

# TCNOpen TRDP

## Prototype

Generated by Doxygen 1.5.6

Thu Jun 14 15:05:45 2012



# 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 . . . . .	10
4.2	MD_ELE Struct Reference . . . . .	11
4.2.1	Detailed Description . . . . .	11
4.3	PD_ELE Struct Reference . . . . .	12
4.3.1	Detailed Description . . . . .	13
4.4	TRDP_CAR_INFO_T Struct Reference . . . . .	14
4.4.1	Detailed Description . . . . .	15
4.4.2	Field Documentation . . . . .	15

4.4.2.1	orient . . . . .	15
4.4.2.2	pDevInfo . . . . .	15
4.5	TRDP_CST_INFO_T Struct Reference . . . . .	16
4.5.1	Detailed Description . . . . .	17
4.5.2	Field Documentation . . . . .	17
4.5.2.1	owner . . . . .	17
4.5.2.2	orient . . . . .	17
4.5.2.3	pFctInfo . . . . .	17
4.5.2.4	pCarInfo . . . . .	17
4.6	TRDP_DATASET_ELEMENT_T Struct Reference . . . . .	18
4.6.1	Detailed Description . . . . .	18
4.7	TRDP_DATASET_T Struct Reference . . . . .	19
4.7.1	Detailed Description . . . . .	19
4.8	TRDP_DBG_CONFIG_T Struct Reference . . . . .	20
4.8.1	Detailed Description . . . . .	20
4.9	TRDP_DEVICE_INFO_T Struct Reference . . . . .	21
4.9.1	Detailed Description . . . . .	22
4.9.2	Field Documentation . . . . .	22
4.9.2.1	orient . . . . .	22
4.10	TRDP_FCT_INFO_T Struct Reference . . . . .	23
4.10.1	Detailed Description . . . . .	23
4.11	TRDP_HANDLE Struct Reference . . . . .	24
4.11.1	Detailed Description . . . . .	24
4.12	TRDP_LIST_STATISTICS_T Struct Reference . . . . .	25
4.12.1	Detailed Description . . . . .	25
4.13	TRDP_MARSHALL_CONFIG_T Struct Reference . . . . .	26
4.13.1	Detailed Description . . . . .	26
4.14	TRDP_MD_CONFIG_T Struct Reference . . . . .	27
4.14.1	Detailed Description . . . . .	27
4.15	TRDP_MD_INFO_T Struct Reference . . . . .	28
4.15.1	Detailed Description . . . . .	29
4.15.2	Field Documentation . . . . .	29
4.15.2.1	msgType . . . . .	29
4.16	TRDP_MD_STATISTICS Struct Reference . . . . .	30
4.16.1	Detailed Description . . . . .	30
4.17	TRDP_MD_STATISTICS_T Struct Reference . . . . .	31

4.17.1 Detailed Description . . . . .	32
4.18 TRDP_MEM_CONFIG_T Struct Reference . . . . .	33
4.18.1 Detailed Description . . . . .	33
4.19 TRDP_MEM_STATISTICS_T Struct Reference . . . . .	34
4.19.1 Detailed Description . . . . .	34
4.20 TRDP_PD_CONFIG_T Struct Reference . . . . .	35
4.20.1 Detailed Description . . . . .	35
4.21 TRDP_PD_INFO_T Struct Reference . . . . .	36
4.21.1 Detailed Description . . . . .	37
4.21.2 Field Documentation . . . . .	37
4.21.2.1 msgType . . . . .	37
4.22 TRDP_PD_STATISTICS Struct Reference . . . . .	38
4.22.1 Detailed Description . . . . .	38
4.23 TRDP_PD_STATISTICS_T Struct Reference . . . . .	39
4.23.1 Detailed Description . . . . .	40
4.24 TRDP_PROCESS_CONFIG_T Struct Reference . . . . .	41
4.24.1 Detailed Description . . . . .	41
4.25 TRDP_PROP_INFO_T Struct Reference . . . . .	42
4.25.1 Detailed Description . . . . .	42
4.26 TRDP_PUB_STATISTICS_T Struct Reference . . . . .	43
4.26.1 Detailed Description . . . . .	43
4.26.2 Field Documentation . . . . .	43
4.26.2.1 destAddr . . . . .	43
4.27 TRDP_RED_STATISTICS_T Struct Reference . . . . .	44
4.27.1 Detailed Description . . . . .	44
4.28 TRDP_SEND_PARAM_T Struct Reference . . . . .	45
4.28.1 Detailed Description . . . . .	45
4.29 TRDP_SESSION Struct Reference . . . . .	46
4.29.1 Detailed Description . . . . .	47
4.30 TRDP_SOCKETS Struct Reference . . . . .	48
4.30.1 Detailed Description . . . . .	48
4.30.2 Field Documentation . . . . .	48
4.30.2.1 usage . . . . .	48
4.31 TRDP_STATISTICS_T Struct Reference . . . . .	49
4.31.1 Detailed Description . . . . .	50
4.32 TRDP_SUBS_STATISTICS_T Struct Reference . . . . .	51

4.32.1	Detailed Description	51
4.32.2	Field Documentation	51
4.32.2.1	filterAddr	51
4.32.2.2	timeout	51
4.32.2.3	toBehav	52
4.32.2.4	numRecv	52
4.33	TRDP_TRAIN_INFO_T Struct Reference	53
4.33.1	Detailed Description	54
4.33.2	Field Documentation	54
4.33.2.1	operator	54
4.33.2.2	topoCnt	54
4.33.2.3	pCstInfo	54
4.34	VOS SOCK_OPT_T Struct Reference	55
4.34.1	Detailed Description	55
4.34.2	Field Documentation	55
4.34.2.1	qos	55
4.35	VOS_TIME_T Struct Reference	56
4.35.1	Detailed Description	56
4.35.2	Field Documentation	56
4.35.2.1	tv_usec	56
<b>5</b>	<b>File Documentation</b>	<b>57</b>
5.1	echoPolling.c File Reference	57
5.1.1	Detailed Description	58
5.1.2	Function Documentation	58
5.1.2.1	dbgOut	58
5.1.2.2	main	58
5.2	echoSelect.c File Reference	61
5.2.1	Detailed Description	61
5.2.2	Function Documentation	62
5.2.2.1	dbgOut	62
5.2.2.2	main	62
5.2.2.3	myPDcallBack	64
5.3	sendHello.c File Reference	65
5.3.1	Detailed Description	65
5.3.2	Function Documentation	66
5.3.2.1	main	66

5.4	tau_addr.h File Reference	68
5.4.1	Detailed Description	70
5.4.2	Function Documentation	70
5.4.2.1	tau_addr2CarId	70
5.4.2.2	tau_addr2CarNo	71
5.4.2.3	tau_addr2CstId	71
5.4.2.4	tau_addr2CstNo	71
5.4.2.5	tau_addr2IecCarNo	72
5.4.2.6	tau_addr2IecCstNo	72
5.4.2.7	tau_addr2Uri	72
5.4.2.8	tau_carNo2Ids	73
5.4.2.9	tau_cstNo2CstId	73
5.4.2.10	tau_getOwnAddr	73
5.4.2.11	tau_getOwnIds	73
5.4.2.12	tau_iecCarNo2Ids	74
5.4.2.13	tau_iecCstNo2CstId	74
5.4.2.14	tau_label2CarId	75
5.4.2.15	tau_label2CarNo	75
5.4.2.16	tau_label2CstId	75
5.4.2.17	tau_label2CstNo	76
5.4.2.18	tau_label2IecCarNo	76
5.4.2.19	tau_label2IecCstNo	76
5.4.2.20	tau_uri2Addr	77
5.5	tau_marshall.h File Reference	78
5.5.1	Detailed Description	79
5.5.2	Typedef Documentation	79
5.5.2.1	tau_calcDatasetSize	79
5.5.2.2	tau_marshall	80
5.5.2.3	tau_marshallIDs	80
5.5.2.4	tau_unmarshall	80
5.5.2.5	tau_unmarshallIDs	81
5.5.3	Function Documentation	81
5.5.3.1	tau_initMarshall	81
5.6	tau_tci.h File Reference	82
5.6.1	Detailed Description	84
5.6.2	Enumeration Type Documentation	84

5.6.2.1	TRDP_FCT_T . . . . .	84
5.6.2.2	TRDP_INAUG_STATE_T . . . . .	85
5.6.3	Function Documentation . . . . .	85
5.6.3.1	tau_getCarDevCnt . . . . .	85
5.6.3.2	tau_getCarInfo . . . . .	85
5.6.3.3	tau_getCarOrient . . . . .	86
5.6.3.4	tau_getCstCarCnt . . . . .	86
5.6.3.5	tau_getCstFctCnt . . . . .	86
5.6.3.6	tau_getCstFctInfo . . . . .	87
5.6.3.7	tau_getCstInfo . . . . .	87
5.6.3.8	tau_getDevInfo . . . . .	88
5.6.3.9	tau_getEtbState . . . . .	88
5.6.3.10	tau_getIecCarOrient . . . . .	88
5.6.3.11	tau_getTrnCarCnt . . . . .	89
5.6.3.12	tau_getTrnCstCnt . . . . .	89
5.6.3.13	tau_getTrnInfo . . . . .	89
5.7	tau_types.h File Reference . . . . .	90
5.7.1	Detailed Description . . . . .	90
5.8	tau_xml.h File Reference . . . . .	91
5.8.1	Detailed Description . . . . .	92
5.8.2	Enumeration Type Documentation . . . . .	92
5.8.2.1	TRDP_DBG_OPTION_T . . . . .	92
5.8.3	Function Documentation . . . . .	93
5.8.3.1	tau_readXmlConfig . . . . .	93
5.8.3.2	tau_readXmlDatasetConfig . . . . .	93
5.9	trdp_if.c File Reference . . . . .	95
5.9.1	Detailed Description . . . . .	97
5.9.2	Function Documentation . . . . .	97
5.9.2.1	tlc_getInterval . . . . .	97
5.9.2.2	tlc_getVersion . . . . .	98
5.9.2.3	tlc_init . . . . .	98
5.9.2.4	tlc_process . . . . .	99
5.9.2.5	tlc_reinit . . . . .	100
5.9.2.6	tlc_setTopoCount . . . . .	101
5.9.2.7	tlc_terminate . . . . .	101
5.9.2.8	tlp_get . . . . .	101



5.9.2.9	<a href="#">tlp_getRedundant</a>	102
5.9.2.10	<a href="#">tlp_publish</a>	103
5.9.2.11	<a href="#">tlp_put</a>	104
5.9.2.12	<a href="#">tlp_setRedundant</a>	105
5.9.2.13	<a href="#">tlp_subscribe</a>	105
5.9.2.14	<a href="#">tlp_unpublish</a>	106
5.9.2.15	<a href="#">tlp_unsubscribe</a>	107
5.9.2.16	<a href="#">trdp_isValidSession</a>	108
5.9.2.17	<a href="#">trdp_sessionQueue</a>	108
5.10	<a href="#">trdp_if.h File Reference</a>	109
5.10.1	<a href="#">Detailed Description</a>	109
5.10.2	<a href="#">Function Documentation</a>	110
5.10.2.1	<a href="#">trdp_isValidSession</a>	110
5.10.2.2	<a href="#">trdp_sessionQueue</a>	110
5.11	<a href="#">trdp_if_light.h File Reference</a>	111
5.11.1	<a href="#">Detailed Description</a>	114
5.11.2	<a href="#">Function Documentation</a>	115
5.11.2.1	<a href="#">tlc_freeBuf</a>	115
5.11.2.2	<a href="#">tlc_getInterval</a>	115
5.11.2.3	<a href="#">tlc_getJoinStatistics</a>	116
5.11.2.4	<a href="#">tlc_getListStatistics</a>	116
5.11.2.5	<a href="#">tlc_getPubStatistics</a>	117
5.11.2.6	<a href="#">tlc_getRedStatistics</a>	117
5.11.2.7	<a href="#">tlc_getStatistics</a>	118
5.11.2.8	<a href="#">tlc_getSubsStatistics</a>	118
5.11.2.9	<a href="#">tlc_getVersion</a>	119
5.11.2.10	<a href="#">tlc_init</a>	119
5.11.2.11	<a href="#">tlc_process</a>	121
5.11.2.12	<a href="#">tlc_reinit</a>	122
5.11.2.13	<a href="#">tlc_resetStatistics</a>	122
5.11.2.14	<a href="#">tlc_setTopoCount</a>	123
5.11.2.15	<a href="#">tlc_terminate</a>	123
5.11.2.16	<a href="#">tlm_abortSession</a>	123
5.11.2.17	<a href="#">tlm_addListener</a>	124
5.11.2.18	<a href="#">tlm_confirm</a>	124
5.11.2.19	<a href="#">tlm_delListener</a>	125

5.11.2.20	tlm_notify	125
5.11.2.21	tlm_reply	126
5.11.2.22	tlm_replyErr	127
5.11.2.23	tlm_replyQuery	128
5.11.2.24	tlm_request	128
5.11.2.25	tlp_get	129
5.11.2.26	tlp_getRedundant	131
5.11.2.27	tlp_publish	131
5.11.2.28	tlp_put	133
5.11.2.29	tlp_request	134
5.11.2.30	tlp_setRedundant	135
5.11.2.31	tlp_subscribe	136
5.11.2.32	tlp_unpublish	137
5.11.2.33	tlp_unsubscribe	137
5.12	trdp_mdcom.c File Reference	139
5.12.1	Detailed Description	139
5.12.2	Function Documentation	140
5.12.2.1	trdp_rcvMD	140
5.12.2.2	trdp_sendMD	140
5.13	trdp_mdcom.h File Reference	141
5.13.1	Detailed Description	141
5.13.2	Function Documentation	142
5.13.2.1	trdp_rcvMD	142
5.13.2.2	trdp_sendMD	142
5.14	trdp_pdcom.c File Reference	143
5.14.1	Detailed Description	144
5.14.2	Function Documentation	144
5.14.2.1	trdp_pdCheck	144
5.14.2.2	trdp_pdInit	145
5.14.2.3	trdp_pdReceive	145
5.14.2.4	trdp_pdSend	146
5.14.2.5	trdp_pdUpdate	146
5.15	trdp_pdcom.h File Reference	148
5.15.1	Detailed Description	149
5.15.2	Function Documentation	149
5.15.2.1	trdp_pdCheck	149

5.15.2.2	trdp_pdInit	150
5.15.2.3	trdp_pdReceive	150
5.15.2.4	trdp_pdSend	151
5.15.2.5	trdp_pdUpdate	151
5.16	trdp_private.h File Reference	153
5.16.1	Detailed Description	155
5.16.2	Enumeration Type Documentation	156
5.16.2.1	TRDP_PRIV_FLAGS_T	156
5.16.2.2	TRDP SOCK_TYPE_T	156
5.17	trdp_stats.c File Reference	157
5.17.1	Detailed Description	158
5.17.2	Function Documentation	158
5.17.2.1	tlc_getJoinStatistics	158
5.17.2.2	tlc_getListStatistics	159
5.17.2.3	tlc_getPubStatistics	159
5.17.2.4	tlc_getRedStatistics	160
5.17.2.5	tlc_getStatistics	160
5.17.2.6	tlc_getSubsStatistics	161
5.17.2.7	tlc_resetStatistics	161
5.18	trdp_stats.h File Reference	162
5.18.1	Detailed Description	162
5.19	trdp_types.h File Reference	163
5.19.1	Detailed Description	167
5.19.2	Define Documentation	168
5.19.2.1	TRDP_MAX_FILE_NAME_LEN	168
5.19.2.2	TRDP_MAX_LABEL_LEN	168
5.19.2.3	TRDP_MAX_URI_HOST_LEN	168
5.19.2.4	TRDP_MAX_URI_LEN	168
5.19.2.5	TRDP_MAX_URI_USER_LEN	168
5.19.3	Typedef Documentation	168
5.19.3.1	TRDP_IP_ADDR_T	168
5.19.3.2	TRDP_MARSHALL_T	169
5.19.3.3	TRDP_MD_CALLBACK_T	169
5.19.3.4	TRDP_PD_CALLBACK_T	169
5.19.3.5	TRDP_PRINT_DBG_T	169
5.19.3.6	TRDP_TIME_T	170

5.19.3.7	TRDP_UNMARSHALL_T . . . . .	170
5.19.4	Enumeration Type Documentation . . . . .	170
5.19.4.1	TRDP_DATA_TYPE_T . . . . .	170
5.19.4.2	TRDP_ERR_T . . . . .	171
5.19.4.3	TRDP_FLAGS_T . . . . .	172
5.19.4.4	TRDP_MSG_T . . . . .	172
5.19.4.5	TRDP_OPTION_T . . . . .	172
5.19.4.6	TRDP_RED_STATE_T . . . . .	172
5.20	trdp_utils.c File Reference . . . . .	173
5.20.1	Detailed Description . . . . .	174
5.20.2	Function Documentation . . . . .	174
5.20.2.1	am_big_endian . . . . .	174
5.20.2.2	trdp_initSockets . . . . .	174
5.20.2.3	trdp_packetSizePD . . . . .	175
5.20.2.4	trdp_queueAppLast . . . . .	175
5.20.2.5	trdp_queueDelElement . . . . .	175
5.20.2.6	trdp_queueFindAddr . . . . .	175
5.20.2.7	trdp_queueFindComId . . . . .	175
5.20.2.8	trdp_queueInsFirst . . . . .	176
5.20.2.9	trdp_releaseSocket . . . . .	176
5.20.2.10	trdp_requestSocket . . . . .	176
5.21	trdp_utils.h File Reference . . . . .	178
5.21.1	Detailed Description . . . . .	179
5.21.2	Function Documentation . . . . .	179
5.21.2.1	am_big_endian . . . . .	179
5.21.2.2	trdp_initSockets . . . . .	180
5.21.2.3	trdp_packetSizePD . . . . .	180
5.21.2.4	trdp_queueAppLast . . . . .	180
5.21.2.5	trdp_queueDelElement . . . . .	180
5.21.2.6	trdp_queueFindAddr . . . . .	180
5.21.2.7	trdp_queueFindComId . . . . .	181
5.21.2.8	trdp_queueInsFirst . . . . .	181
5.21.2.9	trdp_releaseSocket . . . . .	181
5.21.2.10	trdp_requestSocket . . . . .	181
5.22	vos_mem.c File Reference . . . . .	183
5.22.1	Detailed Description . . . . .	184

5.22.2	Function Documentation	184
5.22.2.1	vos_memAlloc	184
5.22.2.2	vos_memCount	185
5.22.2.3	vos_memDelete	185
5.22.2.4	vos_memFree	186
5.22.2.5	vos_memInit	186
5.22.2.6	vos_queueCreate	187
5.22.2.7	vos_queueDestroy	187
5.22.2.8	vos_queueReceive	187
5.22.2.9	vos_queueSend	188
5.22.2.10	vos_sharedClose	188
5.22.2.11	vos_sharedOpen	189
5.23	vos_mem.h File Reference	190
5.23.1	Detailed Description	191
5.23.2	Define Documentation	192
5.23.2.1	VOS_MEM_BLOCKSIZE	192
5.23.2.2	VOS_MEM_PREALLOCATE	192
5.23.3	Function Documentation	192
5.23.3.1	vos_memAlloc	192
5.23.3.2	vos_memCount	193
5.23.3.3	vos_memDelete	193
5.23.3.4	vos_memFree	193
5.23.3.5	vos_memInit	194
5.23.3.6	vos_queueCreate	195
5.23.3.7	vos_queueDestroy	195
5.23.3.8	vos_queueReceive	195
5.23.3.9	vos_queueSend	196
5.23.3.10	vos_sharedClose	196
5.23.3.11	vos_sharedOpen	197
5.24	vos_sock.c File Reference	198
5.24.1	Detailed Description	200
5.24.2	Function Documentation	200
5.24.2.1	vos_htonl	200
5.24.2.2	vos_htons	200
5.24.2.3	vos_isMulticast	201
5.24.2.4	vos_ntohl	201

5.24.2.5	<a href="#">vos_ntohs</a>	201
5.24.2.6	<a href="#">vos_sockAccept</a>	201
5.24.2.7	<a href="#">vos_sockBind</a>	202
5.24.2.8	<a href="#">vos_sockClose</a>	202
5.24.2.9	<a href="#">vos_sockConnect</a>	203
5.24.2.10	<a href="#">vos_sockInit</a>	203
5.24.2.11	<a href="#">vos_sockJoinMC</a>	203
5.24.2.12	<a href="#">vos_sockLeaveMC</a>	204
5.24.2.13	<a href="#">vos_sockListen</a>	204
5.24.2.14	<a href="#">vos_sockOpenTCP</a>	205
5.24.2.15	<a href="#">vos_sockOpenUDP</a>	205
5.24.2.16	<a href="#">vos_sockReceiveTCP</a>	206
5.24.2.17	<a href="#">vos_sockReceiveUDP</a>	206
5.24.2.18	<a href="#">vos_sockSendTCP</a>	207
5.24.2.19	<a href="#">vos_sockSendUDP</a>	207
5.24.2.20	<a href="#">vos_sockSetOptions</a>	208
5.25	<a href="#">vos_sock.h File Reference</a>	209
5.25.1	<a href="#">Detailed Description</a>	211
5.25.2	<a href="#">Function Documentation</a>	211
5.25.2.1	<a href="#">vos_htonl</a>	211
5.25.2.2	<a href="#">vos_htons</a>	211
5.25.2.3	<a href="#">vos_isMulticast</a>	212
5.25.2.4	<a href="#">vos_ntohl</a>	212
5.25.2.5	<a href="#">vos_ntohs</a>	212
5.25.2.6	<a href="#">vos_sockAccept</a>	213
5.25.2.7	<a href="#">vos_sockBind</a>	213
5.25.2.8	<a href="#">vos_sockClose</a>	214
5.25.2.9	<a href="#">vos_sockConnect</a>	214
5.25.2.10	<a href="#">vos_sockInit</a>	215
5.25.2.11	<a href="#">vos_sockJoinMC</a>	215
5.25.2.12	<a href="#">vos_sockLeaveMC</a>	216
5.25.2.13	<a href="#">vos_sockListen</a>	217
5.25.2.14	<a href="#">vos_sockOpenTCP</a>	218
5.25.2.15	<a href="#">vos_sockOpenUDP</a>	218
5.25.2.16	<a href="#">vos_sockReceiveTCP</a>	219
5.25.2.17	<a href="#">vos_sockReceiveUDP</a>	220

5.25.2.18	<code>vos_sockSendTCP</code>	221
5.25.2.19	<code>vos_sockSendUDP</code>	222
5.25.2.20	<code>vos_sockSetOptions</code>	223
5.26	<code>vos_thread.c</code> File Reference	224
5.26.1	Detailed Description	226
5.26.2	Function Documentation	226
5.26.2.1	<code>cyclicThread</code>	226
5.26.2.2	<code>vos_addTime</code>	226
5.26.2.3	<code>vos_clearTime</code>	227
5.26.2.4	<code>vos_cmpTime</code>	227
5.26.2.5	<code>vos_getTime</code>	227
5.26.2.6	<code>vos_getTimeStamp</code>	228
5.26.2.7	<code>vos_getUuid</code>	228
5.26.2.8	<code>vos_mutexCreate</code>	228
5.26.2.9	<code>vos_mutexDelete</code>	229
5.26.2.10	<code>vos_mutexLock</code>	229
5.26.2.11	<code>vos_mutexTryLock</code>	229
5.26.2.12	<code>vos_mutexUnlock</code>	230
5.26.2.13	<code>vos_semaCreate</code>	230
5.26.2.14	<code>vos_semaDelete</code>	230
5.26.2.15	<code>vos_semaGive</code>	231
5.26.2.16	<code>vos_semaTake</code>	231
5.26.2.17	<code>vos_subTime</code>	231
5.26.2.18	<code>vos_threadCreate</code>	232
5.26.2.19	<code>vos_threadDelay</code>	232
5.26.2.20	<code>vos_threadInit</code>	232
5.26.2.21	<code>vos_threadIsActive</code>	233
5.26.2.22	<code>vos_threadTerminate</code>	233
5.27	<code>vos_thread.h</code> File Reference	234
5.27.1	Detailed Description	236
5.27.2	Function Documentation	236
5.27.2.1	<code>vos_addTime</code>	236
5.27.2.2	<code>vos_clearTime</code>	237
5.27.2.3	<code>vos_cmpTime</code>	237
5.27.2.4	<code>vos_getTime</code>	237
5.27.2.5	<code>vos_getTimeStamp</code>	238

5.27.2.6	<code>vos_getUuid</code>	238
5.27.2.7	<code>vos_mutexCreate</code>	238
5.27.2.8	<code>vos_mutexDelete</code>	239
5.27.2.9	<code>vos_mutexLock</code>	240
5.27.2.10	<code>vos_mutexTryLock</code>	240
5.27.2.11	<code>vos_mutexUnlock</code>	241
5.27.2.12	<code>vos_semaCreate</code>	241
5.27.2.13	<code>vos_semaDelete</code>	242
5.27.2.14	<code>vos_semaGive</code>	242
5.27.2.15	<code>vos_semaTake</code>	242
5.27.2.16	<code>vos_subTime</code>	243
5.27.2.17	<code>vos_threadCreate</code>	243
5.27.2.18	<code>vos_threadDelay</code>	244
5.27.2.19	<code>vos_threadInit</code>	245
5.27.2.20	<code>vos_threadIsActive</code>	245
5.27.2.21	<code>vos_threadTerminate</code>	245
5.28	<code>vos_types.h</code> File Reference	247
5.28.1	Detailed Description	248
5.28.2	Typedef Documentation	248
5.28.2.1	<code>VOS_PRINT_DBG_T</code>	248
5.28.3	Enumeration Type Documentation	249
5.28.3.1	<code>VOS_ERR_T</code>	249
5.28.3.2	<code>VOS_LOG_T</code>	249
5.28.4	Function Documentation	250
5.28.4.1	<code>vos_init</code>	250
5.29	<code>vos_utils.c</code> File Reference	251
5.29.1	Detailed Description	251
5.29.2	Function Documentation	251
5.29.2.1	<code>vos_crc32</code>	251
5.29.2.2	<code>vos_init</code>	252
5.30	<code>vos_utils.h</code> File Reference	253
5.30.1	Detailed Description	253
5.30.2	Function Documentation	254
5.30.2.1	<code>vos_crc32</code>	254



# 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.

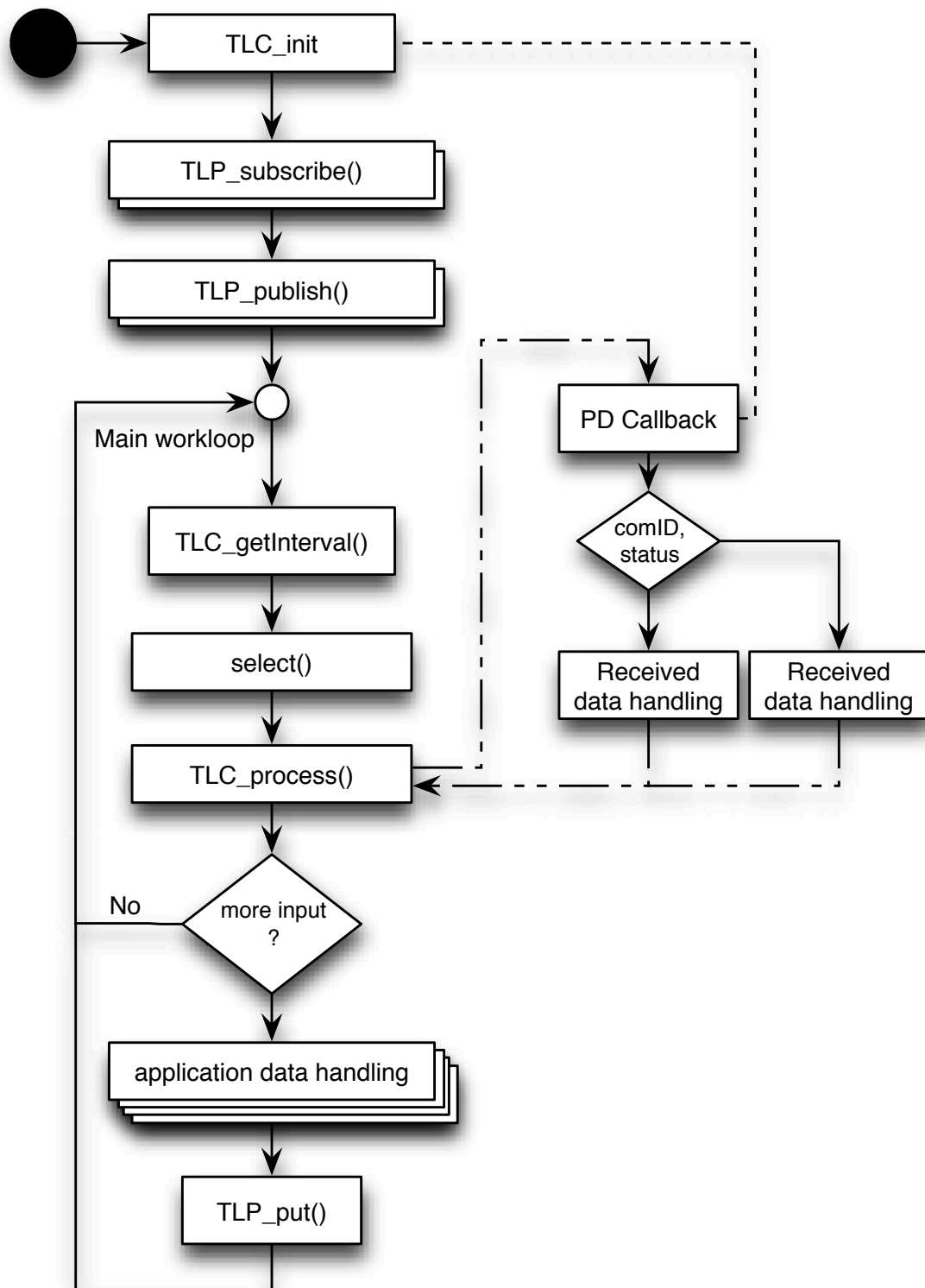


Figure 1.1: Sample client workflow

## 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="#">MD_ELE</a> (Queue element for MD packets to send or receive or acknowledge ) . . . . .	11
<a href="#">PD_ELE</a> (Queue element for PD packets to send or receive ) . . . . .	12
<a href="#">TRDP_CAR_INFO_T</a> (Car information structure ) . . . . .	14
<a href="#">TRDP_CST_INFO_T</a> (Consist information structure ) . . . . .	16
<a href="#">TRDP_DATASET_ELEMENT_T</a> (Dataset element definition ) . . . . .	18
<a href="#">TRDP_DATASET_T</a> (Dataset definition ) . . . . .	19
<a href="#">TRDP_DBG_CONFIG_T</a> (Control for debug output device/file on application level ) . . . . .	20
<a href="#">TRDP_DEVICE_INFO_T</a> (Device information structure ) . . . . .	21
<a href="#">TRDP_FCT_INFO_T</a> (Device information structure ) . . . . .	23
<a href="#">TRDP_HANDLE</a> (Hidden handle definition, used as unique addressing item ) . . . . .	24
<a href="#">TRDP_LIST_STATISTICS_T</a> (Information about a particular MD listener ) . . . . .	25
<a href="#">TRDP_MARSHALL_CONFIG_T</a> (Marshaling/unmarshalling configuration ) . . . . .	26
<a href="#">TRDP_MD_CONFIG_T</a> (Default MD configuration ) . . . . .	27
<a href="#">TRDP_MD_INFO_T</a> (Message data info from received telegram; allows the application to generate responses ) . . . . .	28
<a href="#">TRDP_MD_STATISTICS</a> (Message data statistics ) . . . . .	30
<a href="#">TRDP_MD_STATISTICS_T</a> (Structure containing all general MD statistics information ) . . . . .	31
<a href="#">TRDP_MEM_CONFIG_T</a> (Structure describing memory (and its pre-fragmentation) ) . . . . .	33
<a href="#">TRDP_MEM_STATISTICS_T</a> (TRDP statistics type definitions ) . . . . .	34
<a href="#">TRDP_PD_CONFIG_T</a> (Default PD configuration ) . . . . .	35
<a href="#">TRDP_PD_INFO_T</a> (Process data info from received telegram; allows the application to generate responses ) . . . . .	36
<a href="#">TRDP_PD_STATISTICS</a> (Process data statistics ) . . . . .	38
<a href="#">TRDP_PD_STATISTICS_T</a> (Structure containing all general PD statistics information ) . . . . .	39
<a href="#">TRDP_PROCESS_CONFIG_T</a> (Types to read out the XML configuration ) . . . . .	41
<a href="#">TRDP_PROP_INFO_T</a> (Properties information structure ) . . . . .	42
<a href="#">TRDP_PUB_STATISTICS_T</a> (Table containing particular PD publishing information ) . . . . .	43
<a href="#">TRDP_RED_STATISTICS_T</a> (A table containing PD redundant group information ) . . . . .	44
<a href="#">TRDP_SEND_PARAM_T</a> (Quality/type of service and time to live ) . . . . .	45
<a href="#">TRDP_SESSION</a> (Session/application variables store ) . . . . .	46
<a href="#">TRDP_SOCKETS</a> (Socket item ) . . . . .	48

