

TCNOpen TRDP
ReleaseV1.3

Generated by Doxygen 1.5.6

Mon Mar 7 14:30:17 2016

Contents

1	The TRDP Light Library API Specification	1
1.1	General Information	1
1.1.1	Purpose	1
1.1.2	Scope	1
1.1.3	Related documents	1
1.1.4	Abbreviations and Definitions	1
1.2	Terminology	2
1.3	Conventions of the API	3
2	Data Structure Index	5
2.1	Data Structures	5
3	File Index	7
3.1	File List	7
4	Data Structure Documentation	9
4.1	GNU_PACKED Struct Reference	9
4.1.1	Detailed Description	16
4.1.2	Field Documentation	17
4.1.2.1	trnVehNo	17
4.1.2.2	isLead	17
4.1.2.3	leadDir	17
4.1.2.4	vehOrient	17
4.1.2.5	version	17
4.1.2.6	reserved01	18
4.1.2.7	trnCstNo	18
4.1.2.8	reserved02	18
4.1.2.9	ownOpCstNo	18
4.1.2.10	reserved03	18

4.1.2.11	leadVehOfCst	18
4.1.2.12	reserved04	18
4.1.2.13	reserved06	19
4.1.2.14	confVehCnt	19
4.1.2.15	safetyTrail	19
4.1.2.16	reserved01	19
4.1.2.17	deviceName	19
4.1.2.18	inhibit	19
4.1.2.19	lifesign	19
4.1.2.20	etbInhibit	19
4.1.2.21	etbLength	20
4.1.2.22	etbShort	20
4.1.2.23	reserved02	20
4.1.2.24	trnDirState	20
4.1.2.25	opTrnDirState	20
4.1.2.26	sleepReqCnt	20
4.1.2.27	opTrnTopoCnt	20
4.1.2.28	confVehCnt	21
4.1.2.29	confVehList	21
4.1.2.30	etbTopoCnt	21
4.1.2.31	trnNetDir	21
4.1.2.32	cstUUID	21
4.1.2.33	cstCnt	21
4.1.2.34	cstList	21
4.1.2.35	trnTopoCnt	22
4.1.2.36	etbId	22
4.1.2.37	vehId	22
4.1.2.38	opVehNo	22
4.1.2.39	opCstNo	22
4.1.2.40	trnId	22
4.1.2.41	trnOperator	22
4.1.2.42	opCstCnt	22
4.1.2.43	opCstList	23
4.1.2.44	opVehCnt	23
4.1.2.45	opVehList	23
4.1.2.46	cstNetProp	23

4.1.2.47	protocolVersion	23
4.1.2.48	msgType	23
4.1.2.49	datasetLength	23
4.2	TRDP_CLTR_CST_INFO_T Struct Reference	24
4.2.1	Detailed Description	24
4.2.2	Field Documentation	24
4.2.2.1	cltrCstNo	24
4.3	TRDP_COMID_DSID_MAP_T Struct Reference	25
4.3.1	Detailed Description	25
4.4	TRDP_CONSIST_INFO_T Struct Reference	26
4.4.1	Detailed Description	27
4.4.2	Field Documentation	27
4.4.2.1	cstId	27
4.4.2.2	cstOwner	28
4.4.2.3	etbCnt	28
4.4.2.4	vehCnt	28
4.4.2.5	fctCnt	28
4.4.2.6	cltrCstCnt	28
4.5	TRDP_DATASET Struct Reference	29
4.5.1	Detailed Description	29
4.6	TRDP_DATASET_ELEMENT_T Struct Reference	30
4.6.1	Detailed Description	30
4.6.2	Field Documentation	30
4.6.2.1	type	30
4.7	TRDP_DBG_CONFIG_T Struct Reference	31
4.7.1	Detailed Description	31
4.8	TRDP_ETB_INFO_T Struct Reference	32
4.8.1	Detailed Description	32
4.8.2	Field Documentation	32
4.8.2.1	etbId	32
4.8.2.2	cnCnt	32
4.9	TRDP_FUNCTION_INFO_T Struct Reference	33
4.9.1	Detailed Description	33
4.9.2	Field Documentation	33
4.9.2.1	fctId	33
4.9.2.2	cstVehNo	33

4.9.2.3	etbId	34
4.9.2.4	cnId	34
4.10	TRDP_LIST_STATISTICS_T Struct Reference	35
4.10.1	Detailed Description	35
4.11	TRDP_MARSHALL_CONFIG_T Struct Reference	36
4.11.1	Detailed Description	36
4.12	TRDP_MD_CONFIG_T Struct Reference	37
4.12.1	Detailed Description	38
4.13	TRDP_MD_INFO_T Struct Reference	39
4.13.1	Detailed Description	40
4.13.2	Field Documentation	40
4.13.2.1	msgType	40
4.14	TRDP_MD_STATISTICS_T Struct Reference	41
4.14.1	Detailed Description	42
4.15	TRDP_MEM_CONFIG_T Struct Reference	43
4.15.1	Detailed Description	43
4.16	TRDP_MEM_STATISTICS_T Struct Reference	44
4.16.1	Detailed Description	44
4.17	TRDP_PD_CONFIG_T Struct Reference	45
4.17.1	Detailed Description	45
4.18	TRDP_PD_INFO_T Struct Reference	46
4.18.1	Detailed Description	47
4.18.2	Field Documentation	47
4.18.2.1	msgType	47
4.19	TRDP_PD_STATISTICS_T Struct Reference	48
4.19.1	Detailed Description	49
4.20	TRDP_PROCESS_CONFIG_T Struct Reference	50
4.20.1	Detailed Description	50
4.21	TRDP_PROP_T Struct Reference	51
4.21.1	Detailed Description	51
4.21.2	Field Documentation	51
4.21.2.1	len	51
4.22	TRDP_PUB_STATISTICS_T Struct Reference	52
4.22.1	Detailed Description	52
4.22.2	Field Documentation	52
4.22.2.1	destAddr	52

4.23	TRDP_RED_STATISTICS_T Struct Reference	53
4.23.1	Detailed Description	53
4.24	TRDP_SDT_PAR_T Struct Reference	54
4.24.1	Detailed Description	54
4.25	TRDP_SEND_PARAM_T Struct Reference	55
4.25.1	Detailed Description	55
4.26	TRDP_STATISTICS_T Struct Reference	56
4.26.1	Detailed Description	57
4.27	TRDP_SUBS_STATISTICS_T Struct Reference	58
4.27.1	Detailed Description	58
4.27.2	Field Documentation	58
4.27.2.1	filterAddr	58
4.27.2.2	timeout	59
4.27.2.3	toBehav	59
4.28	TRDP_VEHICLE_INFO_T Struct Reference	60
4.28.1	Detailed Description	60
4.28.2	Field Documentation	60
4.28.2.1	vehId	60
4.28.2.2	cstVehNo	61
4.29	TRDP_XML_DOC_HANDLE_T Struct Reference	62
4.29.1	Detailed Description	62
4.30	VOS SOCK_OPT_T Struct Reference	63
4.30.1	Detailed Description	63
4.30.2	Field Documentation	63
4.30.2.1	qos	63
4.31	VOS_TIME_T Struct Reference	64
4.31.1	Detailed Description	64
4.31.2	Field Documentation	64
4.31.2.1	tv_usec	64
4.32	VOS_VERSION_T Struct Reference	65
4.32.1	Detailed Description	65
5	File Documentation	67
5.1	iec61375-2-3.h File Reference	67
5.1.1	Detailed Description	70
5.1.2	Define Documentation	70
5.1.2.1	ETBN_STATUS_COMID	70

5.1.2.2	TTDB_NET_DIR_REQ_COMID	70
5.1.2.3	TTDB_OP_DIR_INFO_COMID	71
5.1.2.4	TTDB_STAT_CST_REQ_COMID	71
5.1.2.5	TTDB_TRN_DIR_REQ_COMID	71
5.2	tau_ctrl.h File Reference	72
5.2.1	Detailed Description	73
5.2.2	Function Documentation	73
5.2.2.1	tau_getEcspStat	73
5.2.2.2	tau_initEcspCtrl	74
5.2.2.3	tau_requestEcspConfirm	74
5.2.2.4	tau_setEcspCtrl	74
5.2.2.5	tau_terminateEcspCtrl	75
5.3	tau_ctrl_types.h File Reference	76
5.3.1	Detailed Description	77
5.4	tau_dnr.h File Reference	78
5.4.1	Detailed Description	79
5.4.2	Function Documentation	79
5.4.2.1	tau_addr2Uri	79
5.4.2.2	tau_deInitDnr	79
5.4.2.3	tau_DNRstatus	80
5.4.2.4	tau_getOwnAddr	80
5.4.2.5	tau_getOwnIds	80
5.4.2.6	tau_initDnr	81
5.4.2.7	tau_uri2Addr	81
5.5	tau_marshall.h File Reference	82
5.5.1	Detailed Description	83
5.5.2	Function Documentation	83
5.5.2.1	tau_calcDatasetSize	83
5.5.2.2	tau_calcDatasetSizeByComId	84
5.5.2.3	tau_initMarshall	84
5.5.2.4	tau_marshall	84
5.5.2.5	tau_marshallIDs	85
5.5.2.6	tau_unmarshall	86
5.5.2.7	tau_unmarshallIDs	86
5.6	tau_tti.h File Reference	87
5.6.1	Detailed Description	89

