMutex* == NULL

**VOS\_MUTEX\_ERR** no mutex available

### 5.25.2.2 void vos\_mutexLocalDelete (struct VOS\_MUTEX \* *pMutex*)

Delete a mutex.

Release the resources taken by the mutex.

**Parameters:**

← *pMutex* Pointer to mutex struct

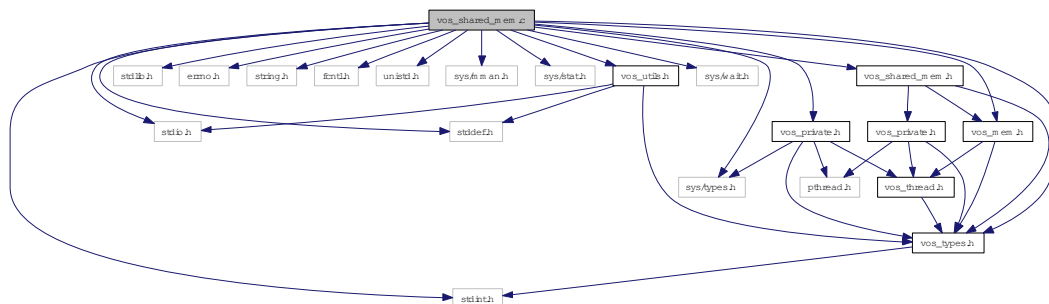


## 5.26 vos\_shared\_mem.c File Reference

Shared Memory functions.

```
#include <stdio.h>
#include <stddef.h>
#include <stdint.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include "vos_types.h"
#include "vos_mem.h"
#include "vos_utils.h"
#include "vos_private.h"
#include "vos_shared_mem.h"
```

Include dependency graph for posix/vos\_shared\_mem.c:



## Functions

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sharedOpen](#) (const CHAR8 \*pKey, VOS\_SHRD\_T \*pHandle, UINT8 \*\*ppMemoryArea, UINT32 \*pSize)  
*Create a shared memory area or attach to existing one.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sharedClose](#) (VOS\_SHRD\_T handle, const UINT8 \*pMemoryArea)  
*Close connection to the shared memory area.*

### 5.26.1 Detailed Description

Shared Memory functions.

OS abstraction of Shared memory access and control

**Note:**

Project: TCNOpen TRDP prototype stack

**Author:**

Kazumasa Aiba, TOSHIBA

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright TOSHIBA, Japan, 2013.

**Id**

[vos\\_mem.h](#) 282 2013-01-11 07:08:44Z 97029

### 5.26.2 Function Documentation

#### 5.26.2.1 EXT\_DECL VOS\_ERR\_T vos\_sharedClose (VOS\_SHRD\_T *handle*, const UINT8 \* *pMemoryArea*)

Close connection to the shared memory area.

If the area was created by the calling process, the area will be closed (freed). If the area was attached, it will be detached. This function is not available in each target implementation.

**Parameters:**

- ← *handle* Returned handle
- ← *pMemoryArea* Pointer to memory area

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_MEM\_ERR* no memory available

#### 5.26.2.2 EXT\_DECL VOS\_ERR\_T vos\_sharedOpen (const CHAR8 \* *pKey*, VOS\_SHRD\_T \* *pHandle*, UINT8 \*\* *ppMemoryArea*, UINT32 \* *pSize*)

Create a shared memory area or attach to existing one.

The first call with the a specified key will create a shared memory area with the supplied size and will return a handle and a pointer to that area. If the area already exists, the area will be attached. This function is not available in each target implementation.

**Parameters:**

- ← *pKey* Unique identifier (file name)

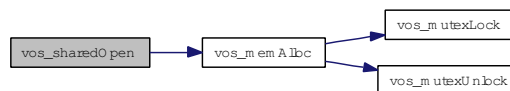
- *pHandle* Pointer to returned handle
- *ppMemoryArea* Pointer to pointer to memory area
- ↔ *pSize* Pointer to size of area to allocate, on return actual size after attach

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_MEM\_ERR* no memory available

Here is the call graph for this function:

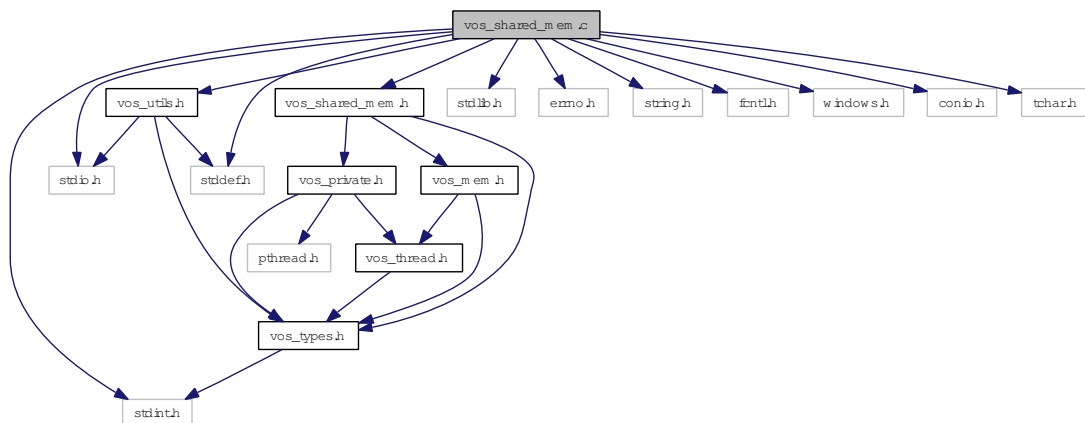


## 5.27 vos\_shared\_mem.c File Reference

Shared Memory functions.

```
#include <stdio.h>
#include <stddef.h>
#include <stdint.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include "vos_shared_mem.h"
#include "vos_utils.h"
#include <windows.h>
#include <conio.h>
#include <tchar.h>
```

Include dependency graph for windows/vos\_shared\_mem.c:



## Functions

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sharedOpen](#) (const CHAR8 \*pKey, VOS\_SHRD\_T \*pHandle, UINT8 \*\*ppMemoryArea, UINT32 \*pSize)

*Create a shared memory area or attach to existing one.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sharedClose](#) (VOS\_SHRD\_T handle, const UINT8 \*pMemoryArea)

*Close connection to the shared memory area.*

### 5.27.1 Detailed Description

Shared Memory functions.

OS abstraction of Shared memory access and control

**Note:**

Project: TCNOpen TRDP prototype stack

**Author:**

Kazumasa Aiba, TOSHIBA

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright TOSHIBA, Japan, 2013.

**Id**

[vos\\_mem.h](#) 282 2013-01-11 07:08:44Z 97029

### 5.27.2 Function Documentation

#### 5.27.2.1 EXT\_DECL VOS\_ERR\_T vos\_sharedClose (VOS\_SHRD\_T *handle*, const UINT8 \* *pMemoryArea*)

Close connection to the shared memory area.

If the area was created by the calling process, the area will be closed (freed). If the area was attached, it will be detached. This function is not available in each target implementation.

**Parameters:**

← *handle* Returned handle

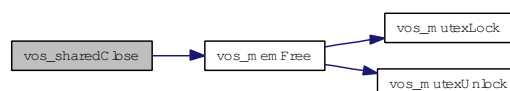
← *pMemoryArea* Pointer to memory area

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_MEM\_ERR* no memory available

Here is the call graph for this function:



#### 5.27.2.2 EXT\_DECL VOS\_ERR\_T vos\_sharedOpen (const CHAR8 \* *pKey*, VOS\_SHRD\_T \* *pHandle*, UINT8 \*\* *ppMemoryArea*, UINT32 \* *pSize*)

Create a shared memory area or attach to existing one.

The first call with the a specified key will create a shared memory area with the supplied size and will return a handle and a pointer to that area. If the area already exists, the area will be opened. This function is not available in each target implementation.

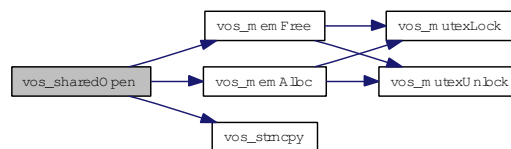
#### Parameters:

- ← ***pKey*** Unique identifier (file name)
- ***pHandle*** Pointer to returned handle
- ***ppMemoryArea*** Pointer to pointer to memory area
- ↔ ***pSize*** Pointer to size of area to allocate, on return actual size after attach. Independent from actual value, always multiples of page size (4k) are allocated

#### Return values:

- VOS\_NO\_ERR*** no error
- VOS\_MEM\_ERR*** no memory available

Here is the call graph for this function:

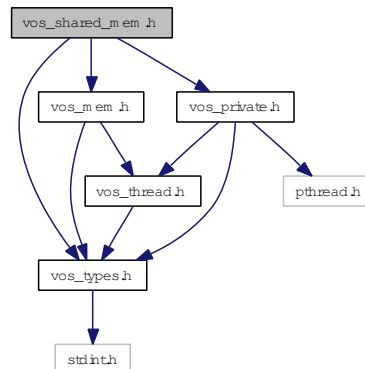


## 5.28 vos\_shared\_mem.h File Reference

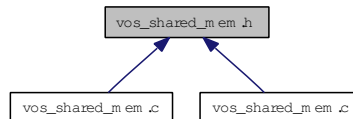
Shared Memory functions for OS abstraction.

```
#include "vos_types.h"
#include "vos_mem.h"
#include "vos_private.h"
```

Include dependency graph for vos\_shared\_mem.h:



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



### Functions

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sharedOpen](#) (const CHAR8 \*pKey, VOS\_SHRD\_T \*pHandle, UINT8 \*\*ppMemoryArea, UINT32 \*pSize)  
*Create a shared memory area or attach to existing one.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sharedClose](#) (VOS\_SHRD\_T handle, const UINT8 \*pMemoryArea)  
*Close connection to the shared memory area.*

### 5.28.1 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:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright TOSHIBA, Japan, 2013.

**Id**

[vos\\_mem.h](#) 282 2013-01-11 07:08:44Z 97029

**5.28.2 Function Documentation****5.28.2.1 EXT\_DECL VOS\_ERR\_T vos\_sharedClose (VOS\_SHRD\_T *handle*, const UINT8 \* *pMemoryArea*)**

Close connection to the shared memory area.

If the area was created by the calling process, the area will be closed (freed). If the area was attached, it will be detached. This function is not available in each target implementation.

**Parameters:**

← *handle* Returned handle

← *pMemoryArea* Pointer to memory area

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_MEM\_ERR** no memory available

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:**

← *handle* Returned handle

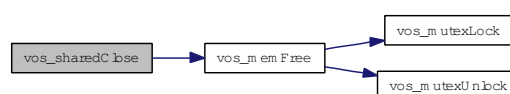
← *pMemoryArea* Pointer to memory area

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_MEM\_ERR** no memory available

Here is the call graph for this function:





### 5.28.2.2 EXT\_DECL VOS\_ERR\_T vos\_sharedOpen (const CHAR8 \* *pKey*, VOS\_SHRD\_T \* *pHandle*, UINT8 \*\* *ppMemoryArea*, UINT32 \* *pSize*)

Create a shared memory area or attach to existing one.

The first call with the a specified key will create a shared memory area with the supplied size and will return a handle and a pointer to that area. If the area already exists, the area will be opened. This function is not available in each target implementation.

#### Parameters:

- ← *pKey* Unique identifier (file name)
- *pHandle* Pointer to returned handle
- *ppMemoryArea* Pointer to pointer to memory area
- ↔ *pSize* Pointer to size of area to allocate, on return actual size after attach

#### Return values:

- VOS\_NO\_ERR* no error
- VOS\_MEM\_ERR* no memory available

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 attached. This function is not available in each target implementation.

#### Parameters:

- ← *pKey* Unique identifier (file name)
- *pHandle* Pointer to returned handle
- *ppMemoryArea* Pointer to pointer to memory area
- ↔ *pSize* Pointer to size of area to allocate, on return actual size after attach

#### Return values:

- VOS\_NO\_ERR* no error
- VOS\_MEM\_ERR* no memory available

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:

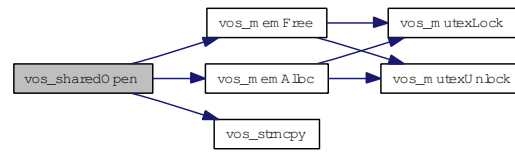
- ← *pKey* Unique identifier (file name)
- *pHandle* Pointer to returned handle
- *ppMemoryArea* Pointer to pointer to memory area
- ↔ *pSize* Pointer to size of area to allocate, on return actual size after attach. Independent from actual value, always multiples of page size (4k) are allocated

#### Return values:

- VOS\_NO\_ERR* no error

***VOS\_MEM\_ERR*** no memory available

Here is the call graph for this function:

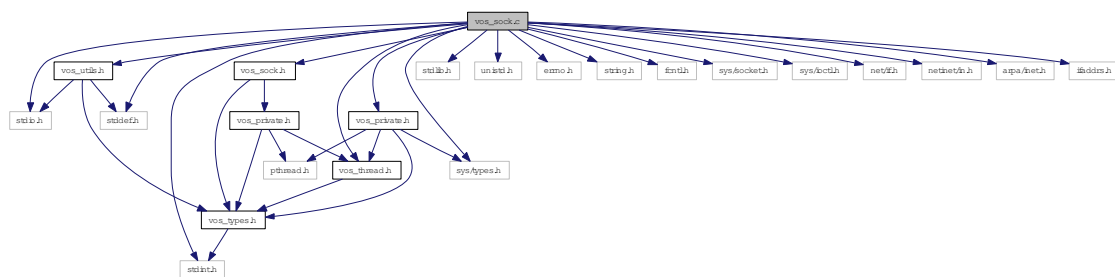


## 5.29 vos\_sock.c File Reference

## Socket functions.

```
#include <stdio.h>
#include <stddef.h>
#include <stdint.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <net/if.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <ifaddrs.h>
#include "vos_utils.h"
#include "vos_sock.h"
#include "vos_thread.h"
#include "vos_private.h"
```

Include dependency graph for posix/vos\_sock.c:



## Functions

- **BOOL** `vos_getMacAddress` (UINT8 \*pMacAddr, const char \*pIfName)  
*Get the MAC address for a named interface.*
- **VOS\_ERR\_T** `vos_sockSetBuffer` (INT32 sock)  
*Enlarge send and receive buffers to TRDP\_SOCKETBUF\_SIZE if necessary.*

- EXT\_DECL UINT16 [vos\\_htons](#) (UINT16 val)  
*Byte swapping.*
- EXT\_DECL UINT16 [vos\\_ntohs](#) (UINT16 val)  
*Byte swapping 2 Bytes.*
- EXT\_DECL UINT32 [vos\\_htonl](#) (UINT32 val)  
*Byte swapping 4 Bytes.*
- EXT\_DECL UINT32 [vos\\_ntohl](#) (UINT32 val)  
*Byte swapping 4 Bytes.*
- EXT\_DECL UINT32 [vos\\_dottedIP](#) (const CHAR8 \*pDottedIP)  
*Convert IP address from dotted dec.*
- EXT\_DECL const CHAR8 \* [vos\\_ipDotted](#) (UINT32 ipAddress)  
*Convert IP address to dotted dec.*
- EXT\_DECL BOOL [vos\\_isMulticast](#) (UINT32 ipAddress)  
*Check if the supplied address is a multicast group address.*
- EXT\_DECL INT32 [vos\\_select](#) (INT32 highDesc, VOS\_FDS\_T \*pReadableFD, VOS\_FDS\_T \*pWritableFD, VOS\_FDS\_T \*pErrorFD, [VOS\\_TIME\\_T](#) \*pTimeOut)  
*select function.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_getInterfaces](#) (UINT32 \*pAddrCnt, VOS\_IF\_REC\_T ifAddrs[ ])
  - Get a list of interface addresses The caller has to provide an array of interface records to be filled.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockInit](#) (void)  
*Initialize the socket library.*
- EXT\_DECL void [vos\\_sockTerm](#) (void)  
*De-Initialize the socket library.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockGetMAC](#) (UINT8 pMAC[VOS\_MAC\_SIZE])  
*Return the MAC address of the default adapter.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockOpenUDP](#) (INT32 \*pSock, const [VOS\\_SOCKET\\_OPT\\_T](#) \*pOptions)  
*Create an UDP socket.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockOpenTCP](#) (INT32 \*pSock, const [VOS\\_SOCKET\\_OPT\\_T](#) \*pOptions)  
*Create a TCP socket.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockClose](#) (INT32 sock)  
*Close a socket.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockSetOptions](#) (INT32 sock, const [VOS\\_SOCKET\\_OPT\\_T](#) \*pOptions)

*Set socket options.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockJoinMC](#) (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)

*Join a multicast group.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockLeaveMC](#) (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)

*Leave a multicast group.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockSendUDP](#) (INT32 sock, const UINT8 \*pBuffer, UINT32 \*pSize, UINT32 ipAddress, UINT16 port)

*Send UDP data.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockReceiveUDP](#) (INT32 sock, UINT8 \*pBuffer, UINT32 \*pSize, UINT32 \*pSrcIPAddr, UINT16 \*pSrcIPPort, UINT32 \*pDstIPAddr, BOOL peek)

*Receive UDP data.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockBind](#) (INT32 sock, UINT32 ipAddress, UINT16 port)

*Bind a socket to an address and port.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockListen](#) (INT32 sock, UINT32 backlog)

*Listen for incoming connections.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockAccept](#) (INT32 sock, INT32 \*pSock, UINT32 \*pIPAddr, UINT16 \*pPort)

*Accept an incoming TCP connection.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockConnect](#) (INT32 sock, UINT32 ipAddress, UINT16 port)

*Open a TCP connection.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockSendTCP](#) (INT32 sock, const UINT8 \*pBuffer, UINT32 \*pSize)

*Send TCP data.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockReceiveTCP](#) (INT32 sock, UINT8 \*pBuffer, UINT32 \*pSize)

*Receive TCP data.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockSetMulticastIf](#) (INT32 sock, UINT32 mcIfAddress)

*Set Using Multicast I/F.*

### 5.29.1 Detailed Description

Socket functions.

OS abstraction of IP socket functions for UDP and TCP

#### Note:

Project: TCNOpen TRDP prototype stack

**Author:**

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012-2013.

**Id**

vos\_sock.c 1000 2013-07-02 08:53:28Z bloehr

**5.29.2 Function Documentation****5.29.2.1 EXT\_DECL UINT32 vos\_dottedIP (const CHAR8 \* *pDottedIP*)**

Convert IP address from dotted dec.

to !host! endianness

**Parameters:**

← *pDottedIP* IP address as dotted decimal.

**Return values:**

*address* in UINT32 in host endianness

Here is the call graph for this function:

**5.29.2.2 EXT\_DECL VOS\_ERR\_T vos\_getInterfaces (UINT32 \* *pAddrCnt*, VOS\_IF\_REC\_T *ifAddrs*[ ])**

Get a list of interface addresses The caller has to provide an array of interface records to be filled.

**Parameters:**

↔ *pAddrCnt* in: pointer to array size of interface record out: pointer to number of interface records read

↔ *ifAddrs* array of interface records

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* pMAC == NULL

Here is the call graph for this function:



**5.29.2.3** `BOOL vos_getMacAddress (UINT8 * pMacAddr, const char * pIfName)`

Get the MAC address for a named interface.

**Parameters:**

- *pMacAddr* pointer to array of MAC address to return
- ← *pIfName* pointer to interface name

**Return values:**

*TRUE* if successfull

**5.29.2.4** `EXT_DECL UINT32 vos_htonl (UINT32 val)`

Byte swapping 4 Bytes.

**Parameters:**

- ← *val* Initial value.

**Return values:**

*swapped* value

**5.29.2.5** `EXT_DECL UINT16 vos_htons (UINT16 val)`

Byte swapping.

Byte swapping 2 Bytes.

**Parameters:**

- ← *val* Initial value.

**Return values:**

*swapped* value

**5.29.2.6** `EXT_DECL const CHAR8* vos_ipDotted (UINT32 ipAddress)`

Convert IP address to dotted dec.

from !host! endianness.

**Parameters:**

- ← *ipAddress* address in UINT32 in host endianness

**Return values:**

*IP* address as dotted decimal.

**5.29.2.7 EXT\_DECL BOOL vos\_isMulticast (UINT32 *ipAddress*)**

Check if the supplied address is a multicast group address.

**Parameters:**

← *ipAddress* IP address to check.

**Return values:**

*TRUE* address is multicast

*FALSE* address is not a multicast address

**5.29.2.8 EXT\_DECL UINT32 vos\_ntohl (UINT32 *val*)**

Byte swapping 4 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

**5.29.2.9 EXT\_DECL UINT16 vos\_ntohs (UINT16 *val*)**

Byte swapping 2 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

**5.29.2.10 EXT\_DECL INT32 vos\_select (INT32 *highDesc*, VOS\_FDS\_T \**pReadableFD*, VOS\_FDS\_T \**pWriteableFD*, VOS\_FDS\_T \**pErrorFD*, VOS\_TIME\_T \**pTimeOut*)**

select function.

Set the ready sockets in the supplied sets. Note: Some target systems might define this function as NOP.

**Parameters:**

← *highDesc* max. socket descriptor + 1

↔ *pReadableFD* pointer to readable socket set

↔ *pWriteableFD* pointer to writeable socket set

↔ *pErrorFD* pointer to error socket set

← *pTimeOut* pointer to time out value

**Return values:**

*number* of ready file descriptors



### 5.29.2.11 EXT\_DECL VOS\_ERR\_T vos\_sockAccept (INT32 *sock*, INT32 \* *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.

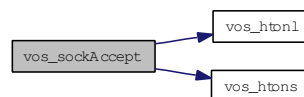
#### Parameters:

- ← *sock* Socket descriptor
- *pSock* Pointer to socket descriptor, on exit new socket
- *pIPAddress* source IP to receive on, 0 for any
- *pPort* port to receive on, 20548 for PD

#### Return values:

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* NULL parameter, parameter error
- VOS\_UNKNOWN\_ERR* sock descriptor unknown error

Here is the call graph for this function:



### 5.29.2.12 EXT\_DECL VOS\_ERR\_T vos\_sockBind (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Bind a socket to an address and port.

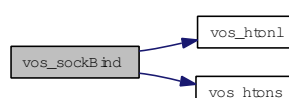
#### Parameters:

- ← *sock* socket descriptor
- ← *ipAddress* source IP to receive on, 0 for any
- ← *port* port to receive on, 20548 for PD

#### Return values:

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* sock descriptor unknown, parameter error
- VOS\_IO\_ERR* Input/Output error
- VOS\_MEM\_ERR* resource error

Here is the call graph for this function:



### 5.29.2.13 EXT\_DECL VOS\_ERR\_T vos\_sockClose (INT32 *sock*)

Close a socket.

Release any resources acquired by this socket

#### Parameters:

← *sock* socket descriptor

#### Return values:

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* sock descriptor unknown

### 5.29.2.14 EXT\_DECL VOS\_ERR\_T vos\_sockConnect (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Open a TCP connection.

#### Parameters:

← *sock* socket descriptor

← *ipAddress* destination IP

← *port* destination port

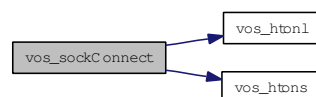
#### Return values:

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* sock descriptor unknown, parameter error

*VOS\_IO\_ERR* Input/Output error

Here is the call graph for this function:



### 5.29.2.15 EXT\_DECL VOS\_ERR\_T vos\_sockGetMAC (UINT8 *pMAC*[VOS\_MAC\_SIZE])

Return the MAC address of the default adapter.

#### Parameters:

→ *pMAC* return MAC address.

#### Return values:

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* pMAC == NULL

**VOS\_SOCK\_ERR** socket not available or option not supported

Here is the call graph for this function:



#### 5.29.2.16 EXT\_DECL VOS\_ERR\_T vos\_sockInit (void)

Initialize the socket library.

Must be called once before any other call

##### Return values:

**VOS\_NO\_ERR** no error

**VOS\_SOCK\_ERR** sockets not supported

#### 5.29.2.17 EXT\_DECL VOS\_ERR\_T vos\_sockJoinMC (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)

Join a multicast group.

Note: Some targeted systems might not support this option.

##### Parameters:

← **sock** socket descriptor

← **mcAddress** multicast group to join

← **ipAddress** depicts interface on which to join, default 0 for any

##### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** sock descriptor unknown, parameter error

**VOS\_SOCK\_ERR** option not supported

Here is the call graph for this function:



#### 5.29.2.18 EXT\_DECL VOS\_ERR\_T vos\_sockLeaveMC (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)

Leave a multicast group.

Note: Some targeted systems might not support this option.

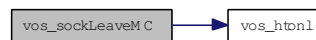
**Parameters:**

- ← *sock* socket descriptor
- ← *mcAddress* multicast group to join
- ← *ipAddress* depicts interface on which to leave, default 0 for any

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* sock descriptor unknown, parameter error
- VOS SOCK\_ERR* option not supported

Here is the call graph for this function:



### 5.29.2.19 EXT\_DECL VOS\_ERR\_T vos\_sockListen (INT32 *sock*, UINT32 *backlog*)

Listen for incoming connections.

Listen for incoming TCP connections.

**Parameters:**

- ← *sock* socket descriptor
- ← *backlog* maximum connection attempts if system is busy

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* sock descriptor unknown, parameter error
- VOS\_IO\_ERR* Input/Output error
- VOS\_MEM\_ERR* resource error

### 5.29.2.20 EXT\_DECL VOS\_ERR\_T vos\_sockOpenTCP (INT32 \* *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:**

- *pSock* pointer to socket descriptor returned
- ← *pOptions* pointer to socket options (optional)

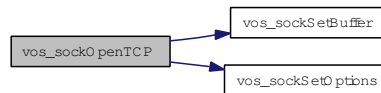
**Return values:**

- VOS\_NO\_ERR* no error

**VOS\_PARAM\_ERR** pSock == NULL

**VOS SOCK\_ERR** socket not available or option not supported

Here is the call graph for this function:



#### 5.29.2.21 EXT\_DECL VOS\_ERR\_T vos\_sockOpenUDP (INT32 \* 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 targeted systems might not support every option.