<a href="#">TRDP_STATISTICS_T</a> (Structure containing all general memory, PD and MD statistics information) . . . . .	<a href="#">49</a>
<a href="#">TRDP_SUBS_STATISTICS_T</a> (Table containing particular PD subscription information) . . .	<a href="#">51</a>
<a href="#">TRDP_TRAIN_INFO_T</a> (Train information structure) . . . . .	<a href="#">53</a>
<a href="#">VOS SOCK_OPT_T</a> (Common socket options) . . . . .	<a href="#">55</a>
<a href="#">VOS_TIME_T</a> (Timer value compatible with timeval / select) . . . . .	<a href="#">56</a>

# Chapter 3

## File Index

### 3.1 File List

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

<a href="#">echoPolling.c</a> (Demo echoing application for TRDP ) . . . . .	57
<a href="#">echoSelect.c</a> (Demo echoing application for TRDP ) . . . . .	61
<a href="#">sendHello.c</a> (Demo application for TRDP ) . . . . .	65
<a href="#">tau_addr.h</a> (TRDP utility interface definitions ) . . . . .	68
<a href="#">tau_marshall.h</a> (TRDP utility interface definitions ) . . . . .	78
<a href="#">tau_tci.h</a> (TRDP utility interface definitions ) . . . . .	82
<a href="#">tau_types.h</a> (TRDP utility interface definitions ) . . . . .	90
<a href="#">tau_xml.h</a> (TRDP utility interface definitions ) . . . . .	91
<a href="#">trdp_if.c</a> (Functions for ECN communication ) . . . . .	95
<a href="#">trdp_if.h</a> (Typedefs for TRDP communication ) . . . . .	109
<a href="#">trdp_if_light.h</a> (TRDP Light interface functions (API) ) . . . . .	111
<a href="#">trdp_mdcom.c</a> (Functions for MD communication ) . . . . .	139
<a href="#">trdp_mdcom.h</a> (Functions for MD communication ) . . . . .	141
<a href="#">trdp_pdcom.c</a> (Functions for PD communication ) . . . . .	143
<a href="#">trdp_pdcom.h</a> (Functions for PD communication ) . . . . .	148
<a href="#">trdp_private.h</a> (Typedefs for TRDP communication ) . . . . .	153
<a href="#">trdp_stats.c</a> (Statistics functions for TRDP communication ) . . . . .	157
<a href="#">trdp_stats.h</a> (Statistics for TRDP communication ) . . . . .	162
<a href="#">trdp_types.h</a> (Typedefs for TRDP communication ) . . . . .	163
<a href="#">trdp_utils.c</a> (Helper functions for TRDP communication ) . . . . .	173
<a href="#">trdp_utils.h</a> (Common utilities for TRDP communication ) . . . . .	178
<a href="#">vos_mem.c</a> (Memory functions ) . . . . .	183
<a href="#">vos_mem.h</a> (Memory and queue functions for OS abstraction ) . . . . .	190
<a href="#">vos_sock.c</a> (Socket functions ) . . . . .	198
<a href="#">vos_sock.h</a> (Typedefs for OS abstraction ) . . . . .	209
<a href="#">vos_thread.c</a> (Multitasking functions ) . . . . .	224
<a href="#">vos_thread.h</a> (Threading functions for OS abstraction ) . . . . .	234
<a href="#">vos_types.h</a> (Typedefs for OS abstraction ) . . . . .	247
<a href="#">vos_utils.c</a> (Common functions for VOS ) . . . . .	251
<a href="#">vos_utils.h</a> (Typedefs for OS abstraction ) . . . . .	253





## 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.*
- UINT16 [subsAndReserved](#)  
*first bit (MSB): indicates substitution transmission*
- UINT16 [offsetAddress](#)  
*for process data in traffic store*
- UINT32 [replyComId](#)  
*used in PD request*

- UINT32 [replyIpAddress](#)  
*used for PD request*
- 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.*

### 4.1.1 Detailed Description

TRDP process data header - network order and alignment.

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 file:

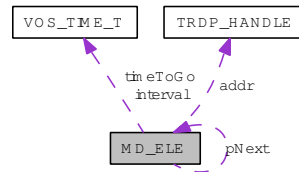
- [trdp\\_private.h](#)

## 4.2 MD\_ELE Struct Reference

Queue element for MD packets to send or receive or acknowledge.

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

Collaboration diagram for MD\_ELE:



### Data Fields

- struct `MD_ELE` \* `pNext`  
*pointer to next element or NULL*
- `TRDP_ADDRESSES` `addr`  
*handle of publisher/subscriber*
- `TRDP_PRIV_FLAGS_T` `privFlags`  
*private 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*
- INT32 `dataSize`  
*net data size*
- INT32 `socketIdx`  
*index into the socket list*
- `MD_HEADER_T` `frameHead`  
*Packet header in network byte order.*
- UINT8 `data` [0]  
*data ready to be sent (with CRCs)*

### 4.2.1 Detailed Description

Queue element for MD packets to send or receive or acknowledge.

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

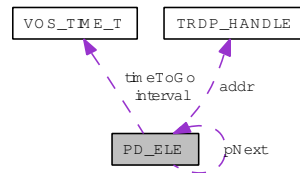
- [trdp\\_private.h](#)

### 4.3 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*
- [TRDP\\_ADDRESSES](#) [addr](#)  
*handle of publisher/subscriber*
- [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*
- [UINT32](#) [dataSize](#)  
*net data size*
- [UINT32](#) [grossSize](#)  
*complete packet size (header, data, padding, FCS)*
- [INT32](#) [socketIdx](#)  
*index into the socket list*
- const void \* [userRef](#)  
*from subscribe()*
- [PD\\_HEADER\\_T](#) [frameHead](#)  
*Packet header in network byte order.*

- `UINT8 data [MAX_PD_PACKET_SIZE]`  
*data ready to be sent or received (with CRCs)*

### 4.3.1 Detailed Description

Queue element for PD packets to send or receive.

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

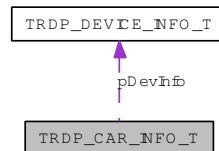
- [trdp\\_private.h](#)

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

car information structure.

```
#include <tau_tci.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:

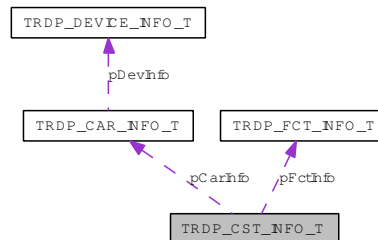
- [tau\\_tci.h](#)

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

consist information structure.

```
#include <tau_tci.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.*

### 4.5.1 Detailed Description

consist information structure.

### 4.5.2 Field Documentation

#### 4.5.2.1 `TRDP_LABEL_T TRDP_CST_INFO_T::owner`

consist owner, e.g.

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

#### 4.5.2.2 `UINT8 TRDP_CST_INFO_T::orient`

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

to train

#### 4.5.2.3 `TRDP_FCT_INFO_T* TRDP_CST_INFO_T::pFctInfo`

Pointer to function info list for application use and convenience.

#### 4.5.2.4 `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\\_tci.h](#)

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

Dataset element definition.

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

### Data Fields

- INT32 [type](#)  
*Data type or dataset id.*
- UINT32 [size](#)  
*Number of items or TDRP\_VAR\_SIZE (0).*

### 4.6.1 Detailed Description

Dataset element definition.

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

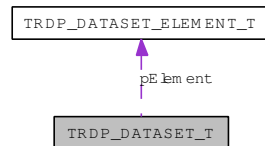
- [trdp\\_types.h](#)

## 4.7 TRDP\_DATASET\_T Struct Reference

Dataset definition.

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

Collaboration diagram for TRDP\_DATASET\_T:



### Data Fields

- INT32 `id`  
*dataset identifier*
- 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\_DBG\_CONFIG\_T Struct Reference

Control for debug output device/file on application level.

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

### Data Fields

- TRDP\_DEBUG\_OPTION\_T [option](#)  
*Debug printout options for application use.*
- UINT32 [maxFileSize](#)  
*Maximal file size.*
- TRDP\_FILE\_NAME\_T [fileName](#)  
*Debug file name and path.*

### 4.8.1 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](#)

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

device information structure

```
#include <tau_tci.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.*

### 4.9.1 Detailed Description

device information structure

### 4.9.2 Field Documentation

#### 4.9.2.1 `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\\_tci.h](#)

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

device information structure

```
#include <tau_tci.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.*

### 4.10.1 Detailed Description

device information structure

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

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

## 4.11 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*

### 4.11.1 Detailed Description

Hidden handle definition, used as unique addressing item.

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

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



## 4.12 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.*

### 4.12.1 Detailed Description

Information about a particular MD listener.

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

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

## 4.13 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.*

### 4.13.1 Detailed Description

Marshaling/unmarshalling configuration.

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

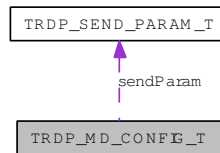
- [trdp\\_types.h](#)

## 4.14 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 timeout in us.*
- UINT32 [confirmTimeout](#)  
*Default timeout in us.*
- UINT32 [udpPort](#)  
*Port to be used for UDP MD communication.*
- UINT32 [tcpPort](#)  
*Port to be used for TCP MD communication.*

### 4.14.1 Detailed Description

Default MD configuration.

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

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

## 4.15 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*
- [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 noOfReplies](#)  
*actual number of replies for the request*

- const void \* [pUserRef](#)  
*User reference given with the local call.*
- [TRDP\\_ERR\\_T](#) `resultCode`  
*error code*

### 4.15.1 Detailed Description

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

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

### 4.15.2 Field Documentation

#### 4.15.2.1 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](#)

## 4.16 TRDP\_MD\_STATISTICS Struct Reference

Message data statistics.

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

### Data Fields

- UINT32 [headerInPackets](#)  
*Incoming packets.*
- UINT32 [headerInCRCErr](#)  
*Incoming CRC errors.*
- UINT32 [headerInProtoErr](#)  
*Incoming protocol errors.*
- UINT32 [headerInTimeOuts](#)  
*Incoming timing errors.*
- UINT32 [headerInFrameErr](#)  
*Incoming timing errors.*
- UINT32 [headerOutPackets](#)  
*Outgoing packets.*
- UINT32 [headerAckErr](#)  
*Missing acknowledge.*

### 4.16.1 Detailed Description

Message data statistics.

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

- [trdp\\_private.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

Structure describing memory (and its pre-fragmentation).

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

### Data Fields

- `UINT8 * p`  
*pointer to static or allocated memory*
- `UINT32 size`  
*size of static or allocated memory*
- `UINT32 prealloc [TRDP_MEM_BLK_524288+1]`  
*memory block structure*

### 4.18.1 Detailed Description

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 allocBlockSize](#) [TRDP\_MEM\_BLK\_524288+1]  
*allocated memory blocks*
- [UINT32 usedBlockSize](#) [TRDP\_MEM\_BLK\_524288+1]  
*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:

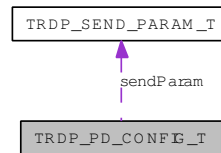
- [trdp\\_types.h](#)

## 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.*
- `UINT32` [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*
- [BOOL subs](#)  
*substitution*
- [UINT16 offsetAddr](#)  
*offset address for ladder architecture*
- [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 Struct Reference

Process data statistics.

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

### Data Fields

- UINT32 [headerInPackets](#)  
*Incoming packets.*
- UINT32 [headerInCRCErr](#)  
*Incoming CRC errors.*
- UINT32 [headerInProtoErr](#)  
*Incoming protocol errors.*
- UINT32 [headerInTimeOuts](#)  
*Incoming timing errors.*
- UINT32 [headerInFrameErr](#)  
*Incoming timing errors.*
- UINT32 [headerOutPackets](#)  
*Outgoing packets.*

### 4.22.1 Detailed Description

Process data statistics.

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

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

## 4.23 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*

### 4.23.1 Detailed Description

Structure containing all general PD statistics information.

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

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



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

Types to read out the XML configuration.

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

### Data Fields

- TRDP\_LABEL\_T [hostName](#)  
*Host name.*
- TRDP\_LABEL\_T [leaderName](#)  
*Leader name dependant on redundanca concept.*
- TRDP\_IP\_ADDR [hostIp](#)  
*Host IP address.*
- TRDP\_IP\_ADDR [leaderIp](#)  
*Leader IP address dependant on redundancy concept.*
- UINT32 [cycleTime](#)  
*TRDP main process cycle time in usec.*
- UINT32 [priority](#)  
*TRDP main process priority.*
- TRDP\_OPTION\_T [options](#)  
*TRDP default options.*

### 4.24.1 Detailed Description

Types to read out the XML configuration.

Configuration of TRDP main process.

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

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

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

properties information structure

```
#include <tau_tci.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*

### 4.25.1 Detailed Description

properties information structure

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

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

## 4.26 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.*

### 4.26.1 Detailed Description

Table containing particular PD publishing information.

### 4.26.2 Field Documentation

#### 4.26.2.1 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](#)

## 4.27 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.*

### 4.27.1 Detailed Description

A table containing PD redundant group information.

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

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

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

Quality/type of service and time to live.

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

### 4.28.1 Detailed Description

Quality/type of service and time to live.

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

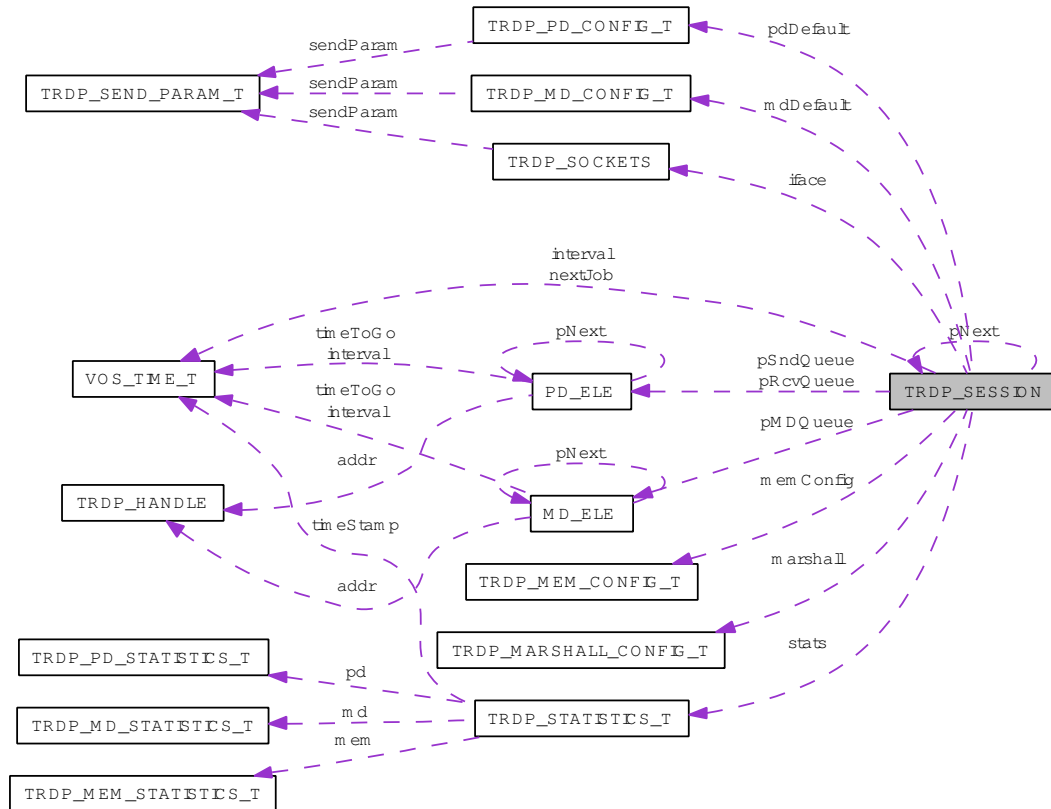
- [trdp\\_types.h](#)

## 4.29 TRDP\_SESSION Struct Reference

Session/application variables store.

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

Collaboration diagram for TRDP\_SESSION:



### Data Fields

- struct [TRDP\\_SESSION](#) \* [pNext](#)  
*Pointer to next session.*
- [VOS\\_MUTEX\\_T](#) [mutex](#)  
*protect this session*
- [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 interval](#)  
*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*
- [MD\\_ELE\\_T \\* pMDQueue](#)  
*pointer to first element of MD 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:

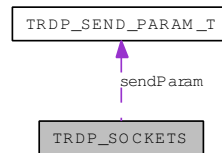
- [trdp\\_private.h](#)

## 4.30 TRDP\_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 SOCK\\_TYPE\\_T type](#)  
*Usage of this socket.*
- [UINT16 usage](#)  
*No.*

### 4.30.1 Detailed Description

Socket item.

### 4.30.2 Field Documentation

#### 4.30.2.1 UINT16 TRDP\_SOCKETS::usage

No.

of current users of this socket

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

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

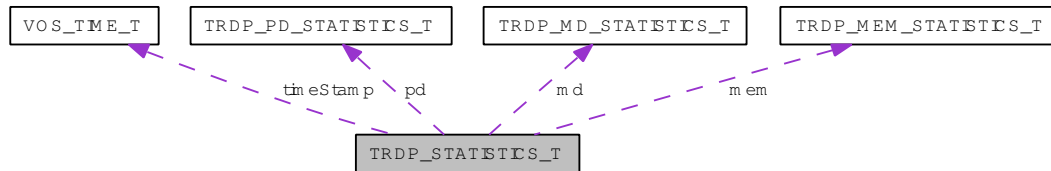


## 4.31 TRDP\_STATISTICS\_T Struct Reference

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.*
- **TRDP\_TIME\_T timeStamp**  
*actual time stamp*
- **UINT32 upTime**  
*time in sec since last initialisation*
- **UINT32 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*
- **TRDP\_MEM\_STATISTICS\_T mem**  
*memory statistics*

- [TRDP\\_PD\\_STATISTICS\\_T](#) `pd`  
*pd statistics*
- [TRDP\\_MD\\_STATISTICS\\_T](#) `md`  
*md statistics*

#### 4.31.1 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](#)

## 4.32 TRDP\_SUBS\_STATISTICS\_T Struct Reference

Table containing particular PD subscription information.

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

### Data Fields

- [UINT32 comId](#)  
*Subscribed ComId.*
- [TRDP\\_IP\\_ADDR\\_T joinedAddr](#)  
*Joined IP address.*
- [TRDP\\_IP\\_ADDR\\_T filterAddr](#)  
*Filter IP address, i.e IP address of the sender for this subscription, 0.0.0.0 in case all senders.*
- [UINT32 callBack](#)  
*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.*

### 4.32.1 Detailed Description

Table containing particular PD subscription information.

### 4.32.2 Field Documentation

#### 4.32.2.1 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.

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

Time-out value in us.

0 = No time-out supervision

#### 4.32.2.3 TRDP\_TO\_BEHAVIOR\_T TRDP\_SUBS\_STATISTICS\_T::toBehav

Behaviour at time-out.

Set data to zero / keep last value

#### 4.32.2.4 UINT32 TRDP\_SUBS\_STATISTICS\_T::numRecv

Number of packets received for this subscription.

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

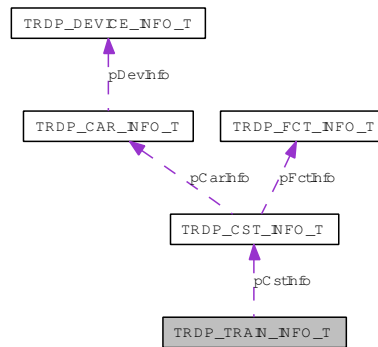
- [trdp\\_types.h](#)

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

train information structure.

```
#include <tau_tci.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.*

### 4.33.1 Detailed Description

train information structure.

### 4.33.2 Field Documentation

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

Train operator e.g.

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

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

IEC (i.e.

TCN) topography counter

#### 4.33.2.3 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\\_tci.h](#)

## 4.34 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*

### 4.34.1 Detailed Description

Common socket options.

### 4.34.2 Field Documentation

#### 4.34.2.1 `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`

## 4.35 VOS\_TIME\_T Struct Reference

Timer value compatible with timeval / select.

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

### Data Fields

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

### 4.35.1 Detailed Description

Timer value compatible with timeval / select.

Relative or absolute date, depending on usage

### 4.35.2 Field Documentation

#### 4.35.2.1 UINT32 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

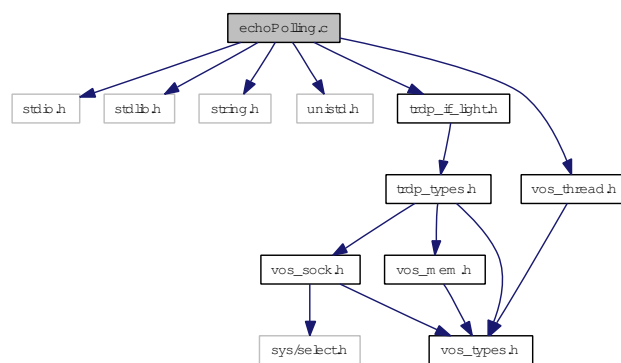
# File Documentation

### 5.1 echoPolling.c File Reference

Demo echoing application for TRDP.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "trdp_if_light.h"
#include "vos_thread.h"
```

Include dependency graph for echoPolling.c:



### Functions

- void `dbgOut` (void \*pRefCon, [TRDP\\_LOG\\_T](#) category, const CHAR8 \*pTime, const CHAR8 \*pFile, UINT16 LineNumber, const CHAR8 \*pMsgStr)  
*callback routine for TRDP logging/error output*
- int `main` (int argc, char \*\*argv)  
*main entry*

### 5.1.1 Detailed Description

Demo echoing application for TRDP.

Receive and send process data, single threaded polling, static memory

**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**

[echoPolling.c](#) 2 2012-06-04 11:25:16Z 97025

### 5.1.2 Function Documentation

**5.1.2.1** void dbgOut (void \* *pRefCon*, TRDP\_LOG\_T *category*, const CHAR8 \* *pTime*, const CHAR8 \* *pFile*, UINT16 *LineNumber*, const CHAR8 \* *pMsgStr*)

callback routine for TRDP logging/error output

**Parameters:**

- ← *pRefCon* user supplied context pointer
- ← *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
- ← *pMsgStr* pointer to NULL-terminated string

**Return values:**

*none*

**5.1.2.2** int main (int *argc*, char \*\* *argv*)

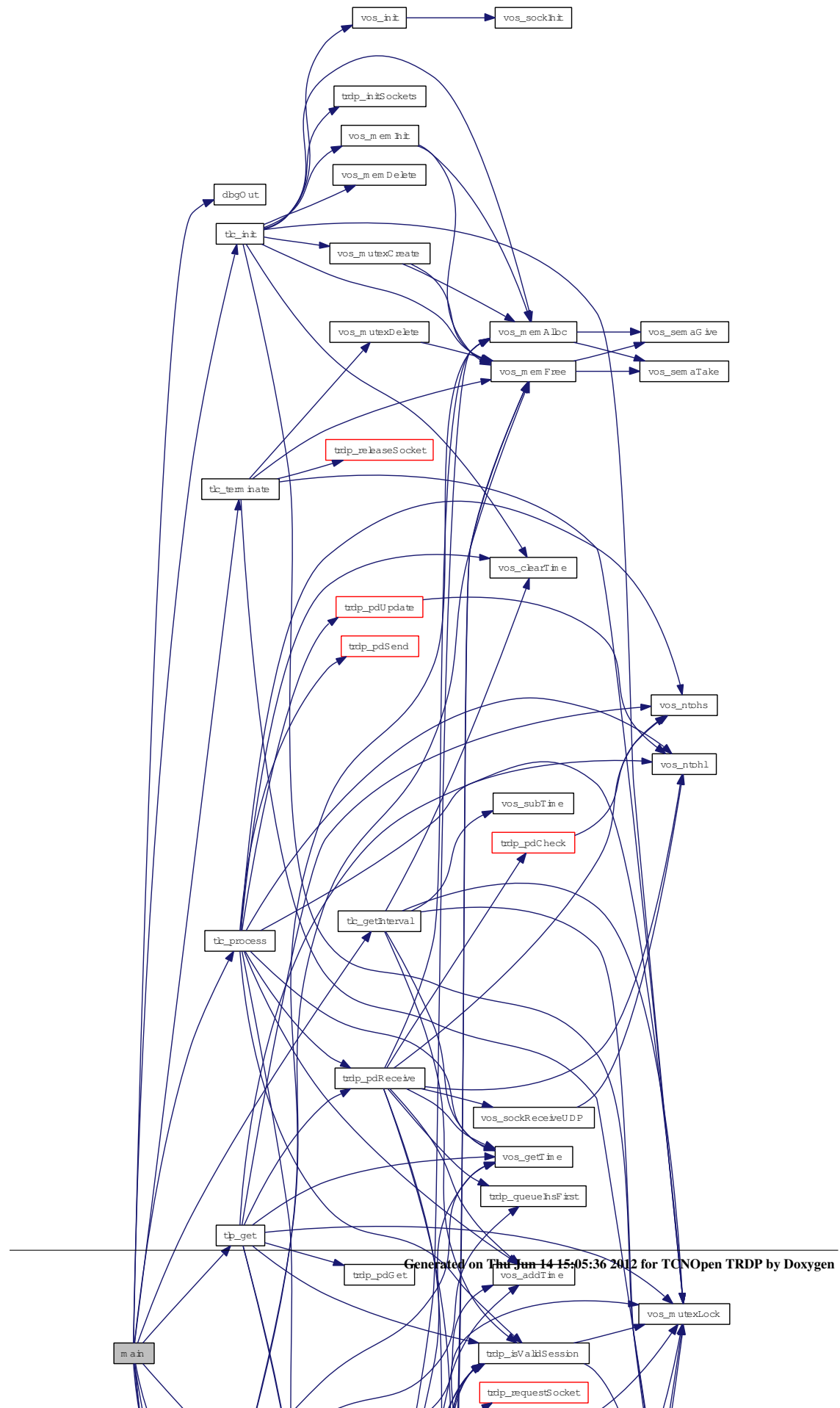
main entry

**Return values:**

0 no error

*I* some error

Here is the call graph for this function:

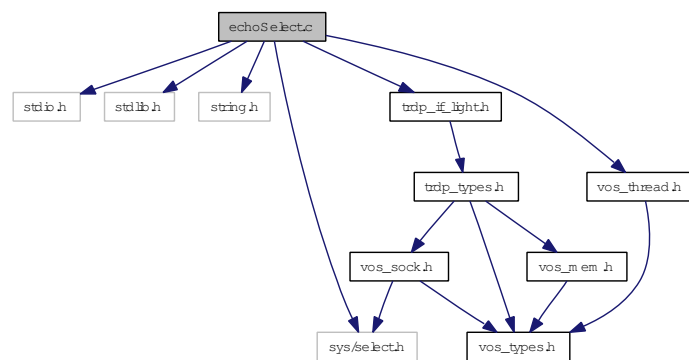


## 5.2 echoSelect.c File Reference

Demo echoing application for TRDP.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/select.h>
#include "trdp_if_light.h"
#include "vos_thread.h"
```

Include dependency graph for echoSelect.c:



### Functions

- void [dbgOut](#) (void \*pRefCon, [TRDP\\_LOG\\_T](#) category, const CHAR8 \*pTime, const CHAR8 \*pFile, UINT16 LineNumber, const CHAR8 \*pMsgStr)  
*callback routine for TRDP logging/error output*
- void [myPDcallback](#) (void \*pRefCon, const [TRDP\\_PD\\_INFO\\_T](#) \*pMsg, UINT8 \*pData, UINT32 dataSize)  
*callback routine for receiving TRDP traffic*
- int [main](#) (int argc, char \*\*argv)  
*main entry*

### 5.2.1 Detailed Description

Demo echoing application for TRDP.