5.6.2	Function Documentation	89
5.6.2.1	tau_deInitTTI	89
5.6.2.2	tau_getCstFctCnt	89
5.6.2.3	tau_getCstFctInfo	90
5.6.2.4	tau_getCstInfo	90
5.6.2.5	tau_getCstVehCnt	90
5.6.2.6	tau_getOpTrDirectory	91
5.6.2.7	tau_getStaticCstInfo	91
5.6.2.8	tau_getTrDirectory	91
5.6.2.9	tau_getTrnCstCnt	92
5.6.2.10	tau_getTrnVehCnt	92
5.6.2.11	tau_getTTI	92
5.6.2.12	tau_getVehInfo	93
5.6.2.13	tau_getVehOrient	93
5.6.2.14	tau_initTTIaccess	93
5.7	tau_tti_types.h File Reference	95
5.7.1	Detailed Description	97
5.8	tau_xml.h File Reference	98
5.8.1	Detailed Description	100
5.8.2	Enumeration Type Documentation	100
5.8.2.1	TRDP_DBG_OPTION_T	100
5.8.2.2	TRDP_EXCHG_OPTION_T	100
5.8.3	Function Documentation	101
5.8.3.1	tau_freeTelegrams	101
5.8.3.2	tau_freeXmlDatasetConfig	101
5.8.3.3	tau_freeXmlDoc	101
5.8.3.4	tau_prepareXmlDoc	102
5.8.3.5	tau_readXmlDatasetConfig	102
5.8.3.6	tau_readXmlDeviceConfig	102
5.8.3.7	tau_readXmlInterfaceConfig	103
5.9	trdp_if_light.h File Reference	104
5.9.1	Detailed Description	108
5.9.2	Function Documentation	108
5.9.2.1	tlc_closeSession	108
5.9.2.2	tlc_configSession	109
5.9.2.3	tlc_freeBuf	109

5.9.2.4	tlc_getInterval	109
5.9.2.5	tlc_getJoinStatistics	110
5.9.2.6	tlc_getOwnIpAddress	110
5.9.2.7	tlc_getPubStatistics	110
5.9.2.8	tlc_getRedStatistics	111
5.9.2.9	tlc_getStatistics	111
5.9.2.10	tlc_getSubsStatistics	112
5.9.2.11	tlc_getTcpListStatistics	112
5.9.2.12	tlc_getUdpListStatistics	112
5.9.2.13	tlc_getVersion	113
5.9.2.14	tlc_getVersionString	113
5.9.2.15	tlc_init	113
5.9.2.16	tlc_openSession	114
5.9.2.17	tlc_process	114
5.9.2.18	tlc_reinitSession	115
5.9.2.19	tlc_resetStatistics	115
5.9.2.20	tlc_setETBTopoCount	115
5.9.2.21	tlc_setOpTrainTopoCount	115
5.9.2.22	tlc_terminate	116
5.9.2.23	tlm_abortSession	116
5.9.2.24	tlm_addListener	116
5.9.2.25	tlm_confirm	117
5.9.2.26	tlm_delListener	117
5.9.2.27	tlm_notify	118
5.9.2.28	tlm_readdListener	118
5.9.2.29	tlm_reply	119
5.9.2.30	tlm_replyErr	119
5.9.2.31	tlm_replyQuery	120
5.9.2.32	tlm_request	121
5.9.2.33	tlp_get	121
5.9.2.34	tlp_getRedundant	122
5.9.2.35	tlp_publish	122
5.9.2.36	tlp_put	123
5.9.2.37	tlp_republish	124
5.9.2.38	tlp_request	124
5.9.2.39	tlp_resubscribe	125

5.9.2.40	tlp_setRedundant	125
5.9.2.41	tlp_subscribe	126
5.9.2.42	tlp_unpublish	126
5.9.2.43	tlp_unsubscribe	127
5.10	trdp_proto.h File Reference	128
5.10.1	Detailed Description	130
5.10.2	Define Documentation	130
5.10.2.1	TRDP_DEST_URI_SIZE	130
5.10.2.2	TRDP_ETBCTRL_COMID	131
5.10.2.3	TRDP_ETBCTRL_DSID	131
5.10.2.4	TRDP_MAX_FILE_NAME_LEN	131
5.10.2.5	TRDP_MAX_LABEL_LEN	131
5.10.2.6	TRDP_MAX_URI_HOST_LEN	131
5.10.2.7	TRDP_MAX_URI_LEN	131
5.10.2.8	TRDP_MAX_URI_USER_LEN	131
5.10.3	Enumeration Type Documentation	131
5.10.3.1	TRDP_MSG_T	131
5.11	trdp_types.h File Reference	133
5.11.1	Detailed Description	138
5.11.2	Typedef Documentation	138
5.11.2.1	TRDP_IP_ADDR_T	138
5.11.2.2	TRDP_MARSHALL_T	139
5.11.2.3	TRDP_MD_CALLBACK_T	139
5.11.2.4	TRDP_PD_CALLBACK_T	139
5.11.2.5	TRDP_PRINT_DBG_T	140
5.11.2.6	TRDP_TIME_T	140
5.11.2.7	TRDP_UNMARSHALL_T	140
5.11.3	Enumeration Type Documentation	140
5.11.3.1	TRDP_DATA_TYPE_T	140
5.11.3.2	TRDP_ERR_T	141
5.11.3.3	TRDP_FLAGS_T	142
5.11.3.4	TRDP_OPTION_T	142
5.11.3.5	TRDP_RED_STATE_T	143
5.11.3.6	TRDP_REPLY_STATUS_T	143
5.11.3.7	TRDP_TO_BEHAVIOR_T	143
5.12	vos_mem.c File Reference	144

5.12.1 Detailed Description	145
5.12.2 Function Documentation	146
5.12.2.1 vos_bsearch	146
5.12.2.2 vos_memAlloc	146
5.12.2.3 vos_memCount	147
5.12.2.4 vos_memDelete	147
5.12.2.5 vos_memFree	147
5.12.2.6 vos_memInit	148
5.12.2.7 vos_qsort	148
5.12.2.8 vos_queueCreate	149
5.12.2.9 vos_queueDestroy	149
5.12.2.10 vos_queueReceive	150
5.12.2.11 vos_queueSend	150
5.12.2.12 vos_strncat	151
5.12.2.13 vos_strncpy	151
5.12.2.14 vos_strncmp	152
5.13 vos_mem.h File Reference	153
5.13.1 Detailed Description	155
5.13.2 Define Documentation	155
5.13.2.1 VOS_MEM_BLOCKSIZE	155
5.13.2.2 VOS_MEM_PREALLOCATE	156
5.13.3 Function Documentation	156
5.13.3.1 vos_bsearch	156
5.13.3.2 vos_memAlloc	156
5.13.3.3 vos_memCount	157
5.13.3.4 vos_memDelete	157
5.13.3.5 vos_memFree	157
5.13.3.6 vos_memInit	158
5.13.3.7 vos_qsort	159
5.13.3.8 vos_queueCreate	159
5.13.3.9 vos_queueDestroy	160
5.13.3.10 vos_queueReceive	160
5.13.3.11 vos_queueSend	161
5.13.3.12 vos_strncat	162
5.13.3.13 vos_strncpy	162
5.13.3.14 vos_strncmp	163

5.14	vos_shared_mem.h File Reference	164
5.14.1	Detailed Description	164
5.14.2	Function Documentation	165
5.14.2.1	vos_sharedClose	165
5.14.2.2	vos_sharedOpen	165
5.15	vos_sock.h File Reference	166
5.15.1	Detailed Description	169
5.15.2	Define Documentation	169
5.15.2.1	VOS_MAX_SOCKET_CNT	169
5.15.2.2	VOS_TTL_MULTICAST	170
5.15.3	Function Documentation	170
5.15.3.1	vos_determineBindAddr	170
5.15.3.2	vos_dottedIP	170
5.15.3.3	vos_getInterfaces	170
5.15.3.4	vos_htonl	171
5.15.3.5	vos_htons	171
5.15.3.6	vos_ipDotted	171
5.15.3.7	vos_isMulticast	171
5.15.3.8	vos_netIfUp	172
5.15.3.9	vos_ntohl	172
5.15.3.10	vos_ntohs	172
5.15.3.11	vos_select	172
5.15.3.12	vos_sockAccept	173
5.15.3.13	vos_sockBind	173
5.15.3.14	vos_sockClose	173
5.15.3.15	vos_sockConnect	174
5.15.3.16	vos_sockGetMAC	174
5.15.3.17	vos_sockInit	174
5.15.3.18	vos_sockJoinMC	174
5.15.3.19	vos_sockLeaveMC	175
5.15.3.20	vos_sockListen	175
5.15.3.21	vos_sockOpenTCP	176
5.15.3.22	vos_sockOpenUDP	176
5.15.3.23	vos_sockReceiveTCP	176
5.15.3.24	vos_sockReceiveUDP	177
5.15.3.25	vos_sockSendTCP	177