##### Parameters:

- **pSock** pointer to socket descriptor returned
- ← **pOptions** pointer to socket options (optional)

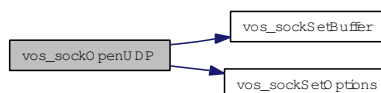
##### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pSock == NULL

**VOS SOCK\_ERR** socket not available or option not supported

Here is the call graph for this function:



#### 5.29.2.22 EXT\_DECL VOS\_ERR\_T vos\_sockReceiveTCP (INT32 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:

- ← **sock** socket descriptor

→ *pBuffer* pointer to applications data buffer

↔ *pSize* pointer to the received data size

#### 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

#### 5.29.2.23 EXT\_DECL VOS\_ERR\_T vos\_sockReceiveUDP (INT32 *sock*, UINT8 \* *pBuffer*, UINT32 \* *pSize*, UINT32 \* *pSrcIPAddr*, UINT16 \* *pSrcIPPort*, UINT32 \* *pDstIPAddr*, BOOL *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:

← *sock* socket descriptor

→ *pBuffer* pointer to applications data buffer

↔ *pSize* pointer to the received data size

→ *pSrcIPAddr* pointer to source IP

→ *pSrcIPPort* pointer to source port

→ *pDstIPAddr* pointer to dest IP

← *peek* if true, leave data in queue

#### Return values:

**VOS\_NO\_ERR** no error

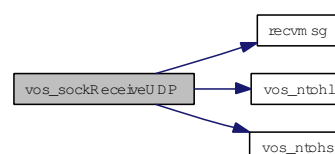
**VOS\_PARAM\_ERR** sock descriptor unknown, parameter error

**VOS\_IO\_ERR** data could not be read

**VOS\_NODATA\_ERR** no data

**VOS\_BLOCK\_ERR** Call would have blocked in blocking mode

Here is the call graph for this function:



### 5.29.2.24 EXT\_DECL VOS\_ERR\_T vos\_sockSendTCP (INT32 *sock*, const UINT8 \* *pBuffer*, UINT32 \* *pSize*)

Send TCP data.

Send data to the supplied address and port.

#### Parameters:

- ← *sock* socket descriptor
- ← *pBuffer* pointer to data to send
- ↔ *pSize* In: size of the data to send, Out: no of bytes sent

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS\_IO\_ERR** data could not be sent
- VOS\_NOCONN\_ERR** no TCP connection
- VOS\_BLOCK\_ERR** Call would have blocked in blocking mode

### 5.29.2.25 EXT\_DECL VOS\_ERR\_T vos\_sockSendUDP (INT32 *sock*, const UINT8 \* *pBuffer*, UINT32 \* *pSize*, UINT32 *ipAddress*, UINT16 *port*)

Send UDP data.

Send data to the supplied address and port.

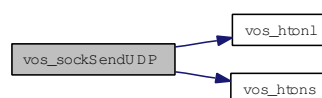
#### Parameters:

- ← *sock* socket descriptor
- ← *pBuffer* pointer to data to send
- ↔ *pSize* In: size of the data to send, Out: no of bytes sent
- ← *ipAddress* destination IP
- ← *port* destination port

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS\_IO\_ERR** data could not be sent
- VOS\_BLOCK\_ERR** Call would have blocked in blocking mode

Here is the call graph for this function:



### 5.29.2.26 VOS\_ERR\_T vos\_sockSetBuffer (INT32 *sock*)

Enlarge send and receive buffers to TRDP\_SOCKETBUF\_SIZE if necessary.

#### Parameters:

← *sock* socket descriptor

#### Return values:

**VOS\_NO\_ERR** no error

**VOS SOCK\_ERR** buffer size can't be set

### 5.29.2.27 EXT\_DECL VOS\_ERR\_T vos\_sockSetMulticastIf (INT32 *sock*, UINT32 *mcIfAddress*)

Set Using Multicast I/F.

#### Parameters:

← *sock* socket descriptor

← *mcIfAddress* using Multicast I/F Address

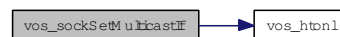
#### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** sock descriptor unknown, parameter error

**VOS SOCK\_ERR** option not supported

Here is the call graph for this function:



### 5.29.2.28 EXT\_DECL VOS\_ERR\_T vos\_sockSetOptions (INT32 *sock*, const VOS\_SOCKET\_OPT\_T \**pOptions*)

Set socket options.

Note: Some targeted systems might not support every option.

#### Parameters:

← *sock* socket descriptor

← *pOptions* pointer to socket options (optional)

#### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** sock descriptor unknown



**5.29.2.29 EXT\_DECL void vos\_sockTerm (void)**

De-Initialize the socket library.