Receive and send process data, single threaded using select() and heap memory

#### 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**

[echoSelect.c](#) 2 2012-06-04 11:25:16Z 97025

## 5.2.2 Function Documentation

### 5.2.2.1 void dbgOut (void \* *pRefCon*, TRDP\_LOG\_T *category*, const CHAR8 \* *pTime*, const CHAR8 \* *pFile*, UINT16 *LineNumber*, const CHAR8 \* *pMsgStr*)

callback routine for TRDP logging/error output

**Parameters:**

- ← *pRefCon* user supplied context pointer
- ← *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
- ← *pMsgStr* pointer to NULL-terminated string

**Return values:**

*none*

### 5.2.2.2 int main (int *argc*, char \*\* *argv*)

main entry

**Return values:**

- 0* no error
- 1* some error



### 5.2.2.3 void myPDcallback (void \* *pRefCon*, const TRDP\_PD\_INFO\_T \* *pMsg*, UINT8 \* *pData*, UINT32 *dataSize*)

callback routine for receiving TRDP traffic

#### Parameters:

- ← *pRefCon* user supplied context pointer
- ← *pMsg* pointer to header/packet infos
- ← *pData* pointer to data block
- ← *dataSize* pointer to data size

#### Return values:

*none*

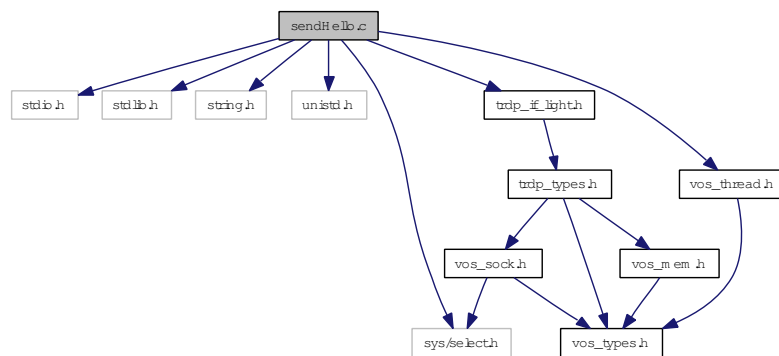


## 5.3 sendHello.c File Reference

Demo application for TRDP.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/select.h>
#include "trdp_if_light.h"
#include "vos_thread.h"
```

Include dependency graph for sendHello.c:



### Functions

- `int main (int argc, char *argv[ ])`  
*main entry*

### 5.3.1 Detailed Description

Demo application for TRDP.

#### Note:

Project: TCNOpen TRDP prototype stack

#### Author:

Bernd Loehr and Florian Weispfenning, 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

[sendHello.c](#) 2 2012-06-04 11:25:16Z 97025

## 5.3.2 Function Documentation

### 5.3.2.1 `int main (int argc, char * argv [ ])`

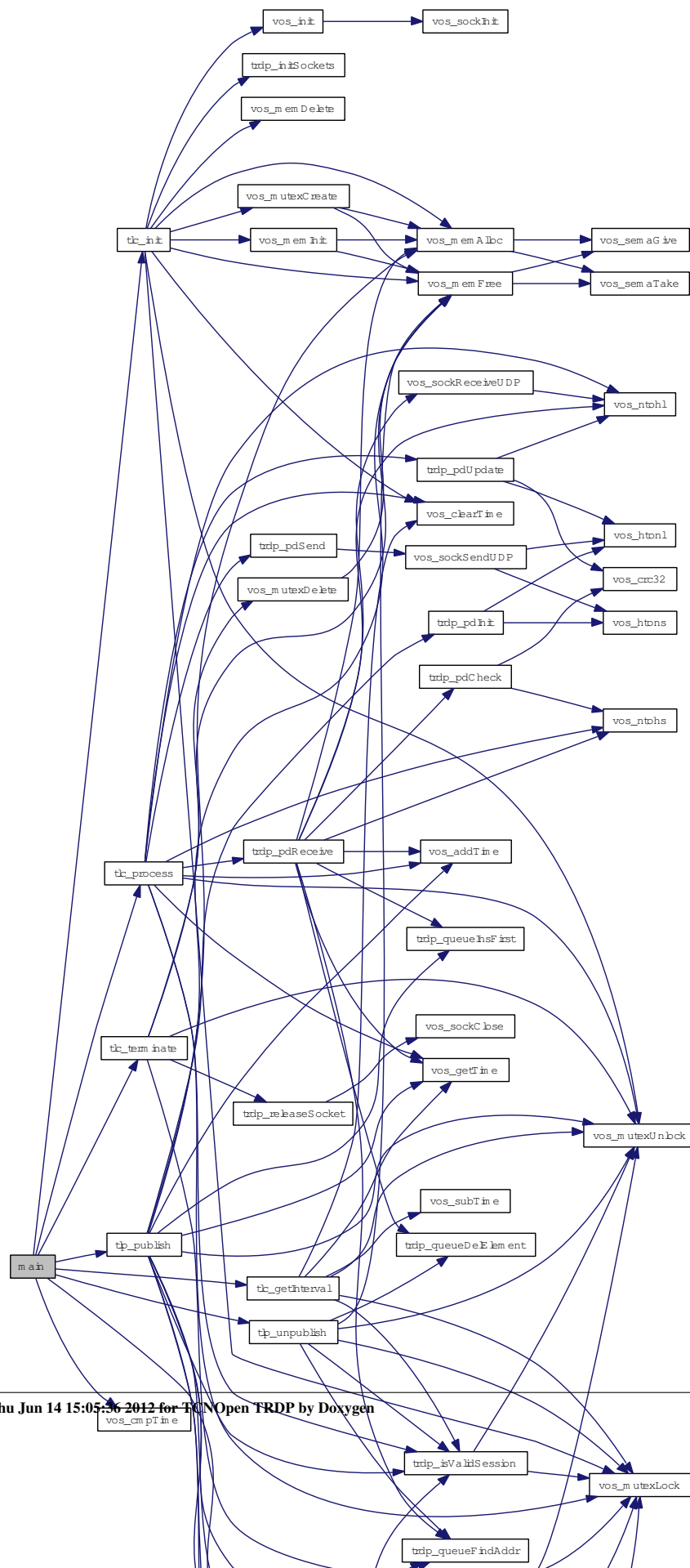
main entry

#### **Return values:**

*0* no error

*1* some error

Here is the call graph for this function:



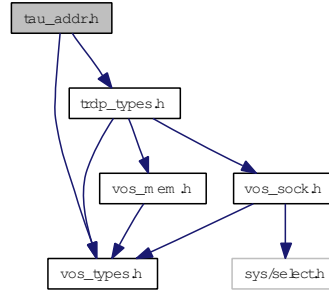
## 5.4 tau\_addr.h File Reference

TRDP utility interface definitions.

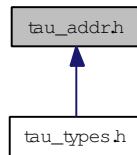
```
#include "vos_types.h"
```

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

Include dependency graph for tau\_addr.h:



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



## 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.*

### 5.4.1 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](#) 8 2012-06-06 16:28:19Z 97025

### 5.4.2 Function Documentation

#### 5.4.2.1 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

#### 5.4.2.2 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

#### 5.4.2.3 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

#### 5.4.2.4 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

#### 5.4.2.5 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

#### 5.4.2.6 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

#### 5.4.2.7 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



#### 5.4.2.8 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

#### 5.4.2.9 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

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

Function to get the own IP address.

##### Return values:

- own* IP address

#### 5.4.2.11 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

#### 5.4.2.12 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

#### 5.4.2.13 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

#### 5.4.2.14 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

#### 5.4.2.15 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

#### 5.4.2.16 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

#### 5.4.2.17 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

#### 5.4.2.18 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

#### 5.4.2.19 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

**5.4.2.20 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:**

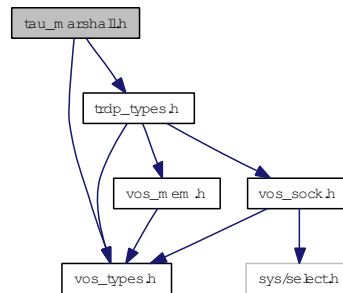
- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

## 5.5 tau\_marshall.h File Reference

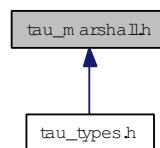
TRDP utility interface definitions.

```
#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:



## Typedefs

- typedef [TRDP\\_ERR\\_T](#) [tau\\_marshall](#) (void \*pRefCon, UINT32 comId, const UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize)  
*marshall function.*
- typedef [TRDP\\_ERR\\_T](#) [tau\\_marshallDs](#) (void \*pRefCon, UINT32 datasetId, const UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize)  
*Marshall data set function.*
- typedef [TRDP\\_ERR\\_T](#) [tau\\_unmarshall](#) (void \*pRefCon, UINT32 comId, const UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize)  
*unmarshall function.*
- typedef [TRDP\\_ERR\\_T](#) [tau\\_unmarshallDs](#) (void \*pRefCon, UINT32 datasetId, const UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize)  
*unmarshall data set function.*
- typedef [TRDP\\_ERR\\_T](#) [tau\\_calcDatasetSize](#) (void \*pRefCon, UINT32 datasetId, UINT8 \*pSrc, UINT32 \*pSize)  
*Calculate data set size.*

## Functions

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tau\\_initMarshall](#) (void \*\*ppRefCon, UINT32 numDataSet, [TRDP\\_DATASET\\_T](#) \*pDataset)

*Types for marshalling / unmarshalling.*

### 5.5.1 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](#) 6 2012-06-06 13:47:09Z 97025

### 5.5.2 Typedef Documentation

#### 5.5.2.1 typedef [TRDP\\_ERR\\_T](#) [tau\\_calcDataSetSize](#)(void \*pRefCon, UINT32 dataSetId, UINT8 \*pSrc, UINT32 \*pSize)

Calculate data set size.

#### Parameters:

- ← *pRefCon* Pointer to user context
- ← *dataSetId* Dataset id to identify the structure out of a configuration
- ← *pSrc* Pointer to received original message
- *pSize* Pointer to the size of the data set

#### Return values:

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

### 5.5.2.2 typedef TRDP\_ERR\_T tau\_marshall(void \*pRefCon, UINT32 comId, const UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize)

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

#### 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

### 5.5.2.3 typedef TRDP\_ERR\_T tau\_marshallDs(void \*pRefCon, UINT32 datasetId, const UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize)

Marshall data set function.

#### Parameters:

- ← *pRefCon* pointer to user context
- ← *datasetId* Dataset 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

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer to small
- TRDP\_INIT\_ERR* marshalling not initialised
- TRDP\_PARAM\_ERR* data set id not existing

### 5.5.2.4 typedef TRDP\_ERR\_T tau\_unmarshall(void \*pRefCon, UINT32 comId, const UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize)

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

**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

### 5.5.2.5 typedef TRDP\_ERR\_T tau\_unmarshallDs(void \*pRefCon, UINT32 datasetId, const UINT8 \*pSrc, UINT8 \*pDest, UINT32 \*pDestSize)

unmarshall data set function.

**Parameters:**

- ← *pRefCon* pointer to user context
- ← *datasetId* Dataset 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

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provided buffer to small
- TRDP\_INIT\_ERR* marshalling not initialised
- TRDP\_PARAM\_ERR* data set id not existing

## 5.5.3 Function Documentation

### 5.5.3.1 EXT\_DECL TRDP\_ERR\_T tau\_initMarshall (void \*\* ppRefCon, 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
- ← *numDataSet* Number of datasets found in the configuration
- ← *pDataset* Pointer to an array of a 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

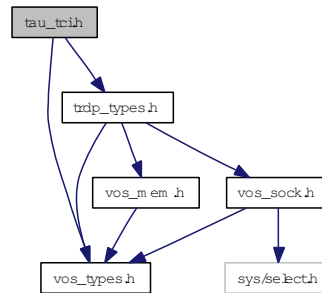
## 5.6 tau\_tci.h File Reference

TRDP utility interface definitions.

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

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

Include dependency graph for tau\_tci.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 \*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.*

### 5.6.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- train configuration 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](#) 8 2012-06-06 16:28:19Z 97025

### 5.6.2 Enumeration Type Documentation

#### 5.6.2.1 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.

### 5.6.2.2 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

## 5.6.3 Function Documentation

### 5.6.3.1 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

### 5.6.3.2 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

**5.6.3.3 \*\*\*\*\***

**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

**5.6.3.4 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

**5.6.3.5 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

#### 5.6.3.6 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

#### 5.6.3.7 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

### 5.6.3.8 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

### 5.6.3.9 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

### 5.6.3.10 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

### 5.6.3.11 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

### 5.6.3.12 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

### 5.6.3.13 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:**

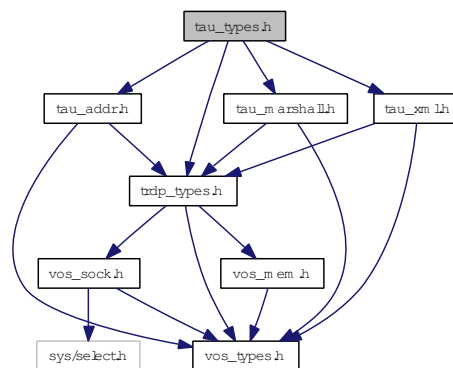
- TRDP\_NO\_ERR* no error
- TRDP\_PARAM\_ERR* Parameter error

## 5.7 tau\_types.h File Reference

TRDP utility interface definitions.

```
#include "trdp_types.h"
#include "tau_addr.h"
#include "tau_marshall.h"
#include "tau_xml.h"
```

Include dependency graph for tau\_types.h:



### 5.7.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- marshall/unmarshall
- xml configuration interpreter
- 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\\_types.h](#) 2 2012-06-04 11:25:16Z 97025

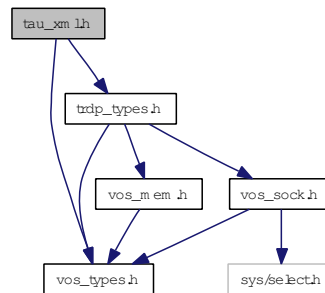
## 5.8 tau\_xml.h File Reference

TRDP utility interface definitions.

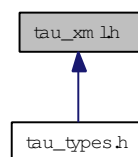
```
#include "vos_types.h"
```

```
#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\\_PROCESS\\_CONFIG\\_T](#)  
*Types to read out the XML configuration.*
- struct [TRDP\\_DBG\\_CONFIG\\_T](#)  
*Control for debug output device/file on application level.*

### 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\\_readXmlConfig](#) (const CHAR8 \*pFileName, [TRDP\\_PROCESS\\_CONFIG\\_T](#) \*pProcessConfig, [TRDP\\_MEM\\_CONFIG\\_T](#) \*pMemConfig, [TRDP\\_PD\\_CONFIG\\_T](#) \*pPdConfig, [TRDP\\_MD\\_CONFIG\\_T](#) \*pMdConfig, UINT32 \*pNumExchgPar, [TRDP\\_EXCHG\\_PAR\\_T](#) \*\*ppExchgPar, UINT32 \*pNumComPar, [TRDP\\_COM\\_PAR\\_T](#) \*\*ppComPar, [TRDP\\_DBG\\_CONFIG\\_T](#) \*pDbgPar)

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

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tau\\_readXmlDatasetConfig](#) (const CHAR8 \*pFileName, UINT32 \*pNumDataset, [TRDP\\_DATASET\\_T](#) \*\*ppDataset)

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

### 5.8.1 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](#) 5587 2012-05-30 09:24:22Z bloehr

### 5.8.2 Enumeration Type Documentation

#### 5.8.2.1 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).

### 5.8.3 Function Documentation

**5.8.3.1** `EXT_DECL TRDP_ERR_T tau_readXmlConfig (const CHAR8 * pFileName,  
TRDP_PROCESS_CONFIG_T * pProcessConfig, TRDP_MEM_CONFIG_T *  
pMemConfig, TRDP_PD_CONFIG_T * pPdConfig, TRDP_MD_CONFIG_T *  
pMdConfig, UINT32 * pNumExchgPar, TRDP_EXCHG_PAR_T ** ppExchgPar, UINT32  
* pNumComPar, TRDP_COM_PAR_T ** ppComPar, TRDP_DBG_CONFIG_T *  
pDbgPar)`

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

#### Parameters:

← *pFileName* Path and filename of the xml configuration file  
→ *pProcessConfig* TRDP main process configuration  
→ *pMemConfig* Memory configuration  
→ *pPdConfig* PD default configuration  
→ *pMdConfig* MD default configuration  
→ *pNumExchgPar* Number of configured telegrams  
→ *ppExchgPar* Pointer to array of telegram configurations  
→ *pNumComPar* Number of configured com parameters  
→ *ppComPar* Pointer to array of com parameters  
→ *pDbgPar* Debug printout options for application use

#### Return values:

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

**5.8.3.2** `EXT_DECL TRDP_ERR_T tau_readXmlDatasetConfig (const CHAR8 * pFileName,  
UINT32 * pNumDataset, TRDP_DATASET_T ** ppDataset)`

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

#### Parameters:

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

→ *pNumDataset* Pointer to the number of datasets found in the configuration

→ *ppDataset* Pointer to an array of a structures of type [TRDP\\_DATASET\\_T](#)

**Return values:**

*TRDP\_NO\_ERR* no error

*TRDP\_MEM\_ERR* provided buffer to small

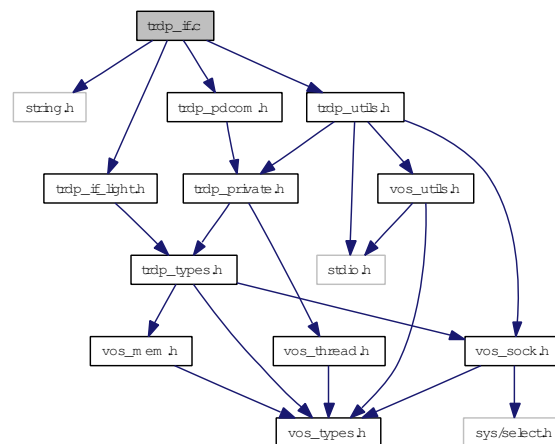
*TRDP\_PARAM\_ERR* File not existing

## 5.9 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 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** (TRDP\_APP\_SESSION\_T \*pAppHandle, TRDP\_IP\_ADDR\_T ownIpAddr, TRDP\_IP\_ADDR\_T leaderIpAddr, const TRDP\_PRINT\_DBG\_T pPrintDebugString, const TRDP\_MARSHALL\_CONFIG\_T \*pMarshall, const TRDP\_PD\_CONFIG\_T \*pPdDefault, const TRDP\_MD\_CONFIG\_T \*pMdDefault, const TRDP\_MEM\_CONFIG\_T \*pMemConfig, TRDP\_OPTION\_T option)  
*Initialize the TRDP stack.*
- **TRDP\_ERR\_T tlc\_terminate** (TRDP\_APP\_SESSION\_T appHandle)  
*Un-Initialize Clean up when app quits.*
- **TRDP\_ERR\_T tlc\_reinit** (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.*
- **const char \* tlc\_getVersion** (void)  
*Return a human readable version representation.*

- [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.*

- [void tlc\\_setTopoCount](#) ([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, [BOOL](#) subs, [UINT16](#) offsetAddress)

*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\\_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, [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\\_FLAGS\\_T](#) pktFlags, [TRDP\\_PD\\_INFO\\_T](#) \*pPdInfo, [UINT8](#) \*pData, [UINT32](#) \*pDataSize)

*Get the last valid PD message.*



## 5.9.1 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](#) 15 2012-06-14 12:52:06Z 97025

## 5.9.2 Function Documentation

### 5.9.2.1 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

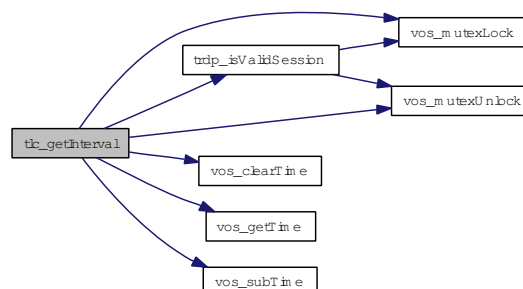
### Parameters:

- ← *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

Here is the call graph for this function:



### 5.9.2.2 `const char* tlc_getVersion (void)`

Return a human readable version representation.

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

#### Return values:

*const* string

### 5.9.2.3 `EXT_DECL TRDP_ERR_T tlc_init (TRDP_APP_SESSION_T * pAppHandle, TRDP_IP_ADDR_T ownIpAddr, TRDP_IP_ADDR_T leaderIpAddr, const TRDP_PRINT_DBG_T pPrintDebugString, const TRDP_MARSHALL_CONFIG_T * pMarshall, const TRDP_PD_CONFIG_T * pPdDefault, const TRDP_MD_CONFIG_T * pMdDefault, const TRDP_MEM_CONFIG_T * pMemConfig, TRDP_OPTION_T option)`

Initialize the TRDP stack.

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

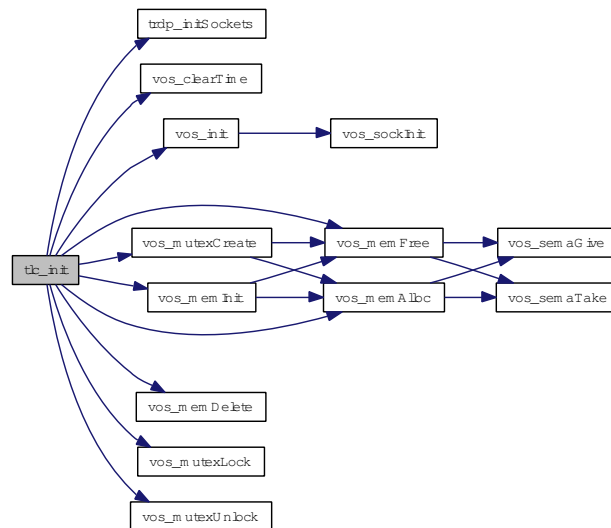
#### Parameters:

- *pAppHandle* A handle for further calls to the trdp stack
- ← *ownIpAddr* Own IP address, can be different for each process in multiprocessing systems
- ← *leaderIpAddr* IP address of redundancy leader
- ← *pPrintDebugString* Pointer to debug print function
- ← *pMarshall* Pointer to marshalling configuration
- ← *pPdDefault* Pointer to default PD configuration
- ← *pMdDefault* Pointer to default MD configuration
- ← *pMemConfig* Pointer to memory configuration
- ← *option* options for library behavior

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* memory allocation failed
- TRDP\_PARAM\_ERR* initialization error
- TRDP SOCK\_ERR* socket error

Here is the call graph for this function:



#### 5.9.2.4 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



### 5.9.2.6 void tlc\_setTopoCount (UINT32 topoCount)

Set new topocount for trainwide communication.

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

#### Parameters:

← *topoCount* New topoCount value

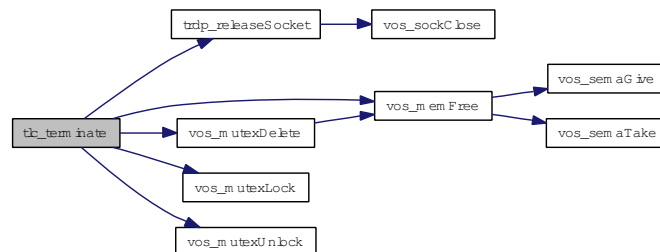
### 5.9.2.7 TRDP\_ERR\_T tlc\_terminate (TRDP\_APP\_SESSION\_T appHandle)

Un-Initialize Clean up when app quits.

Un-Initialize.

Mainly used for debugging/test runs

Here is the call graph for this function:



### 5.9.2.8 EXT\_DECL TRDP\_ERR\_T tlp\_get (TRDP\_APP\_SESSION\_T appHandle, TRDP\_SUB\_T subHandle, TRDP\_FLAGS\_T pktFlags, 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_init`  
 ← *subHandle* the handle returned by subscription  
 ← *pktFlags* OPTION: `TRDP_FLAGS_MARSHALL`  
 ↔ *pPdInfo* pointer to application's info buffer  
 ↔ *pData* pointer to application's data buffer  
 ↔ *pDataSize* in: size of buffer, out: size of data

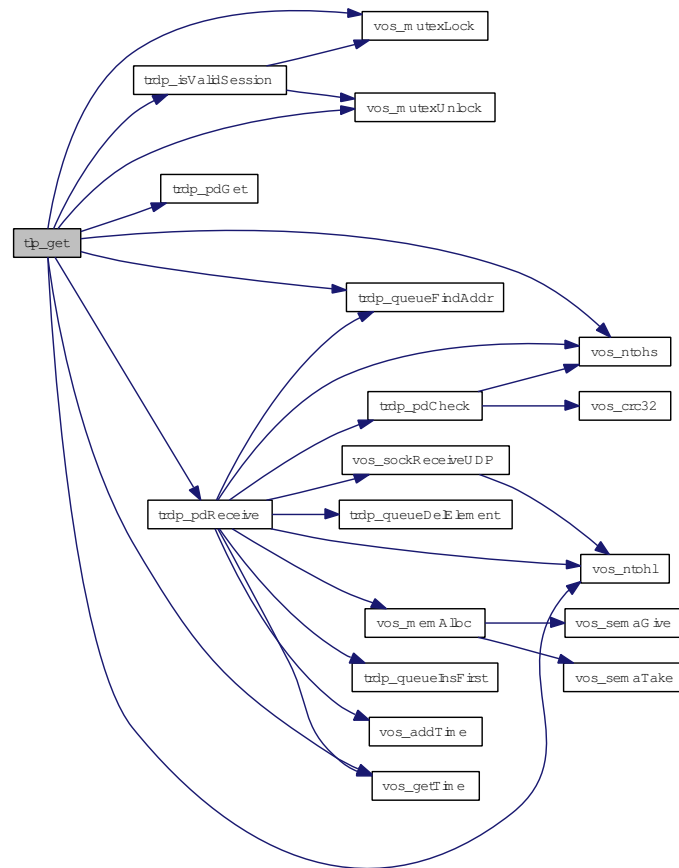
#### Return values:

**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

Here is the call graph for this function:



#### 5.9.2.9 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 returned for all ComID's with the given redId, 0 for all redId

↔ **pLeader** TRUE if we send (leader)

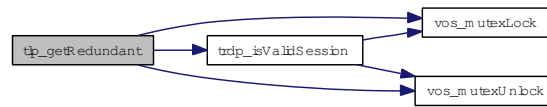
##### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_PARAM\_ERR** parameter error / redId not existing

**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



**5.9.2.10** EXT\_DECL TRDP\_ERR\_T trdp\_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*, BOOL *subs*, UINT16 *offsetAddress*)

Prepare for sending PD messages.

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

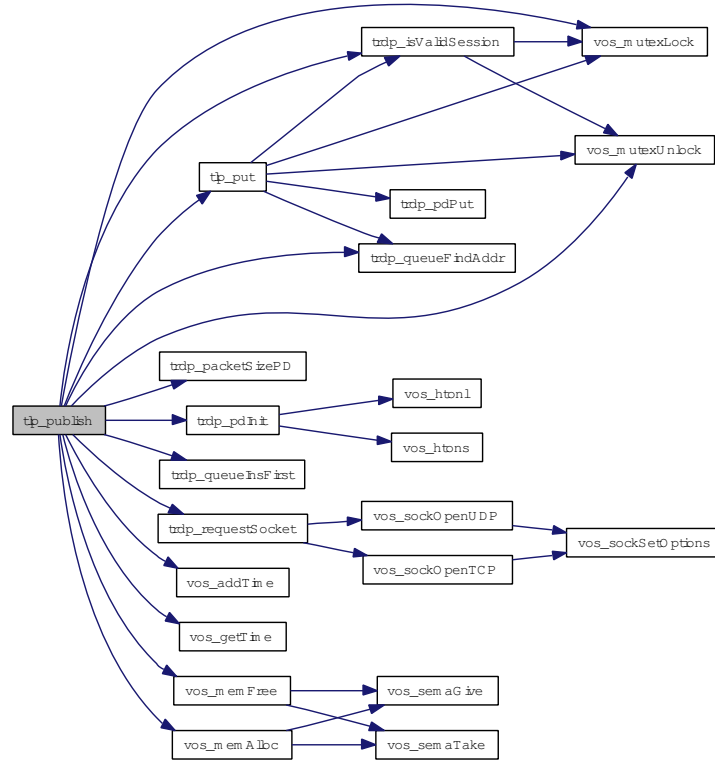
#### Parameters:

- ← *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* OPTIONS: 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
- ← *subs* substitution (Ladder)
- ← *offsetAddress* offset (Ladder)

#### 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:



#### 5.9.2.11 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:

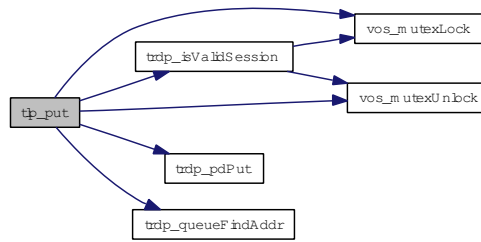
- ← *appHandle* the handle returned by tlc\_init
- ← *pubHandle* the handle returned by publish
- ↔ *pData* pointer to application's data buffer
- ↔ *dataSize* size of data

##### Return values:

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



Here is the call graph for this function:



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

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

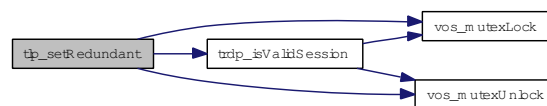
##### Parameters:

- ← *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:



#### 5.9.2.13 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*, 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:

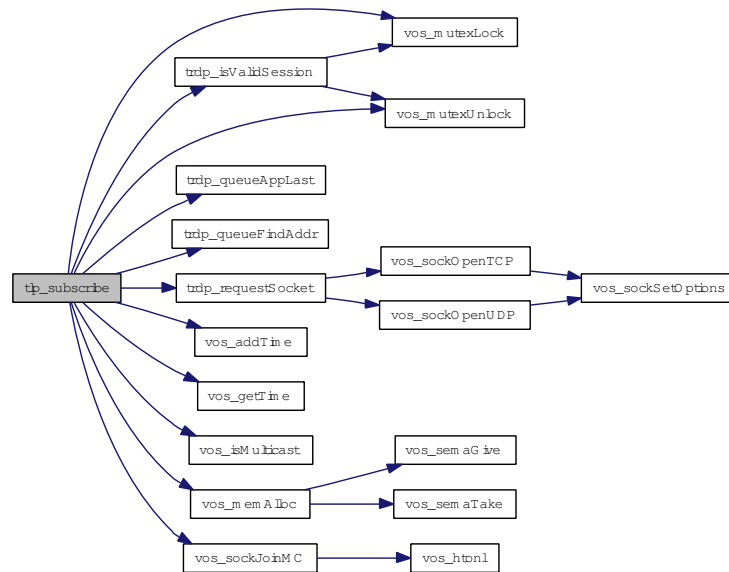
- ← *appHandle* the handle returned by tlc\_init

- **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
- ← **timeout** timeout ( $\geq 10$ ms) 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:



#### 5.9.2.14 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:

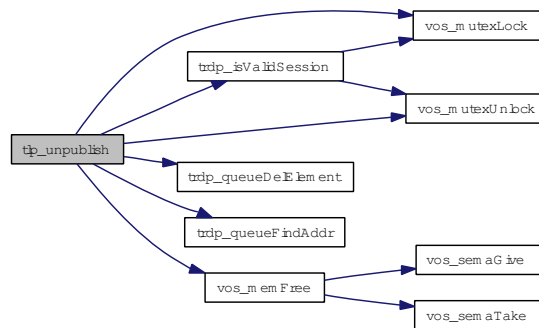
**TRDP\_NO\_ERR** no error

**TRDP\_PARAM\_ERR** parameter error

**TRDP\_NOPUB\_ERR** not published

**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



#### 5.9.2.15 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:

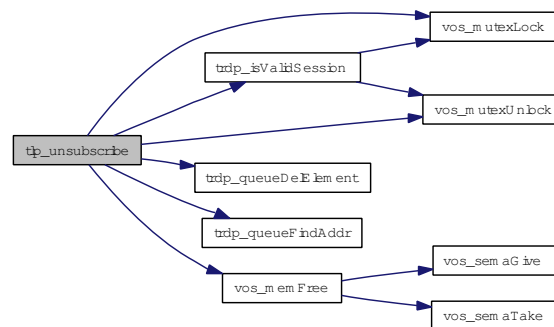
**TRDP\_NO\_ERR** no error

**TRDP\_PARAM\_ERR** parameter error

**TRDP\_SUB\_ERR** not subscribed

**TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



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

Check if the session handle is valid.

##### Parameters:

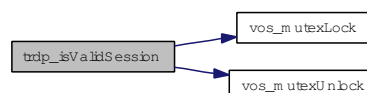
← *pSessionHandle* pointer to packet data (dataset)

##### Return values:

**TRUE** is valid

**FALSE** is invalid

Here is the call graph for this function:



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

Get the session queue head pointer.

##### Return values:

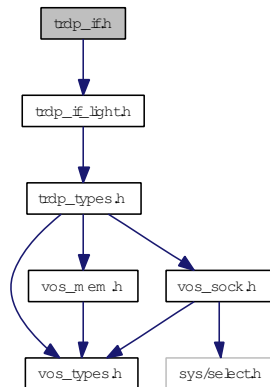
*&sSession*

## 5.10 trdp\_if.h File Reference

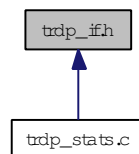
Typedefs for TRDP communication.

```
#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.*

### 5.10.1 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](#) 15 2012-06-14 12:52:06Z 97025

**5.10.2 Function Documentation****5.10.2.1 BOOL trdp\_isValidSession (TRDP\_APP\_SESSION\_T *pSessionHandle*)**

Check if the session handle is valid.

**Parameters:**

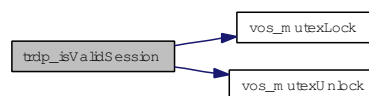
← *pSessionHandle* pointer to packet data (dataset)

**Return values:**

**TRUE** is valid

**FALSE** is invalid

Here is the call graph for this function:

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

Get the session queue head pointer.