5.15.3.26	<code>vos_sockSendUDP</code>	178
5.15.3.27	<code>vos_sockSetMulticastIf</code>	178
5.15.3.28	<code>vos_sockSetOptions</code>	179
5.15.3.29	<code>vos_sockTerm</code>	179
5.16	<code>vos_thread.h</code> File Reference	180
5.16.1	Detailed Description	183
5.16.2	Function Documentation	183
5.16.2.1	<code>vos_addTime</code>	183
5.16.2.2	<code>vos_clearTime</code>	183
5.16.2.3	<code>vos_cmpTime</code>	184
5.16.2.4	<code>vos_cyclicThread</code>	184
5.16.2.5	<code>vos_divTime</code>	184
5.16.2.6	<code>vos_getTime</code>	184
5.16.2.7	<code>vos_getTimeStamp</code>	185
5.16.2.8	<code>vos_getUuid</code>	185
5.16.2.9	<code>vos_mulTime</code>	185
5.16.2.10	<code>vos_mutexCreate</code>	185
5.16.2.11	<code>vos_mutexDelete</code>	185
5.16.2.12	<code>vos_mutexLock</code>	186
5.16.2.13	<code>vos_mutexTryLock</code>	186
5.16.2.14	<code>vos_mutexUnlock</code>	186
5.16.2.15	<code>vos_semaCreate</code>	187
5.16.2.16	<code>vos_semaDelete</code>	187
5.16.2.17	<code>vos_semaGive</code>	187
5.16.2.18	<code>vos_semaTake</code>	187
5.16.2.19	<code>vos_subTime</code>	188
5.16.2.20	<code>vos_threadCreate</code>	188
5.16.2.21	<code>vos_threadDelay</code>	189
5.16.2.22	<code>vos_threadInit</code>	189
5.16.2.23	<code>vos_threadIsActive</code>	189
5.16.2.24	<code>vos_threadTerm</code>	189
5.16.2.25	<code>vos_threadTerminate</code>	190
5.17	<code>vos_types.h</code> File Reference	191
5.17.1	Detailed Description	193
5.17.2	Typedef Documentation	193
5.17.2.1	<code>VOS_PRINT_DBG_T</code>	193

5.17.3	Enumeration Type Documentation	193
5.17.3.1	VOS_ERR_T	193
5.17.3.2	VOS_LOG_T	194
5.18	vos_utils.c File Reference	195
5.18.1	Detailed Description	196
5.18.2	Function Documentation	196
5.18.2.1	vos_crc32	196
5.18.2.2	vos_getVersion	196
5.18.2.3	vos_getVersionString	197
5.18.2.4	vos_init	197
5.18.2.5	vos_initRuntimeConsts	197
5.18.2.6	vos_terminate	197
5.19	vos_utils.h File Reference	199
5.19.1	Detailed Description	200
5.19.2	Define Documentation	201
5.19.2.1	INITFCS	201
5.19.2.2	VOS_MAX_ERR_STR_SIZE	201
5.19.2.3	VOS_MAX_FRMT_SIZE	201
5.19.2.4	VOS_MAX_PRNT_STR_SIZE	201
5.19.3	Function Documentation	201
5.19.3.1	vos_crc32	201
5.19.3.2	vos_getVersion	202
5.19.3.3	vos_getVersionString	202
5.19.3.4	vos_init	202
5.19.3.5	vos_terminate	203

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.

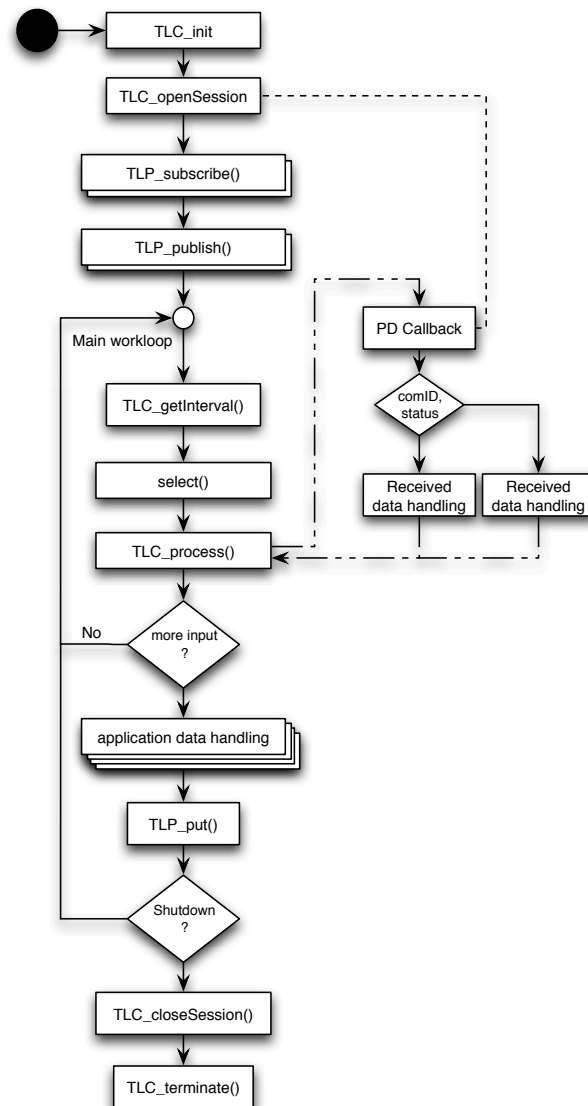


Figure 1.1: Sample client workflow

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:

GNU_PACKED (Types for ETB control)	9
TRDP_CLTR_CST_INFO_T (Closed train consists information)	24
TRDP_COMID_DSID_MAP_T (ComId - data set mapping element definition)	25
TRDP_CONSIST_INFO_T (Consist information structure)	26
TRDP_DATASET (Dataset definition)	29
TRDP_DATASET_ELEMENT_T (Dataset element definition)	30
TRDP_DBG_CONFIG_T (Control for debug output device/file on application level)	31
TRDP_ETB_INFO_T (Types for train configuration information)	32
TRDP_FUNCTION_INFO_T (Function/device information structure)	33
TRDP_LIST_STATISTICS_T (Information about a particular MD listener)	35
TRDP_MARSHALL_CONFIG_T (Marshaling/unmarshalling configuration)	36
TRDP_MD_CONFIG_T (Default MD configuration)	37
TRDP_MD_INFO_T (Message data info from received telegram; allows the application to generate responses)	39
TRDP_MD_STATISTICS_T (Structure containing all general MD statistics information)	41
TRDP_MEM_CONFIG_T (Enumeration type for memory pre-fragmentation, reuse of VOS definition)	43
TRDP_MEM_STATISTICS_T (TRDP statistics type definitions)	44
TRDP_PD_CONFIG_T (Default PD configuration)	45
TRDP_PD_INFO_T (Process data info from received telegram; allows the application to generate responses)	46
TRDP_PD_STATISTICS_T (Structure containing all general PD statistics information)	48
TRDP_PROCESS_CONFIG_T (Various flags/general TRDP options for library initialization) .	50
TRDP_PROP_T (Application defined properties)	51
TRDP_PUB_STATISTICS_T (Table containing particular PD publishing information)	52
TRDP_RED_STATISTICS_T (A table containing PD redundant group information)	53
TRDP_SDT_PAR_T (Types to read out the XML configuration)	54
TRDP_SEND_PARAM_T (Quality/type of service and time to live)	55
TRDP_STATISTICS_T (Structure containing all general memory, PD and MD statistics information)	56
TRDP_SUBS_STATISTICS_T (Table containing particular PD subscription information)	58
TRDP_VEHICLE_INFO_T (Vehicle information structure)	60
TRDP_XML_DOC_HANDLE_T (Parsed XML document handle)	62

VOS_SOCK_OPT_T (Common socket options)	63
VOS_TIME_T (Timer value compatible with timeval / select)	64
VOS_VERSION_T (Version information)	65

Chapter 3

File Index

3.1 File List

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

iec61375-2-3.h (TTDB, CSTINFO Frame typedefs, Telegram definitions)	67
tau_ctrl.h (TRDP utility interface definitions)	72
tau_ctrl_types.h (TRDP utility interface definitions)	76
tau_dnr.h (TRDP utility interface definitions)	78
tau_marshall.h (TRDP utility interface definitions)	82
tau_tti.h (TRDP utility interface definitions)	87
tau_tti_types.h (TRDP utility interface definitions)	95
tau_xml.h (TRDP utility interface definitions)	98
trdp_if_light.h (TRDP Light interface functions (API))	104
trdp_proto.h (Definitions for the TRDP protocol)	128
trdp_types.h (Typedefs for TRDP communication)	133
vos_mem.c (Memory functions)	144
vos_mem.h (Memory and queue functions for OS abstraction)	153
vos_shared_mem.h (Shared Memory functions for OS abstraction)	164
vos_sock.h (Typedefs for OS abstraction)	166
vos_thread.h (Threading functions for OS abstraction)	180
vos_types.h (Typedefs for OS abstraction)	191
vos_utils.c (Common functions for VOS)	195
vos_utils.h (Typedefs for OS abstraction)	199

Chapter 4

Data Structure Documentation

4.1 GNU_PACKED Struct Reference

Types for ETB control.

```
#include <trdp_proto.h>
```

Data Fields

- **UINT8** [trnVehNo](#)
vehicle sequence number within the train with vehicle 01 being the first vehicle in ETB reference direction 1 as defined in IEC61375-2-5 value range: 0.
- **ANTIVALENT8** [isLead](#)
vehicle is leading
- **UINT8** [leadDir](#)
vehicle leading direction 0 = not relevant 1 = leading direction 1 2 = leading direction 2
- **UINT8** [vehOrient](#)
vehicle orientation 0 = not known (corrected vehicle) 1 = same as operational train direction 2 = inverse to operational train direction
- **TRDP_SHORT_VERSION_T** [version](#)
telegram version information, main_version = 1, sub_version = 0
- **UINT16** [reserved01](#)
reserved (=0)
- **UINT8** [trnCstNo](#)
own TCN consist number (= 1.
- **UINT8** [reserved02](#)
reserved (=0)
- **UINT8** [ownOpCstNo](#)

own operational address (= 1.