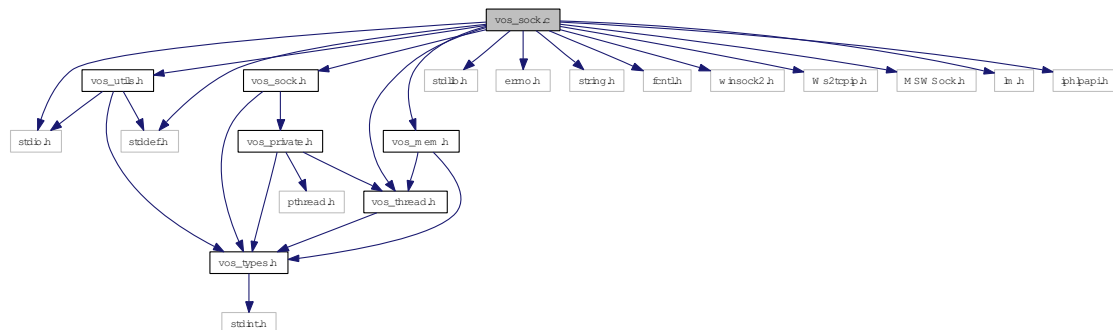
Must be called after last socket call

## 5.30 vos\_sock.c File Reference

Socket functions.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <winsock2.h>
#include <Ws2tcpip.h>
#include <MSWSock.h>
#include <lm.h>
#include <iphlpapi.h>
#include "vos_utils.h"
#include "vos_sock.h"
#include "vos_thread.h"
#include "vos_mem.h"
```

Include dependency graph for windows/vos\_sock.c:



## Functions

- INT32 [recvmsg](#) (int sock, struct msghdr \*pMessage, int flags)  
*Receive a message including sender address information.*
- [VOS\\_ERR\\_T vos\\_sockSetBuffer](#) (INT32 sock)  
*Enlarge send and receive buffers to TRDP\_SOCKETBUF\_SIZE if necessary.*
- EXT\_DECL UINT16 [vos\\_htons](#) (UINT16 val)  
*Byte swapping.*
- EXT\_DECL UINT16 [vos\\_ntohs](#) (UINT16 val)

*Byte swapping 2 Bytes.*

- EXT\_DECL UINT32 [vos\\_htonl](#) (UINT32 val)  
*Byte swapping 4 Bytes.*
- EXT\_DECL UINT32 [vos\\_ntohl](#) (UINT32 val)  
*Byte swapping 4 Bytes.*
- EXT\_DECL UINT32 [vos\\_dottedIP](#) (const CHAR8 \*pDottedIP)  
*Convert IP address from dotted dec.*
- EXT\_DECL const CHAR8 \* [vos\\_ipDotted](#) (UINT32 ipAddress)  
*Convert IP address to dotted dec.*
- EXT\_DECL BOOL [vos\\_isMulticast](#) (UINT32 ipAddress)  
*Check if the supplied address is a multicast group address.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_getInterfaces](#) (UINT32 \*pAddrCnt, [VOS\\_IF\\_REC\\_T](#) ifAddrs[ ])
  - Get a list of interface addresses The caller has to provide an array of interface records to be filled.*
- EXT\_DECL INT32 [vos\\_select](#) (INT32 highDesc, [VOS\\_FDS\\_T](#) \*pReadableFD, [VOS\\_FDS\\_T](#) \*pWriteableFD, [VOS\\_FDS\\_T](#) \*pErrorFD, [VOS\\_TIME\\_T](#) \*pTimeout)  
*select function.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockInit](#) (void)  
*Initialize the socket library.*
- EXT\_DECL void [vos\\_sockTerm](#) (void)  
*De-Initialize the socket library.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockGetMAC](#) (UINT8 pMAC[VOS\_MAC\_SIZE])  
*Return the MAC address of the default adapter.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockOpenUDP](#) (INT32 \*pSock, const [VOS\\_SOCKET\\_OPT\\_T](#) \*pOptions)  
*Create an UDP socket.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockOpenTCP](#) (INT32 \*pSock, const [VOS\\_SOCKET\\_OPT\\_T](#) \*pOptions)  
*Create a TCP socket.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockClose](#) (INT32 sock)  
*Close a socket.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockSetOptions](#) (INT32 sock, const [VOS\\_SOCKET\\_OPT\\_T](#) \*pOptions)  
*Set socket options.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockJoinMC](#) (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)

*Join a multicast group.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockLeaveMC](#) (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)

*Leave a multicast group.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockSendUDP](#) (INT32 sock, const UINT8 \*pBuffer, UINT32 \*pSize, UINT32 ipAddress, UINT16 port)

*Send UDP data.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockReceiveUDP](#) (INT32 sock, UINT8 \*pBuffer, UINT32 \*pSize, UINT32 \*pSrcIPAddr, UINT16 \*pSrcIPPort, UINT32 \*pDstIPAddr, BOOL peek)

*Receive UDP data.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockBind](#) (INT32 sock, UINT32 ipAddress, UINT16 port)

*Bind a socket to an address and port.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockListen](#) (INT32 sock, UINT32 backlog)

*Listen for incoming connections.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockAccept](#) (INT32 sock, INT32 \*pSock, UINT32 \*pIPAddr, UINT16 \*pPort)

*Accept an incoming TCP connection.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockConnect](#) (INT32 sock, UINT32 ipAddress, UINT16 port)

*Open a TCP connection.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockSendTCP](#) (INT32 sock, const UINT8 \*pBuffer, UINT32 \*pSize)

*Send TCP data.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockReceiveTCP](#) (INT32 sock, UINT8 \*pBuffer, UINT32 \*pSize)

*Receive TCP data.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockSetMulticastIf](#) (INT32 sock, UINT32 mcIfAddress)

*Set Using Multicast I/F.*

### 5.30.1 Detailed Description

Socket functions.

OS abstraction of IP socket functions for UDP and TCP

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

vos\_sock.c 1000 2013-07-02 08:53:28Z bloehr

**5.30.2 Function Documentation****5.30.2.1 INT32 recvmsg (int *sock*, struct msghdr \* *pMessage*, int *flags*)**

Receive a message including sender address information.

**Parameters:**

- ← *sock* socket descriptor
- ← *message* Pointer to message header
- ← *flags* Receive flags

**Return values:**

*number* of received bytes, -1 for error

**5.30.2.2 EXT\_DECL UINT32 vos\_dottedIP (const CHAR8 \* *pDottedIP*)**

Convert IP address from dotted dec.

to !host! endianness

**Parameters:**

- ← *pDottedIP* IP address as dotted decimal.

**Return values:**

*address* in UINT32 in host endianness

Here is the call graph for this function:

**5.30.2.3 EXT\_DECL VOS\_ERR\_T vos\_getInterfaces (UINT32 \* *pAddrCnt*, VOS\_IF\_REC\_T *ifAddrs*[ ])**

Get a list of interface addresses The caller has to provide an array of interface records to be filled.

**Parameters:**

- ↔ *pAddrCnt* in: pointer to array size of interface record out: pointer to number of interface records read

↔ *ifAddrs* array of interface records

**Return values:**

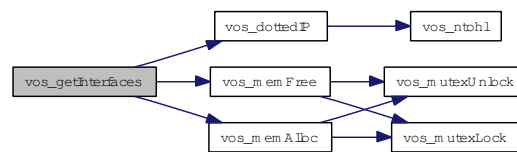
**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pAddrCnt and/or ifAddrs == NULL

**VOS\_MEM\_ERR** memory allocation error

**VOS SOCK\_ERR** GetAdaptersInfo() error

Here is the call graph for this function:



#### 5.30.2.4 EXT\_DECL UINT32 vos\_htonl (UINT32 val)

Byte swapping 4 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

#### 5.30.2.5 EXT\_DECL UINT16 vos\_htons (UINT16 val)

Byte swapping.

Byte swapping 2 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

#### 5.30.2.6 EXT\_DECL const CHAR8\* vos\_ipDotted (UINT32 ipAddress)

Convert IP address to dotted dec.

from !host! endianness.

**Parameters:**

← *ipAddress* address in UINT32 in host endianness

**Return values:**

*IP* address as dotted decimal.

**5.30.2.7 EXT\_DECL BOOL vos\_isMulticast (UINT32 *ipAddress*)**

Check if the supplied address is a multicast group address.

**Parameters:**

← *ipAddress* IP address to check.

**Return values:**

*TRUE* address is multicast

*FALSE* address is not a multicast address

**5.30.2.8 EXT\_DECL UINT32 vos\_ntohl (UINT32 *val*)**

Byte swapping 4 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

**5.30.2.9 EXT\_DECL UINT16 vos\_ntohs (UINT16 *val*)**

Byte swapping 2 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

**5.30.2.10 EXT\_DECL INT32 vos\_select (INT32 *highDesc*, VOS\_FDS\_T \* *pReadableFD*, VOS\_FDS\_T \* *pWriteableFD*, VOS\_FDS\_T \* *pErrorFD*, VOS\_TIME\_T \* *pTimeout*)**

select function.

Set the ready sockets in the supplied sets. Note: Some target systems might define this function as NOP.

**Parameters:**

← *highDesc* max. socket descriptor + 1

↔ *pReadableFD* pointer to readable socket set

- ↔ *pWriteableFD* pointer to writeable socket set
- ↔ *pErrorFD* pointer to error socket set
- ← *pTimeOut* pointer to time out value

**Return values:**

*number* of ready file descriptors

### 5.30.2.11 EXT\_DECL VOS\_ERR\_T vos\_sockAccept (INT32 *sock*, INT32 \**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.

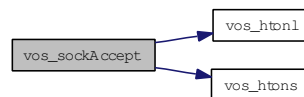
**Parameters:**

- ← *sock* Socket descriptor
- *pSock* Pointer to socket descriptor, on exit new socket
- *pIPAddress* source IP to receive on, 0 for any
- *pPort* port to receive on, 20548 for PD

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* NULL parameter, parameter error
- VOS\_UNKNOWN\_ERR* sock descriptor unknown error

Here is the call graph for this function:



### 5.30.2.12 EXT\_DECL VOS\_ERR\_T vos\_sockBind (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Bind a socket to an address and port.

**Parameters:**

- ← *sock* socket descriptor
- ← *ipAddress* source IP to receive on, 0 for any
- ← *port* port to receive on, 20548 for PD

**Return values:**

- VOS\_NO\_ERR* no error

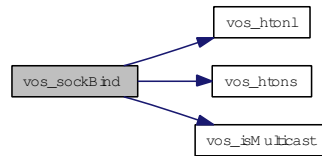


**VOS\_PARAM\_ERR** sock descriptor unknown, parameter error

**VOS\_IO\_ERR** Input/Output error

**VOS\_MEM\_ERR** resource error

Here is the call graph for this function:



### 5.30.2.13 EXT\_DECL VOS\_ERR\_T vos\_sockClose (INT32 *sock*)

Close a socket.

Release any resources acquired by this socket

#### Parameters:

← *sock* socket descriptor

#### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** sock descriptor unknown

### 5.30.2.14 EXT\_DECL VOS\_ERR\_T vos\_sockConnect (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Open a TCP connection.

#### Parameters:

← *sock* socket descriptor

← *ipAddress* destination IP

← *port* destination port

#### Return values:

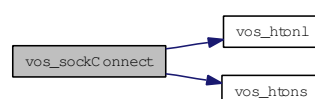
**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** sock descriptor unknown, parameter error

**VOS\_IO\_ERR** Input/Output error

**VOS\_MEM\_ERR** resource error

Here is the call graph for this function:



### 5.30.2.15 EXT\_DECL VOS\_ERR\_T vos\_sockGetMAC (UINT8 *pMAC*[VOS\_MAC\_SIZE])

Return the MAC address of the default adapter.

#### Parameters:

→ *pMAC* return MAC address.

#### Return values:

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* *pMAC* == NULL

*VOS SOCK\_ERR* socket not available or option not supported

### 5.30.2.16 EXT\_DECL VOS\_ERR\_T vos\_sockInit (void)

Initialize the socket library.

Must be called once before any other call

#### Return values:

*VOS\_NO\_ERR* no error

*VOS SOCK\_ERR* sockets not supported

### 5.30.2.17 EXT\_DECL VOS\_ERR\_T vos\_sockJoinMC (INT32 *sock*, UINT32 *mcAddress*, UINT32 *ipAddress*)

Join a multicast group.

Note: Some targeted systems might not support this option.

#### Parameters:

← *sock* socket descriptor

← *mcAddress* multicast group to join

← *ipAddress* depicts interface on which to join, default 0 for any

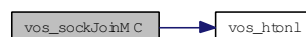
#### Return values:

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* sock descriptor unknown, parameter error

*VOS SOCK\_ERR* option not supported

Here is the call graph for this function:



### 5.30.2.18 EXT\_DECL VOS\_ERR\_T vos\_sockLeaveMC (INT32 *sock*, UINT32 *mcAddress*, UINT32 *ipAddress*)

Leave a multicast group.

Note: Some targeted systems might not support this option.

#### Parameters:

- ← *sock* socket descriptor
- ← *mcAddress* multicast group to join
- ← *ipAddress* depicts interface on which to leave, default 0 for any

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS SOCK\_ERR** option not supported

Here is the call graph for this function:



### 5.30.2.19 EXT\_DECL VOS\_ERR\_T vos\_sockListen (INT32 *sock*, UINT32 *backlog*)

Listen for incoming connections.

Listen for incoming TCP connections.

#### Parameters:

- ← *sock* socket descriptor
- ← *backlog* maximum connection attempts if system is busy

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS\_IO\_ERR** Input/Output error
- VOS\_MEM\_ERR** resource error

### 5.30.2.20 EXT\_DECL VOS\_ERR\_T vos\_sockOpenTCP (INT32 \**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:

- *pSock* pointer to socket descriptor returned

← *pOptions* pointer to socket options (optional)

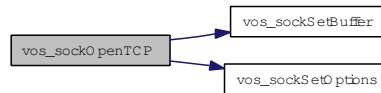
**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pSock == NULL

**VOS SOCK\_ERR** socket not available or option not supported

Here is the call graph for this function:



### 5.30.2.21 EXT\_DECL VOS\_ERR\_T vos\_sockOpenUDP (INT32 \* 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 targeted systems might not support every option.

**Parameters:**

→ *pSock* pointer to socket descriptor returned

← *pOptions* pointer to socket options (optional)

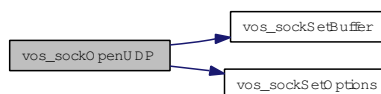
**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pSock == NULL

**VOS SOCK\_ERR** socket not available or option not supported

Here is the call graph for this function:



### 5.30.2.22 EXT\_DECL VOS\_ERR\_T vos\_sockReceiveTCP (INT32 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:**

- ← *sock* socket descriptor
- *pBuffer* pointer to applications data buffer
- ↔ *pSize* pointer to the received data size

**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

### 5.30.2.23 EXT\_DECL VOS\_ERR\_T vos\_sockReceiveUDP (INT32 *sock*, UINT8 \* *pBuffer*, UINT32 \* *pSize*, UINT32 \* *pSrcIPAddr*, UINT16 \* *pSrcIPPort*, UINT32 \* *pDstIPAddr*, BOOL *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.

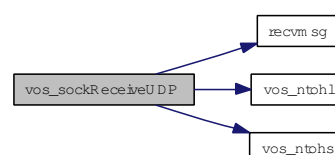
**Parameters:**

- ← *sock* socket descriptor
- *pBuffer* pointer to applications data buffer
- ↔ *pSize* pointer to the received data size
- *pSrcIPAddr* pointer to source IP
- *pSrcIPPort* pointer to source port
- *pDstIPAddr* pointer to dest IP
- ← *peek* 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

Here is the call graph for this function:



### 5.30.2.24 EXT\_DECL VOS\_ERR\_T vos\_sockSendTCP (INT32 *sock*, const UINT8 \* *pBuffer*, UINT32 \* *pSize*)

Send TCP data.

Send data to the supplied address and port.

#### Parameters:

- ← *sock* socket descriptor
- ← *pBuffer* pointer to data to send
- ↔ *pSize* IN: bytes to send, OUT: bytes sent

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS\_IO\_ERR** data could not be sent
- VOS\_NOCONN\_ERR** no TCP connection
- VOS\_BLOCK\_ERR** Call would have blocked in blocking mode

### 5.30.2.25 EXT\_DECL VOS\_ERR\_T vos\_sockSendUDP (INT32 *sock*, const UINT8 \* *pBuffer*, UINT32 \* *pSize*, UINT32 *ipAddress*, UINT16 *port*)

Send UDP data.

Send data to the supplied address and port.

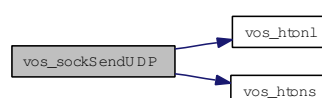
#### Parameters:

- ← *sock* socket descriptor
- ← *pBuffer* pointer to data to send
- ↔ *pSize* IN: bytes to send, OUT: bytes sent
- ← *ipAddress* destination IP
- ← *port* destination port

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS\_IO\_ERR** data could not be sent
- VOS\_BLOCK\_ERR** Call would have blocked in blocking mode

Here is the call graph for this function:



**5.30.2.26 VOS\_ERR\_T vos\_sockSetBuffer (INT32 *sock*)**

Enlarge send and receive buffers to TRDP\_SOCKETBUF\_SIZE if necessary.

**Parameters:**

← *sock* socket descriptor

**Return values:**

**VOS\_NO\_ERR** no error

**VOS SOCK\_ERR** buffer size can't be set

**5.30.2.27 EXT\_DECL VOS\_ERR\_T vos\_sockSetMulticastIf (INT32 *sock*, UINT32 *mcIfAddress*)**

Set Using Multicast I/F.

**Parameters:**

← *sock* socket descriptor

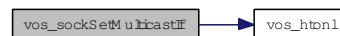
← *mcIfAddress* using Multicast I/F Address

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** sock descriptor unknown, parameter error

Here is the call graph for this function:

**5.30.2.28 EXT\_DECL VOS\_ERR\_T vos\_sockSetOptions (INT32 *sock*, const VOS\_SOCKET\_OPT\_T \**pOptions*)**

Set socket options.

Note: Some targeted systems might not support every option.

**Parameters:**

← *sock* socket descriptor

← *pOptions* pointer to socket options (optional)

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** sock descriptor unknown

**5.30.2.29 EXT\_DECL void vos\_sockTerm (void)**

De-Initialize the socket library.

Must be called after last socket call





- #define [VOS\\_MAX\\_IF\\_NAME\\_SIZE](#) 16  
*The maximum number of IP interface adapters that can be handled by VOS.*
- #define [VOS\\_MAX\\_NUM\\_IF](#) 4  
*The maximum number of unicast addresses that can be handled by VOS.*
- #define [VOS\\_MAX\\_NUM\\_UNICAST](#) 10  
*The MAC size supported by VOS.*
- #define [VOS\\_MAC\\_SIZE](#) 6  
*Size of socket send and receive buffer.*
- #define [VOS\\_INVALID\\_SOCKET](#) -1  
*Invalid socket number.*

## Functions

- EXT\_DECL UINT16 [vos\\_htons](#) (UINT16 val)  
*Byte swapping 2 Bytes.*
- EXT\_DECL UINT16 [vos\\_ntohs](#) (UINT16 val)  
*Byte swapping 2 Bytes.*
- EXT\_DECL UINT32 [vos\\_htonl](#) (UINT32 val)  
*Byte swapping 4 Bytes.*
- EXT\_DECL UINT32 [vos\\_ntohl](#) (UINT32 val)  
*Byte swapping 4 Bytes.*
- EXT\_DECL UINT32 [vos\\_dottedIP](#) (const CHAR8 \*pDottedIP)  
*Convert IP address from dotted dec.*
- EXT\_DECL const CHAR8 \* [vos\\_ipDotted](#) (UINT32 ipAddress)  
*Convert IP address to dotted dec.*
- EXT\_DECL BOOL [vos\\_isMulticast](#) (UINT32 ipAddress)  
*Check if the supplied address is a multicast group address.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_getInterfaces](#) (UINT32 \*pAddrCnt, [VOS\\_IF\\_REC\\_T](#) ifAddrs[ ])   
*Get a list of interface addresses The caller has to provide an array of interface records to be filled.*
- EXT\_DECL INT32 [vos\\_select](#) (INT32 highDesc, [VOS\\_FDS\\_T](#) \*pReadableFD, [VOS\\_FDS\\_T](#) \*pWritableFD, [VOS\\_FDS\\_T](#) \*pErrorFD, [VOS\\_TIME\\_T](#) \*pTimeOut)  
*select function.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockInit](#) (void)  
*Initialize the socket library.*
- EXT\_DECL void [vos\\_sockTerm](#) (void)

*De-Initialize the socket library.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockGetMAC](#) (UINT8 pMAC[VOS\_MAC\_SIZE])  
*Return the MAC address of the default adapter.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockOpenUDP](#) (INT32 \*pSock, const [VOS\\_SOCKET\\_OPT\\_T](#) \*pOptions)  
*Create an UDP socket.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockOpenTCP](#) (INT32 \*pSock, const [VOS\\_SOCKET\\_OPT\\_T](#) \*pOptions)  
*Create a TCP socket.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockClose](#) (INT32 sock)  
*Close a socket.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockSetOptions](#) (INT32 sock, const [VOS\\_SOCKET\\_OPT\\_T](#) \*pOptions)  
*Set socket options.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockJoinMC](#) (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)  
*Join a multicast group.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockLeaveMC](#) (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)  
*Leave a multicast group.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockSendUDP](#) (INT32 sock, const UINT8 \*pBuffer, UINT32 \*pSize, UINT32 ipAddress, UINT16 port)  
*Send UDP data.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockReceiveUDP](#) (INT32 sock, UINT8 \*pBuffer, UINT32 \*pSize, UINT32 \*pSrcIPAddr, UINT16 \*pSrcIPPort, UINT32 \*pDstIPAddr, BOOL peek)  
*Receive UDP data.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockBind](#) (INT32 sock, UINT32 ipAddress, UINT16 port)  
*Bind a socket to an address and port.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockListen](#) (INT32 sock, UINT32 backlog)  
*Listen for incoming TCP connections.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockAccept](#) (INT32 sock, INT32 \*pSock, UINT32 \*pIPAddr, UINT16 \*pPort)  
*Accept an incoming TCP connection.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockConnect](#) (INT32 sock, UINT32 ipAddress, UINT16 port)  
*Open a TCP connection.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockSendTCP](#) (INT32 sock, const UINT8 \*pBuffer, UINT32 \*pSize)

*Send TCP data.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockReceiveTCP](#) (INT32 sock, UINT8 \*pBuffer, UINT32 \*pSize)

*Receive TCP data.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockSetMulticastIf](#) (INT32 sock, UINT32 mcIfAddress)

*Set Using Multicast I/F.*

### 5.31.1 Detailed Description

Typedefs for OS abstraction.

This is the declaration for the OS independend socket interface

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Bernd Loehr, NewTec GmbH

#### Remarks:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

#### Id

[vos\\_sock.h](#) 1015 2013-07-04 07:59:22Z bloehr

### 5.31.2 Define Documentation

#### 5.31.2.1 #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

#### 5.31.2.2 #define VOS\_TTL\_MULTICAST 64

The maximum number of hops a multicast packet can take.

The maximum size for the interface name

### 5.31.3 Function Documentation

#### 5.31.3.1 EXT\_DECL UINT32 vos\_dottedIP (const CHAR8 \*pDottedIP)

Convert IP address from dotted dec.

to !host! endianness

**Parameters:**

← *pDottedIP* IP address as dotted decimal.

**Return values:**

*address* in UINT32 in host endianness

Here is the call graph for this function:



### 5.31.3.2 EXT\_DECL VOS\_ERR\_T vos\_getInterfaces (UINT32 \*pAddrCnt, VOS\_IF\_REC\_T ifAddrs[])

Get a list of interface addresses The caller has to provide an array of interface records to be filled.

**Parameters:**

↔ *pAddrCnt* in: pointer to array size of interface record out: pointer to number of interface records read

↔ *ifAddrs* array of interface records

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* pAddrCnt and/or ifAddrs == NULL

*VOS\_MEM\_ERR* memory allocation error

*VOS SOCK\_ERR* GetAdaptersInfo() error

**Parameters:**

↔ *pAddrCnt* in: pointer to array size of interface record out: pointer to number of interface records read

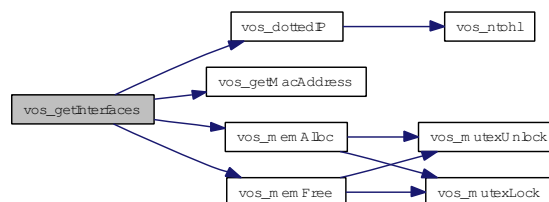
↔ *ifAddrs* array of interface records

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* pMAC == NULL

Here is the call graph for this function:



### 5.31.3.3 EXT\_DECL UINT32 vos\_htonl (UINT32 *val*)

Byte swapping 4 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

### 5.31.3.4 EXT\_DECL UINT16 vos\_htons (UINT16 *val*)

Byte swapping 2 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

Byte swapping 2 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

Byte swapping 2 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

### 5.31.3.5 EXT\_DECL const CHAR8\* vos\_ipDotted (UINT32 *ipAddress*)

Convert IP address to dotted dec.

from !host! endianness

**Parameters:**

← *ipAddress* address in UINT32 in host endianness

**Return values:**

*IP* address as dotted decimal.

from !host! endianness.

**Parameters:**

← *ipAddress* address in UINT32 in host endianness

**Return values:**

*IP* address as dotted decimal.

### 5.31.3.6 EXT\_DECL BOOL vos\_isMulticast (UINT32 *ipAddress*)

Check if the supplied address is a multicast group address.

**Parameters:**

← *ipAddress* IP address to check.

**Return values:**

*TRUE* address is a multicast address

*FALSE* address is not a multicast address

**Parameters:**

← *ipAddress* IP address to check.

**Return values:**

*TRUE* address is multicast

*FALSE* address is not a multicast address

### 5.31.3.7 EXT\_DECL UINT32 vos\_ntohl (UINT32 *val*)

Byte swapping 4 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

### 5.31.3.8 EXT\_DECL UINT16 vos\_ntohs (UINT16 *val*)

Byte swapping 2 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

### 5.31.3.9 EXT\_DECL INT32 vos\_select (INT32 *highDesc*, VOS\_FDS\_T \* *pReadableFD*, VOS\_FDS\_T \* *pWriteableFD*, VOS\_FDS\_T \* *pErrorFD*, VOS\_TIME\_T \* *pTimeOut*)

select function.

Set the ready sockets in the supplied sets. Note: Some target systems might define this function as NOP.

#### Parameters:

- ← *highDesc* max. socket descriptor + 1
- ↔ *pReadableFD* pointer to readable socket set
- ↔ *pWriteableFD* pointer to writeable socket set
- ↔ *pErrorFD* pointer to error socket set
- ← *pTimeOut* pointer to time out value

#### Return values:

*number* of ready file descriptors

### 5.31.3.10 EXT\_DECL VOS\_ERR\_T vos\_sockAccept (INT32 *sock*, INT32 \* *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:

- ← *sock* Socket descriptor
- *pSock* Pointer to socket descriptor, on exit new socket
- *pIPAddress* source IP to receive on, 0 for any
- *pPort* port to receive on, 20548 for PD

#### Return values:

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* NULL parameter, parameter error
- VOS\_UNKNOWN\_ERR* sock descriptor unknown error

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:

- ← *sock* Socket descriptor
- *pSock* Pointer to socket descriptor, on exit new socket
- *pIPAddress* source IP to receive on, 0 for any
- *pPort* port to receive on, 20548 for PD

#### Return values:

- VOS\_NO\_ERR* no error

**VOS\_PARAM\_ERR** NULL parameter, parameter error

**VOS\_UNKNOWN\_ERR** sock descriptor unknown error

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:**

- ← *sock* Socket descriptor
- *pSock* Pointer to socket descriptor, on exit new socket
- *pIPAddress* source IP to receive on, 0 for any
- *pPort* port to receive on, 20548 for PD

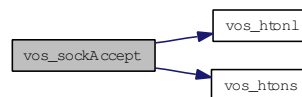
**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** NULL parameter, parameter error

**VOS\_UNKNOWN\_ERR** sock descriptor unknown error

Here is the call graph for this function:



### 5.31.3.11 EXT\_DECL VOS\_ERR\_T vos\_sockBind (INT32 sock, UINT32 ipAddress, UINT16 port)

Bind a socket to an address and port.

**Parameters:**

- ← *sock* socket descriptor
- ← *ipAddress* source IP to receive from, 0 for any
- ← *port* port to receive from

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_IO\_ERR** Input/Output error

**VOS\_MEM\_ERR** resource error

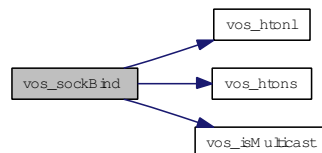
**Parameters:**

- ← *sock* socket descriptor
- ← *ipAddress* source IP to receive on, 0 for any
- ← *port* port to receive on, 20548 for PD



**Return values:***VOS\_NO\_ERR* no error*VOS\_PARAM\_ERR* sock descriptor unknown, parameter error*VOS\_IO\_ERR* Input/Output error*VOS\_MEM\_ERR* resource error

Here is the call graph for this function:

**5.31.3.12 EXT\_DECL VOS\_ERR\_T vos\_sockClose (INT32 sock)**

Close a socket.

Release any resources aquired by this socket

**Parameters:**← *sock* socket descriptor**Return values:***VOS\_NO\_ERR* no error*VOS\_PARAM\_ERR* pSock == NULL

Release any resources aquired by this socket

**Parameters:**← *sock* socket descriptor**Return values:***VOS\_NO\_ERR* no error*VOS\_PARAM\_ERR* sock descriptor unknown

Release any resources aquired by this socket

**Parameters:**← *sock* socket descriptor**Return values:***VOS\_NO\_ERR* no error*VOS\_PARAM\_ERR* sock descriptor unknown

### 5.31.3.13 EXT\_DECL VOS\_ERR\_T vos\_sockConnect (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Open a TCP connection.

#### Parameters:

- ← *sock* socket descriptor
- ← *ipAddress* destination IP
- ← *port* destination port

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** parameter out of range/invalid
- VOS\_IO\_ERR** Input/Output error

#### Parameters:

- ← *sock* socket descriptor
- ← *ipAddress* destination IP
- ← *port* destination port

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS\_IO\_ERR** Input/Output error

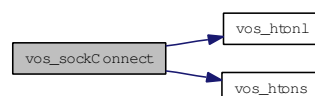
#### Parameters:

- ← *sock* socket descriptor
- ← *ipAddress* destination IP
- ← *port* destination port

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS\_IO\_ERR** Input/Output error
- VOS\_MEM\_ERR** resource error

Here is the call graph for this function:



**5.31.3.14** EXT\_DECL VOS\_ERR\_T vos\_sockGetMAC (UINT8 *pMAC*[VOS\_MAC\_SIZE])

Return the MAC address of the default adapter.

**Parameters:**

→ *pMAC* return MAC address.

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** *pMAC* == NULL

**VOS SOCK\_ERR** socket not available or option not supported

Here is the call graph for this function:

**5.31.3.15** EXT\_DECL VOS\_ERR\_T vos\_sockInit (void)

Initialize the socket library.

Must be called once before any other call

**Return values:**

**VOS\_NO\_ERR** no error

**VOS SOCK\_ERR** sockets not supported

Must be called once before any other call

**Return values:**

**VOS\_NO\_ERR** no error

**VOS SOCK\_ERR** sockets not supported

Must be called once before any other call

**Return values:**

**VOS\_NO\_ERR** no error

**VOS SOCK\_ERR** sockets not supported

**5.31.3.16** EXT\_DECL VOS\_ERR\_T vos\_sockJoinMC (INT32 *sock*, UINT32 *mcAddress*, UINT32 *ipAddress*)

Join a multicast group.

Note: Some target systems might not support this option.

**Parameters:**

- ← *sock* socket descriptor
- ← *mcAddress* multicast group to join
- ← *ipAddress* depicts interface on which to join, default 0 for any

**Return values:**

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** parameter out of range/invalid
- VOS SOCK\_ERR** option not supported

Note: Some targeted systems might not support this option.

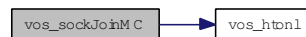
**Parameters:**

- ← *sock* socket descriptor
- ← *mcAddress* multicast group to join
- ← *ipAddress* depicts interface on which to join, default 0 for any

**Return values:**

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS SOCK\_ERR** option not supported

Here is the call graph for this function:



### 5.31.3.17 EXT\_DECL VOS\_ERR\_T vos\_sockLeaveMC (INT32 *sock*, UINT32 *mcAddress*, UINT32 *ipAddress*)

Leave a multicast group.

Note: Some target systems might not support this option.

**Parameters:**

- ← *sock* socket descriptor
- ← *mcAddress* multicast group to join
- ← *ipAddress* depicts interface on which to leave, default 0 for any

**Return values:**

- VOS\_NO\_ERR** no error
- VOS\_INIT\_ERR** module not initialised
- VOS\_NOINIT\_ERR** invalid handle
- VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_SOCK\_ERR** option not supported

Note: Some targeted systems might not support this option.

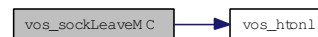
**Parameters:**

- ← *sock* socket descriptor
- ← *mcAddress* multicast group to join
- ← *ipAddress* depicts interface on which to leave, default 0 for any

**Return values:**

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS\_SOCK\_ERR** option not supported

Here is the call graph for this function:



### 5.31.3.18 EXT\_DECL VOS\_ERR\_T vos\_sockListen (INT32 *sock*, UINT32 *backlog*)

Listen for incoming TCP connections.

**Parameters:**

- ← *sock* socket descriptor
- ← *backlog* maximum connection attempts if system is busy

**Return values:**

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** parameter out of range/invalid
- VOS\_IO\_ERR** Input/Output error
- VOS\_MEM\_ERR** resource error

Listen for incoming TCP connections.

**Parameters:**

- ← *sock* socket descriptor
- ← *backlog* maximum connection attempts if system is busy

**Return values:**

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS\_IO\_ERR** Input/Output error
- VOS\_MEM\_ERR** resource error

Listen for incoming TCP connections.

**Parameters:**

- ← *sock* socket descriptor
- ← *backlog* maximum connection attempts if system is busy

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* sock descriptor unknown, parameter error
- VOS\_IO\_ERR* Input/Output error
- VOS\_MEM\_ERR* resource error

### 5.31.3.19 EXT\_DECL VOS\_ERR\_T vos\_sockOpenTCP (INT32 \* *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:**

- *pSock* pointer to socket descriptor returned
- ← *pOptions* pointer to socket options (optional)

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* pSock == NULL
- VOS\_SOCK\_ERR* socket not available or option not supported

Return a socket descriptor for further calls. The socket options are optional and can be applied later.

**Parameters:**

- *pSock* pointer to socket descriptor returned
- ← *pOptions* pointer to socket options (optional)

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* pSock == NULL
- VOS\_SOCK\_ERR* socket not available or option not supported

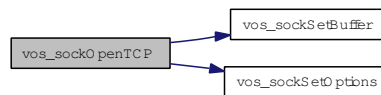
Return a socket descriptor for further calls. The socket options are optional and can be applied later.

**Parameters:**

- *pSock* pointer to socket descriptor returned
- ← *pOptions* pointer to socket options (optional)

**Return values:***VOS\_NO\_ERR* no error*VOS\_PARAM\_ERR* pSock == NULL*VOS SOCK\_ERR* socket not available or option not supported

Here is the call graph for this function:



### 5.31.3.20 EXT\_DECL VOS\_ERR\_T vos\_sockOpenUDP (INT32 \* 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:**

→ *pSock* pointer to socket descriptor returned

← *pOptions* pointer to socket options (optional)

**Return values:***VOS\_NO\_ERR* no error*VOS\_PARAM\_ERR* pSock == NULL*VOS SOCK\_ERR* socket not available or option not supported

Return a socket descriptor for further calls. The socket options are optional and can be applied later. Note: Some targeted systems might not support every option.

**Parameters:**

→ *pSock* pointer to socket descriptor returned

← *pOptions* pointer to socket options (optional)

**Return values:***VOS\_NO\_ERR* no error*VOS\_PARAM\_ERR* pSock == NULL*VOS SOCK\_ERR* socket not available or option not supported

Return a socket descriptor for further calls. The socket options are optional and can be applied later. Note: Some targeted systems might not support every option.

**Parameters:**

→ *pSock* pointer to socket descriptor returned

← *pOptions* pointer to socket options (optional)

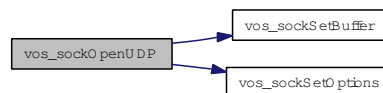
**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pSock == NULL

**VOS SOCK\_ERR** socket not available or option not supported

Here is the call graph for this function:



### 5.31.3.21 EXT\_DECL VOS\_ERR\_T vos\_sockReceiveTCP (INT32 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:**

← *sock* socket descriptor

→ *pBuffer* pointer to applications data buffer

↔ *pSize* pointer to the received data size

**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 in non-blocking

**VOS\_BLOCK\_ERR** call would have blocked in blocking mode

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:**

← *sock* socket descriptor

→ *pBuffer* pointer to applications data buffer



↔ *pSize* pointer to the received data size

#### 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

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:

← *sock* socket descriptor

→ *pBuffer* pointer to applications data buffer

↔ *pSize* pointer to the received data size

#### 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

#### 5.31.3.22 EXT\_DECL VOS\_ERR\_T vos\_sockReceiveUDP (INT32 *sock*, UINT8 \* *pBuffer*, UINT32 \* *pSize*, UINT32 \* *pSrcIPAddr*, UINT16 \* *pSrcIPPort*, UINT32 \* *pDstIPAddr*, BOOL *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:

← *sock* socket descriptor

→ *pBuffer* pointer to applications data buffer

↔ *pSize* pointer to the received data size

→ *pSrcIPAddr* pointer to source IP

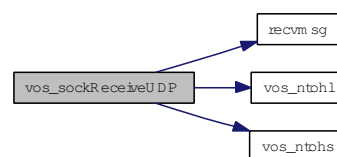
→ *pSrcIPPort* pointer to source port

→ *pDstIPAddr* pointer to dest IP  
 ← *peek* 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

Here is the call graph for this function:



### 5.31.3.23 EXT\_DECL VOS\_ERR\_T vos\_sockSendTCP (INT32 *sock*, const UINT8 \* *pBuffer*, UINT32 \* *pSize*)

Send TCP data.

Send data to the supplied address and port.

**Parameters:**

← *sock* socket descriptor  
 ← *pBuffer* pointer to data to send  
 ↔ *pSize* In: size of the data to send, Out: no of bytes sent

**Return values:**

**VOS\_NO\_ERR** no error  
**VOS\_PARAM\_ERR** sock descriptor unknown, parameter error  
**VOS\_IO\_ERR** data could not be sent  
**VOS\_NOCONN\_ERR** no TCP connection  
**VOS\_BLOCK\_ERR** call would have blocked in blocking mode, data partially sent

Send data to the supplied address and port.

**Parameters:**

← *sock* socket descriptor  
 ← *pBuffer* pointer to data to send  
 ↔ *pSize* In: size of the data to send, Out: no of bytes sent

**Return values:**

***VOS\_NO\_ERR*** no error  
***VOS\_PARAM\_ERR*** sock descriptor unknown, parameter error  
***VOS\_IO\_ERR*** data could not be sent  
***VOS\_NOCONN\_ERR*** no TCP connection  
***VOS\_BLOCK\_ERR*** Call would have blocked in blocking mode

Send data to the supplied address and port.

**Parameters:**

← ***sock*** socket descriptor  
 ← ***pBuffer*** pointer to data to send  
 ↔ ***pSize*** IN: bytes to send, OUT: bytes sent

**Return values:**

***VOS\_NO\_ERR*** no error  
***VOS\_PARAM\_ERR*** sock descriptor unknown, parameter error  
***VOS\_IO\_ERR*** data could not be sent  
***VOS\_NOCONN\_ERR*** no TCP connection  
***VOS\_BLOCK\_ERR*** Call would have blocked in blocking mode

#### 5.31.3.24 EXT\_DECL VOS\_ERR\_T vos\_sockSendUDP (INT32 *sock*, const UINT8 \* *pBuffer*, UINT32 \* *pSize*, UINT32 *ipAddress*, UINT16 *port*)

Send UDP data.

Send data to the given address and port.

**Parameters:**

← ***sock*** socket descriptor  
 ← ***pBuffer*** pointer to data to send  
 ↔ ***pSize*** In: size of the data to send, Out: no of bytes sent  
 ← ***ipAddress*** destination IP  
 ← ***port*** destination port

**Return values:**

***VOS\_NO\_ERR*** no error  
***VOS\_PARAM\_ERR*** parameter out of range/invalid  
***VOS\_IO\_ERR*** data could not be sent  
***VOS\_BLOCK\_ERR*** Call would have blocked in blocking mode

Send data to the supplied address and port.

**Parameters:**

← ***sock*** socket descriptor

← *pBuffer* pointer to data to send  
 ↔ *pSize* In: size of the data to send, Out: no of bytes sent  
 ← *ipAddress* destination IP  
 ← *port* destination port

**Return values:**

***VOS\_NO\_ERR*** no error  
***VOS\_PARAM\_ERR*** sock descriptor unknown, parameter error  
***VOS\_IO\_ERR*** data could not be sent  
***VOS\_BLOCK\_ERR*** Call would have blocked in blocking mode

Send data to the supplied address and port.

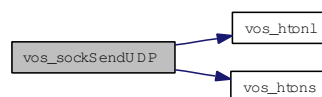
**Parameters:**

← *sock* socket descriptor  
 ← *pBuffer* pointer to data to send  
 ↔ *pSize* IN: bytes to send, OUT: bytes sent  
 ← *ipAddress* destination IP  
 ← *port* destination port

**Return values:**

***VOS\_NO\_ERR*** no error  
***VOS\_PARAM\_ERR*** sock descriptor unknown, parameter error  
***VOS\_IO\_ERR*** data could not be sent  
***VOS\_BLOCK\_ERR*** Call would have blocked in blocking mode

Here is the call graph for this function:



### 5.31.3.25 EXT\_DECL VOS\_ERR\_T vos\_sockSetMulticastIf (INT32 *sock*, UINT32 *mcIfAddress*)

Set Using Multicast I/F.

**Parameters:**

← *sock* socket descriptor  
 ← *mcIfAddress* using Multicast I/F Address

**Return values:**

***VOS\_NO\_ERR*** no error  
***VOS\_PARAM\_ERR*** sock descriptor unknown, parameter error

**Parameters:**

- ← *sock* socket descriptor
- ← *mcIfAddress* using Multicast I/F Address

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* sock descriptor unknown, parameter error
- VOS SOCK\_ERR* option not supported

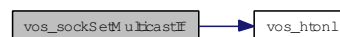
**Parameters:**

- ← *sock* socket descriptor
- ← *mcIfAddress* using Multicast I/F Address

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* sock descriptor unknown, parameter error

Here is the call graph for this function:



### 5.31.3.26 EXT\_DECL VOS\_ERR\_T vos\_sockSetOptions (INT32 *sock*, const VOS\_SOCK\_OPT\_T \**pOptions*)

Set socket options.

Note: Some target systems might not support each option.

**Parameters:**

- ← *sock* socket descriptor
- ← *pOptions* pointer to socket options (optional)

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* parameter out of range/invalid

Note: Some targeted systems might not support every option.

**Parameters:**

- ← *sock* socket descriptor
- ← *pOptions* pointer to socket options (optional)

**Return values:**

- VOS\_NO\_ERR* no error

***VOS\_PARAM\_ERR*** sock descriptor unknown

Note: Some targeted systems might not support every option.

**Parameters:**

← ***sock*** socket descriptor

← ***pOptions*** pointer to socket options (optional)

**Return values:**

***VOS\_NO\_ERR*** no error

***VOS\_PARAM\_ERR*** sock descriptor unknown

**5.31.3.27 EXT\_DECL void vos\_sockTerm (void)**

De-Initialize the socket library.

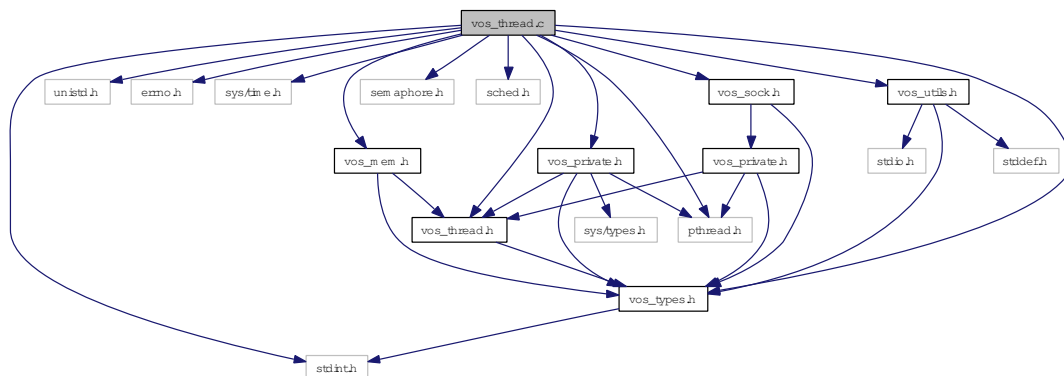
Must be called after last socket call

## 5.32 vos\_thread.c File Reference

Multitasking functions.