**Return values:**

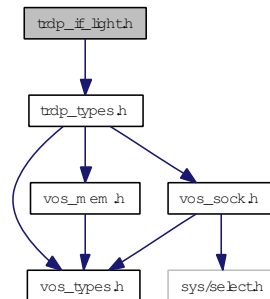
*&sSession*

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

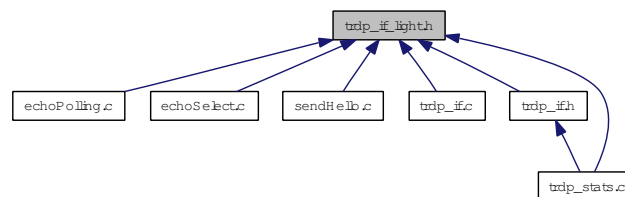
TRDP Light interface functions (API).

```
#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` (`TRDP_APP_SESSION_T *pAppHandle`, `TRDP_IP_ADDR_T ownIpAddr`, `TRDP_IP_ADDR_T leaderIpAddr`, `const TRDP_PRINT_DBG_T pPrintDebugString`, `const TRDP_MARSHALL_CONFIG_T *pMarshall`, `const TRDP_PD_CONFIG_T *pPdDefault`, `const TRDP_MD_CONFIG_T *pMdDefault`, `const TRDP_MEM_CONFIG_T *pMemConfig`, `TRDP_OPTION_T option`)  
*Initialize the TRDP stack.*
- EXT\_DECL `TRDP_ERR_T tlc_reinit` (`TRDP_APP_SESSION_T appHandle`)  
*Re-Initialize.*
- EXT\_DECL `TRDP_ERR_T tlc_terminate` (`TRDP_APP_SESSION_T appHandle`)  
*Un-Initialize.*
- EXT\_DECL `void tlc_setTopoCount` (`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, BOOL subs, UINT16 offsetAddress)  
*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 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 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, BOOL subs, UINT16 offsetAddr)  
*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, 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\\_FLAGS\\_T](#) pktFlags, [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](#) noOfRepliers, [UINT32](#) replyTimeout, [const TRDP\\_SEND\\_PARAM\\_T](#) \*pSendParam, [const UINT8](#) \*pData, [UINT32](#) dataSize, [const TRDP\\_URI\\_USER\\_T](#) srcURI, [const TRDP\\_URI\\_USER\\_T](#) destURI)

*Initiate sending MD request message.*

- 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](#) srcURI, [const TRDP\\_URI\\_USER\\_T](#) destURI)

*Initiate sending MD confirm message.*

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

*Cancel an open session.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlm\\_addListener](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [UINT32](#) \*pListenHandle, [const void](#) \*pUserRef, [UINT32](#) comId, [UINT32](#) topoCount, [TRDP\\_IP\\_ADDR\\_T](#) destIpAddr, [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, [UINT32](#) listenHandle)

*Remove Listener.*

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlm\\_reply](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [void](#) \*pUserRef, [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](#) srcURI, [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, [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 srcURI, const TRDP\_URI\_USER\_T destURI)

*Send a MD reply message.*

- EXT\_DECL [TRDP\\_ERR\\_T tlm\\_replyErr](#) (TRDP\_APP\_SESSION\_T appHandle, [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 srcURI, const TRDP\_URI\_USER\_T destURI)

*Send a MD reply message.*

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

*Return a human readable version representation.*

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

*Return statistics.*

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

*Return PD subscription statistics.*

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

*Return PD publish statistics.*

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

*Return MD listener statistics.*

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

*Return redundancy group statistics.*

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

*Return join statistics.*

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

*Reset statistics.*

### 5.11.1 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](#) 8 2012-06-06 16:28:19Z 97025

## 5.11.2 Function Documentation

### 5.11.2.1 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

### 5.11.2.2 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

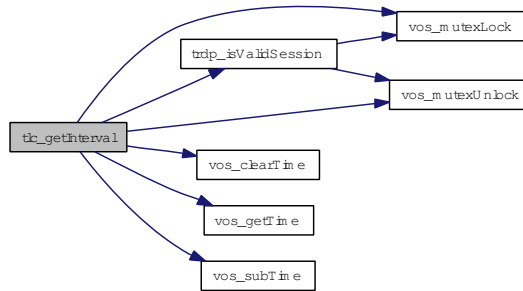
**Parameters:**

- ← *appHandle* The handle returned by tlc\_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

Here is the call graph for this function:



### 5.11.2.3 EXT\_DECL TRDP\_ERR\_T tlc\_getJoinStatistics (TRDP\_APP\_SESSION\_T *appHandle*, UINT16 \**pNumJoin*, UINT32 \*\**ppIpAddr*)

Return join statistics.

Memory for statistics information will be reserved by tlc layer and needs to be freed by the user.

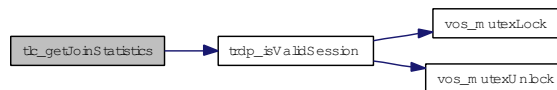
#### Parameters:

- ← *appHandle* the handle returned by tlc\_init
- *pNumJoin* Pointer to the number of joined IP Adresses
- *ppIpAddr* 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

Here is the call graph for this function:



### 5.11.2.4 EXT\_DECL TRDP\_ERR\_T tlc\_getListStatistics (TRDP\_APP\_SESSION\_T *appHandle*, UINT16 \**pNumList*, TRDP\_LIST\_STATISTICS\_T \*\**ppStatistics*)

Return MD listener statistics.

Memory for statistics information will be reserved by tlc layer and needs to be freed by the user.

#### Parameters:

- ← *appHandle* the handle returned by tlc\_init
- *pNumList* Pointer to the number of listeners

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

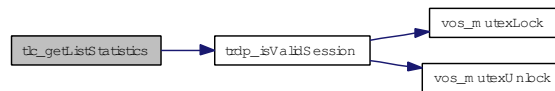
#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

Here is the call graph for this function:



#### 5.11.2.5 EXT\_DECL TRDP\_ERR\_T tlc\_getPubStatistics (TRDP\_APP\_SESSION\_T appHandle, UINT16 \* pNumPub, TRDP\_PUB\_STATISTICS\_T \*\* ppStatistics)

Return PD publish statistics.

Memory for statistics information will be reserved by tlc layer and needs to be freed by the user.

#### Parameters:

← *appHandle* the handle returned by tlc\_init

→ *pNumPub* Pointer to the number of publishers

→ *ppStatistics* Pointer to a list with the publish statistics information

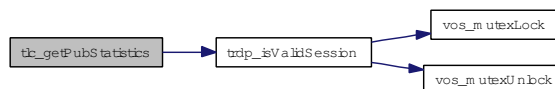
#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

Here is the call graph for this function:



#### 5.11.2.6 EXT\_DECL TRDP\_ERR\_T tlc\_getRedStatistics (TRDP\_APP\_SESSION\_T appHandle, UINT16 \* pNumRed, TRDP\_RED\_STATISTICS\_T \*\* ppStatistics)

Return redundancy group statistics.

Memory for statistics information will be reserved by tlc layer and needs to be freed by the user.

#### Parameters:

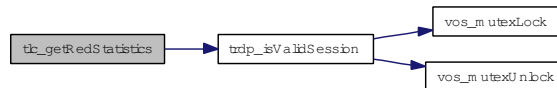
← *appHandle* the handle returned by tlc\_init

- *pNumRed* Pointer to the number of redundancy groups
- *ppStatistics* 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

Here is the call graph for this function:



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

Return statistics.

Memory for statistics information will be reserved by tlc layer and needs to be freed by the user.

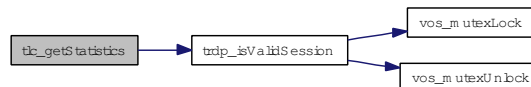
**Parameters:**

- ← *appHandle* the handle returned by tlc\_init
- *ppStatistics* 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:



#### 5.11.2.8 EXT\_DECL TRDP\_ERR\_T tlc\_getSubsStatistics (TRDP\_APP\_SESSION\_T *appHandle*, UINT16 \* *pNumSubs*, TRDP\_SUBS\_STATISTICS\_T \*\* *ppStatistics*)

Return PD subscription statistics.

Memory for statistics information will be reserved by tlc layer and needs to be freed by the user.

**Parameters:**

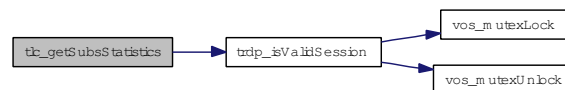
- ← *appHandle* the handle returned by tlc\_init

- *pNumSubs* Pointer to the number of subscriptions
- *ppStatistics* Pointer to a list with the subscription statistics information

**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:

**5.11.2.9 EXT\_DECL const CHAR8\* tlc\_getVersion (void)**

Return a human readable version representation.

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

**Return values:**

- const* string

**5.11.2.10 EXT\_DECL TRDP\_ERR\_T tlc\_init (TRDP\_APP\_SESSION\_T \* pAppHandle, TRDP\_IP\_ADDR\_T ownIpAddr, TRDP\_IP\_ADDR\_T leaderIpAddr, const TRDP\_PRINT\_DBG\_T pPrintDebugString, const TRDP\_MARSHALL\_CONFIG\_T \* pMarshall, const TRDP\_PD\_CONFIG\_T \* pPdDefault, const TRDP\_MD\_CONFIG\_T \* pMdDefault, const TRDP\_MEM\_CONFIG\_T \* pMemConfig, TRDP\_OPTION\_T option)**

Initialize the TRDP stack.

`tlc_init` 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 multiprocessing systems
- ← *leaderIpAddr* IP address of redundancy leader
- ← *pPrintDebugString* Pointer to debug print function
- ← *pMarshall* Pointer to marshall configuration
- ← *pPdDefault* Pointer to default PD configuration
- ← *pMdDefault* Pointer to default MD configuration
- ← *pMemConfig* Pointer to memory configuration
- ← *option* options for library behavior

**Return values:***TRDP\_NO\_ERR* no error*TRDP\_PARAM\_ERR* initialization error*TRDP SOCK\_ERR* socket error

tlc\_init 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 multiprocessing systems

← *leaderIpAddr* IP address of redundancy leader

← *pPrintDebugString* Pointer to debug print function

← *pMarshall* Pointer to marshall configuration

← *pPdDefault* Pointer to default PD configuration

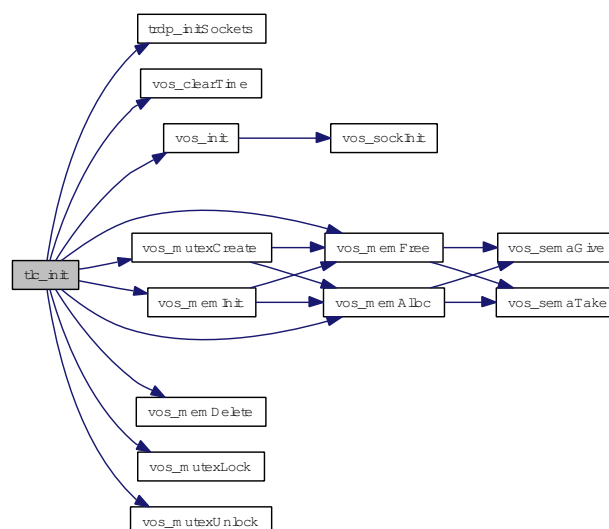
← *pMdDefault* Pointer to default MD configuration

← *pMemConfig* Pointer to memory configuration

← *option* options for library behavior

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

Here is the call graph for this function:





### 5.11.2.11 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)

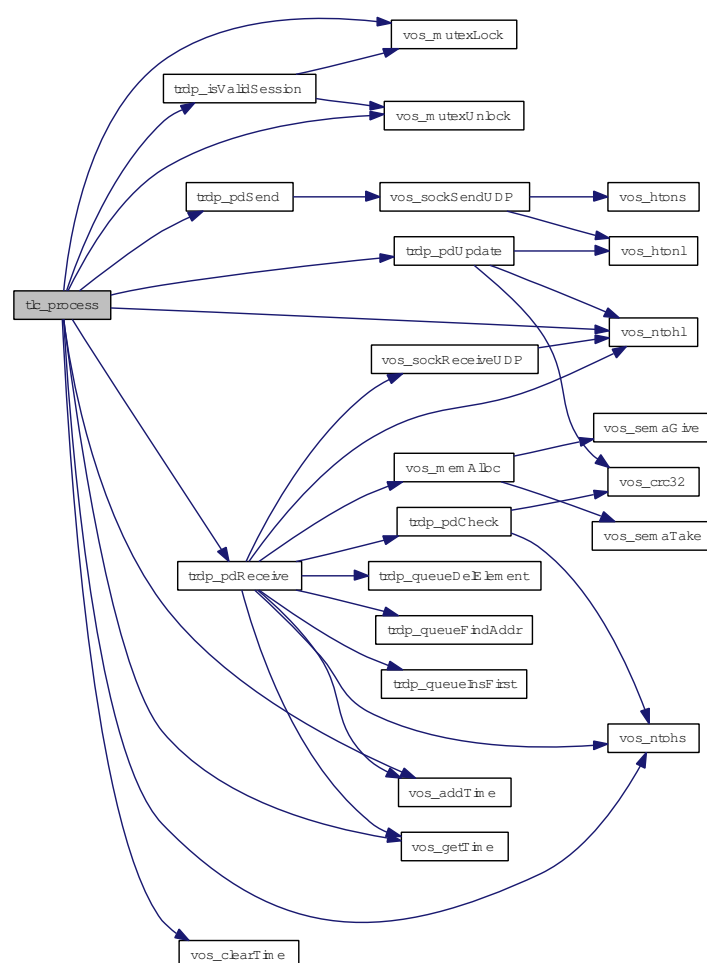
#### Parameters:

- ← *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

Here is the call graph for this function:



### 5.11.2.12 EXT\_DECL TRDP\_ERR\_T tlc\_reinit (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\_init

#### Return values:

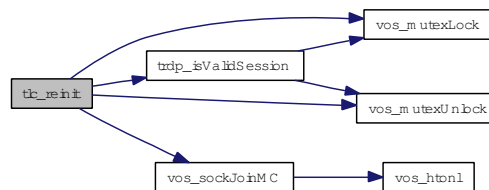
**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

Re-Initialize.

We re-join

Here is the call graph for this function:



### 5.11.2.13 EXT\_DECL TRDP\_ERR\_T tlc\_resetStatistics (TRDP\_APP\_SESSION\_T appHandle)

Reset statistics.

#### Parameters:

← *appHandle* the handle returned by tlc\_init

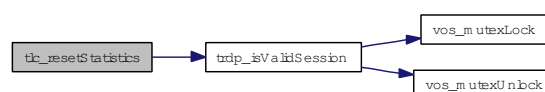
#### Return values:

**TRDP\_NO\_ERR** no error

**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** parameter error

Here is the call graph for this function:



**5.11.2.14 EXT\_DECL void tlc\_setTopoCount (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:**

← *topoCount* New topoCount value

**5.11.2.15 EXT\_DECL TRDP\_ERR\_T tlc\_terminate (TRDP\_APP\_SESSION\_T appHandle)**

Un-Initialize.

Clean up when app quits. Mainly used for debugging/test runs. No further calls to library allowed

**Parameters:**

← *appHandle* The handle returned by tlc\_init

**Return values:**

**TRDP\_NO\_ERR** no error

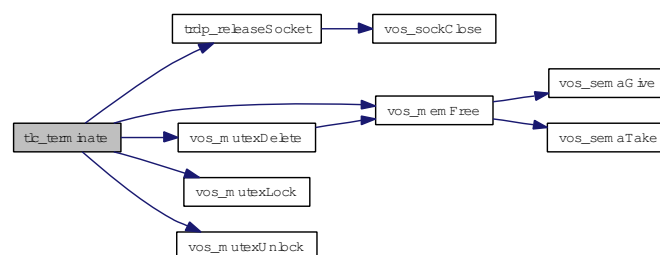
**TRDP\_NOINIT\_ERR** handle invalid

**TRDP\_PARAM\_ERR** handle NULL

Un-Initialize.

Mainly used for debugging/test runs

Here is the call graph for this function:

**5.11.2.16 EXT\_DECL TRDP\_ERR\_T tlm\_abortSession (TRDP\_APP\_SESSION\_T appHandle, TRDP\_UUID\_T \* pSessionId)**

Cancel an open session.

Abort an open session; any pending messages will be dropped; session id set to zero

**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

**5.11.2.17** `EXT_DECL TRDP_ERR_T tlm_addListener (TRDP_APP_SESSION_T appHandle, UINT32 * pListenHandle, const void * pUserRef, UINT32 comId, UINT32 topoCount, TRDP_IP_ADDR_T destIpAddr, 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
- ← *destIpAddr* destination IP address
- ← *pktFlags* optional marshalling
- ← *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

**5.11.2.18** `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 srcURI, 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\_CALLBACK
- ← *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
- ← *srcURI* 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

#### 5.11.2.19 EXT\_DECL TRDP\_ERR\_T tlm\_delListener (TRDP\_APP\_SESSION\_T *appHandle*, UINT32 *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

#### 5.11.2.20 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_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
- ← *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

**5.11.2.21** `EXT_DECL TRDP_ERR_T tlm_reply (TRDP_APP_SESSION_T appHandle, void * pUserRef, 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 srcURI, 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* optional marshalling
- ← *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

← *srcURI* 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

**5.11.2.22** **EXT\_DECL** **TRDP\_ERR\_T** **tlm\_replyErr** (**TRDP\_APP\_SESSION\_T** *appHandle*, **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** *srcURI*, **const** **TRDP\_URI\_USER\_T** *destURI*)

Send a MD 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

← *srcURI* 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

**5.11.2.23** `EXT_DECL TRDP_ERR_T tlm_replyQuery (TRDP_APP_SESSION_T appHandle, void * pUserRef, 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 srcURI, 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* optional marshalling
- ← *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
- ← *srcURI* 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

**5.11.2.24** `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 noOfRepliers, UINT32 replyTimeout, const TRDP_SEND_PARAM_T * pSendParam, const UINT8 * pData, UINT32 dataSize, const TRDP_URI_USER_T srcURI, const TRDP_URI_USER_T destURI)`

Initiate sending MD request message.

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\_MARSHALL, TRDP\_FLAGS\_CALLBACK
- ← *noOfRepliers* number of expected repliers, 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
- ← *srcURI* 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

**5.11.2.25** EXT\_DECL TRDP\_ERR\_T `tlp_get` (TRDP\_APP\_SESSION\_T *appHandle*,  
TRDP\_SUB\_T *subHandle*, TRDP\_FLAGS\_T *pktFlags*, 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
- ← *pktFlags* OPTION: TRDP\_FLAGS\_MARSHALL
- ↔ *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



### 5.11.2.26 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

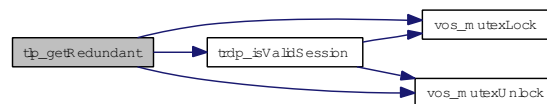
#### Parameters:

- ← *appHandle* the handle returned by tlc\_init
- ← *redId* will be returned 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

Here is the call graph for this function:



### 5.11.2.27 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*, BOOL *subs*, UINT16 *offsetAddress*)

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 ( $\geq 10$ ms) in usec  
 ← *redId* 0 - Non-redundant,  $> 0$  valid redundancy group  
 ← *pktFlags* OPTIONS: 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  
 ← *subs* substitution (Ladder)  
 ← *offsetAddress* offset (Ladder)

#### 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\_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 ( $\geq 10$ ms) in usec  
 ← *redId* 0 - Non-redundant,  $> 0$  valid redundancy group  
 ← *pktFlags* OPTIONS: 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  $\leq 1436$  without FCS  
 ← *subs* substitution (Ladder)  
 ← *offsetAddress* offset (Ladder)

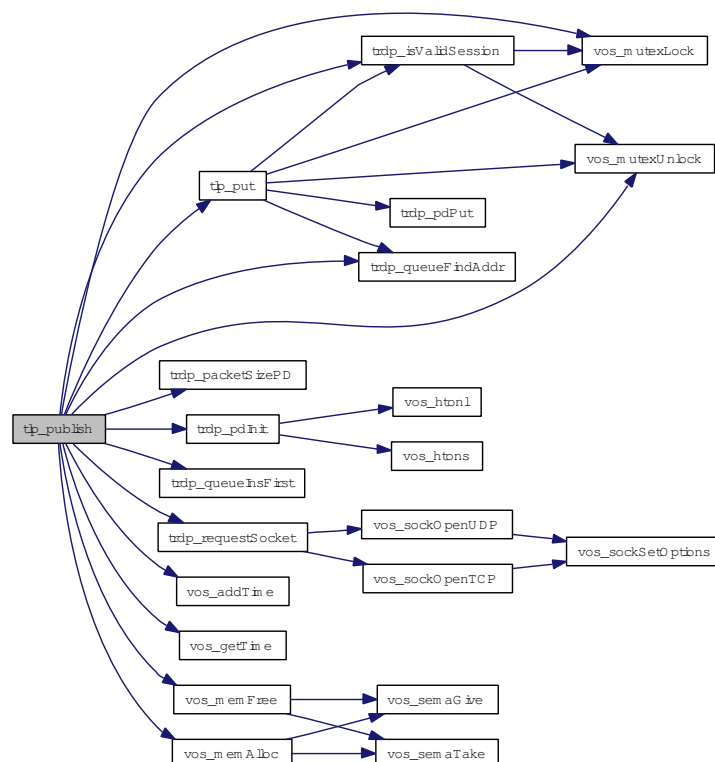
#### Return values:

*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:



#### 5.11.2.28 EXT\_DECL TRDP\_ERR\_T trdp\_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
- TRDP\_PUB\_ERR** not published

**TRDP\_NOINIT\_ERR** handle invalid

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

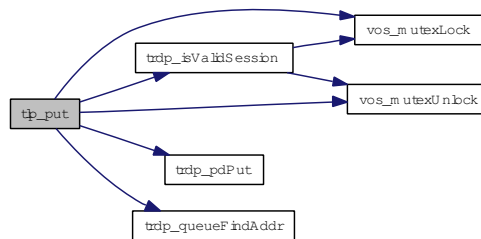
**Parameters:**

- ← **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
- TRDP\_NOPUB\_ERR** not published
- TRDP\_NOINIT\_ERR** handle invalid

Here is the call graph for this function:



**5.11.2.29** 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, BOOL subs, UINT16 offsetAddr)

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: 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
- ← **subs** substitution (Ladder)
- ← **offsetAddr** offset (Ladder)

**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

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

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

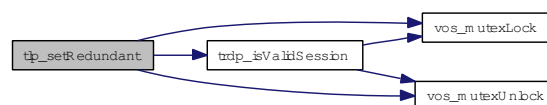
**Parameters:**

- ← **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:



**5.11.2.31 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, 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!

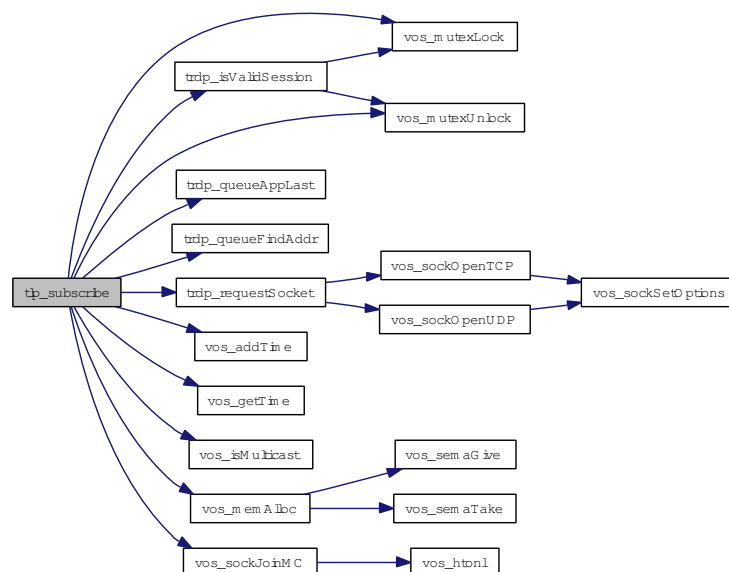
#### Parameters:

- ← **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
- ← **timeout** timeout ( $\geq 10$ ms) 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:





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

Stop sending PD messages.

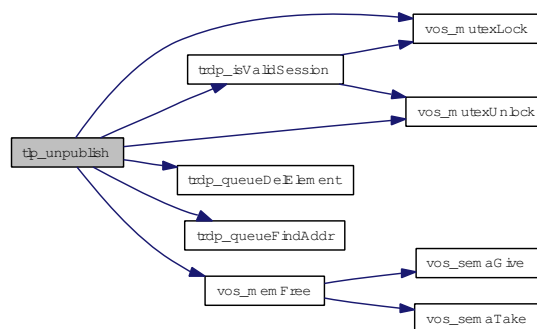
#### Parameters:

- ← *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

Here is the call graph for this function:



### 5.11.2.33 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

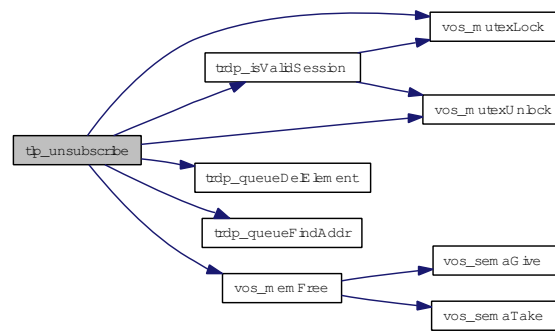
#### Parameters:

- ← *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

Here is the call graph for this function:



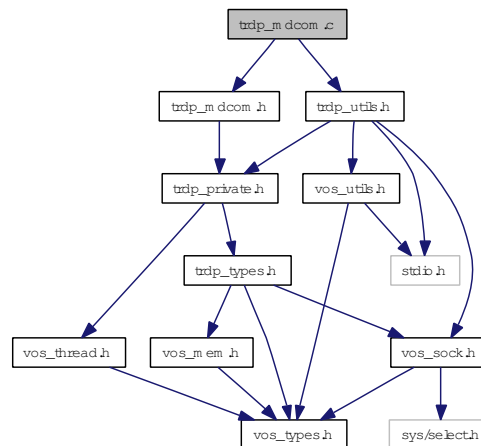
## 5.12 trdp\_mdcom.c File Reference

Functions for MD communication.

```
#include "trdp_utils.h"
```

```
#include "trdp_mdcom.h"
```

Include dependency graph for trdp\_mdcom.c:



### Functions

- **TRDP\_ERR\_T trdp\_sendMD** (int mdSock, const **MD\_ELE\_T** \*pPacket)  
*Send MD packet.*
- **TRDP\_ERR\_T trdp\_rcvMD** (int mdSock, **MD\_HEADER\_T** \*\*ppPacket, **INT32** \*pSize, **UINT32** \*pIPAddr)  
*Receive MD packet.*

### 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, 2012.

#### Id

[trdp\\_mdcom.c](#) 9 2012-06-12 15:30:12Z 97025

## 5.12.2 Function Documentation

### 5.12.2.1 TRDP\_ERR\_T trdp\_rcvMD (int *mdSock*, MD\_HEADER\_T \*\* *ppPacket*, INT32 \* *pSize*, UINT32 \* *pIPAddr*)

Receive MD packet.

#### Parameters:

- ← *mdSock* socket descriptor
- *ppPacket* pointer to pointer to received packet
- *pSize* pointer to size of received packet
- *pIPAddr* pointer to source IP address of packet

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_UNKNOWN\_ERR* error

### 5.12.2.2 TRDP\_ERR\_T trdp\_sendMD (int *mdSock*, const MD\_ELE\_T \* *pPacket*)

Send MD packet.

#### Parameters:

- ← *mdSock* socket descriptor
- ← *pPacket* pointer to packet to be sent

#### Return values:

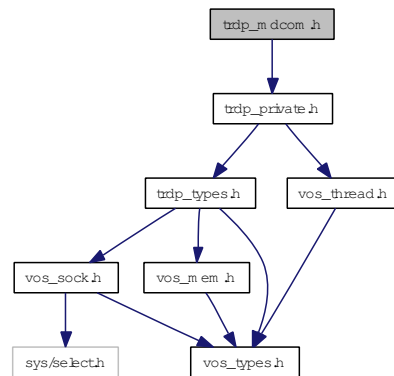
- TRDP\_NO\_ERR* no error
- TRDP\_UNKNOWN\_ERR* error

## 5.13 trdp\_mdcom.h File Reference

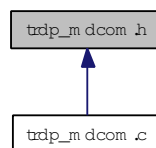
Functions for MD communication.

```
#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\_sendMD** (int sock, const MD\_ELEMENT \*)  
*Send MD packet.*
- **TRDP\_ERR\_T trdp\_rcvMD** (int sock, MD\_HEADER\_T \*\*pPacket, INT32 \*pSize, UINT32 \*pIPAddr)  
*Receive MD packet.*

### 5.13.1 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, 2012.

**Id**

[trdp\\_mdcom.h](#) 9 2012-06-12 15:30:12Z 97025

**5.13.2 Function Documentation****5.13.2.1 TRDP\_ERR\_T trdp\_rcvMD (int *mdSock*, MD\_HEADER\_T \*\**ppPacket*, INT32 \**pSize*, UINT32 \**pIPAddr*)**

Receive MD packet.

**Parameters:**

- ← *mdSock* socket descriptor
- *ppPacket* pointer to pointer to received packet
- *pSize* pointer to size of received packet
- *pIPAddr* pointer to source IP address of packet

**Return values:**

- TRDP\_NO\_ERR* no error
- TRDP\_UNKNOWN\_ERR* error

**5.13.2.2 TRDP\_ERR\_T trdp\_sendMD (int *mdSock*, const MD\_ELE\_T \**pPacket*)**

Send MD packet.