- UINT8 [reserved03](#)
reserved (=0)
- UINT32 [cstTopoCount](#)
Consist topology counter.
- UINT32 [trnTopoCount](#)
Train directory topology counter.
- UINT32 [opTrnTopoCount](#)
Operational Train topology counter.
- ANTIVALENT8 [wasLead](#)
consist was leading, '01'B = false, '10'B = true
- ANTIVALENT8 [reqLead](#)
leading request, '01'B = false, '10'B = true
- UINT8 [reqLeadDir](#)
(request) leading direction, '01'B = consist direction 1, '10'B = consist direction 2
- ANTIVALENT8 [accLead](#)
accept remote leading request, '01'B = false/not accepted, '10'B = true/accepted
- ANTIVALENT8 [clearConfComp](#)
clear confirmed composition, '01'B = false, '10'B = true
- ANTIVALENT8 [corrRequest](#)
request confirmation, '01'B = false, '10'B = true
- ANTIVALENT8 [corrInfoSet](#)
correction info set, '01'B = false, '10'B = true
- ANTIVALENT8 [compStored](#)
corrected composition stored, '01'B = false, '10'B = true
- ANTIVALENT8 [sleepRequest](#)
request sleep mode, '01'B = false, '10'B = true
- UINT8 [leadVehOfCst](#)
position of leading vehicle in consist, 0.
- UINT8 [reserved04](#)
reserved (=0)
- UINT16 [reserved05](#)
reserved (=0)

- UINT8 [reserved06](#)
reserved (=0)
- UINT8 [confVehCnt](#)
number of confirmed vehicles in train (1.
- TRDP_CONF_VEHICLE_T [confVehList](#) [TRDP_MAX_VEH_CNT]
dynamic ordered list of confirmed vehicles in train, starting with vehicle at train head, see sub-clause 5.3.3.2.6
- TRDP_ETB_CTRL_VDP_T [safetyTrail](#)
ETBCTRL-VDP trailer, completely set to 0 == not used.
- UINT8 [reserved01](#)
reserved (=0)
- TRDP_LABEL_T [deviceName](#)
function device of ECSC which sends the telegram
- UINT8 [inhibit](#)
inauguration inhibit 0 = no inhibit request 1 = inhibit request
- UINT8 [leadingReq](#)
leading request 0 = no leading request 1 = leading request
- UINT8 [leadingDir](#)
leading direction 0 = no leading request 1 = leading request direction 1 2 = leading request direction 2
- UINT8 [sleepReq](#)
sleep request 0 = no sleep request 1 = sleep request
- UINT16 [lifesign](#)
wrap-around counter, incremented with each produced datagram.
- UINT8 [ecspState](#)
ECSP state indication 0 = ECSP not operational(initial value) 1 = ECSP in operation.
- UINT8 [etbInhibit](#)
inauguration inhibit indication 0 = n/a (default) 1 = inhibit not requested on ETB 2 = inhibit set on local ETBN 3 = inhibit set on remote ETBN 4 = inhibit set on local and remote ETBN
- UINT8 [etbLength](#)
indicates train lengthening in case train inauguration is inhibit 0 = no lengthening (default) 1 = lengthening detected
- UINT8 [etbShort](#)
indicates train shortening in case train inauguration is inhibit 0 = no shortening (default) 1 = shortening detected
- UINT16 [reserved02](#)

reserved (=0)

- **UINT8 etbLeadState**
indication of local consist leadership 5 = consist not leading (initial value) 6 = consist is leading requesting 9 = consist is leading 10 = leading conflict other values are not allowed
- **UINT8 etbLeadDir**
direction of the leading end car in the local consist 0 = unknown (default) 1 = TCN direction 1 2 = TCN direction 2 other values are not allowed
- **UINT8 ttDbSrvState**
TTDB server state indication 0 = n/a (initial value) 1 = Leader (default) 2 = Follower 3 = Error.
- **UINT8 dnsSrvState**
DNS server state indication 0 = n/a (initial value) 1 = Leader (default) 2 = Follower 3 = Error.
- **UINT8 trnDirState**
train directory state 1 = UNCONFIRMED 2 = CONFIRMED other values are not allowed
- **UINT8 opTrnDirState**
train directory state 1 = INVALID 2 = VALID 4 = SHARED other values are not allowed
- **UINT8 sleepCtrlState**
sleep control state (option) 0 = option not available 1 = RegularOperation 2 = WaitForSleepMode 3 = PrepareForSleepMode
- **UINT8 sleepReqCnt**
number of sleep requests (option) value range: 0.
- **UINT32 opTrnTopoCnt**
operational train topology counter
- **UINT8 command**
confirmation order 1 = confirmation/correction request 2 = un-confirmation request
- **UINT16 confVehCnt**
number of confirmed vehicles in the train (1.
- **TRDP_OP_VEHICLE_T confVehList [TRDP_MAX_VEH_CNT]**
ordered list of confirmed vehicles in the train, starting with vehicle at train head, see chapter 5.3.3.2.10.
- **UINT8 status**
status of storing correction info 0 = correctly stored 1 = not stored
- **UINT32 reqSafetyCode**
SC-32 value of the request message.
- **UINT8 byPassCtrl**
ETBN bypass control 0 = no action (keep old state) 1 = no bypass 2 = activate bypass.

- [UINT8 txCtrl](#)
ETBN transmission control 0 = no action (keep old state) 1 = activate sending on ETB (default) 2 = stop sending on ETB.
- [UINT8 slCtrl](#)
sleep mode control (option) 0 = no action (keep old state) 1 = deactivate sleep mode 2 = activate sleep mode (line activity sensing)
- [UINT8 etbnState](#)
state indication of the (active) ETBN 0 = ETBN not operational(initial value) 1 = ETBN in operation
- [UINT8 etbnInaugState](#)
ETBN inauguration state as defined in IEC61375-2-5 0 = init 1 = not inaugurated 2 = inaugurated 3 = ready for inauguration.
- [UINT8 etbnPosition](#)
position of the ETBN 0 = unknown (default) 1 = single node 2 = middle node 3 = end node TCN direction 1 4 = end node TCN direction 2
- [UINT8 etbnRole](#)
ETBN node role as defined in IEC61375-2-5 0 = undefined 1 = master (redundancy leader) 2 = backup (redundancy follower) 3 = not redundant.
- [BITSET8 etbLineState](#)
indication of ETB line status (FALSE == not trusted, TRUE == trusted) bit0 = line A ETBN direction 1 bit1 = line B ETBN direction 1 bit2 = line C ETBN direction 1 bit3 = line D ETBN direction 1 bit4 = line A ETBN direction 2 bit5 = line B ETBN direction 2 bit6 = line C ETBN direction 2 bit7 = line D ETBN direction 2
- [UINT8 byPassState](#)
state of bypass function 0 = bypass disabled 1 = bypass enabled
- [UINT8 slState](#)
sleep mode state (option) 0 = no sleep mode 1 = sleep mode active (line activity sensing)
- [UINT32 etbTopoCnt](#)
ETB topography counter.
- [TRDP_TRAIN_NET_DIR_T trnNetDir](#)
dynamic train info
- [UINT8 ver](#)
Version - incremented for incompatible changes.
- [UINT8 rel](#)
Release - incremented for compatible changes.
- [UINT32 reserved01](#)
reserved (=0)
- [TRDP_SHORT_VERSION_T userDataVersion](#)

version of the vital ETBCTRL telegram $\text{mainVersion} = 1$, $\text{subVersion} = 0$

- **UINT32** [safeSeqCount](#)
safe sequence counter, as defined in B.9
- **UINT32** [safetyCode](#)
checksum, as defined in B.9
- **TRDP_UUID_T** [cstUUID](#)
UUID of the consist, provided by ETBN (TrainNetworkDirectory) Reference to static consist attributes 0 if not available (e.g.
- **UINT32** [cstTopoCnt](#)
consist topology counter provided with the CSTINFO 0 if no CSTINFO available
- **UINT8** [cstOrient](#)
consist orientation '01'B = same as train direction '10'B = inverse to train direction
- **UINT8** [cstCnt](#)
number of consists in train; range: 1.
- **TRDP_CONSIST_T** [cstList](#) [TRDP_MAX_CST_CNT]
consist list.
- **UINT32** [trnTopoCnt](#)
trnTopoCnt value ctrlType == 0: actual value ctrlType == 1: set to 0
- **UINT8** [etbId](#)
identification of the ETB the TTDB is computed for bit0: ETB0 (operational network) bit1: ETB1 (multimedia network) bit2: ETB2 (other network) bit3: ETB3 (other network)
- **TRDP_LABEL_T** [vehId](#)
Unique vehicle identifier, application defined (e.g.
- **UINT8** [opVehNo](#)
operational vehicle sequence number in train value range 1.
- **UINT8** [opCstNo](#)
operational consist number in train (1.
- **UINT8** [opCstOrient](#)
consist orientation '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction
- **TRDP_LABEL_T** [trnId](#)
train identifier, application defined (e.g.
- **TRDP_LABEL_T** [trnOperator](#)
train operator, e.g.