```
#include <stdint.h>
#include <unistd.h>
#include <errno.h>
#include <sys/time.h>
#include <pthread.h>
#include <semaphore.h>
#include <sched.h>
#include "vos_sock.h"
#include "vos_types.h"
#include "vos_thread.h"
#include "vos_mem.h"
#include "vos_utils.h"
#include "vos_private.h"
```

Include dependency graph for posix/vos\_thread.c:



## Functions

- void [cyclicThread](#) (UINT32 interval, [VOS\\_THREAD\\_FUNC\\_T](#) pFunction, void \*pArguments)  
*Cyclic thread functions.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_threadInit](#) (void)  
*Initialize the thread library.*
- EXT\_DECL void [vos\\_threadTerm](#) (void)  
*De-Initialize the thread library.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_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.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_threadTerminate](#) ([VOS\\_THREAD\\_T](#) thread)  
*Terminate a thread.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_threadIsActive](#) ([VOS\\_THREAD\\_T](#) thread)  
*Is the thread still active? This call will return VOS\_NO\_ERR if the thread is still active, VOS\_PARAM\_ERR in case it ran out.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_threadDelay](#) (UINT32 delay)  
*Delay the execution of the current thread by the given delay in us.*
- EXT\_DECL void [vos\\_getTime](#) ([VOS\\_TIME\\_T](#) \*pTime)  
*Return the current time in sec and us.*
- EXT\_DECL const CHAR8 \* [vos\\_getTimeStamp](#) (void)  
*Get a time-stamp string.*
- EXT\_DECL void [vos\\_clearTime](#) ([VOS\\_TIME\\_T](#) \*pTime)  
*Clear the time stamp.*
- EXT\_DECL void [vos\\_addTime](#) ([VOS\\_TIME\\_T](#) \*pTime, const [VOS\\_TIME\\_T](#) \*pAdd)  
*Add the second to the first time stamp, return sum in first.*
- EXT\_DECL void [vos\\_subTime](#) ([VOS\\_TIME\\_T](#) \*pTime, const [VOS\\_TIME\\_T](#) \*pSub)  
*Subtract the second from the first time stamp, return diff in first.*
- EXT\_DECL void [vos\\_divTime](#) ([VOS\\_TIME\\_T](#) \*pTime, UINT32 divisor)  
*Divide the first time value by the second, return quotient in first.*
- EXT\_DECL void [vos\\_mulTime](#) ([VOS\\_TIME\\_T](#) \*pTime, UINT32 mul)  
*Multiply the first time by the second, return product in first.*
- EXT\_DECL INT32 [vos\\_cmpTime](#) (const [VOS\\_TIME\\_T](#) \*pTime, const [VOS\\_TIME\\_T](#) \*pCmp)  
*Compare the second from the first time stamp, return diff in first.*
- EXT\_DECL void [vos\\_getUuid](#) ([VOS\\_UUID\\_T](#) pUUID)  
*Get a universal unique identifier according to RFC 4122 time based version.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_mutexCreate](#) ([VOS\\_MUTEX\\_T](#) \*pMutex)  
*Create a recursive mutex.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_mutexLocalCreate](#) (struct [VOS\\_MUTEX](#) \*pMutex)  
*Create a recursive mutex.*
- EXT\_DECL void [vos\\_mutexDelete](#) ([VOS\\_MUTEX\\_T](#) pMutex)  
*Delete a mutex.*
- EXT\_DECL void [vos\\_mutexLocalDelete](#) (struct [VOS\\_MUTEX](#) \*pMutex)  
*Delete a mutex.*



- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_mutexLock](#) ([VOS\\_MUTEX\\_T](#) pMutex)  
*Take a mutex.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_mutexTryLock](#) ([VOS\\_MUTEX\\_T](#) pMutex)  
*Try to take a mutex.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_mutexUnlock](#) ([VOS\\_MUTEX\\_T](#) pMutex)  
*Release a mutex.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_semaCreate](#) ([VOS\\_SEMA\\_T](#) \*pSema, [VOS\\_SEMA\\_STATE\\_T](#) initialState)  
*Create a semaphore.*
- EXT\_DECL void [vos\\_semaDelete](#) ([VOS\\_SEMA\\_T](#) sema)  
*Delete a semaphore.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_semaTake](#) ([VOS\\_SEMA\\_T](#) sema, UINT32 timeout)  
*Take a semaphore.*
- EXT\_DECL void [vos\\_semaGive](#) ([VOS\\_SEMA\\_T](#) sema)  
*Give a semaphore.*

### 5.32.1 Detailed Description

Multitasking functions.

OS abstraction of thread-handling functions

**Note:**

Project: TCNOpen TRDP prototype stack

**Author:**

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

vos\_thread.c 1009 2013-07-03 08:58:02Z bloehr

### 5.32.2 Function Documentation

#### 5.32.2.1 void cyclicThread (UINT32 interval, VOS\_THREAD\_FUNC\_T pFunction, void \* pArguments)

Cyclic thread functions.

Wrapper for cyclic threads. The thread function will be called cyclically with interval.

**Parameters:**

- ← *interval* Interval for cyclic threads in us (optional)
- ← *pFunction* Pointer to the thread function
- ← *pArguments* Pointer to the thread function parameters

**Return values:**

*void*

Here is the call graph for this function:



### 5.32.2.2 EXT\_DECL void vos\_addTime (VOS\_TIME\_T \**pTime*, const VOS\_TIME\_T \**pAdd*)

Add the second to the first time stamp, return sum in first.

**Parameters:**

- ↔ *pTime* Pointer to time value
- ← *pAdd* Pointer to time value

### 5.32.2.3 EXT\_DECL void vos\_clearTime (VOS\_TIME\_T \**pTime*)

Clear the time stamp.

**Parameters:**

- *pTime* Pointer to time value

### 5.32.2.4 EXT\_DECL INT32 vos\_cmpTime (const VOS\_TIME\_T \**pTime*, const VOS\_TIME\_T \**pCmp*)

Compare the second from the first time stamp, return diff in first.

**Parameters:**

- ↔ *pTime* Pointer to time value
- ← *pCmp* Pointer to time value to compare

**Return values:**

- 0* *pTime* == *pCmp*
- 1* *pTime* < *pCmp*
- 1* *pTime* > *pCmp*

**5.32.2.5 EXT\_DECL void vos\_divTime (VOS\_TIME\_T \* *pTime*, UINT32 *divisor*)**

Divide the first time value by the second, return quotient in first.

Divide the first time by the second, return quotient in first.

**Parameters:**

↔ *pTime* Pointer to time value

← *divisor* Divisor

**5.32.2.6 EXT\_DECL void vos\_getTime (VOS\_TIME\_T \* *pTime*)**

Return the current time in sec and us.

**Parameters:**

→ *pTime* Pointer to time value

**5.32.2.7 EXT\_DECL const CHAR8\* vos\_getTimeStamp (void)**

Get a time-stamp string.

Get a time-stamp string for debugging in the form "yyyymmdd-hh:mm:ss.ms" Depending on the used OS / hardware the time might not be a real-time stamp but relative from start of system.

**Return values:**

*timestamp* "yyyymmdd-hh:mm:ss.ms"

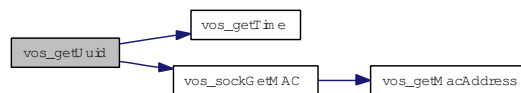
**5.32.2.8 EXT\_DECL void vos\_getUuid (VOS\_UUID\_T *pUUID*)**

Get a universal unique identifier according to RFC 4122 time based version.

**Parameters:**

→ *pUUID* Pointer to a universal unique identifier

Here is the call graph for this function:

**5.32.2.9 EXT\_DECL void vos\_mulTime (VOS\_TIME\_T \* *pTime*, UINT32 *mul*)**

Multiply the first time by the second, return product in first.

**Parameters:**

↔ *pTime* Pointer to time value

← *mul* Factor

### 5.32.2.10 EXT\_DECL VOS\_ERR\_T vos\_mutexCreate (VOS\_MUTEX\_T \* *pMutex*)

Create a recursive mutex.

Create a mutex.

Return a mutex handle. The mutex will be available at creation.

#### Parameters:

→ *pMutex* Pointer to mutex handle

#### Return values:

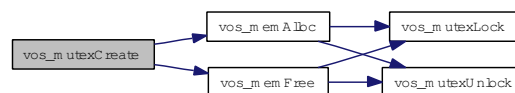
**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_PARAM\_ERR** *pMutex* == NULL

**VOS\_MUTEX\_ERR** no mutex available

Here is the call graph for this function:



### 5.32.2.11 EXT\_DECL void vos\_mutexDelete (VOS\_MUTEX\_T *pMutex*)

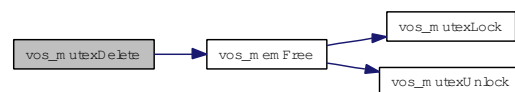
Delete a mutex.

Release the resources taken by the mutex.

#### Parameters:

← *pMutex* mutex handle

Here is the call graph for this function:



### 5.32.2.12 EXT\_DECL VOS\_ERR\_T vos\_mutexLocalCreate (struct VOS\_MUTEX \* *pMutex*)

Create a recursive mutex.

Fill in a mutex handle. The mutex storage must be already allocated.

#### Parameters:

→ *pMutex* Pointer to mutex handle

**Return values:**

*VOS\_NO\_ERR* no error  
*VOS\_INIT\_ERR* module not initialised  
*VOS\_PARAM\_ERR* pMutex == NULL  
*VOS\_MUTEX\_ERR* no mutex available

**5.32.2.13 EXT\_DECL void vos\_mutexLocalDelete (struct VOS\_MUTEX \*pMutex)**

Delete a mutex.

Release the resources taken by the mutex.

**Parameters:**

← *pMutex* Pointer to mutex struct

**5.32.2.14 EXT\_DECL VOS\_ERR\_T vos\_mutexLock (VOS\_MUTEX\_T pMutex)**

Take a mutex.

Wait for the mutex to become available (lock).

**Parameters:**

← *pMutex* mutex handle

**Return values:**

*VOS\_NO\_ERR* no error  
*VOS\_PARAM\_ERR* pMutex == NULL or wrong type  
*VOS\_MUTEX\_ERR* no such mutex

**5.32.2.15 EXT\_DECL VOS\_ERR\_T vos\_mutexTryLock (VOS\_MUTEX\_T pMutex)**

Try to take a mutex.

If mutex is can't be taken VOS\_MUTEX\_ERR is returned.

**Parameters:**

← *pMutex* mutex handle

**Return values:**

*VOS\_NO\_ERR* no error  
*VOS\_PARAM\_ERR* pMutex == NULL or wrong type  
*VOS\_MUTEX\_ERR* mutex not locked

### 5.32.2.16 EXT\_DECL VOS\_ERR\_T vos\_mutexUnlock (VOS\_MUTEX\_T *pMutex*)

Release a mutex.

Unlock the mutex.

#### Parameters:

← *pMutex* mutex handle

### 5.32.2.17 EXT\_DECL VOS\_ERR\_T vos\_semaCreate (VOS\_SEMA\_T \* *pSema*, VOS\_SEMA\_STATE\_T *initialState*)

Create a semaphore.

Return a semaphore handle. Depending on the initial state the semaphore will be available on creation or not.

#### Parameters:

→ *pSema* Pointer to semaphore handle

← *initialState* The initial state of the semaphore

#### Return values:

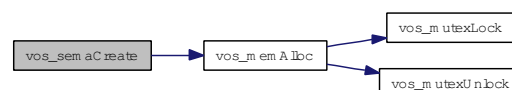
**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_SEMA\_ERR** no semaphore available

Here is the call graph for this function:



### 5.32.2.18 EXT\_DECL void vos\_semaDelete (VOS\_SEMA\_T *sema*)

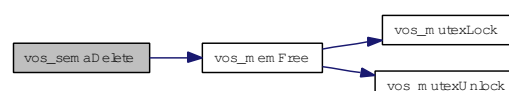
Delete a semaphore.

This will eventually release any processes waiting for the semaphore.

#### Parameters:

← *sema* semaphore handle

Here is the call graph for this function:



**5.32.2.19 EXT\_DECL void vos\_semaGive (VOS\_SEMA\_T *sema*)**

Give a semaphore.

Release (increase) a semaphore.

**Parameters:**

← *sema* semaphore handle

**5.32.2.20 EXT\_DECL VOS\_ERR\_T vos\_semaTake (VOS\_SEMA\_T *sema*, UINT32 *timeout*)**

Take a semaphore.

Try to get (decrease) a semaphore.

**Parameters:**

← *sema* semaphore handle

← *timeout* Max. time in us to wait, 0 means no wait

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

**VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_SEMA\_ERR** could not get semaphore in time

Here is the call graph for this function:

**5.32.2.21 EXT\_DECL void vos\_subTime (VOS\_TIME\_T \**pTime*, const VOS\_TIME\_T \**pSub*)**

Subtract the second from the first time stamp, return diff in first.

**Parameters:**

↔ *pTime* Pointer to time value

← *pSub* Pointer to time value

**5.32.2.22 EXT\_DECL VOS\_ERR\_T vos\_threadCreate (VOS\_THREAD\_T \**pThread*, const CHAR8 \**pName*, VOS\_THREAD\_POLICY\_T *policy*, VOS\_THREAD\_PRIORITY\_T *priority*, UINT32 *interval*, UINT32 *stackSize*, VOS\_THREAD\_FUNC\_T *pFunction*, void \**pArguments*)**

Create a thread.

Create a thread and return a thread handle for further requests. Not each parameter may be supported by all target systems!

**Parameters:**

- *pThread* Pointer to returned thread handle
- ← *pName* Pointer to name of the thread (optional)
- ← *policy* Scheduling policy (FIFO, Round Robin or other)
- ← *priority* Scheduling priority (1...255 (highest), default 0)
- ← *interval* Interval for cyclic threads in us (optional)
- ← *stackSize* Minimum stacksize, default 0: 16kB
- ← *pFunction* Pointer to the thread function
- ← *pArguments* Pointer to the thread function parameters

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_INIT\_ERR* module not initialised
- VOS\_NOINIT\_ERR* invalid handle
- VOS\_PARAM\_ERR* parameter out of range/invalid
- VOS\_THREAD\_ERR* thread creation error

**5.32.2.23 EXT\_DECL VOS\_ERR\_T vos\_threadDelay (UINT32 delay)**

Delay the execution of the current thread by the given delay in us.

**Parameters:**

- ← *delay* Delay in us

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* parameter out of range/invalid

**5.32.2.24 EXT\_DECL VOS\_ERR\_T vos\_threadInit (void)**

Initialize the thread library.

Must be called once before any other call

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_INIT\_ERR* threading not supported



**5.32.2.25 EXT\_DECL VOS\_ERR\_T vos\_threadIsActive (VOS\_THREAD\_T *thread*)**

Is the thread still active? This call will return VOS\_NO\_ERR if the thread is still active, VOS\_PARAM\_ERR in case it ran out.

**Parameters:**

← *thread* Thread handle

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** parameter out of range/invalid

**5.32.2.26 EXT\_DECL void vos\_threadTerm (void)**

De-Initialize the thread library.

Must be called after last thread/timer call

**5.32.2.27 EXT\_DECL VOS\_ERR\_T vos\_threadTerminate (VOS\_THREAD\_T *thread*)**

Terminate a thread.

This call will terminate the thread with the given threadId and release all resources. Depending on the underlying architectures, it may just block until the thread ran out.

**Parameters:**

← *thread* Thread handle (or NULL if current thread)

**Return values:**

**VOS\_NO\_ERR** no error

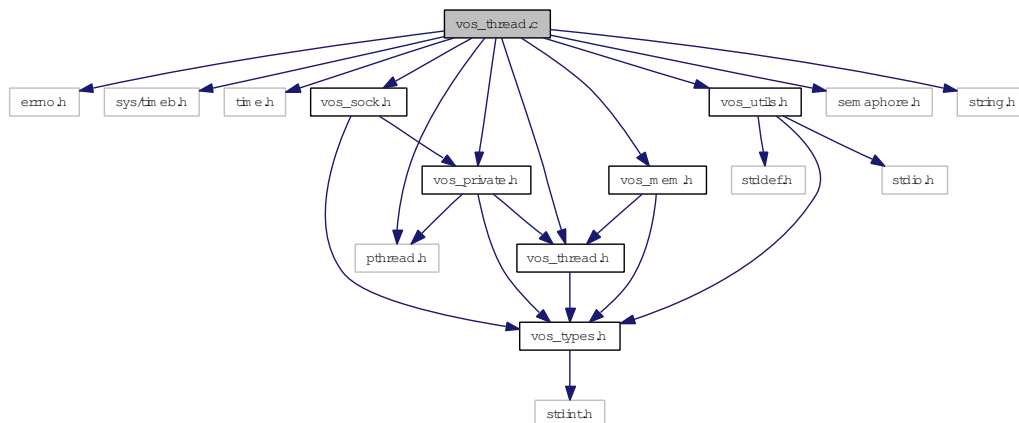
**VOS\_THREAD\_ERR** cancel failed

## 5.33 vos\_thread.c File Reference

Multitasking functions.