**Parameters:**

- ← *mdSock* socket descriptor
- ← *pPacket* pointer to packet to be sent

**Return values:**

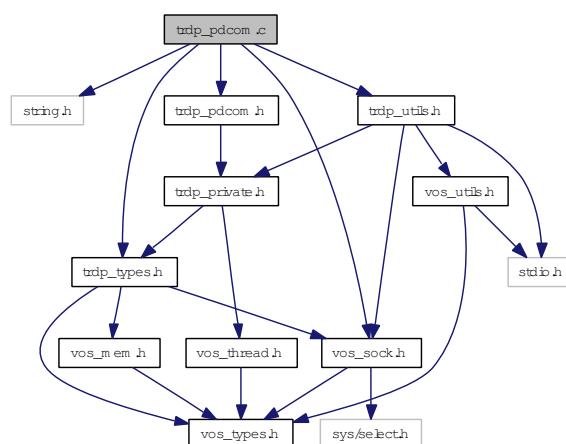
- TRDP\_NO\_ERR* no error
- TRDP\_UNKNOWN\_ERR* error

## 5.14 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 "vos_sock.h"
```

Include dependency graph for trdp\_pdcom.c:



### Functions

- void [trdp\\_pdInit](#) ([PD\\_ELE\\_T](#) \*pPacket, [TRDP\\_MSG\\_T](#) type, UINT32 topoCount)  
*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.*
- [TRDP\\_ERR\\_T](#) [trdp\\_pdGet](#) ([PD\\_ELE\\_T](#) \*pPacket, [TRDP\\_UNMARSHALL\\_T](#) unmarshall, void \*refCon, const [UINT8](#) \*pData, UINT32 dataSize)  
*Copy data Set the header infos.*
- [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\\_pdUpdate](#) ([PD\\_ELE\\_T](#) \*pPacket)  
*Update the header values.*
- [TRDP\\_ERR\\_T](#) [trdp\\_pdCheck](#) ([PD\\_HEADER\\_T](#) \*pPacket, INT32 packetSize)  
*Check if the PD header values are sane.*

- [TRDP\\_ERR\\_T trdp\\_pdSend](#) (INT32 pdSock, const [PD\\_ELE\\_T](#) \*pPacket)

*Send PD packet.*

### 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.c](#) 9 2012-06-12 15:30:12Z 97025

### 5.14.2 Function Documentation

#### 5.14.2.1 TRDP\_ERR\_T trdp\_pdCheck (PD\_HEADER\_T \*pPacket, INT32 packetSize)

Check if the PD header values are sane.

#### Parameters:

← *pPacket* pointer to the packet to update

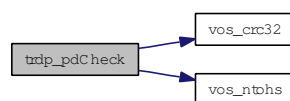
← *packetSize* max size to check

#### Return values:

*TRDP\_NO\_ERR*

*TRDP\_CRC\_ERR*

Here is the call graph for this function:





### 5.14.2.2 void trdp\_pdInit (PD\_ELE\_T \* *pPacket*, TRDP\_MSG\_T *type*, UINT32 *topoCount*)

Initialize/construct the packet Set the header infos.

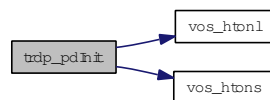
#### Parameters:

← *pPacket* pointer to the packet element to init

← *type* type the packet

← *topoCount* topocount to use for PD frame

Here is the call graph for this function:



### 5.14.2.3 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, discard it (TBD: maybe for another session!). 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

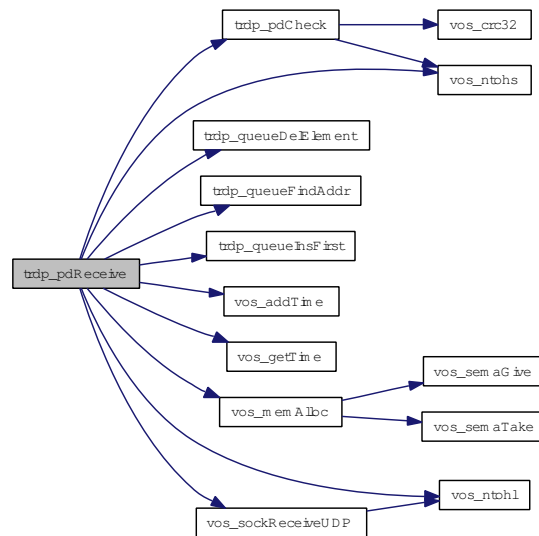
**TRDP\_WIRE\_ERR** protocol error (late packet, version mismatch)

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

**TRDP\_CRC\_ERR** header checksum

**TRDP\_TOPOCOUNT\_ERR** invalid topocount

Here is the call graph for this function:



#### 5.14.2.4 TRDP\_ERR\_T trdp\_pdSend (INT32 *pdSock*, const PD\_ELEM\_T \* *pPacket*)

Send PD packet.

##### Parameters:

- ← *pdSock* socket descriptor
- ← *pPacket* pointer to packet to be sent

##### Return values:

!= NULL error

Here is the call graph for this function:



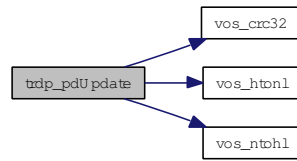
#### 5.14.2.5 void trdp\_pdUpdate (PD\_ELEM\_T \* *pPacket*)

Update the header values.

##### Parameters:

- ← *pPacket* pointer to the packet to update

Here is the call graph for this function:

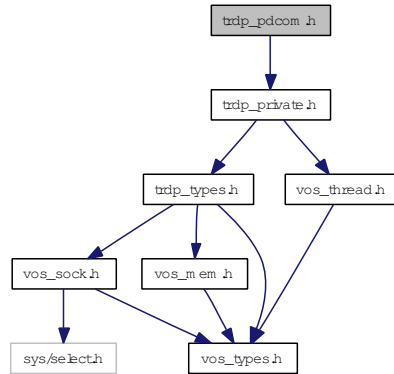


## 5.15 trdp\_pdcom.h File Reference

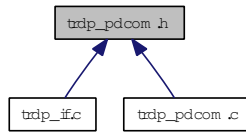
Functions for PD communication.

```
#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)  
*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.*
- [TRDP\\_ERR\\_T](#) [trdp\\_pdCheck](#) ([PD\\_HEADER\\_T](#) \*pPacket, [INT32](#) packetSize)  
*Check if the PD header values are sane.*
- [TRDP\\_ERR\\_T](#) [trdp\\_pdSend](#) ([INT32](#) sock, const [PD\\_ELE\\_T](#) \*)  
*Send PD packet.*
- [TRDP\\_ERR\\_T](#) [trdp\\_pdGet](#) ([PD\\_ELE\\_T](#) \*pPacket, [TRDP\\_UNMARSHALL\\_T](#) unmarshall, void \*refCon, const [UINT8](#) \*pData, [UINT32](#) dataSize)  
*Copy data Set the header infos.*

- [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.*

### 5.15.1 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](#) 2 2012-06-04 11:25:16Z 97025

### 5.15.2 Function Documentation

#### 5.15.2.1 TRDP\_ERR\_T trdp\_pdCheck (PD\_HEADER\_T \* pPacket, INT32 packetSize)

Check if the PD header values are sane.

#### Parameters:

← *pPacket* pointer to the packet to update

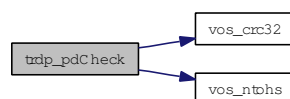
← *packetSize* max size to check

#### Return values:

*TRDP\_NO\_ERR*

*TRDP\_CRC\_ERR*

Here is the call graph for this function:



### 5.15.2.2 void trdp\_pdInit (PD\_ELE\_T \* *pPacket*, TRDP\_MSG\_T *type*, UINT32 *topoCount*)

Initialize/construct the packet Set the header infos.

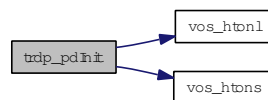
#### Parameters:

← *pPacket* pointer to the packet element to init

← *type* type the packet

← *topoCount* topocount to use for PD frame

Here is the call graph for this function:



### 5.15.2.3 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, discard it (TBD: maybe for another session!). 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

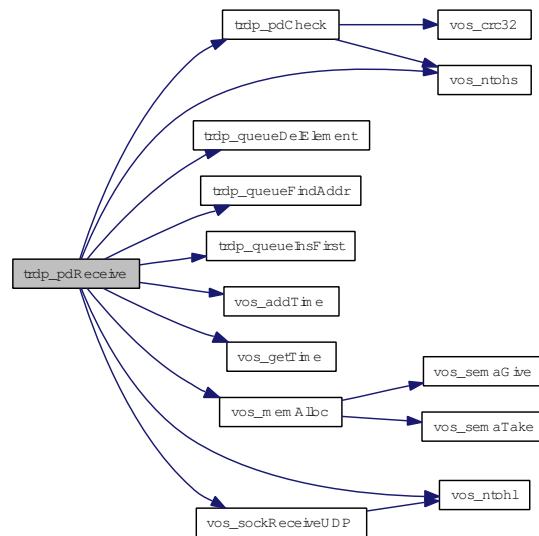
**TRDP\_WIRE\_ERR** protocol error (late packet, version mismatch)

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

**TRDP\_CRC\_ERR** header checksum

**TRDP\_TOPOCOUNT\_ERR** invalid topocount

Here is the call graph for this function:



#### 5.15.2.4 TRDP\_ERR\_T trdp\_pdSend (INT32 *pdSock*, const PD\_ELEMENT \* *pPacket*)

Send PD packet.

##### Parameters:

- ← *pdSock* socket descriptor
- ← *pPacket* pointer to packet to be sent

##### Return values:

!= NULL error

Here is the call graph for this function:



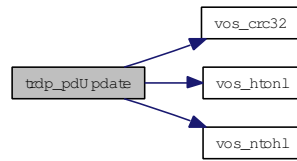
#### 5.15.2.5 void trdp\_pdUpdate (PD\_ELEMENT \* *pPacket*)

Update the header values.

##### Parameters:

- ← *pPacket* pointer to the packet to update

Here is the call graph for this function:





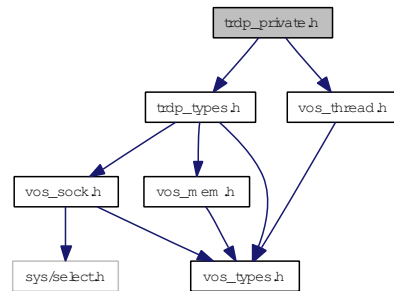
## 5.16 trdp\_private.h File Reference

Typedefs for TRDP communication.

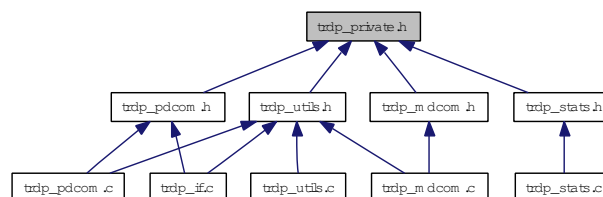
```
#include "trdp_types.h"
```

```
#include "vos_thread.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\\_SOCKETS](#)  
*Socket item.*
- struct [GNU\\_PACKED](#)  
*TRDP process data header - network order and alignment.*
- struct [GNU\\_PACKED](#)  
*TRDP process data header - network order and alignment.*
- struct [PD\\_ELE](#)  
*Queue element for PD packets to send or receive.*
- struct [MD\\_ELE](#)  
*Queue element for MD packets to send or receive or acknowledge.*
- struct [TRDP\\_SESSION](#)

*Session/application variables store.*

- struct [TRDP\\_PD\\_STATISTICS](#)  
*Process data statistics.*
- struct [TRDP\\_MD\\_STATISTICS](#)  
*Message data statistics.*

## Defines

- #define [IP\\_PD\\_UDP\\_PORT](#) 20548  
*process data UDP port*
- #define [IP\\_MD\\_UDP\\_PORT](#) 20550  
*message data UDP port*
- #define [IP\\_PD\\_PROTO\\_VER](#) 0x0100  
*Protocol version.*
- #define [ECHO\\_COMID](#) 110  
*comid used for echo*
- #define [TIMER\\_GRANULARITY](#) 10000  
*granularity in us*
- #define [MD\\_DEFAULT\\_REPLY\\_TIMEOUT](#) 10000000  
*default reply time out 10s*
- #define [MD\\_DEFAULT\\_CONFIRM\\_TIMEOUT](#) 10000000  
*default reply time out 10s*
- #define [MIN\\_PD\\_HEADER\\_SIZE](#) sizeof(PD\_HEADER\_T)  
*PD header size without FCS.*
- #define [ACK\\_TIME\\_OUT\\_VAL\\_DEF](#) 500  
*Default value in milliseconds for waiting on acknowledge message.*

## Typedefs

- typedef struct [TRDP\\_HANDLE](#) [TRDP\\_ADDRESSES](#)  
*Hidden handle definition, used as unique addressing item.*
- 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 [MD\\_ELE MD\\_ELE\\_T](#)  
*Queue element for MD packets to send or receive or acknowledge.*
- typedef struct [TRDP\\_SESSION TRDP\\_SESSION\\_T](#)  
*Session/application variables store.*
- typedef struct [TRDP\\_PD\\_STATISTICS TRDP\\_PD\\_STATS\\_T](#)  
*Process data statistics.*
- typedef struct [TRDP\\_MD\\_STATISTICS TRDP\\_MD\\_STATS\\_T](#)  
*Message data statistics.*

## Enumerations

- enum [TRDP\\_PRIV\\_FLAGS\\_T](#) { , [TRDP\\_TIMED\\_OUT](#) = 0x2 }  
*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.*

### 5.16.1 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](#) 15 2012-06-14 12:52:06Z 97025

## 5.16.2 Enumeration Type Documentation

### 5.16.2.1 enum TRDP\_PRIV\_FLAGS\_T

Internal flags for packets.

**Enumerator:**

*TRDP\_TIMED\_OUT* if set, informed the user

### 5.16.2.2 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.

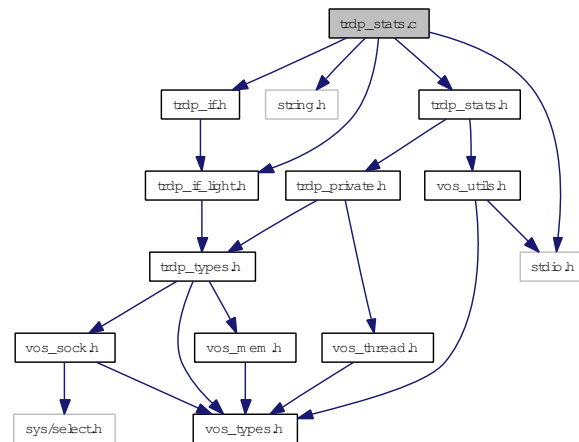
*TRDP SOCK\_MD\_TCP* Socket is used for TCP message data.

## 5.17 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 dependency graph for trdp\_stats.c:



### Functions

- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [TRDP\\_STATISTICS\\_T](#) \*\*ppStatistics)  
*Return statistics.*
- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getSubsStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [UINT16](#) \*pNumSubs, [TRDP\\_SUBS\\_STATISTICS\\_T](#) \*\*ppStatistics)  
*Return PD subscription statistics.*
- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getPubStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [UINT16](#) \*pNumPub, [TRDP\\_PUB\\_STATISTICS\\_T](#) \*\*ppStatistics)  
*Return PD publish statistics.*
- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getListStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [UINT16](#) \*pNumList, [TRDP\\_LIST\\_STATISTICS\\_T](#) \*\*ppStatistics)  
*Return MD listener statistics.*
- EXT\_DECL [TRDP\\_ERR\\_T](#) [tlc\\_getRedStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, [UINT16](#) \*pNumRed, [TRDP\\_RED\\_STATISTICS\\_T](#) \*\*ppStatistics)  
*Return redundancy group statistics.*

- EXT\_DECL [TRDP\\_ERR\\_T tlc\\_getJoinStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle, UINT16 \*pNumJoin, UINT32 \*\*ppIpAddr)  
*Return join statistics.*
- EXT\_DECL [TRDP\\_ERR\\_T tlc\\_resetStatistics](#) ([TRDP\\_APP\\_SESSION\\_T](#) appHandle)  
*Reset statistics.*

### 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](#) 9 2012-06-12 15:30:12Z 97025

### 5.17.2 Function Documentation

#### 5.17.2.1 EXT\_DECL TRDP\_ERR\_T tlc\_getJoinStatistics (TRDP\_APP\_SESSION\_T appHandle, UINT16 \*pNumJoin, UINT32 \*\*ppIpAddr)

Return join statistics.

Memory for statistics information will be reserved by tlc layer and needs to be freed by the user.

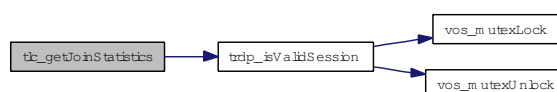
#### Parameters:

- ← **appHandle** the handle returned by tlc\_init
- **pNumJoin** Pointer to the number of joined IP Adresses
- **ppIpAddr** 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

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 \*\**ppStatistics*)

Return MD listener statistics.

Memory for statistics information will be reserved by tlc layer and needs to be freed by the user.

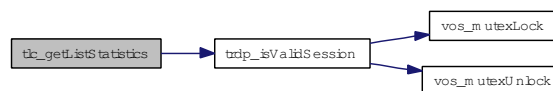
#### Parameters:

- ← *appHandle* the handle returned by tlc\_init
- *pNumList* Pointer to the number of listeners
- *ppStatistics* 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

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 \*\**ppStatistics*)

Return PD publish statistics.

Memory for statistics information will be reserved by tlc layer and needs to be freed by the user.

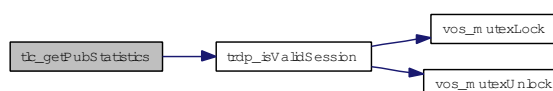
#### Parameters:

- ← *appHandle* the handle returned by tlc\_init
- *pNumPub* Pointer to the number of publishers
- *ppStatistics* 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

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 \*\*ppStatistics)

Return redundancy group statistics.

Memory for statistics information will be reserved by tlc layer and needs to be freed by the user.

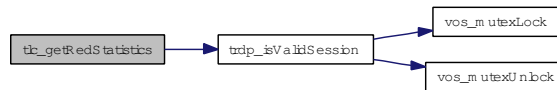
##### Parameters:

- ← *appHandle* the handle returned by tlc\_init
- *pNumRed* Pointer to the number of redundancy groups
- *ppStatistics* 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

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 \*\*ppStatistics)

Return statistics.

Memory for statistics information will be reserved by tlc layer and needs to be freed by the user.

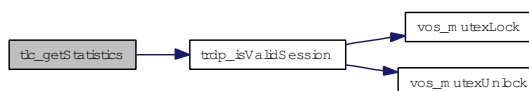
##### Parameters:

- ← *appHandle* the handle returned by tlc\_init
- *ppStatistics* 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 \*\**ppStatistics*)

Return PD subscription statistics.

Memory for statistics information will be reserved by tlc layer and needs to be freed by the user.

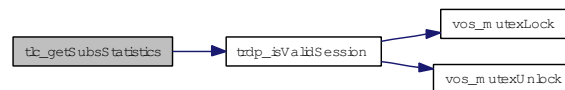
#### Parameters:

- ← *appHandle* the handle returned by tlc\_init
- *pNumSubs* Pointer to the number of subscriptions
- *ppStatistics* Pointer to a list with the subscription statistics information

#### 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:



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

Reset statistics.

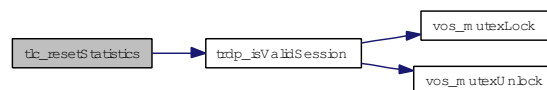
#### Parameters:

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

#### Return values:

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

Here is the call graph for this function:



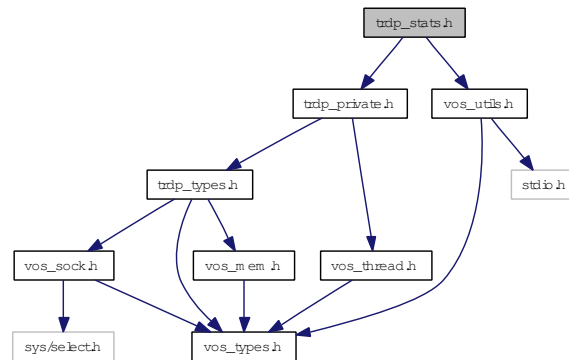
## 5.18 trdp\_stats.h File Reference

Statistics for TRDP communication.

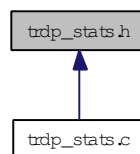
```
#include "trdp_private.h"
```

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

Include dependency graph for trdp\_stats.h:



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



### 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](#) 3 2012-06-04 12:52:54Z 97025

## 5.19 trdp\_types.h File Reference

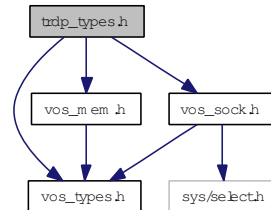
Typedefs for TRDP communication.

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

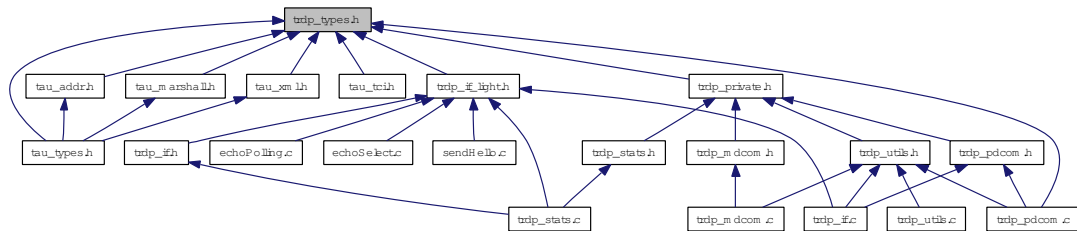
```
#include "vos_mem.h"
```

```
#include "vos_sock.h"
```

Include dependency graph for trdp\_types.h:



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



### Data Structures

- struct [TRDP\\_PD\\_INFO\\_T](#)  
*Process data info from received telegram; allows the application to generate responses.*
- struct [TRDP\\_MD\\_INFO\\_T](#)  
*Message data info from received telegram; allows the application to generate responses.*
- struct [TRDP\\_SEND\\_PARAM\\_T](#)  
*Quality/type of service and time to live.*
- struct [TRDP\\_DATASET\\_ELEMENT\\_T](#)  
*Dataset element definition.*
- struct [TRDP\\_DATASET\\_T](#)  
*Dataset 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](#)  
*Structure describing memory (and its pre-fragmentation).*

## Defines

- #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 [USE\\_HEAP](#) 0  
*If this is set, we can allocate dynamically memory.*

## Typedefs

- typedef UINT32 [TRDP\\_IP\\_ADDR\\_T](#)  
*TRDP general type definitions.*
- typedef [VOS\\_TIME\\_T](#) [TRDP\\_TIME\\_T](#)  
*Timer value compatible with timeval / select.*
- typedef struct fd\_set [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 [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, const UINT8 \*pSrc, UINT8 \*pDst, UINT32 \*pDstSize)  
*Function type for marshallng .*
- typedef [TRDP\\_ERR\\_T](#)(\* [TRDP\\_UNMARSHALL\\_T](#) )(void \*pRefCon, UINT32 comId, const UINT8 \*pSrc, UINT8 \*pDst, UINT32 \*pDstSize)  
*Function type for unmarshalling.*
- typedef void(\* [TRDP\\_PD\\_CALLBACK\\_T](#) )(void \*pRefCon, 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, const [TRDP\\_MD\\_INFO\\_T](#) \*pMsg, UINT8 \*pData, UINT32 dataSize)  
*Callback for receiving indications, timeouts, releases, responses.*
- typedef [VOS\\_MEM\\_BLK\\_T](#) [TRDP\\_MEM\\_BLK\\_T](#)  
*Enumeration type for memory pre-fragmentation, reuse of VOS definition.*

## 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 SOCK_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_NOSESSION_ERR = -13,
TRDP_SESSION_ABORT_ERR = -14,
TRDP_NOSUB_ERR = -15,
TRDP_NOPUB_ERR = -16,
TRDP_NOLIST_ERR = -17,
TRDP_CRC_ERR = -18 ,
TRDP_TOPO_ERR = -20,
TRDP_COMID_ERR = -21,
TRDP_STATE_ERR = -22,
TRDP_UNKNOWN_ERR = -99 }

```

*Return codes for all API functions.*

- enum TRDP\_MSG\_T {
 

```

TRDP_MSG_PD = 0x5064,
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 }

```

*TRDP data transfer type definitions.*

- enum TRDP\_REPLY\_STATUS\_T

*Reply status messages.*

- enum TRDP\_FLAGS\_T { ,
 

```

TRDP_FLAGS_REDUNDANT = 0x1,
TRDP_FLAGS_MARSHALL = 0x2,
TRDP_FLAGS_CALLBACK = 0x4,
TRDP_FLAGS_TCP = 0x8 }

```

*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`  
*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_STRING` = -14,  
`TRDP_ARRAY` = -15,  
`TRDP_RECORD` = -16,  
`TRDP_TIMEDATE32` = -17,  
`TRDP_TIMEDATE48` = -18,  
`TRDP_TIMEDATE64` = -19 }

*TRDP dataset description definitions.*

- enum `TRDP_OPTION_T` { ,  
`TRDP_OPTION_BLOCK` = 0x01,  
`TRDP_OPTION_TRAFFIC_SHAPING` = 0x02 }

*Various flags/general TRDP options for library initialization.*

### 5.19.1 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\\_types.h](#) 9 2012-06-12 15:30:12Z 97025

## 5.19.2 Define Documentation

### 5.19.2.1 #define TRDP\_MAX\_FILE\_NAME\_LEN 128

path and file name length incl.

terminating '0'

### 5.19.2.2 #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'

### 5.19.2.3 #define TRDP\_MAX\_URI\_HOST\_LEN (4 \* TRDP\_MAX\_LABEL\_LEN)

URI host part length incl.

terminating '0'

### 5.19.2.4 #define TRDP\_MAX\_URI\_LEN ((6 \* TRDP\_MAX\_LABEL\_LEN) + 8)

URI length incl.

terminating '0' and 1 padding byte

### 5.19.2.5 #define TRDP\_MAX\_URI\_USER\_LEN (2 \* TRDP\_MAX\_LABEL\_LEN)

URI user part incl.

terminating '0'

## 5.19.3 Typedef Documentation

### 5.19.3.1 typedef UINT32 TRDP\_IP\_ADDR\_T

TRDP general type definitions.



### 5.19.3.2 typedef TRDP\_ERR\_T(\* TRDP\_MARSHALL\_T)(void \*pRefCon, UINT32 comId, const UINT8 \*pSrc, UINT8 \*pDst, UINT32 \*pDstSize)

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

#### Return values:

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

### 5.19.3.3 typedef void(\* TRDP\_MD\_CALLBACK\_T)(void \*pRefCon, const TRDP\_MD\_INFO\_T \*pMsg, UINT8 \*pData, UINT32 dataSize)

Callback for receiving indications, timeouts, releases, responses.

#### Parameters:

- ← *\*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 !!!!

### 5.19.3.4 typedef void(\* TRDP\_PD\_CALLBACK\_T)(void \*pRefCon, const TRDP\_PD\_INFO\_T \*pMsg, UINT8 \*pData, UINT32 dataSize)

Callback for receiving indications, timeouts, releases, responses.

#### Parameters:

- ← *\*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 !!!!

### 5.19.3.5 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.

### 5.19.3.6 typedef VOS\_TIME\_T TRDP\_TIME\_T

Timer value compatible with timeval / select.

Relative or absolute date, depending on usage

### 5.19.3.7 typedef TRDP\_ERR\_T(\* TRDP\_UNMARSHALL\_T)(void \*pRefCon, UINT32 comId, const UINT8 \*pSrc, UINT8 \*pDst, UINT32 \*pDstSize)

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

#### Return values:

- TRDP\_NO\_ERR* no error
- TRDP\_MEM\_ERR* provide buffer to small
- TRDP\_COMID\_ERR* comid not existing

## 5.19.4 Enumeration Type Documentation

### 5.19.4.1 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\_STRING** Zero-terminated array of CHAR8, fixed size.  
**TRDP\_ARRAY** Array.  
**TRDP\_RECORD** Record.  
**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 milliseconds

#### 5.19.4.2 enum TRDP\_ERR\_T

Return codes for all API functions.

##### 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\_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\_TOPO\_ERR** Invalid topo count.  
**TRDP\_COMID\_ERR** Unknown ComId.  
**TRDP\_STATE\_ERR** Call in wrong state.  
**TRDP\_UNKNOWN\_ERR** Unspecified error.

#### 5.19.4.3 enum TRDP\_FLAGS\_T

Various flags for PD and MD packets.

**Enumerator:**

*TRDP\_FLAGS\_REDUNDANT* Redundant.

*TRDP\_FLAGS\_MARSHALL* Optional marshalling/unmarshalling in TRDP stack.

*TRDP\_FLAGS\_CALLBACK* Use of callback function.

*TRDP\_FLAGS\_TCP* Use TCP for message data.

#### 5.19.4.4 enum TRDP\_MSG\_T

TRDP data transfer type definitions.

Message Types

**Enumerator:**

*TRDP\_MSG\_PD* 'Pd' PD Data (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

#### 5.19.4.5 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.

#### 5.19.4.6 enum TRDP\_RED\_STATE\_T

Redundancy states.

**Enumerator:**

*TRDP\_RED\_FOLLOWER* Redundancy follower - redundant PD will be not sent out.

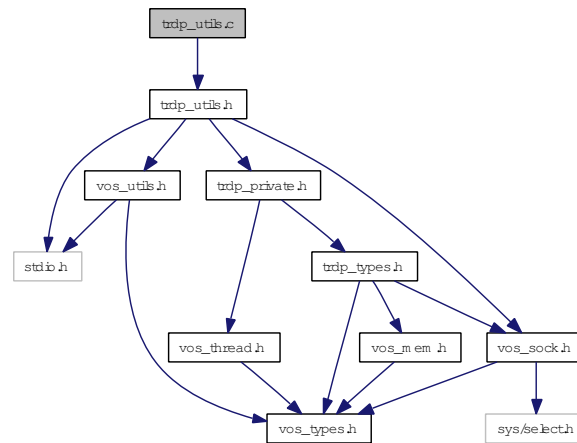
*TRDP\_RED\_LEADER* Redundancy leader - redundant PD will be sent out.

## 5.20 trdp\_utils.c File Reference

Helper functions for TRDP communication.

```
#include "trdp_utils.h"
```

Include dependency graph for trdp\_utils.c:



### Functions

- `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.*
- `PD_ELE_T * trdp_queueFindComId (PD_ELE_T **ppHead, UINT32 comId)`  
*Return the element with same comId.*
- `PD_ELE_T * trdp_queueFindAddr (PD_ELE_T *pHead, TRDP_ADDRESSES *addr)`  
*Return the element with same comId.*
- `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[ ], const [TRDP\\_SEND\\_PARAM\\_T](#) \*params, [TRDP\\_IP\\_ADDR\\_T](#) srcIP, [TRDP SOCK\\_TYPE\\_T](#) usage, [TRDP\\_OPTION\\_T](#) options, INT32 \*pIndex)

*Handle the socket pool: Request a socket from our socket pool.*

- [TRDP\\_ERR\\_T trdp\\_releaseSocket](#) ([TRDP\\_SOCKETS\\_T](#) iface[ ], INT32 index)

*Handle the socket pool: Release a socket from our socket pool.*

### 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, 2012.

#### Id

[trdp\\_utils.c](#) 15 2012-06-14 12:52:06Z 97025

### 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 trdp\_initSockets (TRDP\_SOCKETS\_T iface[ ])

Handle the socket pool: Initialize it.

#### Parameters:

← *iface* pointer to the socket pool

**5.20.2.3   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

**5.20.2.4   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

**5.20.2.5   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

**5.20.2.6   PD\_ELE\_T\* trdp\_queueFindAddr (PD\_ELE\_T \* *pHead*, TRDP\_ADDRESSES \* *addr*)**

Return the element with same comId.

**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

**5.20.2.7   PD\_ELE\_T\* trdp\_queueFindComId (PD\_ELE\_T \*\* *ppHead*, UINT32 *comId*)**

Return the element with same comId.

**Parameters:**

← *ppHead* pointer to pointer to head of queue

← *comId* ComID to search for

**Return values:**

!= NULL pointer to PD element

NULL No PD element found

### 5.20.2.8 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

### 5.20.2.9 TRDP\_ERR\_T trdp\_releaseSocket (TRDP\_SOCKETS\_T *iface*[], INT32 *index*)

Handle the socket pool: Release a socket from our socket pool.

**Parameters:**

↔ *iface* socket pool

← *index* index of socket to release

**Return values:**

TRDP\_NO\_ERR

TRDP\_PARAM\_ERR

Here is the call graph for this function:



### 5.20.2.10 TRDP\_ERR\_T trdp\_requestSocket (TRDP\_SOCKETS\_T *iface*[], const TRDP\_SEND\_PARAM\_T \**params*, TRDP\_IP\_ADDR\_T *srcIP*, TRDP SOCK\_TYPE\_T *usage*, TRDP\_OPTION\_T *options*, INT32 \**pIndex*)

Handle the socket pool: Request a socket from our socket pool.