- `UINT32 crc`
sc-32 computed over record (seed value: 'FFFFFFFF'H)
- `UINT8 opTrnOrient`
operational train orientation '00'B = unknown '01'B = same as train direction '10'B = inverse to train direction
- `UINT8 opCstCnt`
number of consists in train (1).
- `TRDP_OP_CONSIST_T opCstList [TRDP_MAX_CST_CNT]`
operational consist list starting with op.
- `UINT8 reserved05`
reserved for future use (= 0)
- `UINT8 opVehCnt`
number of vehicles in train (1).
- `TRDP_OP_VEHICLE_T opVehList [TRDP_MAX_CST_CNT]`
operational vehicle list starting with op.
- `TRDP_OP_TRAIN_DIR_STATE_T state`
operational state of the train
- `UINT32 cstNetProp`
consist network properties bit0.
- `UINT16 entryCnt`
number of entries in train network directory
- `TRDP_TRAIN_NET_DIR_ENTRY_T trnNetDir [TRDP_MAX_CST_CNT]`
train network directory
- `TRDP_OP_TRAIN_DIR_T opTrnDir`
operational directory
- `TRDP_TRAIN_DIR_T trnDir`
train directory
- `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 [datasetLength](#)
length of the data to transmit 0.
- UINT32 [reserved](#)
before used for ladder support
- UINT32 [replyComId](#)
used in PD request
- UINT32 [replyIpAddress](#)
used for PD request
- UINT32 [frameChecksum](#)
CRC32 of header.
- INT32 [replyStatus](#)
0 = OK
- UINT8 [sessionID](#) [16]
UUID as a byte stream.
- UINT32 [replyTimeout](#)
in us
- UINT8 [sourceURI](#) [32]
User part of URI.
- UINT8 [destinationURI](#) [32]
User part of URI.

4.1.1 Detailed Description

Types for ETB control.

TRDP message data header - network order and alignment.

TRDP process data header - network order and alignment.

Complete TTDB structure.

Train network directory structure.

Train network directory entry structure acc.

Operational Train directory status info structure.

Operational train structure.

Operational train directory state.

Operational consist structure.

Operational vehicle structure.

TCN train directory.

CSTINFO Control telegram.

TCN consist structure.

Version information for communication buffers.

to IEC61375-2-5

4.1.2 Field Documentation

4.1.2.1 UINT8 GNU_PACKED::trnVehNo

vehicle sequence number within the train with vehicle 01 being the first vehicle in ETB reference direction 1 as defined in IEC61375-2-5 value range: 0.

vehicle sequence number within the train with vehicle 01 being the first vehicle in ETB reference direction 1 as defined in IEC61375-2-5, value range: 1.

.63 a value of 0 indicates that this vehicle has been inserted by correction

.63, a value of 0 indicates that this vehicle has been inserted by correction

4.1.2.2 ANTIVALENT8 GNU_PACKED::isLead

vehicle is leading

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

4.1.2.3 UINT8 GNU_PACKED::leadDir

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

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

4.1.2.4 UINT8 GNU_PACKED::vehOrient

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

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

4.1.2.5 TRDP_SHORT_VERSION_T GNU_PACKED::version

telegram version information, main_version = 1, sub_version = 0

Train info structure version.

TrainDirectoryState data structure version parameter 'mainVersion' shall be set to 1.

TrainDirectory data structure version parameter 'mainVersion' shall be set to 1.

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

4.1.2.6 UINT16 GNU_PACKED::reserved01

reserved (=0)

reserved for future use (= 0)

4.1.2.7 UINT8 GNU_PACKED::trnCstNo

own TCN consist number (= 1.

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

Sequence number of consist in train (1.

.32)

.63)

4.1.2.8 UINT16 GNU_PACKED::reserved02

reserved (=0)

reserved (= 0)

reserved for future use (= 0)

4.1.2.9 UINT8 GNU_PACKED::ownOpCstNo

own operational address (= 1.

operational consist number the vehicle belongs to

.32) = 0 if unknown (e.g. after Inauguration)

4.1.2.10 UINT8 GNU_PACKED::reserved03

reserved (=0)

reserved for future use (= 0)

4.1.2.11 UINT8 GNU_PACKED::leadVehOfCst

position of leading vehicle in consist, 0.

position of leading vehicle in consist range 0.

.31 (1: first vehicle in consist in Direction 1, 2: second vehicle, etc.)

..32 0 = not defined 1 = first vehicle in consist in direction 1 2 = second vehicle etc.

4.1.2.12 UINT8 GNU_PACKED::reserved04

reserved (=0)

reserved for future use (= 0)

4.1.2.13 UINT8 GNU_PACKED::reserved06

reserved (=0)

reserved for future use (= 0)

4.1.2.14 UINT8 GNU_PACKED::confVehCnt

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

4.1.2.15 TRDP_ETB_CTRL_VDP_T GNU_PACKED::safetyTrail

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

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

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

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

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

4.1.2.16 UINT8 GNU_PACKED::reserved01

reserved (=0)

reserved for future use (= 0)

4.1.2.17 TRDP_LABEL_T GNU_PACKED::deviceName

function device of ECSC which sends the telegram

function device of ED which sends the telegram

4.1.2.18 UINT8 GNU_PACKED::inhibit

inauguration inhibit 0 = no inhibit request 1 = inhibit request

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

4.1.2.19 UINT16 GNU_PACKED::lifesign

wrap-around counter, incremented with each produced datagram.

4.1.2.20 UINT8 GNU_PACKED::etbInhibit

inauguration inhibit indication 0 = n/a (default) 1 = inhibit not requested on ETB 2 = inhibit set on local ETBN 3 = inhibit set on remote ETBN 4 = inhibit set on local and remote ETBN

inauguration inhibit indication 0 = n/a (default) 1 = inhibit not requested on ETB 2 = inhibit set on local ETBN 3 = inhibit set on remote ETBN 4 = inhibit set on local and remote ETBN

4.1.2.21 UINT8 GNU_PACKED::etbLength

indicates train lengthening in case train inauguration is inhibit 0 = no lengthening (default) 1 = lengthening detected

indicates train lengthening in case train inauguration is inhibit 0 = no lengthening (default) 1 = lengthening detected

4.1.2.22 UINT8 GNU_PACKED::etbShort

indicates train shortening in case train inauguration is inhibit 0 = no shortening (default) 1 = shortening detected

indicates train shortening in case train inauguration is inhibit 0 = no shortening (default) 1 = shortening detected

4.1.2.23 UINT16 GNU_PACKED::reserved02

reserved (=0)

reserved (= 0)

4.1.2.24 UINT8 GNU_PACKED::trnDirState

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

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

4.1.2.25 UINT8 GNU_PACKED::opTrnDirState

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

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

4.1.2.26 UINT8 GNU_PACKED::sleepReqCnt

number of sleep requests (option) value range: 0.

.63, not used = 0

4.1.2.27 UINT32 GNU_PACKED::opTrnTopoCnt

operational train topology counter

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

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

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

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

4.1.2.28 UINT16 GNU_PACKED::confVehCnt

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

4.1.2.29 TRDP_OP_VEHICLE_T GNU_PACKED::confVehList[TRDP_MAX_VEH_CNT]

ordered list of confirmed vehicles in the train, starting with vehicle at train head, see chapter 5.3.3.2.10.
Parameters 'isLead' and 'leadDir' to be set to 0

4.1.2.30 UINT32 GNU_PACKED::etbTopoCnt

ETB topography counter.
set by user: ETB to use, '0' for consist local traffic
train network directory CRC

4.1.2.31 TRDP_TRAIN_NET_DIR_T GNU_PACKED::trnNetDir

dynamic train info
network directory

4.1.2.32 TRDP_UUID_T GNU_PACKED::cstUUID

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

4.1.2.33 UINT8 GNU_PACKED::cstCnt

number of consists in train; range: 1.
.63
.63

4.1.2.34 TRDP_CONSIST_T GNU_PACKED::cstList

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

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

4.1.2.35 `UINT32 GNU_PACKED::trnTopoCnt`

`trnTopoCnt` value `ctrlType == 0`: actual value `ctrlType == 1`: set to 0
computed as defined in 5.3.3.2.16 (seed value: `etbTopoCnt`)

4.1.2.36 `UINT8 GNU_PACKED::etbId`

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

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

4.1.2.37 `TRDP_LABEL_T GNU_PACKED::vehId`

Unique vehicle identifier, application defined (e.g.
UIC Identifier)

4.1.2.38 `UINT8 GNU_PACKED::opVehNo`

operational vehicle sequence number in train value range 1.
.63

4.1.2.39 `UINT8 GNU_PACKED::opCstNo`

operational consist number in train (1.
.63)

4.1.2.40 `TRDP_LABEL_T GNU_PACKED::trnId`

train identifier, application defined (e.g.
'ICE75', 'IC346'), informal

4.1.2.41 `TRDP_LABEL_T GNU_PACKED::trnOperator`

train operator, e.g.
'trenitalia.it', informal

4.1.2.42 `UINT8 GNU_PACKED::opCstCnt`

number of consists in train (1.

.63)

4.1.2.43 TRDP_OP_CONSIST_T GNU_PACKED::opCstList[TRDP_MAX_CST_CNT]

operational consist list starting with op.