```
#include <errno.h>
#include <sys/timeb.h>
#include <time.h>
#include <pthread.h>
#include <semaphore.h>
#include <string.h>
#include "vos_thread.h"
#include "vos_sock.h"
#include "vos_mem.h"
#include "vos_utils.h"
#include "vos_private.h"
```

Include dependency graph for windows/vos\_thread.c:



## Functions

- void [cyclicThread](#) (UINT32 interval, [VOS\\_THREAD\\_FUNC\\_T](#) pFunction, void \*pArguments)  
*Cyclic thread functions.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_threadInit](#) (void)  
*Initialize the thread library.*
- EXT\_DECL void [vos\\_threadTerm](#) (void)  
*De-Initialize the thread library.*
- pthread\_t \* [vos\\_getFreeThreadHandle](#) (void)  
*Search a free Handle place in the thread handle list.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_threadCreate](#) ([VOS\\_THREAD\\_T](#) \*pThread, const [CHAR8](#) \*pName, [VOS\\_THREAD\\_POLICY\\_T](#) policy, [VOS\\_THREAD\\_PRIORITY\\_T](#) priority, [UINT32](#) interval, [UINT32](#) stackSize, [VOS\\_THREAD\\_FUNC\\_T](#) pFunction, void \*pArguments)  
*Create a thread.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_threadTerminate](#) ([VOS\\_THREAD\\_T](#) thread)  
*Terminate a thread.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_threadIsActive](#) ([VOS\\_THREAD\\_T](#) thread)  
*Is the thread still active? This call will return VOS\_NO\_ERR if the thread is still active, VOS\_PARAM\_ERR in case it ran out.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_threadDelay](#) ([UINT32](#) delay)  
*Delay the execution of the current thread by the given delay in us.*
- EXT\_DECL void [vos\\_getTime](#) ([VOS\\_TIME\\_T](#) \*pTime)  
*Return the current time in sec and us.*
- EXT\_DECL const [CHAR8](#) \* [vos\\_getTimeStamp](#) (void)  
*Get a time-stamp string.*
- EXT\_DECL void [vos\\_clearTime](#) ([VOS\\_TIME\\_T](#) \*pTime)  
*Clear the time stamp.*
- EXT\_DECL void [vos\\_addTime](#) ([VOS\\_TIME\\_T](#) \*pTime, const [VOS\\_TIME\\_T](#) \*pAdd)  
*Add the second to the first time stamp, return sum in first.*
- EXT\_DECL void [vos\\_subTime](#) ([VOS\\_TIME\\_T](#) \*pTime, const [VOS\\_TIME\\_T](#) \*pSub)  
*Subtract the second from the first time stamp, return diff in first.*
- EXT\_DECL void [vos\\_divTime](#) ([VOS\\_TIME\\_T](#) \*pTime, [UINT32](#) divisor)  
*Divide the first time value by the second, return quotient in first.*
- EXT\_DECL void [vos\\_mulTime](#) ([VOS\\_TIME\\_T](#) \*pTime, [UINT32](#) mul)  
*Multiply the first time by the second, return product in first.*
- EXT\_DECL [INT32](#) [vos\\_cmpTime](#) (const [VOS\\_TIME\\_T](#) \*pTime, const [VOS\\_TIME\\_T](#) \*pCmp)  
*Compare the second from the first time stamp, return diff in first.*
- EXT\_DECL void [vos\\_getUuid](#) ([VOS\\_UUID\\_T](#) pUuid)  
*Get a universal unique identifier according to RFC 4122 time based version.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_mutexCreate](#) ([VOS\\_MUTEX\\_T](#) \*pMutex)  
*Create a recursive mutex.*
- [VOS\\_ERR\\_T vos\\_mutexLocalCreate](#) (struct [VOS\\_MUTEX](#) \*pMutex)  
*Create a recursive mutex.*
- EXT\_DECL void [vos\\_mutexDelete](#) ([VOS\\_MUTEX\\_T](#) pMutex)  
*Delete a mutex.*

- void `vos_mutexLocalDelete` (struct `VOS_MUTEX` \*pMutex)  
*Delete a mutex.*
- EXT\_DECL `VOS_ERR_T vos_mutexLock` (`VOS_MUTEX_T` pMutex)  
*Take a mutex.*
- EXT\_DECL `VOS_ERR_T vos_mutexTryLock` (`VOS_MUTEX_T` pMutex)  
*Try to take a mutex.*
- EXT\_DECL `VOS_ERR_T vos_mutexUnlock` (`VOS_MUTEX_T` pMutex)  
*Release a mutex.*
- EXT\_DECL `VOS_ERR_T vos_semaCreate` (`VOS_SEMA_T` \*pSema, `VOS_SEMA_STATE_T` initialState)  
*Create a semaphore.*
- EXT\_DECL void `vos_semaDelete` (`VOS_SEMA_T` sema)  
*Delete a semaphore.*
- EXT\_DECL `VOS_ERR_T vos_semaTake` (`VOS_SEMA_T` sema, `UINT32` timeout)  
*Take a semaphore.*
- EXT\_DECL void `vos_semaGive` (`VOS_SEMA_T` sema)  
*Give a semaphore.*

### 5.33.1 Detailed Description

Multitasking functions.

OS abstraction of thread-handling functions

**Note:**

Project: TCNOpen TRDP prototype stack

**Author:**

Bernd Loehr, NewTec GmbH

**Remarks:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2013. `vos_thread.c` uses `pthreads-w32` (<http://sourceware.org/pthreads-win32/>) under LGPL license

**Id**

`vos_thread.c` 1014 2013-07-04 06:51:27Z cschneider

## 5.33.2 Function Documentation

### 5.33.2.1 void cyclicThread (UINT32 *interval*, VOS\_THREAD\_FUNC\_T *pFunction*, void \* *pArguments*)

Cyclic thread functions.

Wrapper for cyclic threads. The thread function will be called cyclically with interval.

#### Parameters:

- ← *interval* Interval for cyclic threads in us (optional)
- ← *pFunction* Pointer to the thread function
- ← *pArguments* Pointer to the thread function parameters

#### Return values:

*void*

Here is the call graph for this function:



### 5.33.2.2 EXT\_DECL void vos\_addTime (VOS\_TIME\_T \* *pTime*, const VOS\_TIME\_T \* *pAdd*)

Add the second to the first time stamp, return sum in first.

#### Parameters:

- ↔ *pTime* Pointer to time value
- ← *pAdd* Pointer to time value

### 5.33.2.3 EXT\_DECL void vos\_clearTime (VOS\_TIME\_T \* *pTime*)

Clear the time stamp.

#### Parameters:

- *pTime* Pointer to time value

### 5.33.2.4 EXT\_DECL INT32 vos\_cmpTime (const VOS\_TIME\_T \* *pTime*, const VOS\_TIME\_T \* *pCmp*)

Compare the second from the first time stamp, return diff in first.

#### Parameters:

- ↔ *pTime* Pointer to time value

← *pCmp* Pointer to time value to compare

**Return values:**

*0* *pTime* == *pCmp*

*-1* *pTime* < *pCmp*

*1* *pTime* > *pCmp*

### 5.33.2.5 EXT\_DECL void vos\_divTime (VOS\_TIME\_T \**pTime*, UINT32 *divisor*)

Divide the first time value by the second, return quotient in first.

Divide the first time by the second, return quotient in first.

**Parameters:**

↔ *pTime* Pointer to time value

← *divisor* Divisor

### 5.33.2.6 pthread\_t\* vos\_getFreeThreadHandle (void)

Search a free Handle place in the thread handle list.

**Return values:**

*pointer* to a free thread handle or NULL if not available

### 5.33.2.7 EXT\_DECL void vos\_getTime (VOS\_TIME\_T \**pTime*)

Return the current time in sec and us.

**Parameters:**

→ *pTime* Pointer to time value

### 5.33.2.8 EXT\_DECL const CHAR8\* vos\_getTimeStamp (void)

Get a time-stamp string.

Get a time-stamp string for debugging in the form "yyyymmdd-hh:mm:ss.ms" Depending on the used OS / hardware the time might not be a real-time stamp but relative from start of system.

**Return values:**

*timestamp* "yyyymmdd-hh:mm:ss.ms"

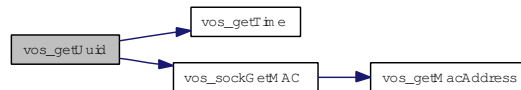
**5.33.2.9 EXT\_DECL void vos\_getUuid (VOS\_UUID\_T *pUuid*)**

Get a universal unique identifier according to RFC 4122 time based version.

**Parameters:**

→ *pUuid* Pointer to a universal unique identifier

Here is the call graph for this function:

**5.33.2.10 EXT\_DECL void vos\_mulTime (VOS\_TIME\_T \**pTime*, UINT32 *mul*)**

Multiply the first time by the second, return product in first.

**Parameters:**

↔ *pTime* Pointer to time value

← *mul* Factor

**5.33.2.11 EXT\_DECL VOS\_ERR\_T vos\_mutexCreate (VOS\_MUTEX\_T \**pMutex*)**

Create a recursive mutex.

Create a mutex.

Return a mutex handle. The mutex will be available at creation.

**Parameters:**

→ *pMutex* Pointer to mutex handle

**Return values:**

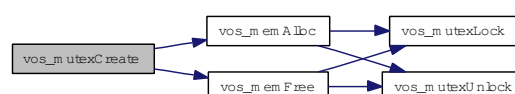
**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_PARAM\_ERR** *pMutex* == NULL

**VOS\_MUTEX\_ERR** no mutex available

Here is the call graph for this function:



### 5.33.2.12 EXT\_DECL void vos\_mutexDelete (VOS\_MUTEX\_T *pMutex*)

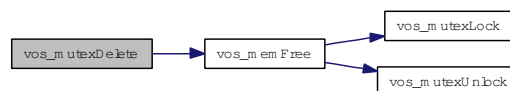
Delete a mutex.

Release the resources taken by the mutex.

#### Parameters:

← *pMutex* mutex handle

Here is the call graph for this function:



### 5.33.2.13 VOS\_ERR\_T vos\_mutexLocalCreate (struct VOS\_MUTEX \* *pMutex*)

Create a recursive mutex.

Fill in a mutex handle. The mutex storage must be already allocated.

#### Parameters:

→ *pMutex* Pointer to mutex handle

#### Return values:

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_PARAM\_ERR** *pMutex* == NULL

**VOS\_MUTEX\_ERR** no mutex available

### 5.33.2.14 void vos\_mutexLocalDelete (struct VOS\_MUTEX \* *pMutex*)

Delete a mutex.

Release the resources taken by the mutex.

#### Parameters:

← *pMutex* Pointer to mutex struct

### 5.33.2.15 EXT\_DECL VOS\_ERR\_T vos\_mutexLock (VOS\_MUTEX\_T *pMutex*)

Take a mutex.

Wait for the mutex to become available (lock).

#### Parameters:

← *pMutex* mutex handle



**Return values:**

**VOS\_NO\_ERR** no error  
**VOS\_PARAM\_ERR** pMutex == NULL or wrong type  
**VOS\_MUTEX\_ERR** no such mutex

**5.33.2.16 EXT\_DECL VOS\_ERR\_T vos\_mutexTryLock (VOS\_MUTEX\_T pMutex)**

Try to take a mutex.

If mutex is can't be taken VOS\_MUTEX\_ERR is returned.

**Parameters:**

← **pMutex** mutex handle

**Return values:**

**VOS\_NO\_ERR** no error  
**VOS\_PARAM\_ERR** pMutex == NULL or wrong type  
**VOS\_MUTEX\_ERR** mutex not locked

**5.33.2.17 EXT\_DECL VOS\_ERR\_T vos\_mutexUnlock (VOS\_MUTEX\_T pMutex)**

Release a mutex.

Unlock the mutex.

**Parameters:**

← **pMutex** mutex handle

**5.33.2.18 EXT\_DECL VOS\_ERR\_T vos\_semaCreate (VOS\_SEMA\_T \* pSema, VOS\_SEMA\_STATE\_T initialState)**

Create a semaphore.

Return a semaphore handle. Depending on the initial state the semaphore will be available on creation or not.

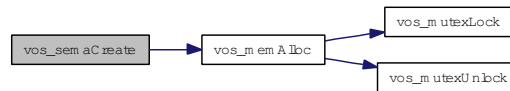
**Parameters:**

→ **pSema** Pointer to semaphore handle  
 ← **initialState** The initial state of the semaphore

**Return values:**

**VOS\_NO\_ERR** no error  
**VOS\_INIT\_ERR** module not initialised  
**VOS\_PARAM\_ERR** parameter out of range/invalid  
**VOS\_SEMA\_ERR** no semaphore available

Here is the call graph for this function:



#### 5.33.2.19 EXT\_DECL void vos\_semaDelete (VOS\_SEMA\_T *sema*)

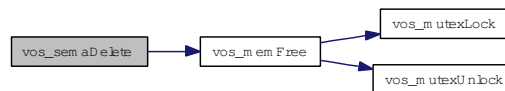
Delete a semaphore.

This will eventually release any processes waiting for the semaphore.

##### Parameters:

← *sema* semaphore handle

Here is the call graph for this function:



#### 5.33.2.20 EXT\_DECL void vos\_semaGive (VOS\_SEMA\_T *sema*)

Give a semaphore.

Release (increase) a semaphore.

##### Parameters:

← *sema* semaphore handle

#### 5.33.2.21 EXT\_DECL VOS\_ERR\_T vos\_semaTake (VOS\_SEMA\_T *sema*, UINT32 *timeout*)

Take a semaphore.

Try to get (decrease) a semaphore.

##### Parameters:

← *sema* semaphore handle

← *timeout* Max. time in us to wait, 0 means no wait

##### Return values:

**VOS\_NO\_ERR** no error

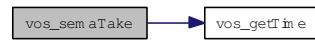
**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

**VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_SEMA\_ERR** could not get semaphore in time

Here is the call graph for this function:



### 5.33.2.22 EXT\_DECL void vos\_subTime (VOS\_TIME\_T \* *pTime*, const VOS\_TIME\_T \* *pSub*)

Subtract the second from the first time stamp, return diff in first.

#### Parameters:

↔ ***pTime*** Pointer to time value

← ***pSub*** Pointer to time value

### 5.33.2.23 EXT\_DECL VOS\_ERR\_T vos\_threadCreate (VOS\_THREAD\_T \* *pThread*, const CHAR8 \* *pName*, VOS\_THREAD\_POLICY\_T *policy*, VOS\_THREAD\_PRIORITY\_T *priority*, UINT32 *interval*, UINT32 *stackSize*, VOS\_THREAD\_FUNC\_T *pFunction*, void \* *pArguments*)

Create a thread.

Create a thread and return a thread handle for further requests. Not each parameter may be supported by all target systems!

#### Parameters:

→ ***pThread*** Pointer to returned thread handle

← ***pName*** Pointer to name of the thread (optional)

← ***policy*** Scheduling policy (FIFO, Round Robin or other)

← ***priority*** Scheduling priority (1...255 (highest), default 0)

← ***interval*** Interval for cyclic threads in us (optional)

← ***stackSize*** Minimum stacksize, default 0: 16kB

← ***pFunction*** Pointer to the thread function

← ***pArguments*** Pointer to the thread function parameters

#### Return values:

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

**VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_THREAD\_ERR** thread creation error

**VOS\_INIT\_ERR** no threads available

Here is the call graph for this function:



#### 5.33.2.24 EXT\_DECL VOS\_ERR\_T vos\_threadDelay (UINT32 *delay*)

Delay the execution of the current thread by the given delay in us.

##### Parameters:

← *delay* Delay in us

##### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** parameter out of range/invalid

#### 5.33.2.25 EXT\_DECL VOS\_ERR\_T vos\_threadInit (void)

Initialize the thread library.

Must be called once before any other call

##### Return values:

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** threading not supported

#### 5.33.2.26 EXT\_DECL VOS\_ERR\_T vos\_threadIsActive (VOS\_THREAD\_T *thread*)

Is the thread still active? This call will return VOS\_NO\_ERR if the thread is still active, VOS\_PARAM\_ERR in case it ran out.

##### Parameters:

← *thread* Thread handle

##### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** parameter out of range/invalid

#### 5.33.2.27 EXT\_DECL void vos\_threadTerm (void)

De-Initialize the thread library.

Must be called after last thread/timer call

**5.33.2.28 EXT\_DECL VOS\_ERR\_T vos\_threadTerminate (VOS\_THREAD\_T *thread*)**

Terminate a thread.

This call will terminate the thread with the given threadId and release all resources. Depending on the underlying architectures, it may just block until the thread ran out.

**Parameters:**

← *thread* Thread handle (or NULL if current thread)

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_THREAD\_ERR* cancel failed



- typedef struct VOS\_SEMA \* [VOS\\_SEMA\\_T](#)  
*Hidden semaphore handle definition.*
- typedef void \* [VOS\\_THREAD\\_T](#)  
*Hidden thread handle definition.*

## Enumerations

- enum [VOS\\_THREAD\\_POLICY\\_T](#)  
*Thread policy matching pthread/Posix defines.*
- enum [VOS\\_SEMA\\_STATE\\_T](#)  
*State of the semaphore.*

## Functions

- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_threadInit](#) (void)  
*Initialize the thread library.*
- EXT\_DECL void [vos\\_threadTerm](#) (void)  
*De-Initialize the thread library.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_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.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_threadTerminate](#) ([VOS\\_THREAD\\_T](#) thread)  
*Terminate a thread.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_threadIsActive](#) ([VOS\\_THREAD\\_T](#) thread)  
*Is the thread still active? This call will return VOS\_NO\_ERR if the thread is still active, VOS\_PARAM\_ERR in case it ran out.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_threadDelay](#) (UINT32 delay)  
*Delay the execution of the current thread by the given delay in us.*
- EXT\_DECL void [vos\\_getTime](#) ([VOS\\_TIME\\_T](#) \*pTime)  
*Return the current time in sec and us.*
- EXT\_DECL const CHAR8 \* [vos\\_getTimeStamp](#) (void)  
*Get a time-stamp string.*
- EXT\_DECL void [vos\\_clearTime](#) ([VOS\\_TIME\\_T](#) \*pTime)  
*Clear the time stamp.*

- EXT\_DECL void [vos\\_addTime](#) (VOS\_TIME\_T \*pTime, const VOS\_TIME\_T \*pAdd)  
*Add the second to the first time stamp, return sum in first.*
- EXT\_DECL void [vos\\_subTime](#) (VOS\_TIME\_T \*pTime, const VOS\_TIME\_T \*pSub)  
*Subtract the second from the first time stamp, return diff in first.*
- EXT\_DECL INT32 [vos\\_cmpTime](#) (const VOS\_TIME\_T \*pTime, const VOS\_TIME\_T \*pCmp)  
*Compare the second from the first time stamp, return diff in first.*
- EXT\_DECL void [vos\\_divTime](#) (VOS\_TIME\_T \*pTime, UINT32 divisor)  
*Divide the first time by the second, return quotient in first.*
- EXT\_DECL void [vos\\_mulTime](#) (VOS\_TIME\_T \*pTime, UINT32 mul)  
*Multiply the first time by the second, return product in first.*
- EXT\_DECL void [vos\\_getUuid](#) (VOS\_UUID\_T pUUID)  
*Get a universal unique identifier according to RFC 4122 time based version.*
- EXT\_DECL VOS\_ERR\_T [vos\\_mutexCreate](#) (VOS\_MUTEX\_T \*pMutex)  
*Create a mutex.*
- EXT\_DECL void [vos\\_mutexDelete](#) (VOS\_MUTEX\_T pMutex)  
*Delete a mutex.*
- EXT\_DECL VOS\_ERR\_T [vos\\_mutexLock](#) (VOS\_MUTEX\_T pMutex)  
*Take a mutex.*
- EXT\_DECL VOS\_ERR\_T [vos\\_mutexTryLock](#) (VOS\_MUTEX\_T pMutex)  
*Try to take a mutex.*
- EXT\_DECL VOS\_ERR\_T [vos\\_mutexUnlock](#) (VOS\_MUTEX\_T pMutex)  
*Release a mutex.*
- EXT\_DECL VOS\_ERR\_T [vos\\_semaCreate](#) (VOS\_SEMA\_T \*pSema, VOS\_SEMA\_STATE\_T initialState)  
*Create a semaphore.*
- EXT\_DECL void [vos\\_semaDelete](#) (VOS\_SEMA\_T sema)  
*Delete a semaphore.*
- EXT\_DECL VOS\_ERR\_T [vos\\_semaTake](#) (VOS\_SEMA\_T sema, UINT32 timeout)  
*Take a semaphore.*
- EXT\_DECL void [vos\\_semaGive](#) (VOS\_SEMA\_T sema)  
*Give a semaphore.*



### 5.34.1 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:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

[vos\\_thread.h](#) 999 2013-07-02 08:51:14Z bloehr

### 5.34.2 Function Documentation

#### 5.34.2.1 EXT\_DECL void vos\_addTime (VOS\_TIME\_T \* *pTime*, const VOS\_TIME\_T \* *pAdd*)

Add the second to the first time stamp, return sum in first.

**Parameters:**

↔ *pTime* Pointer to time value

← *pAdd* Pointer to time value

↔ *pTime* Pointer to time value

← *pAdd* Pointer to time value

#### 5.34.2.2 EXT\_DECL void vos\_clearTime (VOS\_TIME\_T \* *pTime*)

Clear the time stamp.

**Parameters:**

→ *pTime* Pointer to time value

→ *pTime* Pointer to time value

#### 5.34.2.3 EXT\_DECL INT32 vos\_cmpTime (const VOS\_TIME\_T \* *pTime*, const VOS\_TIME\_T \* *pCmp*)

Compare the second from the first time stamp, return diff in first.

**Parameters:**

↔ *pTime* Pointer to time value

← *pCmp* Pointer to time value to compare

**Return values:**

*0* *pTime* == *pCmp*

*-1* *pTime* < *pCmp*

*1* *pTime* > *pCmp*

**Parameters:**

↔ *pTime* Pointer to time value

← *pCmp* Pointer to time value to compare

**Return values:**

*0* *pTime* == *pCmp*

*-1* *pTime* < *pCmp*

*1* *pTime* > *pCmp*

#### 5.34.2.4 EXT\_DECL void vos\_divTime (VOS\_TIME\_T \* *pTime*, UINT32 *divisor*)

Divide the first time by the second, return quotient in first.

**Parameters:**

↔ *pTime* Pointer to time value

← *divisor* Divisor

Divide the first time by the second, return quotient in first.

**Parameters:**

↔ *pTime* Pointer to time value

← *divisor* Divisor

#### 5.34.2.5 EXT\_DECL void vos\_getTime (VOS\_TIME\_T \* *pTime*)

Return the current time in sec and us.

**Parameters:**

→ *pTime* Pointer to time value

→ *pTime* Pointer to time value

**5.34.2.6 EXT\_DECL const CHAR8\* vos\_getTimeStamp (void)**

Get a time-stamp string.

Get a time-stamp string for debugging in the form "yyyymmdd-hh:mm:ss.ms" Depending on the used OS / hardware the time might not be a real-time stamp but relative from start of system.

**Return values:**

*timestamp* "yyyymmdd-hh:mm:ss.ms"

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:**

*timestamp* "yyyymmdd-hh:mm:ss.ms"

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:**

*timestamp* "yyyymmdd-hh:mm:ss.ms"

**5.34.2.7 EXT\_DECL void vos\_getUuid (VOS\_UUID\_T pUUID)**

Get a universal unique identifier according to RFC 4122 time based version.

**Parameters:**

→ *pUUID* Pointer to a universal unique identifier

→ *pUUID* Pointer to a universal unique identifier

Here is the call graph for this function:

**5.34.2.8 EXT\_DECL void vos\_mulTime (VOS\_TIME\_T \*pTime, UINT32 mul)**

Multiply the first time by the second, return product in first.

**Parameters:**

↔ *pTime* Pointer to time value

← *mul* Factor

### 5.34.2.9 EXT\_DECL VOS\_ERR\_T vos\_mutexCreate (VOS\_MUTEX\_T \* *pMutex*)

Create a mutex.

Return a mutex handle. The mutex will be available at creation.

#### Parameters:

→ *pMutex* Pointer to mutex handle

#### Return values:

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_PARAM\_ERR** *pMutex* == NULL

**VOS\_MUTEX\_ERR** no mutex available

Create a mutex.

Return a mutex handle. The mutex will be available at creation.

#### Parameters:

→ *pMutex* Pointer to mutex handle

#### Return values:

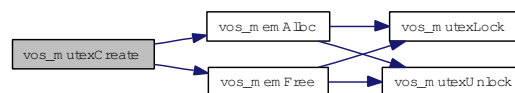
**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_PARAM\_ERR** *pMutex* == NULL

**VOS\_MUTEX\_ERR** no mutex available

Here is the call graph for this function:



### 5.34.2.10 EXT\_DECL void vos\_mutexDelete (VOS\_MUTEX\_T *pMutex*)

Delete a mutex.

Release the resources taken by the mutex.

#### Parameters:

← *pMutex* mutex handle

#### Return values:

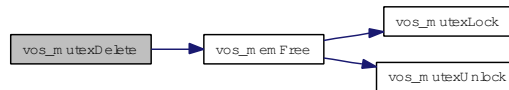
**VOS\_NO\_ERR** no error

Release the resources taken by the mutex.

**Parameters:**

← *pMutex* mutex handle

Here is the call graph for this function:



### 5.34.2.11 EXT\_DECL VOS\_ERR\_T vos\_mutexLock (VOS\_MUTEX\_T *pMutex*)

Take a mutex.

Wait for the mutex to become available (lock).

**Parameters:**

← *pMutex* mutex handle

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

Wait for the mutex to become available (lock).

**Parameters:**

← *pMutex* mutex handle

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** *pMutex* == NULL or wrong type

**VOS\_MUTEX\_ERR** no such mutex

### 5.34.2.12 EXT\_DECL VOS\_ERR\_T vos\_mutexTryLock (VOS\_MUTEX\_T *pMutex*)

Try to take a mutex.

If mutex is can't be taken VOS\_MUTEX\_ERR is returned.

**Parameters:**

← *pMutex* mutex handle

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

**VOS\_MUTEX\_ERR** no mutex available

If mutex is can't be taken VOS\_MUTEX\_ERR is returned.

**Parameters:**

← **pMutex** mutex handle

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pMutex == NULL or wrong type

**VOS\_MUTEX\_ERR** mutex not locked

### 5.34.2.13 EXT\_DECL VOS\_ERR\_T vos\_mutexUnlock (VOS\_MUTEX\_T pMutex)

Release a mutex.

Unlock the mutex.

**Parameters:**

← **pMutex** mutex handle

Unlock the mutex.

**Parameters:**

← **pMutex** mutex handle

Unlock the mutex.

**Parameters:**

← **pMutex** mutex handle

### 5.34.2.14 EXT\_DECL VOS\_ERR\_T vos\_semaCreate (VOS\_SEMA\_T \* pSema, VOS\_SEMA\_STATE\_T initialState)

Create a semaphore.

Return a semaphore handle. Depending on the initial state the semaphore will be available on creation or not.

**Parameters:**

→ **pSema** Pointer to semaphore handle

← **initialState** The initial state of the sempahore

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised  
**VOS\_PARAM\_ERR** parameter out of range/invalid  
**VOS\_SEMA\_ERR** no semaphore available

Return a semaphore handle. Depending on the initial state the semaphore will be available on creation or not.

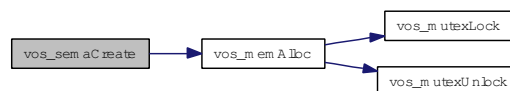
**Parameters:**

→ **pSema** Pointer to semaphore handle  
 ← **initialState** The initial state of the semaphore

**Return values:**

**VOS\_NO\_ERR** no error  
**VOS\_INIT\_ERR** module not initialised  
**VOS\_PARAM\_ERR** parameter out of range/invalid  
**VOS\_SEMA\_ERR** no semaphore available

Here is the call graph for this function:



#### 5.34.2.15 EXT\_DECL void vos\_semaphoreDelete (VOS\_SEMA\_T sema)

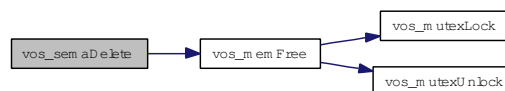
Delete a semaphore.

This will eventually release any processes waiting for the semaphore.

**Parameters:**

← **sema** semaphore handle

Here is the call graph for this function:



#### 5.34.2.16 EXT\_DECL void vos\_semaphoreGive (VOS\_SEMA\_T sema)

Give a semaphore.

Release (increase) a semaphore.

**Parameters:**

← **sema** semaphore handle

### 5.34.2.17 EXT\_DECL VOS\_ERR\_T vos\_semaTake (VOS\_SEMA\_T *sema*, UINT32 *timeout*)

Take a semaphore.

Try to get (decrease) a semaphore.

#### Parameters:

- ← *sema* semaphore handle
- ← *timeout* Max. time in us to wait, 0 means no wait

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_INIT\_ERR** module not initialised
- VOS\_NOINIT\_ERR** invalid handle
- VOS\_PARAM\_ERR** parameter out of range/invalid
- VOS\_SEMA\_ERR** could not get semaphore in time

Try to get (decrease) a semaphore.

#### Parameters:

- ← *sema* semaphore handle
- ← *timeout* Max. time in us to wait, 0 means no wait

#### Return values:

- VOS\_NO\_ERR** no error
- VOS\_INIT\_ERR** module not initialised
- VOS\_NOINIT\_ERR** invalid handle
- VOS\_PARAM\_ERR** parameter out of range/invalid
- VOS\_SEMA\_ERR** could not get semaphore in time

Here is the call graph for this function:



### 5.34.2.18 EXT\_DECL void vos\_subTime (VOS\_TIME\_T \**pTime*, const VOS\_TIME\_T \**pSub*)

Subtract the second from the first time stamp, return diff in first.

#### Parameters:

- ↔ *pTime* Pointer to time value
- ← *pSub* Pointer to time value
- ↔ *pTime* Pointer to time value
- ← *pSub* Pointer to time value



**5.34.2.19** `EXT_DECL VOS_ERR_T vos_threadCreate (VOS_THREAD_T * pThread, const CHAR8 * pName, VOS_THREAD_POLICY_T policy, VOS_THREAD_PRIORITY_T priority, UINT32 interval, UINT32 stackSize, VOS_THREAD_FUNC_T pFunction, void * pArguments)`

Create a thread.

Create a thread and return a thread handle for further requests. Not each parameter may be supported by all target systems!

**Parameters:**

- *pThread* Pointer to returned thread handle
- ← *pName* Pointer to name of the thread (optional)
- ← *policy* Scheduling policy (FIFO, Round Robin or other)
- ← *priority* Scheduling priority (1...255 (highest), default 0)
- ← *interval* Interval for cyclic threads in us (optional)
- ← *stackSize* Minimum stacksize, default 0: 16kB
- ← *pFunction* Pointer to the thread function
- ← *pArguments* Pointer to the thread function parameters

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_INIT\_ERR* module not initialised
- VOS\_NOINIT\_ERR* invalid handle
- VOS\_PARAM\_ERR* parameter out of range/invalid

Create a thread and return a thread handle for further requests. Not each parameter may be supported by all target systems!

**Parameters:**

- *pThread* Pointer to returned thread handle
- ← *pName* Pointer to name of the thread (optional)
- ← *policy* Scheduling policy (FIFO, Round Robin or other)
- ← *priority* Scheduling priority (1...255 (highest), default 0)
- ← *interval* Interval for cyclic threads in us (optional)
- ← *stackSize* Minimum stacksize, default 0: 16kB
- ← *pFunction* Pointer to the thread function
- ← *pArguments* Pointer to the thread function parameters

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_INIT\_ERR* module not initialised
- VOS\_NOINIT\_ERR* invalid handle
- VOS\_PARAM\_ERR* parameter out of range/invalid
- VOS\_THREAD\_ERR* thread creation error

Create a thread and return a thread handle for further requests. Not each parameter may be supported by all target systems!

**Parameters:**

- *pThread* Pointer to returned thread handle
- ← *pName* Pointer to name of the thread (optional)
- ← *policy* Scheduling policy (FIFO, Round Robin or other)
- ← *priority* Scheduling priority (1...255 (highest), default 0)
- ← *interval* Interval for cyclic threads in us (optional)
- ← *stackSize* Minimum stacksize, default 0: 16kB
- ← *pFunction* Pointer to the thread function
- ← *pArguments* Pointer to the thread function parameters

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_INIT\_ERR* module not initialised
- VOS\_NOINIT\_ERR* invalid handle
- VOS\_PARAM\_ERR* parameter out of range/invalid
- VOS\_THREAD\_ERR* thread creation error
- VOS\_INIT\_ERR* no threads available

Here is the call graph for this function:



### 5.34.2.20 EXT\_DECL VOS\_ERR\_T vos\_threadDelay (UINT32 delay)

Delay the execution of the current thread by the given delay in us.

**Parameters:**

- ← *delay* Delay in us

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_INIT\_ERR* module not initialised

**Parameters:**

- ← *delay* Delay in us

**Return values:**

- VOS\_NO\_ERR* no error
- VOS\_PARAM\_ERR* parameter out of range/invalid

**5.34.2.21 EXT\_DECL VOS\_ERR\_T vos\_threadInit (void)**

Initialize the thread library.

Must be called once before any other call

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** threading not supported

Must be called once before any other call

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** threading not supported

Must be called once before any other call

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** threading not supported

**5.34.2.22 EXT\_DECL VOS\_ERR\_T vos\_threadIsActive (VOS\_THREAD\_T thread)**

Is the thread still active? This call will return VOS\_NO\_ERR if the thread is still active, VOS\_PARAM\_ERR in case it ran out.

**Parameters:**

← *thread* Thread handle

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

**VOS\_PARAM\_ERR** parameter out of range/invalid

**Parameters:**

← *thread* Thread handle

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** parameter out of range/invalid

**5.34.2.23 EXT\_DECL void vos\_threadTerm (void)**

De-Initialize the thread library.

Must be called after last thread/timer call

#### 5.34.2.24 EXT\_DECL VOS\_ERR\_T vos\_threadTerminate (VOS\_THREAD\_T *thread*)

Terminate a thread.

This call will terminate the thread with the given threadId and release all resources. Depending on the underlying architectures, it may just block until the thread ran out.

##### Parameters:

← *thread* Thread handle (or NULL if current thread)

##### Return values:

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

**VOS\_PARAM\_ERR** parameter out of range/invalid

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:

← *thread* Thread handle (or NULL if current thread)

##### Return values:

**VOS\_NO\_ERR** no error

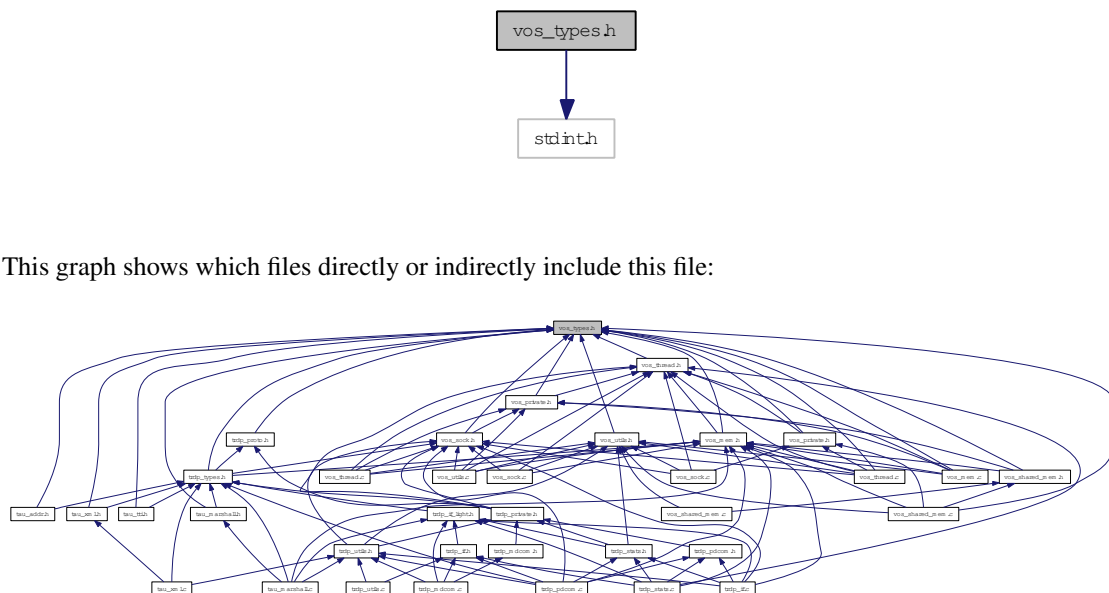
**VOS\_THREAD\_ERR** cancel failed

## 5.35 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\\_TIME\\_T](#)  
*Timer value compatible with timeval / select.*

### Defines

- #define [INLINE](#) inline  
*inline macros*

### Typedefs

- typedef UINT8 [VOS\\_UUID\\_T](#) [16]  
*universal unique identifier according to RFC 4122, time based version*
- typedef void(\* [VOS\\_PRINT\\_DBG\\_T](#))(void \*pRefCon, [VOS\\_LOG\\_T](#) category, const CHAR8 \*pTime, const CHAR8 \*pFile, UINT16 LineNumber, const CHAR8 \*pMsgStr)  
*Function definition for error/debug output.*

## Enumerations

- enum `VOS_ERR_T` {  
    `VOS_NO_ERR` = 0,  
    `VOS_PARAM_ERR` = -1,  
    `VOS_INIT_ERR` = -2,  
    `VOS_NOINIT_ERR` = -3,  
    `VOS_TIMEOUT_ERR` = -4,  
    `VOS_NODATA_ERR` = -5,  
    `VOS SOCK_ERR` = -6,  
    `VOS_IO_ERR` = -7,  
    `VOS_MEM_ERR` = -8,  
    `VOS_SEMA_ERR` = -9,  
    `VOS_QUEUE_ERR` = -10,  
    `VOS_QUEUE_FULL_ERR` = -11,  
    `VOS_MUTEX_ERR` = -12,  
    `VOS_THREAD_ERR` = -13,  
    `VOS_BLOCK_ERR` = -14,  
    `VOS_INTEGRATION_ERR` = -15,  
    `VOS_NOCONN_ERR` = -16,  
    `VOS_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 }

*Categories for logging.*

### 5.35.1 Detailed Description

Typedefs for OS abstraction.

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Bernd Loehr, NewTec GmbH

#### Remarks:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

Id

[vos\\_types.h](#) 1007 2013-07-02 11:28:56Z bloehr

### 5.35.2 Typedef Documentation

#### 5.35.2.1 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:

- ← *pRefCon* pointer to user context
- ← *category* Log category (Error, Warning, Info etc.)
- ← *pTime* pointer to NULL-terminated string of time stamp
- ← *pFile* pointer to NULL-terminated string of source module
- ← *LineNumber* Line number
- ← *pMsgStr* pointer to NULL-terminated string

##### Return values:

*none*

### 5.35.3 Enumeration Type Documentation

#### 5.35.3.1 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\_UNKNOWN\_ERR*** Unknown error.

#### 5.35.3.2 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.

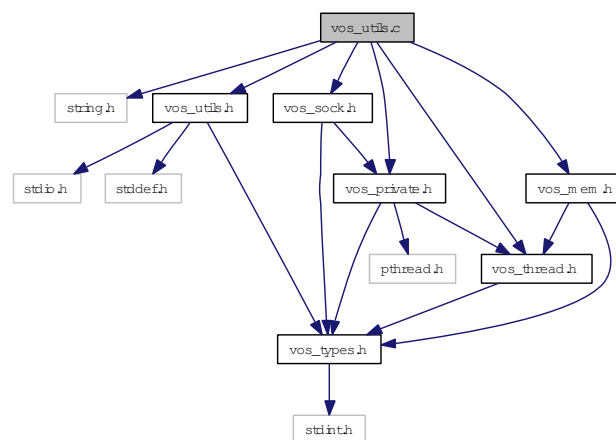


## 5.36 vos\_utils.c File Reference

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

- [VOS\\_ERR\\_T vos\\_initRuntimeConsts](#) (void)  
*Pre-compute alignment and endianness.*
- [VOS\\_ERR\\_T vos\\_init](#) (void \*pRefCon, [VOS\\_PRINT\\_DBG\\_T](#) pDebugOutput)  
*Initialize the virtual operating system.*
- EXT\_DECL void [vos\\_terminate](#) ()  
*DeInitialize the vos library.*
- UINT32 [vos\\_crc32](#) (UINT32 crc, const UINT8 \*pData, UINT32 dataLen)  
*Compute crc32 according to IEEE802.3.*
- INLINE BOOL [vos\\_isBigEndian](#) (void)  
*Return endianness.*

### 5.36.1 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:**

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

**Id**

[vos\\_utils.c](#) 1002 2013-07-02 09:02:41Z bloehr

## 5.36.2 Function Documentation

### 5.36.2.1 UINT32 vos\_crc32 (UINT32 *crc*, const UINT8 \* *pData*, UINT32 *dataLen*)

Compute crc32 according to IEEE802.3.

Calculate CRC for the given buffer and length.

**Parameters:**

← *crc* Initial value.

↔ *pData* Pointer to data.

← *dataLen* length in bytes of data.

**Return values:**

*crc32* according to IEEE802.3

### 5.36.2.2 VOS\_ERR\_T vos\_init (void \* *pRefCon*, VOS\_PRINT\_DBG\_T *pDebugOutput*)

Initialize the virtual operating system.

Initialize the vos library.

**Parameters:**

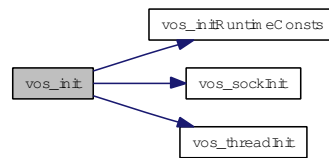
← *pRefCon* context for debug output function

← *pDebugOutput* Pointer to debug output function.

**Return values:**

*VOS\_NO\_ERR* no error *VOS\_INTEGRATION\_ERR* if endianness/alignment mismatch *VOS\_SOCKET\_ERR* sockets not supported *VOS\_UNKNOWN\_ERR* initialisation error

Here is the call graph for this function:



### 5.36.2.3 VOS\_ERR\_T vos\_initRuntimeConsts (void)

Pre-compute alignment and endianness.

**Return values:**

*VOS\_INTEGRATION\_ERR* or *VOS\_NO\_ERR*

### 5.36.2.4 INLINE BOOL vos\_isBigEndian (void)

Return endianness.

**Return values:**

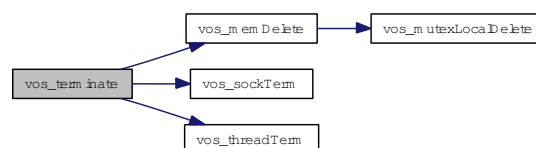
*TRUE* if big endian

### 5.36.2.5 EXT\_DECL void vos\_terminate ()

DeInitialize the vos library.

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

Here is the call graph for this function:

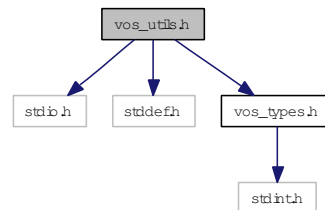


## 5.37 vos\_utils.h File Reference

Typedefs for OS abstraction.