**Parameters:**

↔ *iface* socket pool

← *params* parameters to use

← *srcIP* IP to bind to (0 = any address)

← *usage* type and port to bind to

← *options* blocking/nonblocking



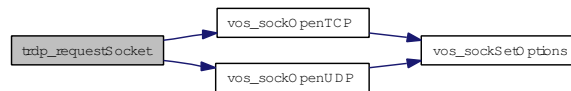
→ *pIndex* returned index of socket pool

**Return values:**

***TRDP\_NO\_ERR***

***TRDP\_PARAM\_ERR***

Here is the call graph for this function:

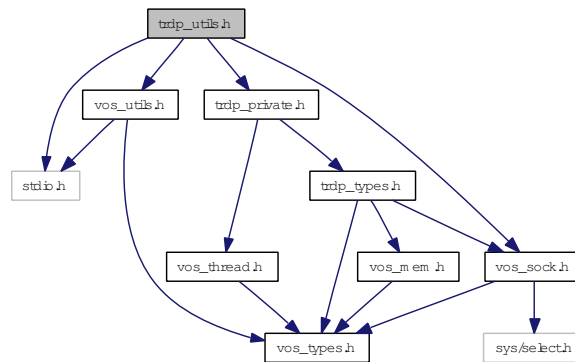


## 5.21 trdp\_utils.h File Reference

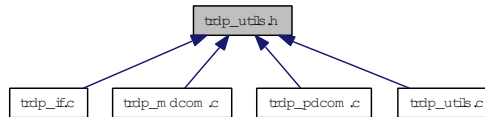
Common utilities for TRDP communication.

```
#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_ELE_T * trdp_queueFindComId (PD_ELE_T **pHead, UINT32 comId)`  
*Return the element with same comId.*
- `PD_ELE_T * trdp_queueFindAddr (PD_ELE_T *pHead, TRDP_ADDRESSES *pAddr)`  
*Return the element with same comId.*
- `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.*

- TRDP\_ERR\_T [trdp\\_requestSocket](#) (TRDP\_SOCKETS\_T iface[ ], const TRDP\_SEND\_PARAM\_T \*params, TRDP\_IP\_ADDR\_T srcIP, TRDP SOCK\_TYPE\_T usage, TRDP\_OPTION\_T options, INT32 \*pIndex)

*Handle the socket pool: Request a socket from our socket pool.*

- TRDP\_ERR\_T [trdp\\_releaseSocket](#) (TRDP\_SOCKETS\_T iface[ ], INT32 index)

*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.*

### 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](#) 15 2012-06-14 12:52:06Z 97025

### 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 void trdp\_initSockets (TRDP\_SOCKETS\_T *iface*[])

Handle the socket pool: Initialize it.

#### Parameters:

← *iface* pointer to the socket pool

### 5.21.2.3 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

### 5.21.2.4 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

### 5.21.2.5 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

### 5.21.2.6 PD\_ELE\_T\* trdp\_queueFindAddr (PD\_ELE\_T \**pHead*, TRDP\_ADDRESSES \**addr*)

Return the element with same comId.

#### 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

**5.21.2.7 PD\_ELE\_T\* trdp\_queueFindComId (PD\_ELE\_T \*\* ppHead, UINT32 comId)**

Return the element with same comId.

**Parameters:**

- ← *ppHead* pointer to pointer to head of queue
- ← *comId* ComID to search for

**Return values:**

- !=* NULL pointer to PD element
- NULL* No PD element found

**5.21.2.8 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

**5.21.2.9 TRDP\_ERR\_T trdp\_releaseSocket (TRDP\_SOCKETS\_T iface[], INT32 index)**

Handle the socket pool: Release a socket from our socket pool.

**Parameters:**

- ↔ *iface* socket pool
- ← *index* index of socket to release

**Return values:**

- TRDP\_NO\_ERR*
- TRDP\_PARAM\_ERR*

Here is the call graph for this function:

**5.21.2.10 TRDP\_ERR\_T trdp\_requestSocket (TRDP\_SOCKETS\_T iface[], const TRDP\_SEND\_PARAM\_T \* params, TRDP\_IP\_ADDR\_T srcIP, TRDP SOCK\_TYPE\_T usage, TRDP\_OPTION\_T options, INT32 \* pIndex)**

Handle the socket pool: Request a socket from our socket pool.

**Parameters:**

- ↔ *iface* socket pool

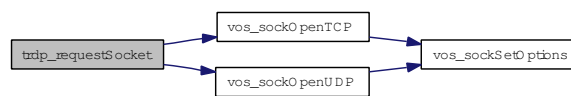
← *params* parameters to use  
 ← *srcIP* IP to bind to (0 = any address)  
 ← *usage* type and port to bind to  
 ← *options* blocking/nonblocking  
 → *pIndex* returned index of socket pool

**Return values:**

*TRDP\_NO\_ERR*

*TRDP\_PARAM\_ERR*

Here is the call graph for this function:

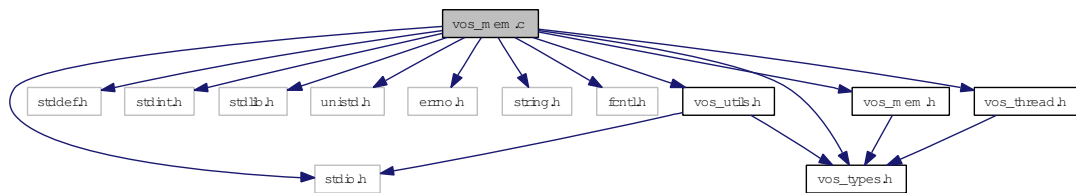


## 5.22 vos\_mem.c File Reference

Memory functions.

```
#include <stdio.h>
#include <stddef.h>
#include <stdint.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include "vos_types.h"
#include "vos_utils.h"
#include "vos_mem.h"
#include "vos_thread.h"
```

Include dependency graph for vos\_mem.c:



### Functions

- EXT\_DECL [VOS\\_ERR\\_T vos\\_memInit](#) (UINT8 \*pMemoryArea, UINT32 size, const UINT32 fragMem[VOS\_MEM\_NBLOCKSIZES])  
*Initialize the memory unit.*
- EXT\_DECL [VOS\\_ERR\\_T 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 [VOS\\_ERR\\_T 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 \*pFragMem[VOS\_MEM\_NBLOCKSIZES])  
*Return used and available memory (of memory area above).*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_queueCreate](#) (const CHAR8 \*pKey, [VOS\\_QUEUE\\_T](#) \*pQueueID, UINT32 maxNoMsg, UINT32 maxLength)  
*Initialize a message queue.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_queueDestroy](#) ([VOS\\_QUEUE\\_T](#) queueID)  
*Destroy a message queue.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_queueSend](#) ([VOS\\_QUEUE\\_T](#) queueID, const UINT8 \*pMsg, UINT32 size)  
*Send a message.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_queueReceive](#) ([VOS\\_QUEUE\\_T](#) queueID, UINT8 \*pMsg, UINT32 \*pSize, UINT32 usTimeout)  
*Get a message.*
- 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.*

### 5.22.1 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](#) 2 2012-06-04 11:25:16Z 97025

### 5.22.2 Function Documentation

#### 5.22.2.1 EXT\_DECL UINT8\* vos\_memAlloc (UINT32 size)

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



**Parameters:**

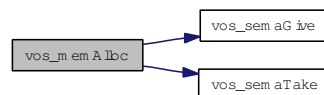
← *size* Size of requested block

**Return values:**

*Pointer* to memory area

*NULL* if no memory available

Here is the call graph for this function:



### 5.22.2.2 EXT\_DECL VOS\_ERR\_T vos\_memCount (UINT32 \* *pAllocatedMemory*, UINT32 \* *pFreeMemory*, UINT32 \* *pFragMem*[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

→ *pFragMem* Pointer to list of used memory blocks

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_INIT\_ERR* module not initialised

*VOS\_PARAM\_ERR* parameter out of range/invalid

### 5.22.2.3 EXT\_DECL VOS\_ERR\_T 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

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_INIT\_ERR* module not initialised

*VOS\_PARAM\_ERR* parameter out of range/invalid

#### 5.22.2.4 EXT\_DECL VOS\_ERR\_T vos\_memFree (void \* *pMemBlock*)

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

##### Parameters:

← *pMemBlock* Pointer to memory block to be freed

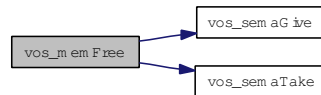
##### Return values:

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_PARAM\_ERR** parameter out of range/invalid

Here is the call graph for this function:



#### 5.22.2.5 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 preallocated block sizes, used to fragment memory for large blocks

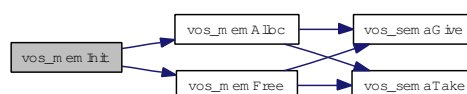
##### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_MEM\_ERR** no memory available

Here is the call graph for this function:



### 5.22.2.6 EXT\_DECL VOS\_ERR\_T vos\_queueCreate (const CHAR8 \* *pKey*, VOS\_QUEUE\_T \* *pQueueID*, UINT32 *maxNoMsg*, UINT32 *maxLength*)

Initialize a message queue.

Returns a handle for further calls

#### Parameters:

- ← *pKey* Unique identifier (file name)
- *pQueueID* Pointer to returned queue handle
- ← *maxNoMsg* maximum number of messages
- ← *maxLength* maximum size of one messages

#### 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

### 5.22.2.7 EXT\_DECL VOS\_ERR\_T vos\_queueDestroy (VOS\_QUEUE\_T *queueID*)

Destroy a message queue.

Free all resources used by this queue

#### Parameters:

- ← *queueID* 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

### 5.22.2.8 EXT\_DECL VOS\_ERR\_T vos\_queueReceive (VOS\_QUEUE\_T *queueID*, UINT8 \* *pMsg*, UINT32 \* *pSize*, UINT32 *usTimeout*)

Get a message.

#### Parameters:

- ← *queueID* Queue handle
- *pMsg* Pointer to message to be received
- ↔ *pSize* Pointer to max. message size on entry, actual size on exit

← *usTimeout* Maximum time to wait for a message in usec

**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\_QUEUE\_ERR* queue is empty

**5.22.2.9 EXT\_DECL VOS\_ERR\_T vos\_queueSend (VOS\_QUEUE\_T *queueID*, const UINT8 \*  
*pMsg*, UINT32 *size*)**

Send a message.

**Parameters:**

← *queueID* Queue handle  
 ← *pMsg* Pointer to message to be sent  
 ← *size* Message size

**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\_QUEUE\_FULL* queue is full

**5.22.2.10 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\_INIT\_ERR* module not initialised  
*VOS\_NOINIT\_ERR* invalid handle  
*VOS\_PARAM\_ERR* parameter out of range/invalid

### 5.22.2.11 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:

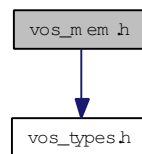
- VOS\_NO\_ERR* no error
- VOS\_INIT\_ERR* module not initialised
- VOS\_NOINIT\_ERR* invalid handle
- VOS\_PARAM\_ERR* parameter out of range/invalid
- VOS\_MEM\_ERR* no memory available

## 5.23 vos\_mem.h File Reference

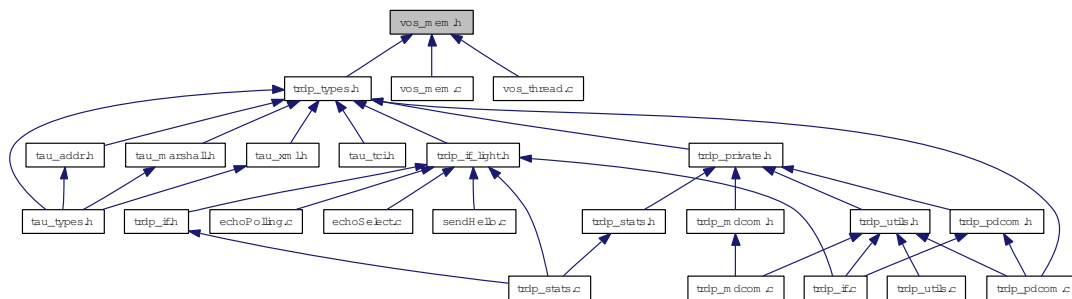
Memory and queue functions for OS abstraction.

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

Include dependency graph for vos\_mem.h:



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



### Defines

- `#define VOS_MEM_BLOCKSIZEs`  
*We internally allocate memory always by these block sizes.*
- `#define VOS_MEM_PREALLOCATE {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 4, 0, 0}`  
*Default pre-allocation of free memory blocks.*

### Typedefs

- `typedef struct VOS_QUEUE_T * VOS_QUEUE_T`  
*Opaque queue define.*

### Enumerations

- `enum VOS_MEM_BLK_T`  
*enumeration for memory block sizes*

## Functions

- EXT\_DECL [VOS\\_ERR\\_T vos\\_memInit](#) (UINT8 \*pMemoryArea, UINT32 size, const UINT32 fragMem[VOS\_MEM\_NBLOCKSIZES])  
*Initialize the memory unit.*
- EXT\_DECL [VOS\\_ERR\\_T 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 [VOS\\_ERR\\_T 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 \*pFragMem[VOS\_MEM\_NBLOCKSIZES])  
*Return used and available memory (of memory area above).*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_queueCreate](#) (const CHAR8 \*pKey, [VOS\\_QUEUE\\_T](#) \*pQueueId, UINT32 maxNoMsg, UINT32 maxLength)  
*Initialize a message queue.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_queueDestroy](#) ([VOS\\_QUEUE\\_T](#) queueID)  
*Destroy a message queue.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_queueSend](#) ([VOS\\_QUEUE\\_T](#) queueID, const UINT8 \*pMsg, UINT32 size)  
*Send a message.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_queueReceive](#) ([VOS\\_QUEUE\\_T](#) queueID, UINT8 \*pMsg, UINT32 \*pSize, UINT32 usTimeout)  
*Get a message.*
- 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.*

### 5.23.1 Detailed Description

Memory and queue functions for OS abstraction.

This module provides three services: 1. A memory control supervisor

- Private memory management with optimised fragmentation handling

- A message queue handler (system-wide on supported systems)
- Access to shared memory (on supported systems only) Within the prototype TRDP stack, only the memory management unit is currently in use.

**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](#) 2 2012-06-04 11:25:16Z 97025

**5.23.2 Define Documentation****5.23.2.1 #define VOS\_MEM\_BLOCKSIZEs****Value:**

```
{32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, \
  16384, 32768, 65536, 131072, 262144, 524288}
```

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, 0, 1, 1, 1, 4, 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 UINT8\* vos\_memAlloc (UINT32 *size*)**

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

**Parameters:**

← *size* Size of requested block

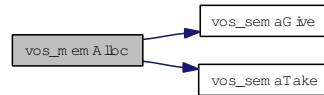
**Return values:**

*Pointer* to memory area



*NULL* if no memory available

Here is the call graph for this function:



### 5.23.3.2 EXT\_DECL VOS\_ERR\_T vos\_memCount (UINT32 \* *pAllocatedMemory*, UINT32 \* *pFreeMemory*, UINT32 \* *pFragMem*[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
- *pFragMem* Pointer to list of used memory blocks

#### Return values:

- VOS\_NO\_ERR* no error
- VOS\_INIT\_ERR* module not initialised
- VOS\_PARAM\_ERR* parameter out of range/invalid

### 5.23.3.3 EXT\_DECL VOS\_ERR\_T 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

#### Return values:

- VOS\_NO\_ERR* no error
- VOS\_INIT\_ERR* module not initialised
- VOS\_PARAM\_ERR* parameter out of range/invalid

### 5.23.3.4 EXT\_DECL VOS\_ERR\_T vos\_memFree (void \* *pMemBlock*)

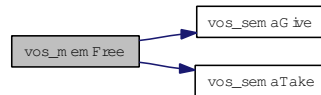
Deallocate a block of memory (from memory area above).

#### Parameters:

- ← *pMemBlock* Pointer to memory block to be freed

**Return values:****VOS\_NO\_ERR** no error**VOS\_INIT\_ERR** module not initialised**VOS\_PARAM\_ERR** parameter out of range/invalid

Here is the call graph for this function:



### 5.23.3.5 EXT\_DECL VOS\_ERR\_T vos\_memInit (UINT8 \*pMemoryArea, UINT32 size, const UINT32 fragMem[VOS\_MEM\_NBLOCKSIZES])

Initialize the memory unit.

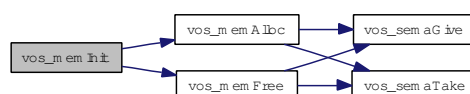
Init a supplied block of memory and prepare it for use with vos\_alloc and vos\_dealloc. The used block sizes can be supplied and will be preallocated.

**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\_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 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

Here is the call graph for this function:



### 5.23.3.6 EXT\_DECL VOS\_ERR\_T vos\_queueCreate (const CHAR8 \* *pKey*, VOS\_QUEUE\_T \* *pQueueID*, UINT32 *maxNoMsg*, UINT32 *maxLength*)

Initialize a message queue.

Returns a handle for further calls

#### Parameters:

- ← *pKey* Unique identifier (file name)
- *pQueueID* Pointer to returned queue handle
- ← *maxNoMsg* maximum number of messages
- ← *maxLength* maximum size of one messages

#### 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

### 5.23.3.7 EXT\_DECL VOS\_ERR\_T vos\_queueDestroy (VOS\_QUEUE\_T *queueID*)

Destroy a message queue.

Free all resources used by this queue

#### Parameters:

- ← *queueID* 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

### 5.23.3.8 EXT\_DECL VOS\_ERR\_T vos\_queueReceive (VOS\_QUEUE\_T *queueID*, UINT8 \* *pMsg*, UINT32 \* *pSize*, UINT32 *usTimeout*)

Get a message.

#### Parameters:

- ← *queueID* Queue handle
- *pMsg* Pointer to message to be received
- ↔ *pSize* Pointer to max. message size on entry, actual size on exit

← *usTimeout* Maximum time to wait for a message in usec

**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\_QUEUE\_ERR* queue is empty

**5.23.3.9 EXT\_DECL VOS\_ERR\_T vos\_queueSend (VOS\_QUEUE\_T *queueID*, const UINT8 \*  
*pMsg*, UINT32 *size*)**

Send a message.

**Parameters:**

← *queueID* Queue handle  
 ← *pMsg* Pointer to message to be sent  
 ← *size* Message size

**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\_QUEUE\_FULL* queue is full

**5.23.3.10 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\_INIT\_ERR* module not initialised  
*VOS\_NOINIT\_ERR* invalid handle  
*VOS\_PARAM\_ERR* parameter out of range/invalid

### 5.23.3.11 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:

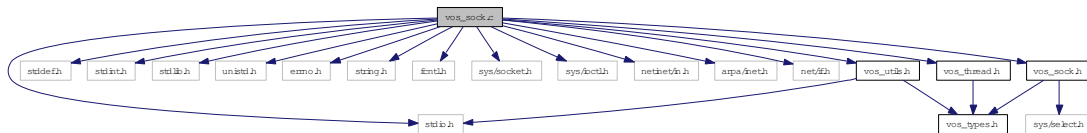
- VOS\_NO\_ERR* no error
- VOS\_INIT\_ERR* module not initialised
- VOS\_NOINIT\_ERR* invalid handle
- VOS\_PARAM\_ERR* parameter out of range/invalid
- VOS\_MEM\_ERR* no memory available

## 5.24 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 <netinet/in.h>
#include <arpa/inet.h>
#include <net/if.h>
#include "vos_utils.h"
#include "vos_sock.h"
#include "vos_thread.h"
```

Include dependency graph for vos\_sock.c:



### Functions

- 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 BOOL [vos\\_isMulticast](#) (UINT32 ipAddress)  
*Check if the supplied address is a multicast group address.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockInit](#) (void)  
*Initialize the socket library.*
- 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 size, UINT32 ipAddress, UINT16 port)  
*Send UDP data.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockReceiveUDP](#) (INT32 sock, UINT8 \*pBuffer, INT32 \*pSize, UINT32 \*pIPAddr)  
*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 \*pIPAddress, 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 size)  
*Send TCP data.*

- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockReceiveTCP](#) (INT32 sock, UINT8 \*pBuffer, INT32 \*pSize)  
*Receive TCP data.*

### 5.24.1 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](#) 10 2012-06-12 15:34:26Z 97025

### 5.24.2 Function Documentation

#### 5.24.2.1 EXT\_DECL UINT32 vos\_htonl (UINT32 *val*)

Byte swapping 4 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

#### 5.24.2.2 EXT\_DECL UINT16 vos\_htons (UINT16 *val*)

Byte swapping.

Byte swapping 2 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value



### 5.24.2.3 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

### 5.24.2.4 EXT\_DECL UINT32 vos\_ntohl (UINT32 *val*)

Byte swapping 4 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

### 5.24.2.5 EXT\_DECL UINT16 vos\_ntohs (UINT16 *val*)

Byte swapping 2 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

### 5.24.2.6 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

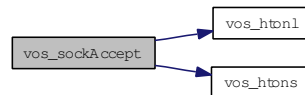
→ *pSock* Pointer to socket descriptor, on exit new socket

→ *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:



#### 5.24.2.7 EXT\_DECL VOS\_ERR\_T vos\_sockBind (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Bind a socket to an address and port.

**Parameters:**

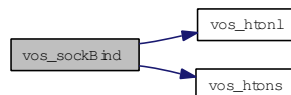
← *sock* socket descriptor

← *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:



#### 5.24.2.8 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** sock descriptor unknown

### 5.24.2.9 EXT\_DECL VOS\_ERR\_T vos\_sockConnect (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Open a TCP connection.

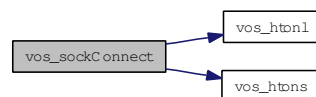
#### Parameters:

- ← *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:



### 5.24.2.10 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

### 5.24.2.11 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:



#### 5.24.2.12 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:



#### 5.24.2.13 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

#### 5.24.2.14 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:



#### 5.24.2.15 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:



#### 5.24.2.16 EXT\_DECL VOS\_ERR\_T vos\_sockReceiveTCP (INT32 *sock*, UINT8 \* *pBuffer*, INT32 \* *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

#### 5.24.2.17 EXT\_DECL VOS\_ERR\_T vos\_sockReceiveUDP (INT32 *sock*, UINT8 \* *pBuffer*, INT32 \* *pSize*, UINT32 \* *pIPAddr*)

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.

##### Parameters:

- ← *sock* socket descriptor
- *pBuffer* pointer to applications data buffer
- ↔ *pSize* pointer to the received data size
- *pIPAddr* source IP

##### 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

Here is the call graph for this function:



#### 5.24.2.18 EXT\_DECL VOS\_ERR\_T vos\_sockSendTCP (INT32 *sock*, const UINT8 \* *pBuffer*, UINT32 *size*)

Send TCP data.

Send data to the supplied address and port.

##### Parameters:

- ← *sock* socket descriptor
- ← *pBuffer* pointer to data to send
- ← *size* size of the data to send

##### Return values:

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS\_IO\_ERR** data could not be sent

#### 5.24.2.19 EXT\_DECL VOS\_ERR\_T vos\_sockSendUDP (INT32 *sock*, const UINT8 \* *pBuffer*, UINT32 *size*, UINT32 *ipAddress*, UINT16 *port*)

Send UDP data.

Send data to the supplied address and port.

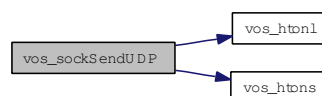
##### Parameters:

- ← *sock* socket descriptor
- ← *pBuffer* pointer to data to send
- ← *size* size of the data to send
- ← *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\_MEM\_ERR** resource error

Here is the call graph for this function:



**5.24.2.20 EXT\_DECL VOS\_ERR\_T vos\_sockSetOptions (INT32 *sock*, const VOS\_SOCK\_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





- EXT\_DECL UINT32 [vos\\_htonl](#) (UINT32 val)  
*Byte swapping 4 Bytes.*
- EXT\_DECL UINT32 [vos\\_ntohl](#) (UINT32 val)  
*Byte swapping 4 Bytes.*
- EXT\_DECL BOOL [vos\\_isMulticast](#) (UINT32 ipAddress)  
*Check if the supplied address is a multicast group address.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockInit](#) (void)  
*Initialize the socket library.*
- 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 size, UINT32 ipAddress, UINT16 port)  
*Send UDP data.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_sockReceiveUDP](#) (INT32 sock, UINT8 \*pBuffer, INT32 \*pSize, UINT32 \*pIPAddr)  
*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 \*pIPAddress, 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 size)  
*Send TCP data.*
- EXT\_DECL [VOS\\_ERR\\_T vos\\_sockReceiveTCP](#) (INT32 sock, UINT8 \*pBuffer, INT32 \*pSize)  
*Receive TCP data.*

### 5.25.1 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](#) 15 2012-06-14 12:52:06Z 97025

### 5.25.2 Function Documentation

#### 5.25.2.1 EXT\_DECL UINT32 vos\_htonl (UINT32 val)

Byte swapping 4 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

#### 5.25.2.2 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

**5.25.2.3 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

**5.25.2.4 EXT\_DECL UINT32 vos\_ntohl (UINT32 *val*)**

Byte swapping 4 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

**5.25.2.5 EXT\_DECL UINT16 vos\_ntohs (UINT16 *val*)**

Byte swapping 2 Bytes.

**Parameters:**

← *val* Initial value.

**Return values:**

*swapped* value

### 5.25.2.6 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.

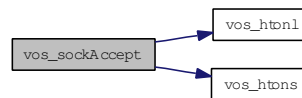
#### Parameters:

- ← *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:



### 5.25.2.7 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\_INIT\_ERR** module not initialised
- VOS\_NOINIT\_ERR** invalid handle
- 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:**

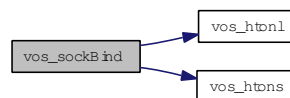
***VOS\_NO\_ERR*** no error

***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:



#### 5.25.2.8 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\_INIT\_ERR*** module not initialised

***VOS\_NOINIT\_ERR*** invalid handle

Release any resources aquired by this socket

**Parameters:**

← *sock* socket descriptor

**Return values:**

***VOS\_NO\_ERR*** no error

***VOS\_PARAM\_ERR*** sock descriptor unknown

#### 5.25.2.9 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\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

**VOS\_PARAM\_ERR** parameter out of range/invalid

**VOS\_IO\_ERR** Input/Output error

**VOS\_MEM\_ERR** resource error

#### Parameters:

← *sock* socket descriptor

← *ipAddress* destination IP

← *port* destination port

#### Return values:

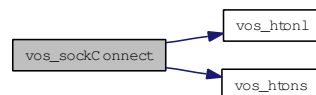
**VOS\_NO\_ERR** no error

**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:



#### 5.25.2.10 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

#### 5.25.2.11 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\_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.

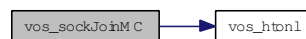
**Parameters:**

- ← *sock* socket descriptor
- ← *mcAddress* multicast group to join
- ← *ipAddress* depicts interface on which to 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:



#### 5.25.2.12 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.

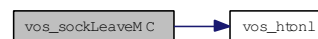
#### Parameters:

← **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:



#### 5.25.2.13 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\_INIT\_ERR** module not initialised  
**VOS\_NOINIT\_ERR** invalid handle  
**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**5.25.2.14 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\_INIT\_ERR* module not initialised*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:

**5.25.2.15 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

Here is the call graph for this function:



### 5.25.2.16 EXT\_DECL VOS\_ERR\_T vos\_sockReceiveTCP (INT32 sock, UINT8 \* pBuffer, INT32 \* 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\_INIT\_ERR** module not initialised

***VOS\_NOINIT\_ERR*** invalid handle  
***VOS\_PARAM\_ERR*** parameter out of range/invalid  
***VOS\_IO\_ERR*** data could not be read  
***VOS\_MEM\_ERR*** resource error  
***VOS\_NODATA\_ERR*** no data in non-blocking

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

#### 5.25.2.17 EXT\_DECL VOS\_ERR\_T vos\_sockReceiveUDP (INT32 *sock*, UINT8 \* *pBuffer*, INT32 \* *pSize*, UINT32 \* *pIPAddr*)

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.

**Parameters:**

← ***sock*** socket descriptor  
 → ***pBuffer*** pointer to applications data buffer  
 ↔ ***pSize*** pointer to the received data size  
 → ***pIPAddr*** source IP

**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\_IO\_ERR** data could not be read

**VOS\_MEM\_ERR** resource error

**VOS\_NODATA\_ERR** no data in non-blocking

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

→ *pIPAddr* source IP

#### 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

Here is the call graph for this function:



#### 5.25.2.18 EXT\_DECL VOS\_ERR\_T vos\_sockSendTCP (INT32 *sock*, const UINT8 \* *pBuffer*, UINT32 *size*)

Send TCP data.

Send data to the given socket.

#### Parameters:

← *sock* socket descriptor

← *pBuffer* pointer to data to send

← *size* size of the data to send

#### 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\_IO\_ERR** data could not be sent

**VOS\_MEM\_ERR** resource error

Send data to the supplied address and port.

**Parameters:**

- ← *sock* socket descriptor
- ← *pBuffer* pointer to data to send
- ← *size* size of the data to send

**Return values:**

- VOS\_NO\_ERR** no error
- VOS\_PARAM\_ERR** sock descriptor unknown, parameter error
- VOS\_IO\_ERR** data could not be sent

**5.25.2.19 EXT\_DECL VOS\_ERR\_T vos\_sockSendUDP (INT32 *sock*, const UINT8 \* *pBuffer*,  
UINT32 *size*, 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
- ← *size* size of the data to send
- ← *ipAddress* destination IP
- ← *port* destination port

**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\_IO\_ERR** data could not be sent
- VOS\_MEM\_ERR** resource error

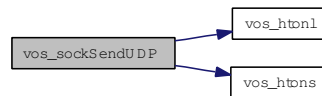
Send data to the supplied address and port.

**Parameters:**

- ← *sock* socket descriptor
- ← *pBuffer* pointer to data to send
- ← *size* size of the data to send
- ← *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\_MEM\_ERR* resource error

Here is the call graph for this function:



### 5.25.2.20 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\_INIT\_ERR* module not initialised*VOS\_NOINIT\_ERR* invalid handle*VOS\_PARAM\_ERR* parameter out of range/invalid*VOS\_SOCK\_ERR* socket not available or option not supported

Note: Some targeted systems might not support every option.

**Parameters:**

← *sock* socket descriptor

← *pOptions* pointer to socket options (optional)

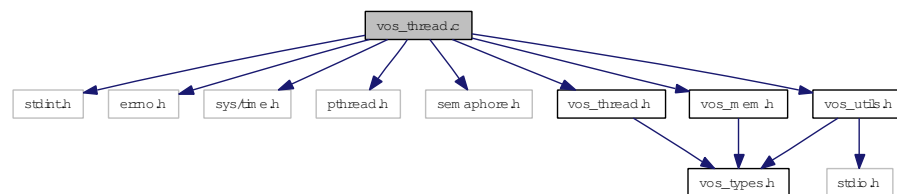
**Return values:***VOS\_NO\_ERR* no error*VOS\_PARAM\_ERR* sock descriptor unknown

## 5.26 vos\_thread.c File Reference

Multitasking functions.

```
#include <stdint.h>
#include <errno.h>
#include <sys/time.h>
#include <pthread.h>
#include <semaphore.h>
#include "vos_thread.h"
#include "vos_mem.h"
#include "vos_utils.h"
```

Include dependency graph for 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 [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 [VOS\\_ERR\\_T](#) [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 VOS\_ERR\_T vos\_clearTime (VOS\_TIME\_T \*pTime)  
*Clear the time stamp.*
- EXT\_DECL VOS\_ERR\_T 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 VOS\_ERR\_T 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 VOS\_ERR\_T 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\_mutexDelete (VOS\_MUTEX\_T mutex)  
*Delete a mutex.*
- EXT\_DECL VOS\_ERR\_T vos\_mutexLock (VOS\_MUTEX\_T mutex)  
*Take a mutex.*
- EXT\_DECL VOS\_ERR\_T vos\_mutexTryLock (VOS\_MUTEX\_T mutex)  
*Try to take a mutex.*
- EXT\_DECL VOS\_ERR\_T vos\_mutexUnlock (VOS\_MUTEX\_T mutex)  
*Release a mutex.*
- EXT\_DECL VOS\_ERR\_T vos\_semaCreate (VOS\_SEMA\_T \*pSema, VOS\_SEMA\_STATE\_T initialState)  
*Create a semaphore.*
- EXT\_DECL VOS\_ERR\_T 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 VOS\_ERR\_T vos\_semaGive (VOS\_SEMA\_T sema)  
*Give a semaphore.*

### 5.26.1 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](#) 15 2012-06-14 12:52:06Z 97025

### 5.26.2 Function Documentation

#### 5.26.2.1 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:



#### 5.26.2.2 EXT\_DECL VOS\_ERR\_T 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

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* parameter must not be NULL

### 5.26.2.3 EXT\_DECL VOS\_ERR\_T vos\_clearTime (VOS\_TIME\_T \* *pTime*)

Clear the time stamp.