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

4.1.2.44 UINT8 GNU_PACKED::opVehCnt

number of vehicles in train (1.

.63)

4.1.2.45 TRDP_OP_VEHICLE_T GNU_PACKED::opVehList[TRDP_MAX_CST_CNT]

operational vehicle list starting with op.

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

4.1.2.46 UINT32 GNU_PACKED::cstNetProp

consist network properties bit0.

.1: consist orientation bit2..7: 0 bit8..13: ETBN Id bit14..15: 0 bit16..21: subnet Id bit24..29: CN Id bit30..31: 0

4.1.2.47 UINT16 GNU_PACKED::protocolVersion

fix value for compatibility (set by the API)

fix value for compatibility

4.1.2.48 UINT16 GNU_PACKED::msgType

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

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

4.1.2.49 UINT32 GNU_PACKED::datasetLength

length of the data to transmit 0.

defined by user: length of data to transmit

..1432

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

- [tau_ctrl_types.h](#)
- [tau_tti_types.h](#)
- [trdp_proto.h](#)

4.2 TRDP_CLTR_CST_INFO_T Struct Reference

Closed train consists information.

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

Data Fields

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

4.2.1 Detailed Description

Closed train consists information.

4.2.2 Field Documentation

4.2.2.1 [UINT8 TRDP_CLTR_CST_INFO_T::cltrCstNo](#)

sequence number of the consist within the closed train, value range 1.

.32

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

- [tau_tti_types.h](#)

4.3 TRDP_COMID_DSID_MAP_T Struct Reference

ComId - data set mapping element definition.

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

Data Fields

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

4.3.1 Detailed Description

ComId - data set mapping element definition.

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

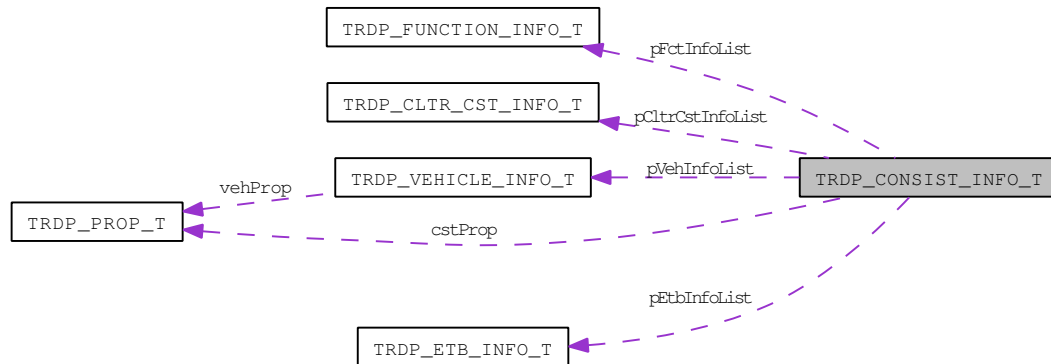
- [trdp_types.h](#)

4.4 TRDP_CONSIST_INFO_T Struct Reference

consist information structure

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

Collaboration diagram for TRDP_CONSIST_INFO_T:



Data Fields

- `TRDP_SHORT_VERSION_T` [version](#)
ConsistInfo data structure version, application defined mainVersion = 1, subVersion = 0.
- `UINT8` [cstClass](#)
consist info classification 1 = (single) consist 2 = closed train 3 = closed train consist
- `UINT8` [reserved01](#)
reserved for future use (= 0)
- `TRDP_LABEL_T` [cstId](#)
application defined consist identifier, e.g.
- `TRDP_LABEL_T` [cstType](#)
consist type, application defined
- `TRDP_LABEL_T` [cstOwner](#)
consist owner, e.g.
- `TRDP_UUID_T` [cstUUID](#)
consist UUID
- `UINT32` [reserved02](#)
reserved for future use (= 0)
- `TRDP_PROP_T` [cstProp](#)
static consist properties
- `UINT16` [reserved03](#)

reserved for future use (= 0)

- **UINT16 etbCnt**
number of ETB's, range: 1.
- **TRDP_ETB_INFO_T * pEtbInfoList**
ETB information list for the consist Ordered list starting with lowest etbId.
- **UINT16 reserved04**
reserved for future use (= 0)
- **UINT16 vehCnt**
number of vehicles in consist 1.
- **TRDP_VEHICLE_INFO_T * pVehInfoList**
vehicle info list for the vehicles in the consist Ordered list starting with cstVehNo==1
- **UINT16 reserved05**
reserved for future use (= 0)
- **UINT16 fctCnt**
number of consist functions value range 0.
- **TRDP_FUNCTION_INFO_T * pFctInfoList**
function info list for the functions in consist lexicographical ordered by fctName
- **UINT16 reserved06**
reserved for future use (= 0)
- **UINT16 cltrCstCnt**
number of original consists in closed train value range: 0.
- **TRDP_CLTR_CST_INFO_T * pCltrCstInfoList**
info on closed train composition Ordered list starting with cltrCstNo == 1
- **UINT32 cstTopoCnt**
consist topology counter computed as defined in 5.3.3.2.16, seed value: 'FFFFFFFF'H

4.4.1 Detailed Description

consist information structure

4.4.2 Field Documentation

4.4.2.1 TRDP_LABEL_T TRDP_CONSIST_INFO_T::cstId

application defined consist identifier, e.g.

UIC identifier

4.4.2.2 TRDP_LABEL_T TRDP_CONSIST_INFO_T::cstOwner

consist owner, e.g.

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

4.4.2.3 UINT16 TRDP_CONSIST_INFO_T::etbCnt

number of ETB's, range: 1.

.4

4.4.2.4 UINT16 TRDP_CONSIST_INFO_T::vehCnt

number of vehicles in consist 1.

.32

4.4.2.5 UINT16 TRDP_CONSIST_INFO_T::fctCnt

number of consist functions value range 0.

.1024

4.4.2.6 UINT16 TRDP_CONSIST_INFO_T::cltrCstCnt

number of original consists in closed train value range: 0.

.32, 0 = consist is no closed train

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

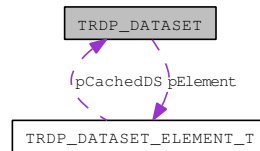
- [tau_tti_types.h](#)

4.5 TRDP_DATASET Struct Reference

Dataset definition.

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

Collaboration diagram for TRDP_DATASET:



Data Fields

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

4.5.1 Detailed Description

Dataset definition.

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

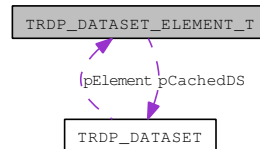
- [trdp_types.h](#)

4.6 TRDP_DATASET_ELEMENT_T Struct Reference

Dataset element definition.

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

Collaboration diagram for TRDP_DATASET_ELEMENT_T:



Data Fields

- **UINT32** [type](#)
Data type (TRDP_DATA_TYPE_T 1).
- **UINT32** [size](#)
Number of items or TDRP_VAR_SIZE (0).
- **CHAR8 *** [unit](#)
Unit text for visualisation.
- **REAL32** [scale](#)
Factor for visualisation.
- **INT32** [offset](#)
*Offset for visualisation ($val = scale * x + offset$).*
- **struct** [TRDP_DATASET](#) * [pCachedDS](#)
Used internally for marshallng speed-up.

4.6.1 Detailed Description

Dataset element definition.

4.6.2 Field Documentation

4.6.2.1 **UINT32** TRDP_DATASET_ELEMENT_T::type

Data type (TRDP_DATA_TYPE_T 1.

..99) or dataset id > 1000

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

- [trdp_types.h](#)

4.7 TRDP_DBG_CONFIG_T Struct Reference

Control for debug output device/file on application level.

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

Data Fields

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

4.7.1 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](#)

4.8 TRDP_ETB_INFO_T Struct Reference

Types for train configuration information.

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

Data Fields

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

4.8.1 Detailed Description

Types for train configuration information.

ETB information

4.8.2 Field Documentation

4.8.2.1 **UINT8 TRDP_ETB_INFO_T::etbId**

identification of train backbone; value range: 0.

.3

4.8.2.2 **UINT8 TRDP_ETB_INFO_T::cnCnt**

number of CNs within consist connected to this ETB value range 1.

.16 referring to cnId 0..15 acc. IEC61375-2-5

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

- [tau_tti_types.h](#)

4.9 TRDP_FUNCTION_INFO_T Struct Reference

function/device information structure

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

Data Fields

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

4.9.1 Detailed Description

function/device information structure

4.9.2 Field Documentation

4.9.2.1 UINT16 TRDP_FUNCTION_INFO_T::fctId

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

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

4.9.2.2 UINT8 TRDP_FUNCTION_INFO_T::cstVehNo

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

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

4.9.2.3 `UINT8 TRDP_FUNCTION_INFO_T::etbId`

number of connected train backbone.

Value range: 0..3

4.9.2.4 `UINT8 TRDP_FUNCTION_INFO_T::cnId`

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

Value range: 0..31

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

- [tau_tti_types.h](#)

4.10 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 if used.
- UINT32 [userRef](#)
User reference if used.
- UINT32 [numSessions](#)
Number of sessions.

4.10.1 Detailed Description

Information about a particular MD listener.

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

- [trdp_types.h](#)

4.11 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.

4.11.1 Detailed Description

Marshaling/unmarshalling configuration.