```
#include <stdio.h>
#include <stddef.h>
#include "vos_types.h"
```

Include dependency graph for vos\_utils.h:



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



## Defines

- **#define VOS\_MAX\_PRNT\_STR\_SIZE** 256  
*String size definitions for the debug output functions.*
- **#define VOS\_MAX\_FRMT\_SIZE** 64  
*Max.*
- **#define VOS\_MAX\_ERR\_STR\_SIZE** (VOS\_MAX\_PRNT\_STR\_SIZE - VOS\_MAX\_FRMT\_SIZE)  
*Max.*
- **#define vos\_snprintf**(str, size, format, args...) snprintf(str, size, format, ## args)  
*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.*

## Functions

- EXT\_DECL UINT32 [vos\\_crc32](#) (UINT32 crc, const UINT8 \*pData, UINT32 dataLen)  
*Calculate CRC for the given buffer and length.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_init](#) (void \*pRefCon, [VOS\\_PRINT\\_DBG\\_T](#) pDebugOutput)  
*Initialize the vos library.*
- EXT\_DECL void [vos\\_terminate](#) ()  
*DeInitialize the vos library.*

### 5.37.1 Detailed Description

Typedefs for OS abstraction.

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Bernd Loehr, NewTec GmbH

#### Remarks:

All rights reserved. Reproduction, modification, use or disclosure to third parties without express authority is forbidden, Copyright Bombardier Transportation GmbH, Germany, 2012.

#### Id

[vos\\_utils.h](#) 1003 2013-07-02 09:05:06Z bloehr

### 5.37.2 Define Documentation

#### 5.37.2.1 #define VOS\_MAX\_ERR\_STR\_SIZE (VOS\_MAX\_PRNT\_STR\_SIZE - VOS\_MAX\_FRMT\_SIZE)

Max.

size of the error part

#### 5.37.2.2 #define VOS\_MAX\_FRMT\_SIZE 64

Max.

size of the 'format' part

#### 5.37.2.3 #define VOS\_MAX\_PRNT\_STR\_SIZE 256

String size definitions for the debug output functions.

Max. size of the debug/error string of debug function

### 5.37.3 Function Documentation

#### 5.37.3.1 EXT\_DECL\_UINT32 vos\_crc32 (UINT32 *crc*, const UINT8 \* *pData*, UINT32 *dataLen*)

Calculate CRC for the given buffer and length.

For TRDP FCS CRC calculation the CRC32 according to IEEE802.3 with start value 0xffffffff is used.

##### Parameters:

- ← *crc* Initial value.
- ↔ *pData* Pointer to data.
- ← *dataLen* length in bytes of data.

##### Return values:

*crc32* according to IEEE802.3

Calculate CRC for the given buffer and length.

##### Parameters:

- ← *crc* Initial value.
- ↔ *pData* Pointer to data.
- ← *dataLen* length in bytes of data.

##### Return values:

*crc32* according to IEEE802.3

#### 5.37.3.2 EXT\_DECL\_VOS\_ERR\_T vos\_init (void \* *pRefCon*, VOS\_PRINT\_DBG\_T *pDebugOutput*)

Initialize the vos library.

This is used to set the output function for all VOS error and debug output.

##### Parameters:

- ← *\*pRefCon* user context
- ← *\*pDebugOutput* pointer to debug output function

##### Return values:

*VOS\_NO\_ERR* no error  
*VOS\_INIT\_ERR* unsupported

Initialize the vos library.

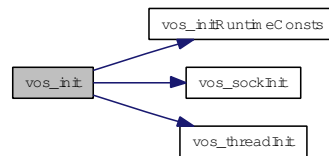
##### Parameters:

- ← *pRefCon* context for debug output function
- ← *pDebugOutput* Pointer to debug output function.

**Return values:**

**VOS\_NO\_ERR** no error  
**VOS\_INTEGRATION\_ERR** if endianess/alignment mismatch  
**VOS\_SOCKET\_ERR** sockets not supported  
**VOS\_UNKNOWN\_ERR** initialisation error

Here is the call graph for this function:

**5.37.3.3 EXT\_DECL void vos\_terminate ()**

DeInitialize the vos library.

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

**Parameters:**

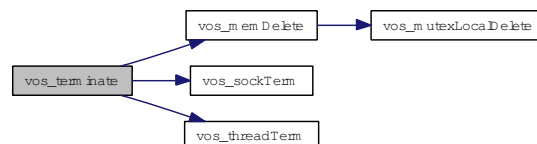
← *\*pRefCon* user context

**Return values:**

**VOS\_NO\_ERR** no error  
**VOS\_INIT\_ERR** not inited

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

Here is the call graph for this function:



# Index

- am\_big\_endian
  - trdp\_utils.c, [237](#)
  - trdp\_utils.h, [246](#)
- cyclicThread
  - posix/vos\_thread.c, [339](#)
  - windows/vos\_thread.c, [351](#)
- datasetLength
  - GNU\_PACKED, [10](#)
- destAddr
  - TRDP\_PUB\_STATISTICS\_T, [45](#)
- filterAddr
  - TRDP\_SUBS\_STATISTICS\_T, [56](#)
- GNU\_PACKED, [9](#)
  - datasetLength, [10](#)
  - msgType, [10](#)
  - protocolVersion, [10](#)
- msgType
  - GNU\_PACKED, [10](#)
  - TRDP\_MD\_INFO\_T, [33](#)
  - TRDP\_PD\_INFO\_T, [40](#)
- numRecv
  - TRDP\_SUBS\_STATISTICS\_T, [57](#)
- operator
  - TRDP\_TRAIN\_INFO\_T, [59](#)
- orient
  - TRDP\_CAR\_INFO\_T, [17](#)
  - TRDP\_CST\_INFO\_T, [20](#)
  - TRDP\_DEVICE\_INFO\_T, [25](#)
- owner
  - TRDP\_CST\_INFO\_T, [20](#)
- pCarInfo
  - TRDP\_CST\_INFO\_T, [20](#)
- pCstInfo
  - TRDP\_TRAIN\_INFO\_T, [59](#)
- PD\_ELE, [12](#)
  - pFrame, [13](#)
- pDevInfo
  - TRDP\_CAR\_INFO\_T, [17](#)
- pFctInfo
  - TRDP\_CST\_INFO\_T, [20](#)
- pFrame
  - PD\_ELE, [13](#)
- posix/vos\_private.h
  - vos\_mutexLocalCreate, [272](#)
  - vos\_mutexLocalDelete, [272](#)
- posix/vos\_shared\_mem.c
  - vos\_sharedClose, [276](#)
  - vos\_sharedOpen, [276](#)
- posix/vos\_sock.c
  - vos\_dottedIP, [288](#)
  - vos\_getInterfaces, [288](#)
  - vos\_getMacAddress, [288](#)
  - vos\_htonl, [289](#)
  - vos\_htons, [289](#)
  - vos\_ipDotted, [289](#)
  - vos\_isMulticast, [289](#)
  - vos\_ntohl, [290](#)
  - vos\_ntohs, [290](#)
  - vos\_select, [290](#)
  - vos\_sockAccept, [290](#)
  - vos\_sockBind, [291](#)
  - vos\_sockClose, [291](#)
  - vos\_sockConnect, [292](#)
  - vos\_sockGetMAC, [292](#)
  - vos\_sockInit, [293](#)
  - vos\_sockJoinMC, [293](#)
  - vos\_sockLeaveMC, [293](#)
  - vos\_sockListen, [294](#)
  - vos\_sockOpenTCP, [294](#)
  - vos\_sockOpenUDP, [295](#)
  - vos\_sockReceiveTCP, [295](#)
  - vos\_sockReceiveUDP, [296](#)
  - vos\_sockSendTCP, [296](#)
  - vos\_sockSendUDP, [297](#)
  - vos\_sockSetBuffer, [297](#)
  - vos\_sockSetMulticastIf, [298](#)
  - vos\_sockSetOptions, [298](#)
  - vos\_sockTerm, [298](#)
- posix/vos\_thread.c
  - cyclicThread, [339](#)
  - vos\_addTime, [340](#)
  - vos\_clearTime, [340](#)
  - vos\_cmpTime, [340](#)



- vos\_divTime, 340
- vos\_getTime, 341
- vos\_getTimeStamp, 341
- vos\_getUuid, 341
- vos\_mulTime, 341
- vos\_mutexCreate, 341
- vos\_mutexDelete, 342
- vos\_mutexLocalCreate, 342
- vos\_mutexLocalDelete, 343
- vos\_mutexLock, 343
- vos\_mutexTryLock, 343
- vos\_mutexUnlock, 343
- vos\_semaCreate, 344
- vos\_semaDelete, 344
- vos\_semaGive, 344
- vos\_semaTake, 345
- vos\_subTime, 345
- vos\_threadCreate, 345
- vos\_threadDelay, 346
- vos\_threadInit, 346
- vos\_threadIsActive, 346
- vos\_threadTerm, 347
- vos\_threadTerminate, 347
- printSocketUsage
  - trdp\_utils.c, 237
- protocolVersion
  - GNU\_PACKED, 10
- qos
  - VOS\_SOCK\_OPT\_T, 62
- recvmsg
  - windows/vos\_sock.c, 303
- tau\_tti.h
  - TRDP\_FCT\_CAR, 90
  - TRDP\_FCT\_CST, 90
  - TRDP\_FCT\_INVALID, 90
  - TRDP\_FCT\_TRAIN, 90
  - TRDP\_INAUG\_INVALID, 90
  - TRDP\_INAUG\_LEAD\_CONF, 90
  - TRDP\_INAUG\_LEAD\_UNCONF, 90
  - TRDP\_INAUG\_NOLEAD\_UNCONF, 90
- tau\_xml.h
  - TRDP\_DBG\_CAT, 102
  - TRDP\_DBG\_DBG, 102
  - TRDP\_DBG\_DEFAULT, 102
  - TRDP\_DBG\_ERR, 102
  - TRDP\_DBG\_INFO, 102
  - TRDP\_DBG\_LOC, 102
  - TRDP\_DBG\_OFF, 102
  - TRDP\_DBG\_TIME, 102
  - TRDP\_DBG\_WARN, 102
- tau\_addr.h, 65
  - tau\_addr2CarId, 67
  - tau\_addr2CarNo, 68
  - tau\_addr2CstId, 68
  - tau\_addr2CstNo, 68
  - tau\_addr2IecCarNo, 69
  - tau\_addr2IecCstNo, 69
  - tau\_addr2Uri, 69
  - tau\_carNo2Ids, 70
  - tau\_cstNo2CstId, 70
  - tau\_getOwnAddr, 70
  - tau\_getOwnIds, 70
  - tau\_iecCarNo2Ids, 71
  - tau\_iecCstNo2CstId, 71
  - tau\_label2CarId, 71
  - tau\_label2CarNo, 72
  - tau\_label2CstId, 72
  - tau\_label2CstNo, 72
  - tau\_label2IecCarNo, 73
  - tau\_label2IecCstNo, 73
  - tau\_uri2Addr, 73
- tau\_addr2CarId
  - tau\_addr.h, 67
- tau\_addr2CarNo
  - tau\_addr.h, 68
- tau\_addr2CstId
  - tau\_addr.h, 68
- tau\_addr2CstNo
  - tau\_addr.h, 68
- tau\_addr2IecCarNo
  - tau\_addr.h, 69
- tau\_addr2IecCstNo
  - tau\_addr.h, 69
- tau\_addr2Uri
  - tau\_addr.h, 69
- tau\_calcDatasetSize
  - tau\_marshall.c, 76
  - tau\_marshall.h, 82
- tau\_calcDatasetSizeByComId
  - tau\_marshall.c, 77
  - tau\_marshall.h, 83
- tau\_carNo2Ids
  - tau\_addr.h, 70
- tau\_cstNo2CstId
  - tau\_addr.h, 70
- tau\_freeTelegrams
  - tau\_xml.c, 97
  - tau\_xml.h, 102
- tau\_freeXmlDoc
  - tau\_xml.c, 97
  - tau\_xml.h, 103
- tau\_getCarDevCnt
  - tau\_tti.h, 90
- tau\_getCarInfo
  - tau\_tti.h, 90

- tau\_getCarOrient
  - tau\_tti.h, 91
- tau\_getCstCarCnt
  - tau\_tti.h, 91
- tau\_getCstFctCnt
  - tau\_tti.h, 91
- tau\_getCstFctInfo
  - tau\_tti.h, 92
- tau\_getCstInfo
  - tau\_tti.h, 92
- tau\_getDevInfo
  - tau\_tti.h, 92
- tau\_getEtbState
  - tau\_tti.h, 93
- tau\_getIecCarOrient
  - tau\_tti.h, 93
- tau\_getOwnAddr
  - tau\_addr.h, 70
- tau\_getOwnIds
  - tau\_addr.h, 70
- tau\_getTrnCarCnt
  - tau\_tti.h, 94
- tau\_getTrnCstCnt
  - tau\_tti.h, 94
- tau\_getTrnInfo
  - tau\_tti.h, 94
- tau\_iecCarNo2Ids
  - tau\_addr.h, 71
- tau\_iecCstNo2CstId
  - tau\_addr.h, 71
- tau\_initMarshall
  - tau\_marshall.c, 77
  - tau\_marshall.h, 83
- tau\_label2CarId
  - tau\_addr.h, 71
- tau\_label2CarNo
  - tau\_addr.h, 72
- tau\_label2CstId
  - tau\_addr.h, 72
- tau\_label2CstNo
  - tau\_addr.h, 72
- tau\_label2IecCarNo
  - tau\_addr.h, 73
- tau\_label2IecCstNo
  - tau\_addr.h, 73
- tau\_marshall
  - tau\_marshall.c, 78
  - tau\_marshall.h, 84
- tau\_marshall.c, 75
  - tau\_calcDatasetSize, 76
  - tau\_calcDatasetSizeByComId, 77
  - tau\_initMarshall, 77
  - tau\_marshall, 78
  - tau\_marshallIDs, 78
  - tau\_unmarshall, 79
  - tau\_unmarshallIDs, 79
- tau\_marshall.h, 81
  - tau\_calcDatasetSize, 82
  - tau\_calcDatasetSizeByComId, 83
  - tau\_initMarshall, 83
  - tau\_marshall, 84
  - tau\_marshallIDs, 84
  - tau\_unmarshall, 85
  - tau\_unmarshallIDs, 85
- TAU\_MARSHALL\_INFO\_T, 15
- tau\_marshallDs
  - tau\_marshall.c, 78
  - tau\_marshall.h, 84
- tau\_prepareXmlDoc
  - tau\_xml.c, 97
  - tau\_xml.h, 103
- tau\_readXmlDatasetConfig
  - tau\_xml.c, 97
  - tau\_xml.h, 103
- tau\_readXmlDeviceConfig
  - tau\_xml.c, 98
  - tau\_xml.h, 103
- tau\_readXmlInterfaceConfig
  - tau\_xml.c, 98
  - tau\_xml.h, 104
- tau\_tti.h, 87
  - tau\_getCarDevCnt, 90
  - tau\_getCarInfo, 90
  - tau\_getCarOrient, 91
  - tau\_getCstCarCnt, 91
  - tau\_getCstFctCnt, 91
  - tau\_getCstFctInfo, 92
  - tau\_getCstInfo, 92
  - tau\_getDevInfo, 92
  - tau\_getEtbState, 93
  - tau\_getIecCarOrient, 93
  - tau\_getTrnCarCnt, 94
  - tau\_getTrnCstCnt, 94
  - tau\_getTrnInfo, 94
  - TRDP\_FCT\_T, 89
  - TRDP\_INAUG\_STATE\_T, 90
- tau\_unmarshall
  - tau\_marshall.c, 79
  - tau\_marshall.h, 85
- tau\_unmarshallDs
  - tau\_marshall.c, 79
  - tau\_marshall.h, 85
- tau\_uri2Addr
  - tau\_addr.h, 73
- tau\_xml.c, 95
  - tau\_freeTelegrams, 97
  - tau\_freeXmlDoc, 97
  - tau\_prepareXmlDoc, 97

- tau\_readXmlDatasetConfig, 97
- tau\_readXmlDeviceConfig, 98
- tau\_readXmlInterfaceConfig, 98
- TRDP\_SDT\_DEFAULT\_CMTHR, 97
- tau\_xml.h, 100
  - tau\_freeTelegrams, 102
  - tau\_freeXmlDoc, 103
  - tau\_prepareXmlDoc, 103
  - tau\_readXmlDatasetConfig, 103
  - tau\_readXmlDeviceConfig, 103
  - tau\_readXmlInterfaceConfig, 104
  - TRDP\_DBG\_OPTION\_T, 102
- timeout
  - TRDP\_SUBS\_STATISTICS\_T, 56
- tlc\_closeSession
  - trdp\_if.c, 110
  - trdp\_if\_light.h, 135
- tlc\_freeBuf
  - trdp\_if\_light.h, 136
- tlc\_getInterval
  - trdp\_if.c, 110
  - trdp\_if\_light.h, 136
- tlc\_getJoinStatistics
  - trdp\_if\_light.h, 137
  - trdp\_stats.c, 217
- tlc\_getListStatistics
  - trdp\_if\_light.h, 138
  - trdp\_stats.c, 217
- tlc\_getPubStatistics
  - trdp\_if\_light.h, 139
  - trdp\_stats.c, 218
- tlc\_getRedStatistics
  - trdp\_if\_light.h, 140
  - trdp\_stats.c, 218
- tlc\_getStatistics
  - trdp\_if\_light.h, 141
  - trdp\_stats.c, 219
- tlc\_getSubsStatistics
  - trdp\_if\_light.h, 141
  - trdp\_stats.c, 219
- tlc\_getVersion
  - trdp\_if.c, 111
  - trdp\_if\_light.h, 142
- tlc\_getVersionString
  - trdp\_if.c, 111
  - trdp\_if\_light.h, 143
- tlc\_init
  - trdp\_if.c, 111
  - trdp\_if\_light.h, 143
- tlc\_openSession
  - trdp\_if.c, 112
  - trdp\_if\_light.h, 144
- tlc\_process
  - trdp\_if.c, 114
  - trdp\_if\_light.h, 147
- tlc\_reinitSession
  - trdp\_if.c, 116
  - trdp\_if\_light.h, 149
- tlc\_resetStatistics
  - trdp\_if\_light.h, 149
  - trdp\_stats.c, 220
- tlc\_setTopoCount
  - trdp\_if.c, 116
  - trdp\_if\_light.h, 150
- tlc\_terminate
  - trdp\_if.c, 116
  - trdp\_if\_light.h, 150
- tlim\_abortSession
  - trdp\_if\_light.h, 151
- tlim\_addListener
  - trdp\_if\_light.h, 152
- tlim\_confirm
  - trdp\_if\_light.h, 152
- tlim\_delListener
  - trdp\_if\_light.h, 153
- tlim\_notify
  - trdp\_if\_light.h, 153
- tlim\_reply
  - trdp\_if\_light.h, 154
- tlim\_replyErr
  - trdp\_if\_light.h, 155
- tlim\_replyQuery
  - trdp\_if\_light.h, 155
- tlim\_request
  - trdp\_if\_light.h, 156
- tlp\_get
  - trdp\_if.c, 117
  - trdp\_if\_light.h, 157
- tlp\_getRedundant
  - trdp\_if.c, 118
  - trdp\_if\_light.h, 159
- tlp\_publish
  - trdp\_if.c, 119
  - trdp\_if\_light.h, 160
- tlp\_put
  - trdp\_if.c, 121
  - trdp\_if\_light.h, 162
- tlp\_request
  - trdp\_if.c, 122
  - trdp\_if\_light.h, 163
- tlp\_setRedundant
  - trdp\_if.c, 124
  - trdp\_if\_light.h, 166
- tlp\_subscribe
  - trdp\_if.c, 125
  - trdp\_if\_light.h, 167
- tlp\_unpublish
  - trdp\_if.c, 126

- trdp\_if\_light.h, 169
- tlp\_unsubscribe
  - trdp\_if.c, 127
  - trdp\_if\_light.h, 170
- toBehav
  - TRDP\_SUBS\_STATISTICS\_T, 56
- topoCnt
  - TRDP\_TRAIN\_INFO\_T, 59
- TRDP\_APP\_CONFIRMTO\_ERR
  - trdp\_types.h, 233
- TRDP\_APP\_REPLYTO\_ERR
  - trdp\_types.h, 233
- TRDP\_APP\_TIMEOUT\_ERR
  - trdp\_types.h, 233
- TRDP\_BLOCK\_ERR
  - trdp\_types.h, 233
- TRDP\_BOOLEAN
  - trdp\_types.h, 232
- TRDP\_CHAR8
  - trdp\_types.h, 232
- TRDP\_COMID\_ERR
  - trdp\_types.h, 233
- TRDP\_CONFIRMTO\_ERR
  - trdp\_types.h, 233
- TRDP\_CRC\_ERR
  - trdp\_types.h, 233
- TRDP\_DBG\_CAT
  - tau\_xml.h, 102
- TRDP\_DBG\_DBG
  - tau\_xml.h, 102
- TRDP\_DBG\_DEFAULT
  - tau\_xml.h, 102
- TRDP\_DBG\_ERR
  - tau\_xml.h, 102
- TRDP\_DBG\_INFO
  - tau\_xml.h, 102
- TRDP\_DBG\_LOC
  - tau\_xml.h, 102
- TRDP\_DBG\_OFF
  - tau\_xml.h, 102
- TRDP\_DBG\_TIME
  - tau\_xml.h, 102
- TRDP\_DBG\_WARN
  - tau\_xml.h, 102
- TRDP\_FCT\_CAR
  - tau\_tti.h, 90
- TRDP\_FCT\_CST
  - tau\_tti.h, 90
- TRDP\_FCT\_INVALID
  - tau\_tti.h, 90
- TRDP\_FCT\_TRAIN
  - tau\_tti.h, 90
- TRDP\_FLAGS\_CALLBACK
  - trdp\_types.h, 233
- TRDP\_FLAGS\_DEFAULT
  - trdp\_types.h, 233
- TRDP\_FLAGS\_MARSHALL
  - trdp\_types.h, 233
- TRDP\_FLAGS\_NONE
  - trdp\_types.h, 233
- TRDP\_FLAGS\_TCP
  - trdp\_types.h, 233
- TRDP\_INAUG\_INVALID
  - tau\_tti.h, 90
- TRDP\_INAUG\_LEAD\_CONF
  - tau\_tti.h, 90
- TRDP\_INAUG\_LEAD\_UNCONF
  - tau\_tti.h, 90
- TRDP\_INAUG\_NOLEAD\_UNCONF
  - tau\_tti.h, 90
- TRDP\_INIT\_ERR
  - trdp\_types.h, 233
- TRDP\_INT16
  - trdp\_types.h, 232
- TRDP\_INT32
  - trdp\_types.h, 232
- TRDP\_INT64
  - trdp\_types.h, 232
- TRDP\_INT8
  - trdp\_types.h, 232
- TRDP\_INTEGRATION\_ERR
  - trdp\_types.h, 233
- TRDP\_INVALID\_DATA
  - trdp\_private.h, 209
- TRDP\_IO\_ERR
  - trdp\_types.h, 233
- TRDP\_MEM\_ERR
  - trdp\_types.h, 233
- TRDP\_MSG\_MC
  - trdp\_proto.h, 214
- TRDP\_MSG\_ME
  - trdp\_proto.h, 214
- TRDP\_MSG\_MN
  - trdp\_proto.h, 213