**Parameters:**

→ *pTime* Pointer to time value

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* parameter must not be NULL

### 5.26.2.4 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*

### 5.26.2.5 EXT\_DECL VOS\_ERR\_T vos\_getTime (VOS\_TIME\_T \* *pTime*)

Return the current time in sec and us.

**Parameters:**

→ *pTime* Pointer to time value

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* parameter out of range/invalid

### 5.26.2.6 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"

### 5.26.2.7 EXT\_DECL VOS\_ERR\_T 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

#### Return values:

*VOS\_NO\_ERR* no error

*VOS\_INIT\_ERR* module not initialised

Here is the call graph for this function:



### 5.26.2.8 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:

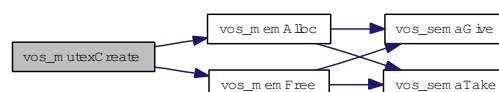
*VOS\_NO\_ERR* no error

*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:



### 5.26.2.9 EXT\_DECL VOS\_ERR\_T vos\_mutexDelete (VOS\_MUTEX\_T mutex)

Delete a mutex.

Release the resources taken by the mutex.

#### Parameters:

← *mutex* mutex handle

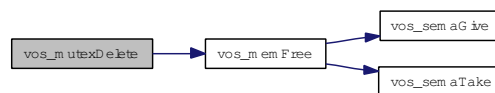
#### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pMutex == NULL or wrong type

**VOS\_MUTEX\_ERR** no such mutex

Here is the call graph for this function:



### 5.26.2.10 EXT\_DECL VOS\_ERR\_T vos\_mutexLock (VOS\_MUTEX\_T mutex)

Take a mutex.

Wait for the mutex to become available (lock).

#### Parameters:

← *mutex* mutex handle

#### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pMutex == NULL or wrong type

**VOS\_MUTEX\_ERR** no such mutex

### 5.26.2.11 EXT\_DECL VOS\_ERR\_T vos\_mutexTryLock (VOS\_MUTEX\_T mutex)

Try to take a mutex.

If mutex is can't be taken VOS\_MUTEX\_ERR is returned.

#### Parameters:

← *mutex* mutex handle

#### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pMutex == NULL or wrong type

**VOS\_MUTEX\_ERR** mutex not locked

#### 5.26.2.12 EXT\_DECL VOS\_ERR\_T vos\_mutexUnlock (VOS\_MUTEX\_T *mutex*)

Release a mutex.

Unlock the mutex.

##### Parameters:

← *mutex* mutex handle

##### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pMutex == NULL or wrong type

**VOS\_MUTEX\_ERR** no such mutex

#### 5.26.2.13 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

#### 5.26.2.14 EXT\_DECL VOS\_ERR\_T vos\_semaDelete (VOS\_SEMA\_T *sema*)

Delete a semaphore.

This will eventually release any processes waiting for the semaphore.

##### Parameters:

← *sema* semaphore handle

##### Return values:

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

**5.26.2.15 EXT\_DECL VOS\_ERR\_T vos\_semaGive (VOS\_SEMA\_T *sema*)**

Give a semaphore.

Release (increase) a semaphore.

**Parameters:**

← *sema* semaphore handle

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_INIT\_ERR* module not initialised

*VOS\_NOINIT\_ERR* invalid handle

*VOS\_SEM\_ERR* could not release semaphore

**5.26.2.16 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 forever

**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

**5.26.2.17 EXT\_DECL VOS\_ERR\_T 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

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* parameter must not be NULL

**5.26.2.18** `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

**5.26.2.19** `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

**5.26.2.20** `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



**5.26.2.21 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

**5.26.2.22 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

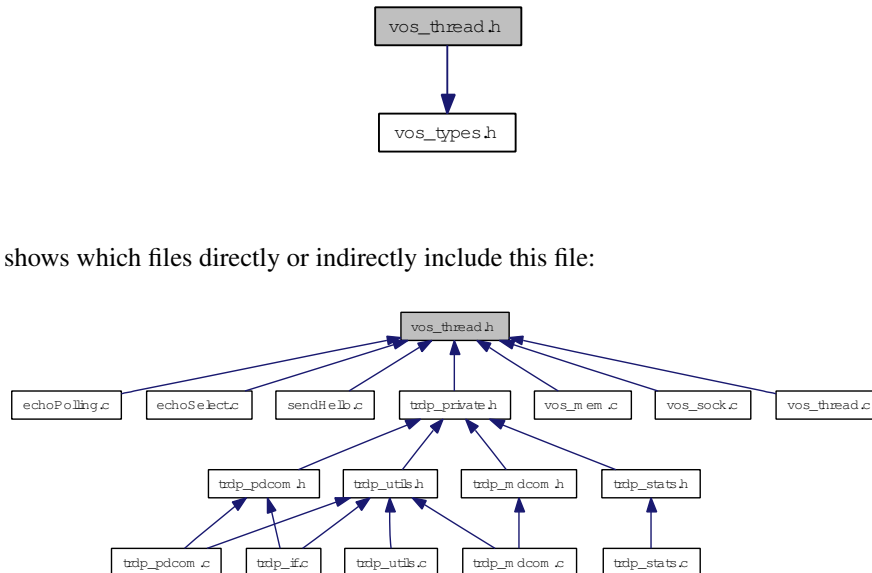
**VOS\_THREAD\_ERR** cancel failed

## 5.27 vos\_thread.h File Reference

Threading functions for OS abstraction.

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

Include dependency graph for vos\_thread.h:



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

## Typedefs

- typedef uint8 **VOS\_THREAD\_PRIORITY\_T**  
Thread priority range from 1 (highest) to 255 (lowest), 0 default of the target system.
- typedef void(\_\_cdecl \* **VOS\_THREAD\_FUNC\_T**)(void \*pArg)  
Thread function definition.
- typedef struct **VOS\_MUTEX\_T** \* **VOS\_MUTEX\_T**  
Hidden mutex handle definition.
- typedef struct **VOS\_SEMA\_T** \* **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 [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 [VOS\\_ERR\\_T](#) [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 [VOS\\_ERR\\_T](#) [vos\\_clearTime](#) ([VOS\\_TIME\\_T](#) \*pTime)  
*Clear the time stamp.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [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 [VOS\\_ERR\\_T](#) [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 [VOS\\_ERR\\_T](#) [vos\\_getUuid](#) ([VOS\\_UUID\\_T](#) pUulD)  
*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 [VOS\\_ERR\\_T](#) [vos\\_mutexDelete](#) ([VOS\\_MUTEX\\_T](#) mutex)  
*Delete a mutex.*

- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_mutexLock](#) ([VOS\\_MUTEX\\_T](#) mutex)  
*Take a mutex.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_mutexTryLock](#) ([VOS\\_MUTEX\\_T](#) mutex)  
*Try to take a mutex.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_mutexUnlock](#) ([VOS\\_MUTEX\\_T](#) mutex)  
*Release a mutex.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [vos\\_semaCreate](#) ([VOS\\_SEMA\\_T](#) \*pSema, [VOS\\_SEMA\\_STATE\\_T](#) initialState)  
*Create a semaphore.*
- EXT\_DECL [VOS\\_ERR\\_T](#) [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 [VOS\\_ERR\\_T](#) [vos\\_semaGive](#) ([VOS\\_SEMA\\_T](#) sema)  
*Give a semaphore.*

### 5.27.1 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](#) 15 2012-06-14 12:52:06Z 97025

### 5.27.2 Function Documentation

#### 5.27.2.1 EXT\_DECL [VOS\\_ERR\\_T](#) [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**Return values:***VOS\_NO\_ERR* no error*VOS\_PARAM\_ERR* parameter must not be NULL**5.27.2.2 EXT\_DECL VOS\_ERR\_T vos\_clearTime (VOS\_TIME\_T \* *pTime*)**

Clear the time stamp.

**Parameters:**→ *pTime* Pointer to time value**Return values:***VOS\_NO\_ERR* no error*VOS\_PARAM\_ERR* parameter must not be NULL**5.27.2.3 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***5.27.2.4 EXT\_DECL VOS\_ERR\_T vos\_getTime (VOS\_TIME\_T \* *pTime*)**

Return the current time in sec and us.

**Parameters:**→ *pTime* Pointer to time value**Return values:***VOS\_NO\_ERR* no error*VOS\_INIT\_ERR* module not initialised

**Parameters:**

→ *pTime* Pointer to time value

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** parameter out of range/invalid

**5.27.2.5 EXT\_DECL const CHAR8\* vos\_getTimeStamp (void)**

Get a time-stamp string.

Get a time-stamp string for debugging in the form "yyymmdd-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* "yyymmdd-hh:mm:ss.ms"

**5.27.2.6 EXT\_DECL VOS\_ERR\_T 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

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

Here is the call graph for this function:

**5.27.2.7 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.

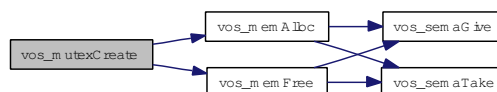
**Parameters:**

→ **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:



### 5.27.2.8 EXT\_DECL VOS\_ERR\_T vos\_mutexDelete (VOS\_MUTEX\_T mutex)

Delete a mutex.

Release the resources taken by the mutex.

**Parameters:**

← **mutex** mutex handle

**Return values:**

**VOS\_NO\_ERR** no error  
**VOS\_INIT\_ERR** module not initialised  
**VOS\_NOINIT\_ERR** invalid handle  
**VOS\_MUTEX\_ERR** no such mutex

Release the resources taken by the mutex.

**Parameters:**

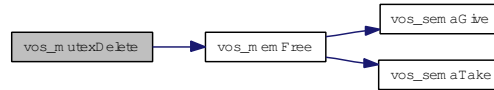
← **mutex** mutex handle

**Return values:**

**VOS\_NO\_ERR** no error  
**VOS\_PARAM\_ERR** pMutex == NULL or wrong type

**VOS\_MUTEX\_ERR** no such mutex

Here is the call graph for this function:



#### 5.27.2.9 EXT\_DECL VOS\_ERR\_T vos\_mutexLock (VOS\_MUTEX\_T mutex)

Take a mutex.

Wait for the mutex to become available (lock).

##### Parameters:

← **mutex** 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:

← **mutex** mutex handle

##### Return values:

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pMutex == NULL or wrong type

**VOS\_MUTEX\_ERR** no such mutex

#### 5.27.2.10 EXT\_DECL VOS\_ERR\_T vos\_mutexTryLock (VOS\_MUTEX\_T mutex)

Try to take a mutex.

If mutex is can't be taken VOS\_MUTEX\_ERR is returned.

##### Parameters:

← **mutex** 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:**

← *mutex* mutex handle

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pMutex == NULL or wrong type

**VOS\_MUTEX\_ERR** mutex not locked

### 5.27.2.11 EXT\_DECL VOS\_ERR\_T vos\_mutexUnlock (VOS\_MUTEX\_T *mutex*)

Release a mutex.

Unlock the mutex.

**Parameters:**

← *mutex* mutex handle

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_INIT\_ERR** module not initialised

**VOS\_NOINIT\_ERR** invalid handle

Unlock the mutex.

**Parameters:**

← *mutex* mutex handle

**Return values:**

**VOS\_NO\_ERR** no error

**VOS\_PARAM\_ERR** pMutex == NULL or wrong type

**VOS\_MUTEX\_ERR** no such mutex

### 5.27.2.12 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

**5.27.2.13 EXT\_DECL VOS\_ERR\_T vos\_semaDelete (VOS\_SEMA\_T *sema*)**

Delete a semaphore.

This will eventually release any processes waiting for the semaphore.

**Parameters:**

← *sema* semaphore handle

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_INIT\_ERR* module not initialised

*VOS\_NOINIT\_ERR* invalid handle

**5.27.2.14 EXT\_DECL VOS\_ERR\_T vos\_semaGive (VOS\_SEMA\_T *sema*)**

Give a semaphore.

Release (increase) a semaphore.

**Parameters:**

← *sema* semaphore handle

**Return values:**

*VOS\_NO\_ERR* no error

*VOS\_INIT\_ERR* module not initialised

*VOS\_NOINIT\_ERR* invalid handle

*VOS\_SEM\_ERR* could not release semaphore

**5.27.2.15 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 forever

#### 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

#### 5.27.2.16 EXT\_DECL VOS\_ERR\_T 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

#### Return values:

*VOS\_NO\_ERR* no error

*VOS\_PARAM\_ERR* parameter must not be NULL

#### 5.27.2.17 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\_INIT\_ERR*** no threads available

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

#### 5.27.2.18 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

### 5.27.2.19 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

### 5.27.2.20 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

### 5.27.2.21 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:**

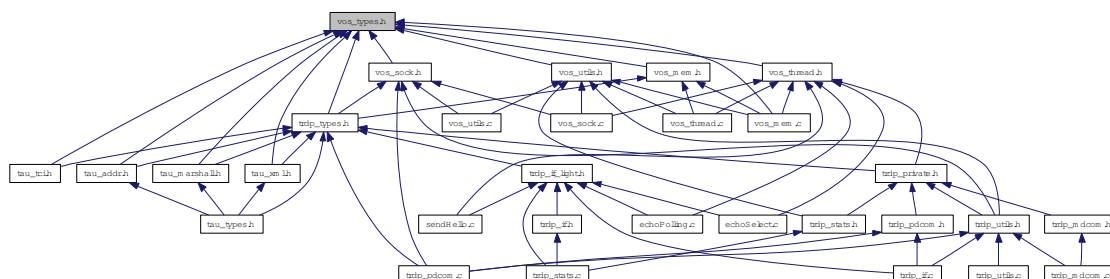
*VOS\_NO\_ERR* no error

*VOS\_THREAD\_ERR* cancel failed

## 5.28 vos\_types.h File Reference

Typedefs for OS abstraction.

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



## Data Structures

- struct `VOS_TIME_T`  
*Timer value compatible with timeval / select.*

## 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\_SOCKET\_ERR = -6,  
    VOS\_IO\_ERR = -7,  
    VOS\_MEM\_ERR = -8,  
    VOS\_SEMA\_ERR = -9,  
    VOS\_QUEUE\_ERR = -10,  
};

```

VOS_QUEUE_FULL_ERR = -11,
VOS_MUTEX_ERR = -12,
VOS_THREAD_ERR = -13,
VOS_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.*

## Functions

- EXT\_DECL `VOS_ERR_T vos_init` (void \*pRefCon, `VOS_PRINT_DBG_T` pDebugOutput)

*Initialize the vos library.*

### 5.28.1 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](#) 15 2012-06-14 12:52:06Z 97025

### 5.28.2 Typedef Documentation

#### 5.28.2.1 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*

### 5.28.3 Enumeration Type Documentation

#### 5.28.3.1 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\_UNKNOWN\_ERR** Unknown error.

#### 5.28.3.2 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.

## 5.28.4 Function Documentation

### 5.28.4.1 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

Here is the call graph for this function:



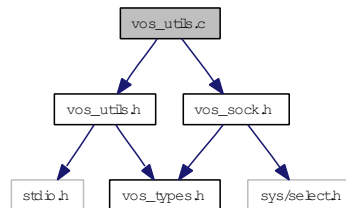
## 5.29 vos\_utils.c File Reference

Common functions for VOS.

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

```
#include "vos_sock.h"
```

Include dependency graph for vos\_utils.c:



### Functions

- **VOS\_ERR\_T vos\_init** (void \*pRefCon, **VOS\_PRINT\_DBG\_T** pDebugOutput)  
*Initialize the vos library.*
- **UINT32 vos\_crc32** (UINT32 crc, const **UINT8** \*pData, **UINT32** dataLen)  
*Compute crc32 according to IEEE802.3.*

### 5.29.1 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](#) 2 2012-06-04 11:25:16Z 97025

### 5.29.2 Function Documentation

#### 5.29.2.1 **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

### 5.29.2.2 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

Here is the call graph for this function:

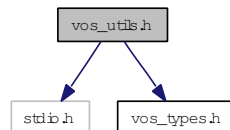


## 5.30 vos\_utils.h File Reference

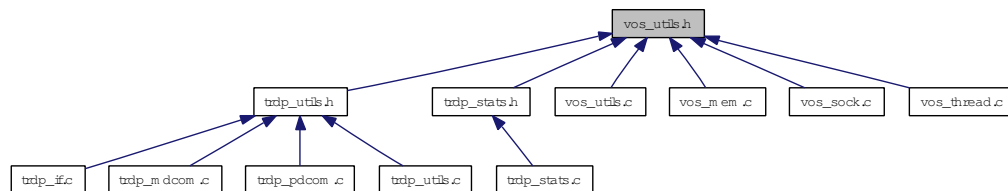
Typedefs for OS abstraction.

```
#include <stdio.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\_print**(level, string)  
*Debug output macro without formatting options.*
- #define **vos\_printf**(level, format, args...)  
*Debug output macro with formatting options.*

### Functions

- EXT\_DECL UINT32 **vos\_crc32** (UINT32 crc, const UINT8 \*pData, UINT32 dataLen)  
*Calculate CRC for the given buffer and length.*

#### 5.30.1 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](#) 15 2012-06-14 12:52:06Z 97025

**5.30.2 Function Documentation****5.30.2.1 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

# Index

- am\_big\_endian
  - trdp\_utils.c, [174](#)
  - trdp\_utils.h, [179](#)
- cyclicThread
  - vos\_thread.c, [226](#)
- datasetLength
  - GNU\_PACKED, [10](#)
- dbgOut
  - echoPolling.c, [58](#)
  - echoSelect.c, [62](#)
- destAddr
  - TRDP\_PUB\_STATISTICS\_T, [43](#)
- echoPolling.c, [57](#)
  - dbgOut, [58](#)
  - main, [58](#)
- echoSelect.c, [61](#)
  - dbgOut, [62](#)
  - main, [62](#)
  - myPDcallBack, [64](#)
- filterAddr
  - TRDP\_SUBS\_STATISTICS\_T, [51](#)
- GNU\_PACKED, [9](#)
  - datasetLength, [10](#)
  - msgType, [10](#)
  - protocolVersion, [10](#)
- main
  - echoPolling.c, [58](#)
  - echoSelect.c, [62](#)
  - sendHello.c, [66](#)
- MD\_ELE, [11](#)
- msgType
  - GNU\_PACKED, [10](#)
  - TRDP\_MD\_INFO\_T, [29](#)
  - TRDP\_PD\_INFO\_T, [37](#)
- myPDcallBack
  - echoSelect.c, [64](#)
- numRecv
  - TRDP\_SUBS\_STATISTICS\_T, [52](#)
- operator
  - TRDP\_TRAIN\_INFO\_T, [54](#)
- orient
  - TRDP\_CAR\_INFO\_T, [15](#)
  - TRDP\_CST\_INFO\_T, [17](#)
  - TRDP\_DEVICE\_INFO\_T, [22](#)
- owner
  - TRDP\_CST\_INFO\_T, [17](#)
- pCarInfo
  - TRDP\_CST\_INFO\_T, [17](#)
- pCstInfo
  - TRDP\_TRAIN\_INFO\_T, [54](#)
- PD\_ELE, [12](#)
- pDevInfo
  - TRDP\_CAR\_INFO\_T, [15](#)
- pFctInfo
  - TRDP\_CST\_INFO\_T, [17](#)
- protocolVersion
  - GNU\_PACKED, [10](#)
- qos
  - VOS SOCK\_OPT\_T, [55](#)
- sendHello.c, [65](#)
  - main, [66](#)
- tau\_tci.h
  - TRDP\_FCT\_CAR, [84](#)
  - TRDP\_FCT\_CST, [84](#)
  - TRDP\_FCT\_INVALID, [84](#)
  - TRDP\_FCT\_TRAIN, [84](#)
  - TRDP\_INAUG\_INVALID, [85](#)
  - TRDP\_INAUG\_LEAD\_CONF, [85](#)
  - TRDP\_INAUG\_LEAD\_UNCONF, [85](#)
  - TRDP\_INAUG\_NOLEAD\_UNCONF, [85](#)
- tau\_xml.h
  - TRDP\_DBG\_CAT, [93](#)
  - TRDP\_DBG\_DBG, [93](#)
  - TRDP\_DBG\_DEFAULT, [92](#)
  - TRDP\_DBG\_ERR, [93](#)
  - TRDP\_DBG\_INFO, [93](#)
  - TRDP\_DBG\_LOC, [93](#)
  - TRDP\_DBG\_OFF, [92](#)
  - TRDP\_DBG\_TIME, [93](#)
  - TRDP\_DBG\_WARN, [93](#)
- tau\_addr.h, [68](#)

- tau\_addr2CarId, 70
- tau\_addr2CarNo, 70
- tau\_addr2CstId, 71
- tau\_addr2CstNo, 71
- tau\_addr2IecCarNo, 71
- tau\_addr2IecCstNo, 72
- tau\_addr2Uri, 72
- tau\_carNo2Ids, 72
- tau\_cstNo2CstId, 73
- tau\_getOwnAddr, 73
- tau\_getOwnIds, 73
- tau\_iecCarNo2Ids, 74
- tau\_iecCstNo2CstId, 74
- tau\_label2CarId, 74
- tau\_label2CarNo, 75
- tau\_label2CstId, 75
- tau\_label2CstNo, 75
- tau\_label2IecCarNo, 76
- tau\_label2IecCstNo, 76
- tau\_uri2Addr, 76
- tau\_addr2CarId
  - tau\_addr.h, 70
- tau\_addr2CarNo
  - tau\_addr.h, 70
- tau\_addr2CstId
  - tau\_addr.h, 71
- tau\_addr2CstNo
  - tau\_addr.h, 71
- tau\_addr2IecCarNo
  - tau\_addr.h, 71
- tau\_addr2IecCstNo
  - tau\_addr.h, 72
- tau\_addr2Uri
  - tau\_addr.h, 72
- tau\_calcDatasetSize
  - tau\_marshall.h, 79
- tau\_carNo2Ids
  - tau\_addr.h, 72
- tau\_cstNo2CstId
  - tau\_addr.h, 73
- tau\_getCarDevCnt
  - tau\_tci.h, 85
- tau\_getCarInfo
  - tau\_tci.h, 85
- tau\_getCarOrient
  - tau\_tci.h, 86
- tau\_getCstCarCnt
  - tau\_tci.h, 86
- tau\_getCstFctCnt
  - tau\_tci.h, 86
- tau\_getCstFctInfo
  - tau\_tci.h, 87
- tau\_getCstInfo
  - tau\_tci.h, 87
- tau\_getDevInfo
  - tau\_tci.h, 87
- tau\_getEtbState
  - tau\_tci.h, 88
- tau\_getIecCarOrient
  - tau\_tci.h, 88
- tau\_getOwnAddr
  - tau\_addr.h, 73
- tau\_getOwnIds
  - tau\_addr.h, 73
- tau\_getTrnCarCnt
  - tau\_tci.h, 89
- tau\_getTrnCstCnt
  - tau\_tci.h, 89
- tau\_getTrnInfo
  - tau\_tci.h, 89
- tau\_iecCarNo2Ids
  - tau\_addr.h, 74
- tau\_iecCstNo2CstId
  - tau\_addr.h, 74
- tau\_initMarshall
  - tau\_marshall.h, 81
- tau\_label2CarId
  - tau\_addr.h, 74
- tau\_label2CarNo
  - tau\_addr.h, 75
- tau\_label2CstId
  - tau\_addr.h, 75
- tau\_label2CstNo
  - tau\_addr.h, 75
- tau\_label2IecCarNo
  - tau\_addr.h, 76
- tau\_label2IecCstNo
  - tau\_addr.h, 76
- tau\_marshall
  - tau\_marshall.h, 79
- tau\_marshall.h, 78
  - tau\_calcDatasetSize, 79
  - tau\_initMarshall, 81
  - tau\_marshall, 79
  - tau\_marshallIDs, 80
  - tau\_unmarshall, 80
  - tau\_unmarshallIDs, 81
- tau\_marshallIDs
  - tau\_marshall.h, 80
- tau\_readXmlConfig
  - tau\_xml.h, 93
- tau\_readXmlDatasetConfig
  - tau\_xml.h, 93
- tau\_tci.h, 82
  - tau\_getCarDevCnt, 85
  - tau\_getCarInfo, 85
  - tau\_getCarOrient, 86
  - tau\_getCstCarCnt, 86



- tau\_getCstFctCnt, 86
- tau\_getCstFctInfo, 87
- tau\_getCstInfo, 87
- tau\_getDevInfo, 87
- tau\_getEtbState, 88
- tau\_getIecCarOrient, 88
- tau\_getTrnCarCnt, 89
- tau\_getTrnCstCnt, 89
- tau\_getTrnInfo, 89
- TRDP\_FCT\_T, 84
- TRDP\_INAUG\_STATE\_T, 84
- tau\_types.h, 90
- tau\_unmarshall
  - tau\_marshall.h, 80
- tau\_unmarshallDs
  - tau\_marshall.h, 81
- tau\_uri2Addr
  - tau\_addr.h, 76
- tau\_xml.h, 91
  - tau\_readXmlConfig, 93
  - tau\_readXmlDatasetConfig, 93
  - TRDP\_DBG\_OPTION\_T, 92
- timeout
  - TRDP\_SUBS\_STATISTICS\_T, 51
- tlc\_freeBuf
  - trdp\_if\_light.h, 115
- tlc\_getInterval
  - trdp\_if.c, 97
  - trdp\_if\_light.h, 115
- tlc\_getJoinStatistics
  - trdp\_if\_light.h, 116
  - trdp\_stats.c, 158
- tlc\_getListStatistics
  - trdp\_if\_light.h, 116
  - trdp\_stats.c, 158
- tlc\_getPubStatistics
  - trdp\_if\_light.h, 117
  - trdp\_stats.c, 159
- tlc\_getRedStatistics
  - trdp\_if\_light.h, 117
  - trdp\_stats.c, 159
- tlc\_getStatistics
  - trdp\_if\_light.h, 118
  - trdp\_stats.c, 160
- tlc\_getSubsStatistics
  - trdp\_if\_light.h, 118
  - trdp\_stats.c, 160
- tlc\_getVersion
  - trdp\_if.c, 97
  - trdp\_if\_light.h, 119
- tlc\_init
  - trdp\_if.c, 98
  - trdp\_if\_light.h, 119
- tlc\_process
  - trdp\_if.c, 99
  - trdp\_if\_light.h, 120
- tlc\_reinit
  - trdp\_if.c, 100
  - trdp\_if\_light.h, 121
- tlc\_resetStatistics
  - trdp\_if\_light.h, 122
  - trdp\_stats.c, 161
- tlc\_setTopoCount
  - trdp\_if.c, 100
  - trdp\_if\_light.h, 122
- tlc\_terminate
  - trdp\_if.c, 101
  - trdp\_if\_light.h, 123
- tlm\_abortSession
  - trdp\_if\_light.h, 123
- tlm\_addListener
  - trdp\_if\_light.h, 124
- tlm\_confirm
  - trdp\_if\_light.h, 124
- tlm\_delListener
  - trdp\_if\_light.h, 125
- tlm\_notify
  - trdp\_if\_light.h, 125
- tlm\_reply
  - trdp\_if\_light.h, 126
- tlm\_replyErr
  - trdp\_if\_light.h, 127
- tlm\_replyQuery
  - trdp\_if\_light.h, 127
- tlm\_request
  - trdp\_if\_light.h, 128
- tlp\_get
  - trdp\_if.c, 101
  - trdp\_if\_light.h, 129
- tlp\_getRedundant
  - trdp\_if.c, 102
  - trdp\_if\_light.h, 131
- tlp\_publish
  - trdp\_if.c, 103
  - trdp\_if\_light.h, 131
- tlp\_put
  - trdp\_if.c, 104
  - trdp\_if\_light.h, 133
- tlp\_request
  - trdp\_if\_light.h, 134
- tlp\_setRedundant
  - trdp\_if.c, 105
  - trdp\_if\_light.h, 135
- tlp\_subscribe
  - trdp\_if.c, 105
  - trdp\_if\_light.h, 135
- tlp\_unpublish
  - trdp\_if.c, 106

- trdp\_if\_light.h, 137
- tlp\_unsubscribe
  - trdp\_if.c, 107
  - trdp\_if\_light.h, 137
- toBehav
  - TRDP\_SUBS\_STATISTICS\_T, 51
- topoCnt
  - TRDP\_TRAIN\_INFO\_T, 54
- TRDP\_ARRAY
  - trdp\_types.h, 171
- TRDP\_BOOLEAN
  - trdp\_types.h, 170
- TRDP\_CHAR8
  - trdp\_types.h, 170
- TRDP\_COMID\_ERR
  - trdp\_types.h, 171
- TRDP\_CRC\_ERR
  - trdp\_types.h, 171
- TRDP\_DBG\_CAT
  - tau\_xml.h, 93
- TRDP\_DBG\_DBG
  - tau\_xml.h, 93
- TRDP\_DBG\_DEFAULT
  - tau\_xml.h, 92
- TRDP\_DBG\_ERR
  - tau\_xml.h, 93
- TRDP\_DBG\_INFO
  - tau\_xml.h, 93
- TRDP\_DBG\_LOC
  - tau\_xml.h, 93
- TRDP\_DBG\_OFF
  - tau\_xml.h, 92
- TRDP\_DBG\_TIME
  - tau\_xml.h, 93
- TRDP\_DBG\_WARN
  - tau\_xml.h, 93
- TRDP\_FCT\_CAR
  - tau\_tci.h, 84
- TRDP\_FCT\_CST
  - tau\_tci.h, 84
- TRDP\_FCT\_INVALID
  - tau\_tci.h, 84
- TRDP\_FCT\_TRAIN
  - tau\_tci.h, 84
- TRDP\_FLAGS\_CALLBACK
  - trdp\_types.h, 172
- TRDP\_FLAGS\_MARSHALL
  - trdp\_types.h, 172
- TRDP\_FLAGS\_REDUNDANT
  - trdp\_types.h, 172
- TRDP\_FLAGS\_TCP
  - trdp\_types.h, 172
- TRDP\_INAUG\_INVALID
  - tau\_tci.h, 85
- TRDP\_INAUG\_LEAD\_CONF
  - tau\_tci.h, 85
- TRDP\_INAUG\_LEAD\_UNCONF
  - tau\_tci.h, 85
- TRDP\_INAUG\_NOLEAD\_UNCONF
  - tau\_tci.h, 85
- TRDP\_INIT\_ERR
  - trdp\_types.h, 171
- TRDP\_INT16
  - trdp\_types.h, 170
- TRDP\_INT32
  - trdp\_types.h, 170
- TRDP\_INT64
  - trdp\_types.h, 170
- TRDP\_INT8
  - trdp\_types.h, 170
- TRDP\_IO\_ERR
  - trdp\_types.h, 171
- TRDP\_MEM\_ERR
  - trdp\_types.h, 171
- TRDP\_MSG\_MC
  - trdp\_types.h, 172
- TRDP\_MSG\_ME
  - trdp\_types.h, 172
- TRDP\_MSG\_MN
  - trdp\_types.h, 172
- TRDP\_MSG\_MP
  - trdp\_types.h, 172
- TRDP\_MSG\_MQ
  - trdp\_types.h, 172
- TRDP\_MSG\_MR
  - trdp\_types.h, 172
- TRDP\_MSG\_PD
  - trdp\_types.h, 172
- TRDP\_MSG\_PE
  - trdp\_types.h, 172
- TRDP\_MSG\_PR
  - trdp\_types.h, 172
- TRDP\_Mutex\_ERR
  - trdp\_types.h, 171
- TRDP\_NO\_ERR
  - trdp\_types.h, 171
- TRDP\_NODATA\_ERR
  - trdp\_types.h, 171
- TRDP\_NOINIT\_ERR
  - trdp\_types.h, 171
- TRDP\_NOLIST\_ERR
  - trdp\_types.h, 171
- TRDP\_NOPUB\_ERR
  - trdp\_types.h, 171
- TRDP\_NOSESSION\_ERR
  - trdp\_types.h, 171
- TRDP\_NOSUB\_ERR
  - trdp\_types.h, 171