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

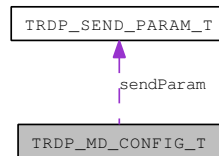
- [trdp_types.h](#)

4.12 TRDP_MD_CONFIG_T Struct Reference

Default MD configuration.

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

Collaboration diagram for TRDP_MD_CONFIG_T:



Data Fields

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

4.12.1 Detailed Description

Default MD configuration.

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

- [trdp_types.h](#)

4.13 TRDP_MD_INFO_T Struct Reference

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

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

Data Fields

- [TRDP_IP_ADDR_T srcIpAddr](#)
source IP address for filtering
- [TRDP_IP_ADDR_T destIpAddr](#)
destination IP address for filtering
- [UINT32 seqCount](#)
sequence counter
- [UINT16 protVersion](#)
Protocol version.
- [TRDP_MSG_T msgType](#)
Protocol ('PD', 'MD', .
- [UINT32 comId](#)
ComID.
- [UINT32 etbTopoCnt](#)
received topocount
- [UINT32 opTrnTopoCnt](#)
received topocount
- [BOOL8 aboutToDie](#)
session is about to die
- [UINT32 numRepliesQuery](#)
number of ReplyQuery received
- [UINT32 numConfirmSent](#)
number of Confirm sent
- [UINT32 numConfirmTimeout](#)
number of Confirm Timeouts (incremented by listeners
- [UINT16 userStatus](#)
error code, user stat
- [TRDP_REPLY_STATUS_T replyStatus](#)
reply status

- [TRDP_UUID_T sessionId](#)
for response
- [UINT32 replyTimeout](#)
reply timeout in us given with the request
- [TRDP_URI_USER_T srcUserURI](#)
source URI user part from MD header
- [TRDP_URI_HOST_T srcHostURI](#)
source URI host part (unused)
- [TRDP_URI_USER_T destUserURI](#)
destination URI user part from MD header
- [TRDP_URI_HOST_T destHostURI](#)
destination URI host part (unused)
- [UINT32 numExpReplies](#)
number of expected replies, 0 if unknown
- [UINT32 numReplies](#)
actual number of replies for the request
- `const void *` [pUserRef](#)
User reference given with the local call.
- [TRDP_ERR_T resultCode](#)
error code

4.13.1 Detailed Description

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

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

4.13.2 Field Documentation

4.13.2.1 [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](#)

4.14 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

4.14.1 Detailed Description

Structure containing all general MD statistics information.

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

- [trdp_types.h](#)

4.15 TRDP_MEM_CONFIG_T Struct Reference

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

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

Data Fields

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

4.15.1 Detailed Description

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

Structure describing memory (and its pre-fragmentation)

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

- [trdp_types.h](#)

4.16 TRDP_MEM_STATISTICS_T Struct Reference

TRDP statistics type definitions.

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

Data Fields

- UINT32 [total](#)
total memory size
- UINT32 [free](#)
free memory size
- UINT32 [minFree](#)
minimal free memory size in statistics interval
- UINT32 [numAllocBlocks](#)
allocated memory blocks
- UINT32 [numAllocErr](#)
allocation errors
- UINT32 [numFreeErr](#)
free errors
- UINT32 [blockSize](#) [VOS_MEM_NBLOCKSIZES]
preallocated memory blocks
- UINT32 [usedBlockSize](#) [VOS_MEM_NBLOCKSIZES]
used memory blocks

4.16.1 Detailed Description

TRDP statistics type definitions.

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

- PD subscr table: ComId, sourceIpAddr, destIpAddr, cbFct?, timeout, toBehavior, counter
- PD publish table: ComId, destIpAddr, redId, redState cycle, ttl, qos, counter
- PD join table: joined MC address table
- MD listener table: ComId destIpAddr, destUri, cbFct?, counter
- Memory usage Structure containing all general memory statistics information.

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

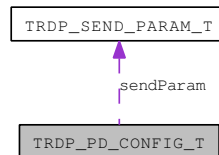
- [trdp_types.h](#)

4.17 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 behavior.
- `UINT16` [port](#)
Port to be used for PD communication.

4.17.1 Detailed Description

Default PD configuration.

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

- [trdp_types.h](#)

4.18 TRDP_PD_INFO_T Struct Reference

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

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

Data Fields

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

4.18.1 Detailed Description

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

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

4.18.2 Field Documentation

4.18.2.1 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](#)

4.19 TRDP_PD_STATISTICS_T Struct Reference

Structure containing all general PD statistics information.

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

Data Fields

- UINT32 [defQos](#)
default QoS for PD
- UINT32 [defTtl](#)
default TTL for PD
- UINT32 [defTimeout](#)
default timeout in us for PD
- UINT32 [numSubs](#)
number of subscribed ComId's
- UINT32 [numPub](#)
number of published ComId's
- UINT32 [numRcv](#)
number of received PD packets
- UINT32 [numCrcErr](#)
number of received PD packets with CRC err
- UINT32 [numProtErr](#)
number of received PD packets with protocol err
- UINT32 [numTopoErr](#)
number of received PD packets with wrong topo count
- UINT32 [numNoSubs](#)
number of received PD push packets without subscription
- UINT32 [numNoPub](#)
number of received PD pull packets without publisher
- UINT32 [numTimeout](#)
number of PD timeouts
- UINT32 [numSend](#)
number of sent PD packets
- UINT32 [numMissed](#)
number of packets skipped

4.19.1 Detailed Description

Structure containing all general PD statistics information.

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

- [trdp_types.h](#)

4.20 TRDP_PROCESS_CONFIG_T Struct Reference

Various flags/general TRDP options for library initialization.

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

Data Fields

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

4.20.1 Detailed Description

Various flags/general TRDP options for library initialization.

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

- [trdp_types.h](#)

4.21 TRDP_PROP_T Struct Reference

Application defined properties.

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

Data Fields

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

4.21.1 Detailed Description

Application defined properties.

4.21.2 Field Documentation

4.21.2.1 UINT16 TRDP_PROP_T::len

properties length in number of octets, application defined, must be a multiple of 4 octets for alignment reasons value range: 0.

.32768

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

- [tau_tti_types.h](#)

4.22 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.

4.22.1 Detailed Description

Table containing particular PD publishing information.

4.22.2 Field Documentation

4.22.2.1 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](#)

4.23 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.

4.23.1 Detailed Description

A table containing PD redundant group information.

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

- [trdp_types.h](#)

4.24 TRDP_SDT_PAR_T Struct Reference

Types to read out the XML configuration.

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

Data Fields

- [UINT32 smi1](#)
Safe message identifier - unique for this message at consist level.
- [UINT32 smi2](#)
Safe message identifier - unique for this message at consist level.
- [UINT32 cmThr](#)
Channel monitoring threshold.
- [UINT16 udv](#)
User data version.
- [UINT16 rxPeriod](#)
Sink cycle time.
- [UINT16 txPeriod](#)
Source cycle time.
- [UINT16 nGuard](#)
Initial timeout cycles.
- [UINT8 nrxSafe](#)
Timeout cycles.
- [UINT8 reserved1](#)
Reserved for future use.
- [UINT16 reserved2](#)
Reserved for future use.

4.24.1 Detailed Description

Types to read out the XML configuration.

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

- [tau_xml.h](#)

4.25 TRDP_SEND_PARAM_T Struct Reference

Quality/type of service and time to live.

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

Data Fields

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

4.25.1 Detailed Description

Quality/type of service and time to live.

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

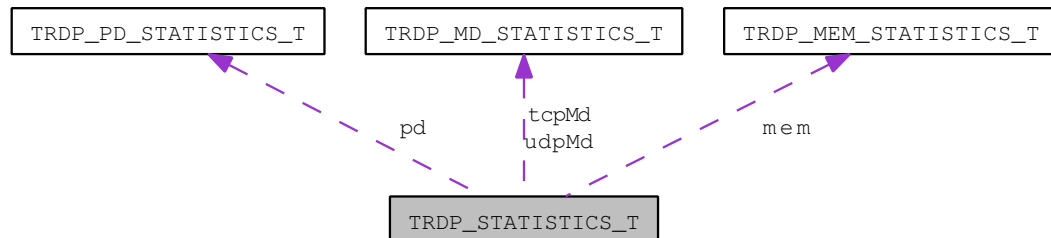
- [trdp_types.h](#)

4.26 TRDP_STATISTICS_T Struct Reference

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

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

Collaboration diagram for TRDP_STATISTICS_T:



Data Fields

- `UINT32` [version](#)
TRDP version.
- `TIMEDATE64` [timeStamp](#)
actual time stamp
- `TIMEDATE32` [upTime](#)
time in sec since last initialisation
- `TIMEDATE32` [statisticTime](#)
time in sec since last reset of statistics
- `TRDP_LABEL_T` [hostName](#)
host name
- `TRDP_LABEL_T` [leaderName](#)
leader host name
- `TRDP_IP_ADDR_T` [ownIpAddr](#)
own IP address
- `TRDP_IP_ADDR_T` [leaderIpAddr](#)
leader IP address
- `UINT32` [processPrio](#)
priority of TRDP process
- `UINT32` [processCycle](#)
cycle time of TRDP process in microseconds
- `UINT32` [numJoin](#)

number of joins

- [UINT32 numRed](#)
number of redundancy groups
- [TRDP_MEM_STATISTICS_T mem](#)
memory statistics
- [TRDP_PD_STATISTICS_T pd](#)
pd statistics
- [TRDP_MD_STATISTICS_T udpMd](#)
UDP md statistics.
- [TRDP_MD_STATISTICS_T tcpMd](#)
TCP md statistics.