- TRDP\_MSG\_MP
  - trdp\_proto.h, 214
- TRDP\_MSG\_MQ
  - trdp\_proto.h, 214
- TRDP\_MSG\_MR
  - trdp\_proto.h, 214
- TRDP\_MSG\_PD
  - trdp\_proto.h, 213
- TRDP\_MSG\_PE
  - trdp\_proto.h, 213
- TRDP\_MSG\_PP
  - trdp\_proto.h, 213
- TRDP\_MSG\_PR
  - trdp\_proto.h, 213

- TRDP\_MUTEX\_ERR
  - trdp\_types.h, 233
- TRDP\_NO\_ERR
  - trdp\_types.h, 232
- TRDP\_NOCONN\_ERR
  - trdp\_types.h, 233
- TRDP\_NODATA\_ERR
  - trdp\_types.h, 233
- TRDP\_NOINIT\_ERR
  - trdp\_types.h, 233
- TRDP\_NOLIST\_ERR
  - trdp\_types.h, 233
- TRDP\_NOPUB\_ERR
  - trdp\_types.h, 233
- TRDP\_NOSESSION\_ERR
  - trdp\_types.h, 233
- TRDP\_NOSUB\_ERR
  - trdp\_types.h, 233
- TRDP\_OPTION\_BLOCK
  - trdp\_types.h, 234
- TRDP\_OPTION\_NO\_REUSE\_ADDR
  - trdp\_types.h, 234
- TRDP\_OPTION\_TRAFFIC\_SHAPING
  - trdp\_types.h, 234
- TRDP\_PACKET\_ERR
  - trdp\_types.h, 233
- TRDP\_PARAM\_ERR
  - trdp\_types.h, 232
- trdp\_private.h
  - TRDP\_INVALID\_DATA, 209
  - TRDP\_PULL\_SUB, 209
  - TRDP\_REDUNDANT, 209
  - TRDP\_REQ\_2B\_SENT, 209
  - TRDP SOCK\_MD\_TCP, 209
  - TRDP SOCK\_MD\_UDP, 209
  - TRDP SOCK\_PD, 209
  - TRDP\_ST\_NONE, 208
  - TRDP\_ST\_RX\_CONF\_RECEIVED, 208
  - TRDP\_ST\_RX\_NOTIFY\_RECEIVED, 208
  - TRDP\_ST\_RX\_READY, 208
  - TRDP\_ST\_RX\_REPLY\_SENT, 208
  - TRDP\_ST\_RX\_REPLYQUERY\_W4C, 208
  - TRDP\_ST\_RX\_REQ\_W4AP\_REPLY, 208
  - TRDP\_ST\_TX\_CONFIRM\_ARM, 208
  - TRDP\_ST\_TX\_NOTIFY\_ARM, 208
  - TRDP\_ST\_TX\_REPLY\_ARM, 208
  - TRDP\_ST\_TX\_REPLY\_RECEIVED, 208
  - TRDP\_ST\_TX\_REPLYQUERY\_ARM, 208
  - TRDP\_ST\_TX\_REQ\_W4AP\_CONFIRM, 208
  - TRDP\_ST\_TX\_REQUEST\_ARM, 208
  - TRDP\_ST\_TX\_REQUEST\_W4REPLY, 208
  - TRDP\_TIMED\_OUT, 209
- trdp\_proto.h
  - TRDP\_MSG\_MC, 214
  - TRDP\_MSG\_ME, 214
  - TRDP\_MSG\_MN, 213
  - TRDP\_MSG\_MP, 214
  - TRDP\_MSG\_MQ, 214
  - TRDP\_MSG\_MR, 214
  - TRDP\_MSG\_PD, 213
  - TRDP\_MSG\_PE, 213
  - TRDP\_MSG\_PP, 213
  - TRDP\_MSG\_PR, 213
  - TRDP\_PULL\_SUB
    - trdp\_private.h, 209
  - TRDP\_QUEUE\_ERR
    - trdp\_types.h, 233
  - TRDP\_QUEUE\_FULL\_ERR
    - trdp\_types.h, 233
  - TRDP\_REAL32
    - trdp\_types.h, 232
  - TRDP\_REAL64
    - trdp\_types.h, 232
  - TRDP\_RED\_FOLLOWER
    - trdp\_types.h, 234
  - TRDP\_RED\_LEADER
    - trdp\_types.h, 234
  - TRDP\_REDUNDANT
    - trdp\_private.h, 209
  - TRDP\_REPLYTO\_ERR
    - trdp\_types.h, 233
  - TRDP\_REQ\_2B\_SENT
    - trdp\_private.h, 209
  - TRDP\_REQCONFIRMTO\_ERR
    - trdp\_types.h, 233
  - TRDP\_SEMA\_ERR
    - trdp\_types.h, 233
  - TRDP\_SESSION\_ABORT\_ERR
    - trdp\_types.h, 233
  - TRDP SOCK\_ERR
    - trdp\_types.h, 233
  - TRDP SOCK\_MD\_TCP
    - trdp\_private.h, 209
  - TRDP SOCK\_MD\_UDP
    - trdp\_private.h, 209
  - TRDP SOCK\_PD
    - trdp\_private.h, 209
  - TRDP\_ST\_NONE
    - trdp\_private.h, 208
  - TRDP\_ST\_RX\_CONF\_RECEIVED
    - trdp\_private.h, 208
  - TRDP\_ST\_RX\_NOTIFY\_RECEIVED
    - trdp\_private.h, 208
  - TRDP\_ST\_RX\_READY
    - trdp\_private.h, 208
  - TRDP\_ST\_RX\_REPLY\_SENT
    - trdp\_private.h, 208

- TRDP\_ST\_RX\_REPLYQUERY\_W4C  
trdp\_private.h, 208
- TRDP\_ST\_RX\_REQ\_W4AP\_REPLY  
trdp\_private.h, 208
- TRDP\_ST\_TX\_CONFIRM\_ARM  
trdp\_private.h, 208
- TRDP\_ST\_TX\_NOTIFY\_ARM  
trdp\_private.h, 208
- TRDP\_ST\_TX\_REPLY\_ARM  
trdp\_private.h, 208
- TRDP\_ST\_TX\_REPLY\_RECEIVED  
trdp\_private.h, 208
- TRDP\_ST\_TX\_REPLYQUERY\_ARM  
trdp\_private.h, 208
- TRDP\_ST\_TX\_REQ\_W4AP\_CONFIRM  
trdp\_private.h, 208
- TRDP\_ST\_TX\_REQUEST\_ARM  
trdp\_private.h, 208
- TRDP\_ST\_TX\_REQUEST\_W4REPLY  
trdp\_private.h, 208
- TRDP\_STATE\_ERR  
trdp\_types.h, 233
- TRDP\_THREAD\_ERR  
trdp\_types.h, 233
- TRDP\_TIMED\_OUT  
trdp\_private.h, 209
- TRDP\_TIMEDATE32  
trdp\_types.h, 232
- TRDP\_TIMEDATE48  
trdp\_types.h, 232
- TRDP\_TIMEDATE64  
trdp\_types.h, 232
- TRDP\_TIMEOUT\_ERR  
trdp\_types.h, 233
- TRDP\_TO\_DEFAULT  
trdp\_types.h, 234
- TRDP\_TO\_KEEP\_LAST\_VALUE  
trdp\_types.h, 234
- TRDP\_TO\_SET\_TO\_ZERO  
trdp\_types.h, 234
- TRDP\_TOPO\_ERR  
trdp\_types.h, 233
- TRDP\_TYPE\_MAX  
trdp\_types.h, 232
- trdp\_types.h
  - TRDP\_APP\_CONFIRMTO\_ERR, 233
  - TRDP\_APP\_REPLYTO\_ERR, 233
  - TRDP\_APP\_TIMEOUT\_ERR, 233
  - TRDP\_BLOCK\_ERR, 233
  - TRDP\_BOOLEAN, 232
  - TRDP\_CHAR8, 232
  - TRDP\_COMID\_ERR, 233
  - TRDP\_CONFIRMTO\_ERR, 233
  - TRDP\_CRC\_ERR, 233
  - TRDP\_FLAGS\_CALLBACK, 233
  - TRDP\_FLAGS\_DEFAULT, 233
  - TRDP\_FLAGS\_MARSHALL, 233
  - TRDP\_FLAGS\_NONE, 233
  - TRDP\_FLAGS\_TCP, 233
  - TRDP\_INIT\_ERR, 233
  - TRDP\_INT16, 232
  - TRDP\_INT32, 232
  - TRDP\_INT64, 232
  - TRDP\_INT8, 232
  - TRDP\_INTEGRATION\_ERR, 233
  - TRDP\_IO\_ERR, 233
  - TRDP\_MEM\_ERR, 233
  - TRDP\_Mutex\_ERR, 233
  - TRDP\_NO\_ERR, 232
  - TRDP\_NOCONN\_ERR, 233
  - TRDP\_NODATA\_ERR, 233
  - TRDP\_NOINIT\_ERR, 233
  - TRDP\_NOLIST\_ERR, 233
  - TRDP\_NOPUB\_ERR, 233
  - TRDP\_NOSESSION\_ERR, 233
  - TRDP\_NOSUB\_ERR, 233
  - TRDP\_OPTION\_BLOCK, 234
  - TRDP\_OPTION\_NO\_REUSE\_ADDR, 234
  - TRDP\_OPTION\_TRAFFIC\_SHAPING, 234
  - TRDP\_PACKET\_ERR, 233
  - TRDP\_PARAM\_ERR, 232
  - TRDP\_QUEUE\_ERR, 233
  - TRDP\_QUEUE\_FULL\_ERR, 233
  - TRDP\_REAL32, 232
  - TRDP\_REAL64, 232
  - TRDP\_RED\_FOLLOWER, 234
  - TRDP\_RED\_LEADER, 234
  - TRDP\_REPLYTO\_ERR, 233
  - TRDP\_REQCONFIRMTO\_ERR, 233
  - TRDP\_SEMA\_ERR, 233
  - TRDP\_SESSION\_ABORT\_ERR, 233
  - TRDP SOCK\_ERR, 233
  - TRDP\_STATE\_ERR, 233
  - TRDP\_THREAD\_ERR, 233
  - TRDP\_TIMEDATE32, 232
  - TRDP\_TIMEDATE48, 232
  - TRDP\_TIMEDATE64, 232
  - TRDP\_TIMEOUT\_ERR, 233
  - TRDP\_TO\_DEFAULT, 234
  - TRDP\_TO\_KEEP\_LAST\_VALUE, 234
  - TRDP\_TO\_SET\_TO\_ZERO, 234
  - TRDP\_TOPO\_ERR, 233
  - TRDP\_TYPE\_MAX, 232
  - TRDP\_UINT16, 232
  - TRDP\_UINT32, 232
  - TRDP\_UINT64, 232
  - TRDP\_UINT8, 232
  - TRDP\_UNKNOWN\_ERR, 233

- TRDP\_UTF16, 232
- TRDP\_WIRE\_ERR, 233
- TRDP\_UINT16
  - trdp\_types.h, 232
- TRDP\_UINT32
  - trdp\_types.h, 232
- TRDP\_UINT64
  - trdp\_types.h, 232
- TRDP\_UINT8
  - trdp\_types.h, 232
- TRDP\_UNKNOWN\_ERR
  - trdp\_types.h, 233
- TRDP\_UTF16
  - trdp\_types.h, 232
- TRDP\_WIRE\_ERR
  - trdp\_types.h, 233
- TRDP\_CAR\_INFO\_T, 16
  - orient, 17
  - pDevInfo, 17
- trdp\_closeMDSessions
  - trdp\_mdcom.c, 174
  - trdp\_mdcom.h, 182
- TRDP\_COMID\_DSID\_MAP\_T, 18
- TRDP\_COMID\_ECHO
  - trdp\_proto.h, 212
- TRDP\_CST\_INFO\_T, 19
  - orient, 20
  - owner, 20
  - pCarInfo, 20
  - pFctInfo, 20
- TRDP\_DATA\_TYPE\_T
  - trdp\_types.h, 232
- TRDP\_DATASET, 21
- TRDP\_DATASET\_ELEMENT\_T, 22
  - type, 22
- TRDP\_DBG\_CONFIG\_T, 23
- TRDP\_DBG\_OPTION\_T
  - tau\_xml.h, 102
- TRDP\_DEST\_URI\_SIZE
  - trdp\_proto.h, 212
- TRDP\_DEVICE\_INFO\_T, 24
  - orient, 25
- trdp\_dllmain.c, 106
- TRDP\_ERR\_T
  - trdp\_types.h, 232
- TRDP\_FCT\_INFO\_T, 26
- TRDP\_FCT\_T
  - tau\_tti.h, 89
- TRDP\_FLAGS\_T
  - trdp\_types.h, 233
- trdp\_getSeqCnt
  - trdp\_utils.c, 237
  - trdp\_utils.h, 246
- trdp\_getTCPSocket
  - trdp\_mdcom.c, 174
  - trdp\_mdcom.h, 183
- TRDP\_HANDLE, 27
- trdp\_if.c, 107
  - tlc\_closeSession, 110
  - tlc\_getInterval, 110
  - tlc\_getVersion, 111
  - tlc\_getVersionString, 111
  - tlc\_init, 111
  - tlc\_openSession, 112
  - tlc\_process, 114
  - tlc\_reinitSession, 116
  - tlc\_setTopoCount, 116
  - tlc\_terminate, 116
  - tlp\_get, 117
  - tlp\_getRedundant, 118
  - tlp\_publish, 119
  - tlp\_put, 121
  - tlp\_request, 122
  - tlp\_setRedundant, 124
  - tlp\_subscribe, 125
  - tlp\_unpublish, 126
  - tlp\_unsubscribe, 127
  - trdp\_isValidSession, 127
  - trdp\_sessionQueue, 128
- trdp\_if.h, 129
  - trdp\_isValidSession, 130
  - trdp\_sessionQueue, 130
- trdp\_if\_light.h, 131
  - tlc\_closeSession, 135
  - tlc\_freeBuf, 136
  - tlc\_getInterval, 136
  - tlc\_getJoinStatistics, 137
  - tlc\_getListStatistics, 138
  - tlc\_getPubStatistics, 139
  - tlc\_getRedStatistics, 140
  - tlc\_getStatistics, 141
  - tlc\_getSubsStatistics, 141
  - tlc\_getVersion, 142
  - tlc\_getVersionString, 143
  - tlc\_init, 143
  - tlc\_openSession, 144
  - tlc\_process, 147
  - tlc\_reinitSession, 149
  - tlc\_resetStatistics, 149
  - tlc\_setTopoCount, 150
  - tlc\_terminate, 150
  - tlim\_abortSession, 151
  - tlim\_addListener, 152
  - tlim\_confirm, 152
  - tlim\_delListener, 153
  - tlim\_notify, 153
  - tlim\_reply, 154
  - tlim\_replyErr, 155

- tlm\_replyQuery, 155
- tlm\_request, 156
- tlp\_get, 157
- tlp\_getRedundant, 159
- tlp\_publish, 160
- tlp\_put, 162
- tlp\_request, 163
- tlp\_setRedundant, 166
- tlp\_subscribe, 167
- tlp\_unpublish, 169
- tlp\_unsubscribe, 170
- TRDP\_INAUG\_STATE\_T
  - tau\_tti.h, 90
- trdp\_initSockets
  - trdp\_utils.c, 238
  - trdp\_utils.h, 246
- trdp\_initStats
  - trdp\_stats.c, 220
  - trdp\_stats.h, 223
- trdp\_initUncompletedTCP
  - trdp\_utils.h, 247
- TRDP\_IP\_ADDR\_T
  - trdp\_types.h, 230
- trdp\_isAddressed
  - trdp\_utils.c, 238
  - trdp\_utils.h, 247
- trdp\_isRcvSeqCnt
  - trdp\_utils.c, 238
  - trdp\_utils.h, 247
- trdp\_isValidSession
  - trdp\_if.c, 127
  - trdp\_if.h, 130
- TRDP\_LIST\_STATISTICS\_T, 28
- TRDP\_MARSHALL\_CONFIG\_T, 29
- TRDP\_MARSHALL\_T
  - trdp\_types.h, 230
- TRDP\_MAX\_FILE\_NAME\_LEN
  - trdp\_proto.h, 212
- TRDP\_MAX\_LABEL\_LEN
  - trdp\_proto.h, 213
- TRDP\_MAX\_URI\_HOST\_LEN
  - trdp\_proto.h, 213
- TRDP\_MAX\_URI\_LEN
  - trdp\_proto.h, 213
- TRDP\_MAX\_URI\_USER\_LEN
  - trdp\_proto.h, 213
- TRDP\_MD\_CALLBACK\_T
  - trdp\_types.h, 230
- TRDP\_MD\_CONFIG\_T, 30
- TRDP\_MD\_ELE\_ST\_T
  - trdp\_private.h, 208
- TRDP\_MD\_INFO\_T, 32
  - msgType, 33
- TRDP\_MD\_STATISTICS\_T, 34
- trdp\_mdCheck
  - trdp\_mdcom.c, 174
- trdp\_mdCheckListenSocks
  - trdp\_mdcom.c, 175
  - trdp\_mdcom.h, 183
- trdp\_mdCheckPending
  - trdp\_mdcom.c, 176
  - trdp\_mdcom.h, 184
- trdp\_mdCheckTimeouts
  - trdp\_mdcom.c, 177
  - trdp\_mdcom.h, 185
- trdp\_mdcom.c, 172
  - trdp\_closeMDSessions, 174
  - trdp\_getTCPSocket, 174
  - trdp\_mdCheck, 174
  - trdp\_mdCheckListenSocks, 175
  - trdp\_mdCheckPending, 176
  - trdp\_mdCheckTimeouts, 177
  - trdp\_mdFreeSession, 177
  - trdp\_mdRecv, 177
  - trdp\_mdRecvPacket, 178
  - trdp\_mdSend, 179
  - trdp\_mdSendPacket, 179
  - trdp\_mdSetSessionTimeout, 180
  - trdp\_mdUpdatePacket, 180
- trdp\_mdcom.h, 181
  - trdp\_closeMDSessions, 182
  - trdp\_getTCPSocket, 183
  - trdp\_mdCheckListenSocks, 183
  - trdp\_mdCheckPending, 184
  - trdp\_mdCheckTimeouts, 185
  - trdp\_mdFreeSession, 185
  - trdp\_mdRecv, 185
  - trdp\_mdSend, 186
  - trdp\_mdSendPacket, 187
  - trdp\_mdSetSessionTimeout, 187
  - trdp\_mdUpdatePacket, 188
- trdp\_mdFreeSession
  - trdp\_mdcom.c, 177
  - trdp\_mdcom.h, 185
- trdp\_mdRecv
  - trdp\_mdcom.c, 177
  - trdp\_mdcom.h, 185
- trdp\_mdRecvPacket
  - trdp\_mdcom.c, 178
- trdp\_mdSend
  - trdp\_mdcom.c, 179
  - trdp\_mdcom.h, 186
- trdp\_mdSendPacket
  - trdp\_mdcom.c, 179
  - trdp\_mdcom.h, 187
- trdp\_mdSetSessionTimeout
  - trdp\_mdcom.c, 180
  - trdp\_mdcom.h, 187



- trdp\_mdUpdatePacket
  - trdp\_mdcom.c, 180
  - trdp\_mdcom.h, 188
- TRDP\_MEM\_CONFIG\_T, 36
- TRDP\_MEM\_STATISTICS\_T, 37
- TRDP\_MSG\_T
  - trdp\_proto.h, 213
- TRDP\_OPTION\_T
  - trdp\_types.h, 233
- trdp\_packetSizeMD
  - trdp\_utils.c, 239
  - trdp\_utils.h, 248
- trdp\_packetSizePD
  - trdp\_utils.c, 239
  - trdp\_utils.h, 248
- TRDP\_PD\_CALLBACK\_T
  - trdp\_types.h, 231
- TRDP\_PD\_CONFIG\_T, 38
- TRDP\_PD\_INFO\_T, 39
  - msgType, 40
- TRDP\_PD\_STATISTICS\_T, 41
- trdp\_pdCheck
  - trdp\_pdcom.c, 191
  - trdp\_pdcom.h, 199
- trdp\_pdCheckListenSocks
  - trdp\_pdcom.c, 191
  - trdp\_pdcom.h, 199
- trdp\_pdCheckPending
  - trdp\_pdcom.c, 192
  - trdp\_pdcom.h, 200
- trdp\_pdcom.c, 189
  - trdp\_pdCheck, 191
  - trdp\_pdCheckListenSocks, 191
  - trdp\_pdCheckPending, 192
  - trdp\_pdDataUpdate, 192
  - trdp\_pdDistribute, 193
  - trdp\_pdHandleTimeOuts, 193
  - trdp\_pdInit, 193
  - trdp\_pdReceive, 194
  - trdp\_pdSend, 195
  - trdp\_pdSendQueued, 196
  - trdp\_pdUpdate, 196
- trdp\_pdcom.h, 197
  - trdp\_pdCheck, 199
  - trdp\_pdCheckListenSocks, 199
  - trdp\_pdCheckPending, 200
  - trdp\_pdDataUpdate, 200
  - trdp\_pdDistribute, 201
  - trdp\_pdHandleTimeOuts, 201
  - trdp\_pdInit, 201
  - trdp\_pdReceive, 202
  - trdp\_pdSend, 203
  - trdp\_pdSendQueued, 204
  - trdp\_pdUpdate, 204
- trdp\_pdDataUpdate
  - trdp\_pdcom.c, 192
  - trdp\_pdcom.h, 200
- trdp\_pdDistribute
  - trdp\_pdcom.c, 193
  - trdp\_pdcom.h, 201
- trdp\_pdHandleTimeOuts
  - trdp\_pdcom.c, 193
  - trdp\_pdcom.h, 201
- trdp\_pdInit
  - trdp\_pdcom.c, 193
  - trdp\_pdcom.h, 201
- trdp\_pdPrepareStats
  - trdp\_stats.c, 221
  - trdp\_stats.h, 223
- trdp\_pdReceive
  - trdp\_pdcom.c, 194
  - trdp\_pdcom.h, 202
- trdp\_pdSend
  - trdp\_pdcom.c, 195
  - trdp\_pdcom.h, 203
- trdp\_pdSendQueued
  - trdp\_pdcom.c, 196
  - trdp\_pdcom.h, 204
- trdp\_pdUpdate
  - trdp\_pdcom.c, 196
  - trdp\_pdcom.h, 204
- TRDP\_PRINT\_DBG\_T
  - trdp\_types.h, 231
- TRDP\_PRIV\_FLAGS\_T
  - trdp\_private.h, 208
- trdp\_private.h, 205
  - TRDP\_MD\_ELE\_ST\_T, 208
  - TRDP\_PRIV\_FLAGS\_T, 208
  - TRDP SOCK\_TYPE\_T, 209
- TRDP\_PROCESS\_CONFIG\_T, 43
- TRDP\_PROP\_INFO\_T, 44
- trdp\_proto.h, 210
  - TRDP\_COMID\_ECHO, 212
  - TRDP\_DEST\_URI\_SIZE, 212
  - TRDP\_MAX\_FILE\_NAME\_LEN, 212
  - TRDP\_MAX\_LABEL\_LEN, 213
  - TRDP\_MAX\_URI\_HOST\_LEN, 213
  - TRDP\_MAX\_URI\_LEN, 213
  - TRDP\_MAX\_URI\_USER\_LEN, 213
  - TRDP\_MSG\_T, 213
  - TRDP\_STATISTICS\_REQUEST\_DSID, 213
- TRDP\_PUB\_STATISTICS\_T, 45
  - destAddr, 45
- trdp\_queueAppLast
  - trdp\_utils.c, 239
  - trdp\_utils.h, 248
- trdp\_queueDelElement
  - trdp\_utils.c, 239

- trdp\_utils.h, 248
- trdp\_queueFindComId
  - trdp\_utils.c, 240
  - trdp\_utils.h, 248
- trdp\_queueFindPubAddr
  - trdp\_utils.c, 240
  - trdp\_utils.h, 249
- trdp\_queueFindSubAddr
  - trdp\_utils.c, 240
  - trdp\_utils.h, 249
- trdp\_queueInsFirst
  - trdp\_utils.c, 240
  - trdp\_utils.h, 249
- TRDP\_RED\_STATE\_T
  - trdp\_types.h, 234
- TRDP\_RED\_STATISTICS\_T, 46
- trdp\_releaseSocket
  - trdp\_utils.c, 241
  - trdp\_utils.h, 249
- TRDP\_REPLY\_STATUS\_T
  - trdp\_types.h, 234
- trdp\_requestSocket
  - trdp\_utils.c, 241
  - trdp\_utils.h, 250
- TRDP\_SDT\_DEFAULT\_CMTHR
  - tau\_xml.c, 97
- TRDP\_SDT\_PAR\_T, 47
- TRDP\_SEND\_PARAM\_T, 48
- TRDP\_SESSION, 49
- trdp\_sessionQueue
  - trdp\_if.c, 128
  - trdp\_if.h, 130
- TRDP SOCK\_TYPE\_T
  - trdp\_private.h, 209
- trdp\_SockAddJoin
  - trdp\_utils.c, 242
- trdp\_SockDelJoin
  - trdp\_utils.c, 242
- TRDP\_SOCKET\_TCP, 51
- TRDP\_SOCKETS, 52
  - usage, 53
- trdp\_SockIsJoined
  - trdp\_utils.c, 243
- TRDP\_STATISTICS\_REQUEST\_DSID
  - trdp\_proto.h, 213
- TRDP\_STATISTICS\_T, 54
- trdp\_stats.c, 215
  - tlc\_getJoinStatistics, 217
  - tlc\_getListStatistics, 217
  - tlc\_getPubStatistics, 218
  - tlc\_getRedStatistics, 218
  - tlc\_getStatistics, 219
  - tlc\_getSubsStatistics, 219
  - tlc\_resetStatistics, 220
- trdp\_initStats, 220
- trdp\_pdPrepareStats, 221
- trdp\_UpdateStats, 221
- trdp\_stats.h, 222
  - trdp\_initStats, 223
  - trdp\_pdPrepareStats, 223
- TRDP\_SUBS\_STATISTICS\_T, 56
  - filterAddr, 56
  - numRecv, 57
  - timeout, 56
  - toBehav, 56