- TRDP\_OPTION\_BLOCK
  - trdp\_types.h, 172
- TRDP\_OPTION\_TRAFFIC\_SHAPING
  - trdp\_types.h, 172
- TRDP\_PARAM\_ERR
  - trdp\_types.h, 171
- trdp\_private.h
  - TRDP\_SOCKET\_MD\_TCP, 156
  - TRDP\_SOCKET\_MD\_UDP, 156
  - TRDP\_SOCKET\_PD, 156
  - TRDP\_TIMED\_OUT, 156
- TRDP\_QUEUE\_ERR
  - trdp\_types.h, 171
- TRDP\_QUEUE\_FULL\_ERR
  - trdp\_types.h, 171
- TRDP\_REAL32
  - trdp\_types.h, 170
- TRDP\_REAL64
  - trdp\_types.h, 170
- TRDP\_RECORD
  - trdp\_types.h, 171
- TRDP\_RED\_FOLLOWER
  - trdp\_types.h, 172
- TRDP\_RED\_LEADER
  - trdp\_types.h, 172
- TRDP\_SEMA\_ERR
  - trdp\_types.h, 171
- TRDP\_SESSION\_ABORT\_ERR
  - trdp\_types.h, 171
- TRDP\_SOCKET\_ERR
  - trdp\_types.h, 171
- TRDP\_SOCKET\_MD\_TCP
  - trdp\_private.h, 156
- TRDP\_SOCKET\_MD\_UDP
  - trdp\_private.h, 156
- TRDP\_SOCKET\_PD
  - trdp\_private.h, 156
- TRDP\_STATE\_ERR
  - trdp\_types.h, 171
- TRDP\_STRING
  - trdp\_types.h, 171
- TRDP\_TIMED\_OUT
  - trdp\_private.h, 156
- TRDP\_TIMESTAMP32
  - trdp\_types.h, 171
- TRDP\_TIMESTAMP48
  - trdp\_types.h, 171
- TRDP\_TIMESTAMP64
  - trdp\_types.h, 171
- TRDP\_TIMEOUT\_ERR
  - trdp\_types.h, 171
- TRDP\_TOPO\_ERR
  - trdp\_types.h, 171
- trdp\_types.h
  - TRDP\_ARRAY, 171
  - TRDP\_BOOLEAN, 170
  - TRDP\_CHAR8, 170
  - TRDP\_COMID\_ERR, 171
  - TRDP\_CRC\_ERR, 171
  - TRDP\_FLAGS\_CALLBACK, 172
  - TRDP\_FLAGS\_MARSHALL, 172
  - TRDP\_FLAGS\_REDUNDANT, 172
  - TRDP\_FLAGS\_TCP, 172
  - TRDP\_INIT\_ERR, 171
  - TRDP\_INT16, 170
  - TRDP\_INT32, 170
  - TRDP\_INT64, 170
  - TRDP\_INT8, 170
  - TRDP\_IO\_ERR, 171
  - TRDP\_MEM\_ERR, 171
  - TRDP\_MSG\_MC, 172
  - TRDP\_MSG\_ME, 172
  - TRDP\_MSG\_MN, 172
  - TRDP\_MSG\_MP, 172
  - TRDP\_MSG\_MQ, 172
  - TRDP\_MSG\_MR, 172
  - TRDP\_MSG\_PD, 172
  - TRDP\_MSG\_PE, 172
  - TRDP\_MSG\_PR, 172
  - TRDP\_MUTEX\_ERR, 171
  - TRDP\_NO\_ERR, 171
  - TRDP\_NODATA\_ERR, 171
  - TRDP\_NOINIT\_ERR, 171
  - TRDP\_NOLIST\_ERR, 171
  - TRDP\_NOPUB\_ERR, 171
  - TRDP\_NOSESSION\_ERR, 171
  - TRDP\_NOSUB\_ERR, 171
  - TRDP\_OPTION\_BLOCK, 172
  - TRDP\_OPTION\_TRAFFIC\_SHAPING, 172
  - TRDP\_PARAM\_ERR, 171
  - TRDP\_QUEUE\_ERR, 171
  - TRDP\_QUEUE\_FULL\_ERR, 171
  - TRDP\_REAL32, 170
  - TRDP\_REAL64, 170
  - TRDP\_RECORD, 171
  - TRDP\_RED\_FOLLOWER, 172
  - TRDP\_RED\_LEADER, 172
  - TRDP\_SEMA\_ERR, 171
  - TRDP\_SESSION\_ABORT\_ERR, 171
  - TRDP\_SOCKET\_ERR, 171
  - TRDP\_STATE\_ERR, 171
  - TRDP\_STRING, 171
  - TRDP\_TIMESTAMP32, 171
  - TRDP\_TIMESTAMP48, 171
  - TRDP\_TIMESTAMP64, 171
  - TRDP\_TIMEOUT\_ERR, 171
  - TRDP\_TOPO\_ERR, 171
  - TRDP\_UINT16, 170

- TRDP\_UINT32, 170
- TRDP\_UINT64, 170
- TRDP\_UINT8, 170
- TRDP\_UNKNOWN\_ERR, 171
- TRDP\_UTF16, 170
- TRDP\_UINT16
  - trdp\_types.h, 170
- TRDP\_UINT32
  - trdp\_types.h, 170
- TRDP\_UINT64
  - trdp\_types.h, 170
- TRDP\_UINT8
  - trdp\_types.h, 170
- TRDP\_UNKNOWN\_ERR
  - trdp\_types.h, 171
- TRDP\_UTF16
  - trdp\_types.h, 170
- TRDP\_CAR\_INFO\_T, 14
  - orient, 15
  - pDevInfo, 15
- TRDP\_CST\_INFO\_T, 16
  - orient, 17
  - owner, 17
  - pCarInfo, 17
  - pFctInfo, 17
- TRDP\_DATA\_TYPE\_T
  - trdp\_types.h, 170
- TRDP\_DATASET\_ELEMENT\_T, 18
- TRDP\_DATASET\_T, 19
- TRDP\_DBG\_CONFIG\_T, 20
- TRDP\_DBG\_OPTION\_T
  - tau\_xml.h, 92
- TRDP\_DEVICE\_INFO\_T, 21
  - orient, 22
- TRDP\_ERR\_T
  - trdp\_types.h, 171
- TRDP\_FCT\_INFO\_T, 23
- TRDP\_FCT\_T
  - tau\_tci.h, 84
- TRDP\_FLAGS\_T
  - trdp\_types.h, 171
- TRDP\_HANDLE, 24
- trdp\_if.c, 95
  - tlc\_getInterval, 97
  - tlc\_getVersion, 97
  - tlc\_init, 98
  - tlc\_process, 99
  - tlc\_reinit, 100
  - tlc\_setTopoCount, 100
  - tlc\_terminate, 101
  - tlp\_get, 101
  - tlp\_getRedundant, 102
  - tlp\_publish, 103
  - tlp\_put, 104
  - tlp\_setRedundant, 105
  - tlp\_subscribe, 105
  - tlp\_unpublish, 106
  - tlp\_unsubscribe, 107
  - trdp\_isValidSession, 108
  - trdp\_sessionQueue, 108
- trdp\_if.h, 109
  - trdp\_isValidSession, 110
  - trdp\_sessionQueue, 110
- trdp\_if\_light.h, 111
  - tlc\_freeBuf, 115
  - tlc\_getInterval, 115
  - tlc\_getJoinStatistics, 116
  - tlc\_getListStatistics, 116
  - tlc\_getPubStatistics, 117
  - tlc\_getRedStatistics, 117
  - tlc\_getStatistics, 118
  - tlc\_getSubsStatistics, 118
  - tlc\_getVersion, 119
  - tlc\_init, 119
  - tlc\_process, 120
  - tlc\_reinit, 121
  - tlc\_resetStatistics, 122
  - tlc\_setTopoCount, 122
  - tlc\_terminate, 123
  - tlm\_abortSession, 123
  - tlm\_addListener, 124
  - tlm\_confirm, 124
  - tlm\_delListener, 125
  - tlm\_notify, 125
  - tlm\_reply, 126
  - tlm\_replyErr, 127
  - tlm\_replyQuery, 127
  - tlm\_request, 128
  - tlp\_get, 129
  - tlp\_getRedundant, 131
  - tlp\_publish, 131
  - tlp\_put, 133
  - tlp\_request, 134
  - tlp\_setRedundant, 135
  - tlp\_subscribe, 135
  - tlp\_unpublish, 137
  - tlp\_unsubscribe, 137
- TRDP\_INAUG\_STATE\_T
  - tau\_tci.h, 84
- trdp\_initSockets
  - trdp\_utils.c, 174
  - trdp\_utils.h, 179
- TRDP\_IP\_ADDR\_T
  - trdp\_types.h, 168
- trdp\_isValidSession
  - trdp\_if.c, 108
  - trdp\_if.h, 110
- TRDP\_LIST\_STATISTICS\_T, 25

- TRDP\_MARSHALL\_CONFIG\_T, 26
- TRDP\_MARSHALL\_T
  - trdp\_types.h, 168
- TRDP\_MAX\_FILE\_NAME\_LEN
  - trdp\_types.h, 168
- TRDP\_MAX\_LABEL\_LEN
  - trdp\_types.h, 168
- TRDP\_MAX\_URI\_HOST\_LEN
  - trdp\_types.h, 168
- TRDP\_MAX\_URI\_LEN
  - trdp\_types.h, 168
- TRDP\_MAX\_URI\_USER\_LEN
  - trdp\_types.h, 168
- TRDP\_MD\_CALLBACK\_T
  - trdp\_types.h, 169
- TRDP\_MD\_CONFIG\_T, 27
- TRDP\_MD\_INFO\_T, 28
  - msgType, 29
- TRDP\_MD\_STATISTICS, 30
- TRDP\_MD\_STATISTICS\_T, 31
- trdp\_mdcom.c, 139
  - trdp\_rcvMD, 140
  - trdp\_sendMD, 140
- trdp\_mdcom.h, 141
  - trdp\_rcvMD, 142
  - trdp\_sendMD, 142
- TRDP\_MEM\_CONFIG\_T, 33
- TRDP\_MEM\_STATISTICS\_T, 34
- TRDP\_MSG\_T
  - trdp\_types.h, 172
- TRDP\_OPTION\_T
  - trdp\_types.h, 172
- trdp\_packetSizePD
  - trdp\_utils.c, 174
  - trdp\_utils.h, 180
- TRDP\_PD\_CALLBACK\_T
  - trdp\_types.h, 169
- TRDP\_PD\_CONFIG\_T, 35
- TRDP\_PD\_INFO\_T, 36
  - msgType, 37
- TRDP\_PD\_STATISTICS, 38
- TRDP\_PD\_STATISTICS\_T, 39
- trdp\_pdCheck
  - trdp\_pdcom.c, 144
  - trdp\_pdcom.h, 149
- trdp\_pdcom.c, 143
  - trdp\_pdCheck, 144
  - trdp\_pdInit, 144
  - trdp\_pdReceive, 145
  - trdp\_pdSend, 146
  - trdp\_pdUpdate, 146
- trdp\_pdcom.h, 148
  - trdp\_pdCheck, 149
  - trdp\_pdInit, 149
  - trdp\_pdReceive, 150
  - trdp\_pdSend, 151
  - trdp\_pdUpdate, 151
- trdp\_pdInit
  - trdp\_pdcom.c, 144
  - trdp\_pdcom.h, 149
- trdp\_pdReceive
  - trdp\_pdcom.c, 145
  - trdp\_pdcom.h, 150
- trdp\_pdSend
  - trdp\_pdcom.c, 146
  - trdp\_pdcom.h, 151
- trdp\_pdUpdate
  - trdp\_pdcom.c, 146
  - trdp\_pdcom.h, 151
- TRDP\_PRINT\_DBG\_T
  - trdp\_types.h, 169
- TRDP\_PRIV\_FLAGS\_T
  - trdp\_private.h, 156
- trdp\_private.h, 153
  - TRDP\_PRIV\_FLAGS\_T, 156
  - TRDP SOCK\_TYPE\_T, 156
- TRDP\_PROCESS\_CONFIG\_T, 41
- TRDP\_PROP\_INFO\_T, 42
- TRDP\_PUB\_STATISTICS\_T, 43
  - destAddr, 43
- trdp\_queueAppLast
  - trdp\_utils.c, 175
  - trdp\_utils.h, 180
- trdp\_queueDelElement
  - trdp\_utils.c, 175
  - trdp\_utils.h, 180
- trdp\_queueFindAddr
  - trdp\_utils.c, 175
  - trdp\_utils.h, 180
- trdp\_queueFindComId
  - trdp\_utils.c, 175
  - trdp\_utils.h, 180
- trdp\_queueInsFirst
  - trdp\_utils.c, 176
  - trdp\_utils.h, 181
- trdp\_rcvMD
  - trdp\_mdcom.c, 140
  - trdp\_mdcom.h, 142
- TRDP\_RED\_STATE\_T
  - trdp\_types.h, 172
- TRDP\_RED\_STATISTICS\_T, 44
- trdp\_releaseSocket
  - trdp\_utils.c, 176
  - trdp\_utils.h, 181
- trdp\_requestSocket
  - trdp\_utils.c, 176
  - trdp\_utils.h, 181
- TRDP\_SEND\_PARAM\_T, 45

- trdp\_sendMD
  - trdp\_mdcom.c, 140
  - trdp\_mdcom.h, 142
- TRDP\_SESSION, 46
- trdp\_sessionQueue
  - trdp\_if.c, 108
  - trdp\_if.h, 110
- TRDP SOCK\_TYPE\_T
  - trdp\_private.h, 156
- TRDP\_SOCKETS, 48
  - usage, 48
- TRDP\_STATISTICS\_T, 49
- trdp\_stats.c, 157
  - tlc\_getJoinStatistics, 158
  - tlc\_getListStatistics, 158
  - tlc\_getPubStatistics, 159
  - tlc\_getRedStatistics, 159
  - tlc\_getStatistics, 160
  - tlc\_getSubsStatistics, 160
  - tlc\_resetStatistics, 161
- trdp\_stats.h, 162
- TRDP\_SUBS\_STATISTICS\_T, 51
  - filterAddr, 51
  - numRecv, 52
  - timeout, 51
  - toBehav, 51
- TRDP\_TIME\_T
  - trdp\_types.h, 169
- TRDP\_TRAIN\_INFO\_T, 53
  - operator, 54
  - pCstInfo, 54
  - topoCnt, 54
- trdp\_types.h, 163
  - TRDP\_DATA\_TYPE\_T, 170
  - TRDP\_ERR\_T, 171
  - TRDP\_FLAGS\_T, 171
  - TRDP\_IP\_ADDR\_T, 168
  - TRDP\_MARSHALL\_T, 168
  - TRDP\_MAX\_FILE\_NAME\_LEN, 168
  - TRDP\_MAX\_LABEL\_LEN, 168
  - TRDP\_MAX\_URI\_HOST\_LEN, 168
  - TRDP\_MAX\_URI\_LEN, 168
  - TRDP\_MAX\_URI\_USER\_LEN, 168
  - TRDP\_MD\_CALLBACK\_T, 169
  - TRDP\_MSG\_T, 172
  - TRDP\_OPTION\_T, 172
  - TRDP\_PD\_CALLBACK\_T, 169
  - TRDP\_PRINT\_DBG\_T, 169
  - TRDP\_RED\_STATE\_T, 172
  - TRDP\_TIME\_T, 169
  - TRDP\_UNMARSHALL\_T, 170
- TRDP\_UNMARSHALL\_T
  - trdp\_types.h, 170
- trdp\_utils.c, 173
  - am\_big\_endian, 174
  - trdp\_initSockets, 174
  - trdp\_packetSizePD, 174
  - trdp\_queueAppLast, 175
  - trdp\_queueDelElement, 175
  - trdp\_queueFindAddr, 175
  - trdp\_queueFindComId, 175
  - trdp\_queueInsFirst, 176
  - trdp\_releaseSocket, 176
  - trdp\_requestSocket, 176
- trdp\_utils.h, 178
  - am\_big\_endian, 179
  - trdp\_initSockets, 179
  - trdp\_packetSizePD, 180
  - trdp\_queueAppLast, 180
  - trdp\_queueDelElement, 180
  - trdp\_queueFindAddr, 180
  - trdp\_queueFindComId, 180
  - trdp\_queueInsFirst, 181
  - trdp\_releaseSocket, 181
  - trdp\_requestSocket, 181
- tv\_usec
  - VOS\_TIME\_T, 56
- usage
  - TRDP\_SOCKETS, 48
- VOS\_INIT\_ERR
  - vos\_types.h, 249
- VOS\_IO\_ERR
  - vos\_types.h, 249
- VOS\_LOG\_DBG
  - vos\_types.h, 249
- VOS\_LOG\_ERROR
  - vos\_types.h, 249
- VOS\_LOG\_INFO
  - vos\_types.h, 249
- VOS\_LOG\_WARNING
  - vos\_types.h, 249
- VOS\_MEM\_ERR
  - vos\_types.h, 249
- VOS\_MUTEX\_ERR
  - vos\_types.h, 249
- VOS\_NO\_ERR
  - vos\_types.h, 249
- VOS\_NODATA\_ERR
  - vos\_types.h, 249
- VOS\_NOINIT\_ERR
  - vos\_types.h, 249
- VOS\_PARAM\_ERR
  - vos\_types.h, 249
- VOS\_QUEUE\_ERR
  - vos\_types.h, 249
- VOS\_QUEUE\_FULL\_ERR

- vos\_types.h, 249
- VOS\_SEMA\_ERR
  - vos\_types.h, 249
- VOS SOCK\_ERR
  - vos\_types.h, 249
- VOS\_THREAD\_ERR
  - vos\_types.h, 249
- VOS\_TIMEOUT\_ERR
  - vos\_types.h, 249
- vos\_types.h
  - VOS\_INIT\_ERR, 249
  - VOS\_IO\_ERR, 249
  - VOS\_LOG\_DBG, 249
  - VOS\_LOG\_ERROR, 249
  - VOS\_LOG\_INFO, 249
  - VOS\_LOG\_WARNING, 249
  - VOS\_MEM\_ERR, 249
  - VOS\_MUTEX\_ERR, 249
  - VOS\_NO\_ERR, 249
  - VOS\_NODATA\_ERR, 249
  - VOS\_NOINIT\_ERR, 249
  - VOS\_PARAM\_ERR, 249
  - VOS\_QUEUE\_ERR, 249
  - VOS\_QUEUE\_FULL\_ERR, 249
  - VOS\_SEMA\_ERR, 249
  - VOS SOCK\_ERR, 249
  - VOS\_THREAD\_ERR, 249
  - VOS\_TIMEOUT\_ERR, 249
  - VOS\_UNKNOWN\_ERR, 249
- VOS\_UNKNOWN\_ERR
  - vos\_types.h, 249
- vos\_addTime
  - vos\_thread.c, 226
  - vos\_thread.h, 236
- vos\_clearTime
  - vos\_thread.c, 227
  - vos\_thread.h, 237
- vos\_cmpTime
  - vos\_thread.c, 227
  - vos\_thread.h, 237
- vos\_crc32
  - vos\_utils.c, 251
  - vos\_utils.h, 254
- VOS\_ERR\_T
  - vos\_types.h, 249
- vos\_getTime
  - vos\_thread.c, 227
  - vos\_thread.h, 237
- vos\_getTimeStamp
  - vos\_thread.c, 227
  - vos\_thread.h, 238
- vos\_getUuid
  - vos\_thread.c, 228
  - vos\_thread.h, 238
- vos\_htonl
  - vos\_sock.c, 200
  - vos\_sock.h, 211
- vos\_htons
  - vos\_sock.c, 200
  - vos\_sock.h, 211
- vos\_init
  - vos\_types.h, 250
  - vos\_utils.c, 252
- vos\_isMulticast
  - vos\_sock.c, 200
  - vos\_sock.h, 212
- VOS\_LOG\_T
  - vos\_types.h, 249
- vos\_mem.c, 183
  - vos\_memAlloc, 184
  - vos\_memCount, 185
  - vos\_memDelete, 185
  - vos\_memFree, 185
  - vos\_memInit, 186
  - vos\_queueCreate, 186
  - vos\_queueDestroy, 187
  - vos\_queueReceive, 187
  - vos\_queueSend, 188
  - vos\_sharedClose, 188
  - vos\_sharedOpen, 188
- vos\_mem.h, 190
  - VOS\_MEM\_BLOCKSIZEs, 192
  - VOS\_MEM\_PREALLOCATE, 192
  - vos\_memAlloc, 192
  - vos\_memCount, 193
  - vos\_memDelete, 193
  - vos\_memFree, 193
  - vos\_memInit, 194
  - vos\_queueCreate, 194
  - vos\_queueDestroy, 195
  - vos\_queueReceive, 195
  - vos\_queueSend, 196
  - vos\_sharedClose, 196
  - vos\_sharedOpen, 196
- VOS\_MEM\_BLOCKSIZEs
  - vos\_mem.h, 192
- VOS\_MEM\_PREALLOCATE
  - vos\_mem.h, 192
- vos\_memAlloc
  - vos\_mem.c, 184
  - vos\_mem.h, 192
- vos\_memCount
  - vos\_mem.c, 185
  - vos\_mem.h, 193
- vos\_memDelete
  - vos\_mem.c, 185
  - vos\_mem.h, 193
- vos\_memFree

- [vos\\_mem.c, 185](#)
  - [vos\\_mem.h, 193](#)
- [vos\\_memInit](#)
  - [vos\\_mem.c, 186](#)
  - [vos\\_mem.h, 194](#)
- [vos\\_mutexCreate](#)
  - [vos\\_thread.c, 228](#)
  - [vos\\_thread.h, 238](#)
- [vos\\_mutexDelete](#)
  - [vos\\_thread.c, 228](#)
  - [vos\\_thread.h, 239](#)
- [vos\\_mutexLock](#)
  - [vos\\_thread.c, 229](#)
  - [vos\\_thread.h, 240](#)
- [vos\\_mutexTryLock](#)
  - [vos\\_thread.c, 229](#)
  - [vos\\_thread.h, 240](#)
- [vos\\_mutexUnlock](#)
  - [vos\\_thread.c, 229](#)
  - [vos\\_thread.h, 241](#)
- [vos\\_ntohl](#)
  - [vos\\_sock.c, 201](#)
  - [vos\\_sock.h, 212](#)
- [vos\\_ntohs](#)
  - [vos\\_sock.c, 201](#)
  - [vos\\_sock.h, 212](#)
- [VOS\\_PRINT\\_DBG\\_T](#)
  - [vos\\_types.h, 248](#)
- [vos\\_queueCreate](#)
  - [vos\\_mem.c, 186](#)
  - [vos\\_mem.h, 194](#)
- [vos\\_queueDestroy](#)
  - [vos\\_mem.c, 187](#)
  - [vos\\_mem.h, 195](#)
- [vos\\_queueReceive](#)
  - [vos\\_mem.c, 187](#)
  - [vos\\_mem.h, 195](#)
- [vos\\_queueSend](#)
  - [vos\\_mem.c, 188](#)
  - [vos\\_mem.h, 196](#)
- [vos\\_semaCreate](#)
  - [vos\\_thread.c, 230](#)
  - [vos\\_thread.h, 241](#)
- [vos\\_semaDelete](#)
  - [vos\\_thread.c, 230](#)
  - [vos\\_thread.h, 242](#)
- [vos\\_semaGive](#)
  - [vos\\_thread.c, 230](#)
  - [vos\\_thread.h, 242](#)
- [vos\\_semaTake](#)
  - [vos\\_thread.c, 231](#)
  - [vos\\_thread.h, 242](#)
- [vos\\_sharedClose](#)
  - [vos\\_mem.c, 188](#)
- [vos\\_mem.h, 196](#)
- [vos\\_sharedOpen](#)
  - [vos\\_mem.c, 188](#)
  - [vos\\_mem.h, 196](#)
- [vos\\_sock.c, 198](#)
  - [vos\\_htonl, 200](#)
  - [vos\\_htons, 200](#)
  - [vos\\_isMulticast, 200](#)
  - [vos\\_ntohl, 201](#)
  - [vos\\_ntohs, 201](#)
  - [vos\\_sockAccept, 201](#)
  - [vos\\_sockBind, 202](#)
  - [vos\\_sockClose, 202](#)
  - [vos\\_sockConnect, 202](#)
  - [vos\\_sockInit, 203](#)
  - [vos\\_sockJoinMC, 203](#)
  - [vos\\_sockLeaveMC, 204](#)
  - [vos\\_sockListen, 204](#)
  - [vos\\_sockOpenTCP, 204](#)
  - [vos\\_sockOpenUDP, 205](#)
  - [vos\\_sockReceiveTCP, 205](#)
  - [vos\\_sockReceiveUDP, 206](#)
  - [vos\\_sockSendTCP, 206](#)
  - [vos\\_sockSendUDP, 207](#)
  - [vos\\_sockSetOptions, 207](#)
- [vos\\_sock.h, 209](#)
  - [vos\\_htonl, 211](#)
  - [vos\\_htons, 211](#)
  - [vos\\_isMulticast, 212](#)
  - [vos\\_ntohl, 212](#)
  - [vos\\_ntohs, 212](#)
  - [vos\\_sockAccept, 212](#)
  - [vos\\_sockBind, 213](#)
  - [vos\\_sockClose, 214](#)
  - [vos\\_sockConnect, 214](#)
  - [vos\\_sockInit, 215](#)
  - [vos\\_sockJoinMC, 215](#)
  - [vos\\_sockLeaveMC, 216](#)
  - [vos\\_sockListen, 217](#)
  - [vos\\_sockOpenTCP, 218](#)
  - [vos\\_sockOpenUDP, 218](#)
  - [vos\\_sockReceiveTCP, 219](#)
  - [vos\\_sockReceiveUDP, 220](#)
  - [vos\\_sockSendTCP, 221](#)
  - [vos\\_sockSendUDP, 222](#)
  - [vos\\_sockSetOptions, 223](#)
- [VOS\\_SOCKET\\_OPT\\_T, 55](#)
  - [qos, 55](#)
- [vos\\_sockAccept](#)
  - [vos\\_sock.c, 201](#)
  - [vos\\_sock.h, 212](#)
- [vos\\_sockBind](#)
  - [vos\\_sock.c, 202](#)
  - [vos\\_sock.h, 213](#)



vos\_sockClose  
  vos\_sock.c, 202  
  vos\_sock.h, 214  
vos\_sockConnect  
  vos\_sock.c, 202  
  vos\_sock.h, 214  
vos\_sockInit  
  vos\_sock.c, 203  
  vos\_sock.h, 215  
vos\_sockJoinMC  
  vos\_sock.c, 203  
  vos\_sock.h, 215  
vos\_sockLeaveMC  
  vos\_sock.c, 204  
  vos\_sock.h, 216  
vos\_sockListen  
  vos\_sock.c, 204  
  vos\_sock.h, 217  
vos\_sockOpenTCP  
  vos\_sock.c, 204  
  vos\_sock.h, 218  
vos\_sockOpenUDP  
  vos\_sock.c, 205  
  vos\_sock.h, 218  
vos\_sockReceiveTCP  
  vos\_sock.c, 205  
  vos\_sock.h, 219  
vos\_sockReceiveUDP  
  vos\_sock.c, 206  
  vos\_sock.h, 220  
vos\_sockSendTCP  
  vos\_sock.c, 206  
  vos\_sock.h, 221  
vos\_sockSendUDP  
  vos\_sock.c, 207  
  vos\_sock.h, 222  
vos\_sockSetOptions  
  vos\_sock.c, 207  
  vos\_sock.h, 223  
vos\_subTime  
  vos\_thread.c, 231  
  vos\_thread.h, 243  
vos\_thread.c, 224  
  cyclicThread, 226  
  vos\_addTime, 226  
  vos\_clearTime, 227  
  vos\_cmpTime, 227  
  vos\_getTime, 227  
  vos\_getTimeStamp, 227  
  vos\_getUuid, 228  
  vos\_mutexCreate, 228  
  vos\_mutexDelete, 228  
  vos\_mutexLock, 229  
  vos\_mutexTryLock, 229  
  vos\_mutexUnlock, 229  
  vos\_semaCreate, 230  
  vos\_semaDelete, 230  
  vos\_semaGive, 230  
  vos\_semaTake, 231  
  vos\_subTime, 231  
  vos\_threadCreate, 231  
  vos\_threadDelay, 232  
  vos\_threadInit, 232  
  vos\_threadIsActive, 232  
  vos\_threadTerminate, 233  
vos\_thread.h, 234  
  vos\_addTime, 236  
  vos\_clearTime, 237  
  vos\_cmpTime, 237  
  vos\_getTime, 237  
  vos\_getTimeStamp, 238  
  vos\_getUuid, 238  
  vos\_mutexCreate, 238  
  vos\_mutexDelete, 239  
  vos\_mutexLock, 240  
  vos\_mutexTryLock, 240  
  vos\_mutexUnlock, 241  
  vos\_semaCreate, 241  
  vos\_semaDelete, 242  
  vos\_semaGive, 242  
  vos\_semaTake, 242  
  vos\_subTime, 243  
  vos\_threadCreate, 243  
  vos\_threadDelay, 244  
  vos\_threadInit, 244  
  vos\_threadIsActive, 245  
  vos\_threadTerminate, 245  
vos\_threadCreate  
  vos\_thread.c, 231  
  vos\_thread.h, 243  
vos\_threadDelay  
  vos\_thread.c, 232  
  vos\_thread.h, 244  
vos\_threadInit  
  vos\_thread.c, 232  
  vos\_thread.h, 244  
vos\_threadIsActive  
  vos\_thread.c, 232  
  vos\_thread.h, 245  
vos\_threadTerminate  
  vos\_thread.c, 233  
  vos\_thread.h, 245  
VOS\_TIME\_T, 56  
  tv\_usec, 56  
vos\_types.h, 247  
  VOS\_ERR\_T, 249  
  vos\_init, 250  
  VOS\_LOG\_T, 249

VOS\_PRINT\_DBG\_T, [248](#)  
vos\_utils.c, [251](#)  
    vos\_crc32, [251](#)  
    vos\_init, [252](#)  
vos\_utils.h, [253](#)  
    vos\_crc32, [254](#)