- TRDP\_TIME\_T
  - trdp\_types.h, 231
- TRDP\_TO\_BEHAVIOR\_T
  - trdp\_types.h, 234
- TRDP\_TRAIN\_INFO\_T, 58
  - operator, 59
  - pCstInfo, 59
  - topoCnt, 59
- trdp\_types.h, 225
  - TRDP\_DATA\_TYPE\_T, 232
  - TRDP\_ERR\_T, 232
  - TRDP\_FLAGS\_T, 233
  - TRDP\_IP\_ADDR\_T, 230
  - TRDP\_MARSHALL\_T, 230
  - TRDP\_MD\_CALLBACK\_T, 230
  - TRDP\_OPTION\_T, 233
  - TRDP\_PD\_CALLBACK\_T, 231
  - TRDP\_PRINT\_DBG\_T, 231
  - TRDP\_RED\_STATE\_T, 234
  - TRDP\_REPLY\_STATUS\_T, 234
  - TRDP\_TIME\_T, 231
  - TRDP\_TO\_BEHAVIOR\_T, 234
  - TRDP\_UNMARSHALL\_T, 231
- TRDP\_UNMARSHALL\_T
  - trdp\_types.h, 231
- trdp\_UpdateStats
  - trdp\_stats.c, 221
- trdp\_utils.c, 235
  - am\_big\_endian, 237
  - printSocketUsage, 237
  - trdp\_getSeqCnt, 237
  - trdp\_initSockets, 238
  - trdp\_isAddressed, 238
  - trdp\_isRcvSeqCnt, 238
  - trdp\_packetSizeMD, 239
  - trdp\_packetSizePD, 239
  - trdp\_queueAppLast, 239
  - trdp\_queueDelElement, 239
  - trdp\_queueFindComId, 240
  - trdp\_queueFindPubAddr, 240
  - trdp\_queueFindSubAddr, 240
  - trdp\_queueInsFirst, 240
  - trdp\_releaseSocket, 241

- trdp\_requestSocket, 241
- trdp\_SockAddJoin, 242
- trdp\_SockDelJoin, 242
- trdp\_SockIsJoined, 243
- trdp\_utils.h, 244
  - am\_big\_endian, 246
  - trdp\_getSeqCnt, 246
  - trdp\_initSockets, 246
  - trdp\_initUncompletedTCP, 247
  - trdp\_isAddressed, 247
  - trdp\_isRcvSeqCnt, 247
  - trdp\_packetSizeMD, 248
  - trdp\_packetSizePD, 248
  - trdp\_queueAppLast, 248
  - trdp\_queueDelElement, 248
  - trdp\_queueFindComId, 248
  - trdp\_queueFindPubAddr, 249
  - trdp\_queueFindSubAddr, 249
  - trdp\_queueInsFirst, 249
  - trdp\_releaseSocket, 249
  - trdp\_requestSocket, 250
- TRDP\_VERSION\_T, 60
- TRDP\_XML\_DOC\_HANDLE\_T, 61
- tv\_usec
  - VOS\_TIME\_T, 63
- type
  - TRDP\_DATASET\_ELEMENT\_T, 22
- usage
  - TRDP\_SOCKETS, 53
- VOS\_BLOCK\_ERR
  - vos\_types.h, 377
- VOS\_INIT\_ERR
  - vos\_types.h, 377
- VOS\_INTEGRATION\_ERR
  - vos\_types.h, 378
- VOS\_IO\_ERR
  - vos\_types.h, 377
- VOS\_LOG\_DBG
  - vos\_types.h, 378
- VOS\_LOG\_ERROR
  - vos\_types.h, 378
- VOS\_LOG\_INFO
  - vos\_types.h, 378
- VOS\_LOG\_WARNING
  - vos\_types.h, 378
- VOS\_MEM\_ERR
  - vos\_types.h, 377
- VOS\_MUTEX\_ERR
  - vos\_types.h, 377
- VOS\_NO\_ERR
  - vos\_types.h, 377
- VOS\_NOCONN\_ERR
  - vos\_types.h, 378
- VOS\_NODATA\_ERR
  - vos\_types.h, 377
- VOS\_NOINIT\_ERR
  - vos\_types.h, 377
- VOS\_PARAM\_ERR
  - vos\_types.h, 377
- VOS\_QUEUE\_ERR
  - vos\_types.h, 377
- VOS\_QUEUE\_FULL\_ERR
  - vos\_types.h, 377
- VOS\_SEMA\_ERR
  - vos\_types.h, 377
- VOS SOCK\_ERR
  - vos\_types.h, 377
- VOS\_THREAD\_ERR
  - vos\_types.h, 377
- VOS\_TIMEOUT\_ERR
  - vos\_types.h, 377
- vos\_types.h
  - VOS\_BLOCK\_ERR, 377
  - VOS\_INIT\_ERR, 377
  - VOS\_INTEGRATION\_ERR, 378
  - VOS\_IO\_ERR, 377
  - VOS\_LOG\_DBG, 378
  - VOS\_LOG\_ERROR, 378
  - VOS\_LOG\_INFO, 378
  - VOS\_LOG\_WARNING, 378
  - VOS\_MEM\_ERR, 377
  - VOS\_MUTEX\_ERR, 377
  - VOS\_NO\_ERR, 377
  - VOS\_NOCONN\_ERR, 378
  - VOS\_NODATA\_ERR, 377
  - VOS\_NOINIT\_ERR, 377
  - VOS\_PARAM\_ERR, 377
  - VOS\_QUEUE\_ERR, 377
  - VOS\_QUEUE\_FULL\_ERR, 377
  - VOS\_SEMA\_ERR, 377
  - VOS SOCK\_ERR, 377
  - VOS\_THREAD\_ERR, 377
  - VOS\_TIMEOUT\_ERR, 377
  - VOS\_UNKNOWN\_ERR, 378
- VOS\_UNKNOWN\_ERR
  - vos\_types.h, 378
- vos\_addTime
  - posix/vos\_thread.c, 340
  - vos\_thread.h, 363
  - windows/vos\_thread.c, 351
- vos\_bsearch
  - vos\_mem.c, 254
  - vos\_mem.h, 263
- vos\_clearTime
  - posix/vos\_thread.c, 340
  - vos\_thread.h, 363

- windows/vos\_thread.c, 351
- vos\_cmpTime
  - posix/vos\_thread.c, 340
  - vos\_thread.h, 363
  - windows/vos\_thread.c, 351
- vos\_crc32
  - vos\_utils.c, 380
  - vos\_utils.h, 384
- vos\_divTime
  - posix/vos\_thread.c, 340
  - vos\_thread.h, 364
  - windows/vos\_thread.c, 352
- vos\_dottedIP
  - posix/vos\_sock.c, 288
  - vos\_sock.h, 317
  - windows/vos\_sock.c, 303
- VOS\_ERR\_T
  - vos\_types.h, 377
- vos\_getFreeThreadHandle
  - windows/vos\_thread.c, 352
- vos\_getInterfaces
  - posix/vos\_sock.c, 288
  - vos\_sock.h, 318
  - windows/vos\_sock.c, 303
- vos\_getMacAddress
  - posix/vos\_sock.c, 288
- vos\_getTime
  - posix/vos\_thread.c, 341
  - vos\_thread.h, 364
  - windows/vos\_thread.c, 352
- vos\_getTimeStamp
  - posix/vos\_thread.c, 341
  - vos\_thread.h, 364
  - windows/vos\_thread.c, 352
- vos\_getUuid
  - posix/vos\_thread.c, 341
  - vos\_thread.h, 365
  - windows/vos\_thread.c, 352
- vos\_htonl
  - posix/vos\_sock.c, 289
  - vos\_sock.h, 318
  - windows/vos\_sock.c, 304
- vos\_htons
  - posix/vos\_sock.c, 289
  - vos\_sock.h, 319
  - windows/vos\_sock.c, 304
- vos\_init
  - vos\_utils.c, 380
  - vos\_utils.h, 384
- vos\_initRuntimeConsts
  - vos\_utils.c, 381
- vos\_ipDotted
  - posix/vos\_sock.c, 289
  - vos\_sock.h, 319
- windows/vos\_sock.c, 304
- vos\_isBigEndian
  - vos\_utils.c, 381
- vos\_isMulticast
  - posix/vos\_sock.c, 289
  - vos\_sock.h, 320
  - windows/vos\_sock.c, 305
- VOS\_LOG\_T
  - vos\_types.h, 378
- VOS\_MAX\_ERR\_STR\_SIZE
  - vos\_utils.h, 383
- VOS\_MAX\_FRMT\_SIZE
  - vos\_utils.h, 383
- VOS\_MAX\_PRNT\_STR\_SIZE
  - vos\_utils.h, 383
- VOS\_MAX\_SOCKET\_CNT
  - vos\_sock.h, 317
- vos\_mem.c, 252
  - vos\_bsearch, 254
  - vos\_memAlloc, 254
  - vos\_memCount, 254
  - vos\_memDelete, 255
  - vos\_memFree, 255
  - vos\_memInit, 256
  - vos\_mutexLocalCreate, 256
  - vos\_mutexLocalDelete, 257
  - vos\_qsort, 257
  - vos\_queueCreate, 257
  - vos\_queueDestroy, 258
  - vos\_queueReceive, 258
  - vos\_queueSend, 259
  - vos\_strncpy, 259
  - vos\_strncmp, 260
- vos\_mem.h, 261
  - vos\_bsearch, 263
  - VOS\_MEM\_BLOCKSIZE, 263
  - VOS\_MEM\_PREALLOCATE, 263
  - vos\_memAlloc, 264
  - vos\_memCount, 264
  - vos\_memDelete, 264
  - vos\_memFree, 265
  - vos\_memInit, 265
  - vos\_qsort, 266
  - vos\_queueCreate, 266
  - vos\_queueDestroy, 267
  - vos\_queueReceive, 268
  - vos\_queueSend, 269
  - vos\_strncpy, 269
  - vos\_strncmp, 270
- VOS\_MEM\_BLOCKSIZE
  - vos\_mem.h, 263
- VOS\_MEM\_PREALLOCATE
  - vos\_mem.h, 263
- vos\_memAlloc

- vos\_mem.c, 254
- vos\_mem.h, 264
- vos\_memCount
  - vos\_mem.c, 254
  - vos\_mem.h, 264
- vos\_memDelete
  - vos\_mem.c, 255
  - vos\_mem.h, 264
- vos\_memFree
  - vos\_mem.c, 255
  - vos\_mem.h, 265
- vos\_memInit
  - vos\_mem.c, 256
  - vos\_mem.h, 265
- vos\_mulTime
  - posix/vos\_thread.c, 341
  - vos\_thread.h, 365
  - windows/vos\_thread.c, 353
- vos\_mutexCreate
  - posix/vos\_thread.c, 341
  - vos\_thread.h, 365
  - windows/vos\_thread.c, 353
- vos\_mutexDelete
  - posix/vos\_thread.c, 342
  - vos\_thread.h, 366
  - windows/vos\_thread.c, 353
- vos\_mutexLocalCreate
  - posix/vos\_private.h, 272
  - posix/vos\_thread.c, 342
  - vos\_mem.c, 256
  - windows/vos\_private.h, 274
  - windows/vos\_thread.c, 354
- vos\_mutexLocalDelete
  - posix/vos\_private.h, 272
  - posix/vos\_thread.c, 343
  - vos\_mem.c, 257
  - windows/vos\_private.h, 274
  - windows/vos\_thread.c, 354
- vos\_mutexLock
  - posix/vos\_thread.c, 343
  - vos\_thread.h, 367
  - windows/vos\_thread.c, 354
- vos\_mutexTryLock
  - posix/vos\_thread.c, 343
  - vos\_thread.h, 367
  - windows/vos\_thread.c, 355
- vos\_mutexUnlock
  - posix/vos\_thread.c, 343
  - vos\_thread.h, 368
  - windows/vos\_thread.c, 355
- vos\_ntohl
  - posix/vos\_sock.c, 290
  - vos\_sock.h, 320
  - windows/vos\_sock.c, 305
- vos\_ntohs
  - posix/vos\_sock.c, 290
  - vos\_sock.h, 320
  - windows/vos\_sock.c, 305
- VOS\_PRINT\_DBG\_T
  - vos\_types.h, 377
- vos\_private.h, 271, 273
- vos\_qsort
  - vos\_mem.c, 257
  - vos\_mem.h, 266
- vos\_queueCreate
  - vos\_mem.c, 257
  - vos\_mem.h, 266
- vos\_queueDestroy
  - vos\_mem.c, 258
  - vos\_mem.h, 267
- vos\_queueReceive
  - vos\_mem.c, 258
  - vos\_mem.h, 268
- vos\_queueSend
  - vos\_mem.c, 259
  - vos\_mem.h, 269
- vos\_select
  - posix/vos\_sock.c, 290
  - vos\_sock.h, 320
  - windows/vos\_sock.c, 305
- vos\_semaCreate
  - posix/vos\_thread.c, 344
  - vos\_thread.h, 368
  - windows/vos\_thread.c, 355
- vos\_semaDelete
  - posix/vos\_thread.c, 344
  - vos\_thread.h, 369
  - windows/vos\_thread.c, 356
- vos\_semaGive
  - posix/vos\_thread.c, 344
  - vos\_thread.h, 369
  - windows/vos\_thread.c, 356
- vos\_semaTake
  - posix/vos\_thread.c, 345
  - vos\_thread.h, 369
  - windows/vos\_thread.c, 356
- vos\_shared\_mem.c, 275, 278
- vos\_shared\_mem.h, 281
  - vos\_sharedClose, 282
  - vos\_sharedOpen, 282
- vos\_sharedClose
  - posix/vos\_shared\_mem.c, 276
  - vos\_shared\_mem.h, 282
  - windows/vos\_shared\_mem.c, 279
- vos\_sharedOpen
  - posix/vos\_shared\_mem.c, 276
  - vos\_shared\_mem.h, 282
  - windows/vos\_shared\_mem.c, 279

- vos\_sock.c, 285, 300
- vos\_sock.h, 314
  - vos\_dottedIP, 317
  - vos\_getInterfaces, 318
  - vos\_htonl, 318
  - vos\_htons, 319
  - vos\_ipDotted, 319
  - vos\_isMulticast, 320
  - VOS\_MAX\_SOCKET\_CNT, 317
  - vos\_ntohl, 320
  - vos\_ntohs, 320
  - vos\_select, 320
  - vos\_sockAccept, 321
  - vos\_sockBind, 322
  - vos\_sockClose, 323
  - vos\_sockConnect, 323
  - vos\_sockGetMAC, 324
  - vos\_sockInit, 325
  - vos\_sockJoinMC, 325
  - vos\_sockLeaveMC, 326
  - vos\_sockListen, 327
  - vos\_sockOpenTCP, 328
  - vos\_sockOpenUDP, 329
  - vos\_sockReceiveTCP, 330
  - vos\_sockReceiveUDP, 331
  - vos\_sockSendTCP, 332
  - vos\_sockSendUDP, 333
  - vos\_sockSetMulticastIf, 334
  - vos\_sockSetOptions, 335
  - vos\_sockTerm, 336
  - VOS\_TTL\_MULTICAST, 317
- VOS\_SOCKET\_OPT\_T, 62
  - qos, 62
- vos\_sockAccept
  - posix/vos\_sock.c, 290
  - vos\_sock.h, 321
  - windows/vos\_sock.c, 306
- vos\_sockBind
  - posix/vos\_sock.c, 291
  - vos\_sock.h, 322
  - windows/vos\_sock.c, 306
- vos\_sockClose
  - posix/vos\_sock.c, 291
  - vos\_sock.h, 323
  - windows/vos\_sock.c, 307
- vos\_sockConnect
  - posix/vos\_sock.c, 292
  - vos\_sock.h, 323
  - windows/vos\_sock.c, 307
- vos\_sockGetMAC
  - posix/vos\_sock.c, 292
  - vos\_sock.h, 324
  - windows/vos\_sock.c, 307
- vos\_sockInit
  - posix/vos\_sock.c, 293
  - vos\_sock.h, 325
  - windows/vos\_sock.c, 308
- vos\_sockJoinMC
  - posix/vos\_sock.c, 293
  - vos\_sock.h, 325
  - windows/vos\_sock.c, 308
- vos\_sockLeaveMC
  - posix/vos\_sock.c, 293
  - vos\_sock.h, 326
  - windows/vos\_sock.c, 308
- vos\_sockListen
  - posix/vos\_sock.c, 294
  - vos\_sock.h, 327
  - windows/vos\_sock.c, 309
- vos\_sockOpenTCP
  - posix/vos\_sock.c, 294
  - vos\_sock.h, 328
  - windows/vos\_sock.c, 309
- vos\_sockOpenUDP
  - posix/vos\_sock.c, 295
  - vos\_sock.h, 329
  - windows/vos\_sock.c, 310
- vos\_sockReceiveTCP
  - posix/vos\_sock.c, 295
  - vos\_sock.h, 330
  - windows/vos\_sock.c, 310
- vos\_sockReceiveUDP
  - posix/vos\_sock.c, 296
  - vos\_sock.h, 331
  - windows/vos\_sock.c, 311
- vos\_sockSendTCP
  - posix/vos\_sock.c, 296
  - vos\_sock.h, 332
  - windows/vos\_sock.c, 311
- vos\_sockSendUDP
  - posix/vos\_sock.c, 297
  - vos\_sock.h, 333
  - windows/vos\_sock.c, 312
- vos\_sockSetBuffer
  - posix/vos\_sock.c, 297
  - windows/vos\_sock.c, 312
- vos\_sockSetMulticastIf
  - posix/vos\_sock.c, 298
  - vos\_sock.h, 334
  - windows/vos\_sock.c, 313
- vos\_sockSetOptions
  - posix/vos\_sock.c, 298
  - vos\_sock.h, 335
  - windows/vos\_sock.c, 313
- vos\_sockTerm
  - posix/vos\_sock.c, 298
  - vos\_sock.h, 336
  - windows/vos\_sock.c, 313

- vos\_strncpy
  - vos\_mem.c, 259
  - vos\_mem.h, 269
- vos\_strncmp
  - vos\_mem.c, 260
  - vos\_mem.h, 270
- vos\_subTime
  - posix/vos\_thread.c, 345
  - vos\_thread.h, 370
  - windows/vos\_thread.c, 357
- vos\_terminate
  - vos\_utils.c, 381
  - vos\_utils.h, 385
- vos\_thread.c, 337, 348
- vos\_thread.h, 360
  - vos\_addTime, 363
  - vos\_clearTime, 363
  - vos\_cmpTime, 363
  - vos\_divTime, 364
  - vos\_getTime, 364
  - vos\_getTimeStamp, 364
  - vos\_getUuid, 365
  - vos\_mulTime, 365
  - vos\_mutexCreate, 365
  - vos\_mutexDelete, 366
  - vos\_mutexLock, 367
  - vos\_mutexTryLock, 367
  - vos\_mutexUnlock, 368
  - vos\_semaCreate, 368
  - vos\_semaDelete, 369
  - vos\_semaGive, 369
  - vos\_semaTake, 369
  - vos\_subTime, 370
  - vos\_threadCreate, 370
  - vos\_threadDelay, 372
  - vos\_threadInit, 372
  - vos\_threadIsActive, 373
  - vos\_threadTerm, 373
  - vos\_threadTerminate, 373
- vos\_threadCreate
  - posix/vos\_thread.c, 345
  - vos\_thread.h, 370
  - windows/vos\_thread.c, 357
- vos\_threadDelay
  - posix/vos\_thread.c, 346
  - vos\_thread.h, 372
  - windows/vos\_thread.c, 358
- vos\_threadInit
  - posix/vos\_thread.c, 346
  - vos\_thread.h, 372
  - windows/vos\_thread.c, 358
- vos\_threadIsActive
  - posix/vos\_thread.c, 346
  - vos\_thread.h, 373
  - windows/vos\_thread.c, 358
- vos\_threadTerm
  - posix/vos\_thread.c, 347
  - vos\_thread.h, 373
  - windows/vos\_thread.c, 358
- vos\_threadTerminate
  - posix/vos\_thread.c, 347
  - vos\_thread.h, 373
  - windows/vos\_thread.c, 358
- VOS\_TIME\_T, 63
  - tv\_usec, 63
- VOS\_TTL\_MULTICAST
  - vos\_sock.h, 317
- vos\_types.h, 375
  - VOS\_ERR\_T, 377
  - VOS\_LOG\_T, 378
  - VOS\_PRINT\_DBG\_T, 377
- vos\_utils.c, 379
  - vos\_crc32, 380
  - vos\_init, 380
  - vos\_initRuntimeConsts, 381
  - vos\_isBigEndian, 381
  - vos\_terminate, 381
- vos\_utils.h, 382
  - vos\_crc32, 384
  - vos\_init, 384
  - VOS\_MAX\_ERR\_STR\_SIZE, 383
  - VOS\_MAX\_FRMT\_SIZE, 383
  - VOS\_MAX\_PRNT\_STR\_SIZE, 383
  - vos\_terminate, 385
- windows/vos\_private.h
  - vos\_mutexLocalCreate, 274
  - vos\_mutexLocalDelete, 274
- windows/vos\_shared\_mem.c
  - vos\_sharedClose, 279
  - vos\_sharedOpen, 279
- windows/vos\_sock.c
  - recvmsg, 303
  - vos\_dottedIP, 303
  - vos\_getInterfaces, 303
  - vos\_htonl, 304
  - vos\_htons, 304
  - vos\_ipDotted, 304
  - vos\_isMulticast, 305
  - vos\_ntohl, 305
  - vos\_ntohs, 305
  - vos\_select, 305
  - vos\_sockAccept, 306
  - vos\_sockBind, 306
  - vos\_sockClose, 307
  - vos\_sockConnect, 307
  - vos\_sockGetMAC, 307
  - vos\_sockInit, 308

- vos\_sockJoinMC, [308](#)
- vos\_sockLeaveMC, [308](#)
- vos\_sockListen, [309](#)
- vos\_sockOpenTCP, [309](#)
- vos\_sockOpenUDP, [310](#)
- vos\_sockReceiveTCP, [310](#)
- vos\_sockReceiveUDP, [311](#)
- vos\_sockSendTCP, [311](#)
- vos\_sockSendUDP, [312](#)
- vos\_sockSetBuffer, [312](#)
- vos\_sockSetMulticastIf, [313](#)
- vos\_sockSetOptions, [313](#)
- vos\_sockTerm, [313](#)
- windows/vos\_thread.c
  - cyclicThread, [351](#)
  - vos\_addTime, [351](#)
  - vos\_clearTime, [351](#)
  - vos\_cmpTime, [351](#)
  - vos\_divTime, [352](#)
  - vos\_getFreeThreadHandle, [352](#)
  - vos\_getTime, [352](#)
  - vos\_getTimeStamp, [352](#)
  - vos\_getUuid, [352](#)
  - vos\_mulTime, [353](#)
  - vos\_mutexCreate, [353](#)
  - vos\_mutexDelete, [353](#)
  - vos\_mutexLocalCreate, [354](#)
  - vos\_mutexLocalDelete, [354](#)
  - vos\_mutexLock, [354](#)
  - vos\_mutexTryLock, [355](#)
  - vos\_mutexUnlock, [355](#)
  - vos\_semaCreate, [355](#)
  - vos\_semaDelete, [356](#)
  - vos\_semaGive, [356](#)
  - vos\_semaTake, [356](#)
  - vos\_subTime, [357](#)
  - vos\_threadCreate, [357](#)
  - vos\_threadDelay, [358](#)
  - vos\_threadInit, [358](#)
  - vos\_threadIsActive, [358](#)
  - vos\_threadTerm, [358](#)
  - vos\_threadTerminate, [358](#)