4.26.1 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](#)

4.27 TRDP_SUBS_STATISTICS_T Struct Reference

Table containing particular PD subscription information.

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

Data Fields

- [UINT32 comId](#)
Subscribed ComId.
- [TRDP_IP_ADDR_T joinedAddr](#)
Joined IP address.
- [TRDP_IP_ADDR_T filterAddr](#)
Filter IP address, i.e IP address of the sender for this subscription, 0.0.0.0 in case all senders.
- [UINT32 callBack](#)
call back function if used
- [UINT32 userRef](#)
User reference if used.
- [UINT32 timeout](#)
Time-out value in us.
- [TRDP_ERR_T status](#)
Receive status information TRDP_NO_ERR, TRDP_TIMEOUT_ERR.
- [TRDP_TO_BEHAVIOR_T toBehav](#)
Behavior at time-out.
- [UINT32 numRecv](#)
Number of packets received for this subscription.
- [UINT32 numMissed](#)
number of packets skipped for this subscription

4.27.1 Detailed Description

Table containing particular PD subscription information.

4.27.2 Field Documentation

4.27.2.1 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.

4.27.2.2 UINT32 TRDP_SUBS_STATISTICS_T::timeout

Time-out value in us.

0 = No time-out supervision

4.27.2.3 TRDP_TO_BEHAVIOR_T TRDP_SUBS_STATISTICS_T::toBehav

Behavior at time-out.

Set data to zero / keep last value

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

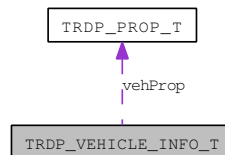
- [trdp_types.h](#)

4.28 TRDP_VEHICLE_INFO_T Struct Reference

vehicle information structure

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

Collaboration diagram for TRDP_VEHICLE_INFO_T:



Data Fields

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

4.28.1 Detailed Description

vehicle information structure

4.28.2 Field Documentation

4.28.2.1 TRDP_LABEL_T TRDP_VEHICLE_INFO_T::vehId

vehicle identifier label,application defined (e.g.

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

4.28.2.2 UINT8 TRDP_VEHICLE_INFO_T::cstVehNo

Sequence number of vehicle in consist(1.

.16)

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

- [tau_tti_types.h](#)

4.29 TRDP_XML_DOC_HANDLE_T Struct Reference

Parsed XML document handle.

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

Data Fields

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

4.29.1 Detailed Description

Parsed XML document handle.

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

- [tau_xml.h](#)

4.30 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
- [BOOL8 reuseAddrPort](#)
allow reuse of address and port
- [BOOL8 nonBlocking](#)
use non blocking calls
- [BOOL8 no_mc_loop](#)
no multicast loop back
- [BOOL8 no_udp_crc](#)
supress udp crc computation

4.30.1 Detailed Description

Common socket options.

4.30.2 Field Documentation

4.30.2.1 [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](#)

4.31 VOS_TIME_T Struct Reference

Timer value compatible with timeval / select.

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

Data Fields

- UINT32 [tv_sec](#)
full seconds
- INT32 [tv_usec](#)
Micro seconds (max.

4.31.1 Detailed Description

Timer value compatible with timeval / select.

Relative or absolute date, depending on usage

4.31.2 Field Documentation

4.31.2.1 INT32 VOS_TIME_T::tv_usec

Micro seconds (max.

value 999999)

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

- [vos_types.h](#)

4.32 VOS_VERSION_T Struct Reference

Version information.

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

Data Fields

- `UINT8 ver`
Version - incremented for incompatible changes.
- `UINT8 rel`
Release - incremented for compatible changes.
- `UINT8 upd`
Update - incremented for bug fixes.
- `UINT8 evo`
Evolution - incremented for build.

4.32.1 Detailed Description

Version information.

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

- [vos_types.h](#)

Chapter 5

File Documentation

5.1 iec61375-2-3.h File Reference

TTDB, CSTINFO Frame typedefs, Telegram definitions.

Defines

- #define [ETB_CTRL_COMID](#) 1
ETB Control telegram.
- #define [ETB_CTRL_CYC](#) 500
0.5s
- #define [ETB_CTRL_TO](#) 3000
3s
- #define [CSTINFO_COMID](#) 2
Consist Info telegram (Message data notification 'Mn').
- #define [CSTINFOCTRL_COMID](#) 3
Consist Info control/request telegram (Message data notification 'Mn').
- #define [TTDB_STATUS_COMID](#) 100
TTDB manager telegram PD.
- #define [TTDB_STATUS_CYC](#) 1000
Push.
- #define [TTDB_STATUS_TO](#) 5000
5s
- #define [TTDB_OP_DIR_INFO_COMID](#) 101
TTDB manager telegram MD: Push the OP_TRAIN_DIRECTORY.
- #define [TTDB_OP_DIR_INFO_DS](#) "TTDB_OP_TRAIN_DIR_INFO"

OP_TRAIN_DIRECTORY.

- #define [TTDB_TRN_DIR_REQ_COMID](#) 102
TTDB manager telegram MD: Get the TRAIN_DIRECTORY.
- #define [TTDB_TRN_DIR_REQ_TO](#) 3000
3s timeout
- #define [TTDB_TRN_DIR_REP_COMID](#) 103
MD reply.
- #define [TTDB_TRN_DIR_REP_DS](#) "TTDB_TRAIN_DIRECTORY_INFO_REPLY"
TRAIN_DIRECTORY.
- #define [TTDB_STAT_CST_REQ_COMID](#) 104
TTDB manager telegram MD: Get the static consist information.
- #define [TTDB_STAT_CST_REQ_TO](#) 3000
3s timeout
- #define [TTDB_STAT_CST_REP_DS](#) "TTDB_STATIC_CONSIST_INFO_REPLY"
CONSIST_INFO.
- #define [TTDB_NET_DIR_REQ_COMID](#) 106
TTDB manager telegram MD: Get the NETWORK_TRAIN_DIRECTORY.
- #define [TTDB_NET_DIR_REQ_TO](#) 3000
3s timeout
- #define [TTDB_NET_DIR_REP_COMID](#) 107
MD reply.
- #define [TTDB_NET_DIR_REP_DS](#) "TTDB_TRAIN_NETWORK_DIRECTORY_INFO_REPLY"

TRAIN_NETWORK_DIRECTORY.
- #define [TTDB_OP_DIR_INFO_REQ_COMID](#) 108
TTDB manager telegram MD: Get the OP_TRAIN_DIRECTORY.
- #define [TTDB_OP_DIR_INFO_REQ_TO](#) 3000
3s timeout
- #define [TTDB_OP_DIR_INFO_REP_DS](#) "TTDB_OP_TRAIN_DIR_INFO"
OP_TRAIN_DIRECTORY.
- #define [TTDB_READ_CMPLT_REQ_COMID](#) 110
TTDB manager telegram MD: Get the TTDB.
- #define [TTDB_READ_CMPLT_REQ_DS](#) "TTDB_READ_COMPLETE_REQUEST"
ETBx.

- #define [TTDB_READ_CMPLT_REQ_TO](#) 3000
3s timeout
- #define [TTDB_READ_CMPLT_REP_COMID](#) 111
MD reply.
- #define [TTDB_READ_CMPLT_REP_DS](#) "TTDB_READ_COMPLETE_REPLY"
TRDP_READ_COMPLETE_REPLY_T.
- #define [ECSP_CTRL_COMID](#) 120
ECSP Control telegram.
- #define [ECSP_CTRL_CYC](#) 1000
1s
- #define [ECSP_CTRL_TO](#) 5000
5s
- #define [ECSP_CTRL_DEST_URI](#) "devECSP.anyVeh.ICst"
10.0.0.1
- #define [ECSP_STATUS_COMID](#) 121
ECSP status telegram.
- #define [ECSP_STATUS_CYC](#) 1000
1s
- #define [ECSP_STATUS_TO](#) 5000
5s
- #define [ECSP_STATUS_DEST_URI](#) "devECSC.anyVeh.ICst"
10.0.0.100
- #define [ETBN_STATUS_COMID](#) 122
ETBN STATUS Telegram PD.
- #define [ETBN_STATUS_CYC](#) 1000
1s cycle time
- #define [ETBN_STATUS_TO](#) 5000
5s timeout
- #define [ETBN_CTRL_REQ_COMID](#) 130
ETBN Control Telegram MD.
- #define [ETBN_CTRL_REQ_DS](#) "ETBN_CTRL"
ETBx.
- #define [ETBN_CTRL_REQ_TO](#) 3000

3s timeout

- #define [ETBN_CTRL_REP_DS](#) "ETBN_STATUS"
ETBN status reply.
- #define [ETBN_TRN_NET_DIR_REQ_COMID](#) 132
ETBN Control Telegram MD.
- #define [ETBN_TRN_NET_DIR_REQ_TO](#) 3000
3s timeout
- #define [ETBN_TRN_NET_DIR_REP_DS](#) "ETBN_TRAIN_NETWORK_DIRECTORY_INFO_-REPLY"
ETBx.

5.1.1 Detailed Description

TTDB, CSTINFO Frame typedefs, Telegram definitions.

Note:

Project: TCNOpen TRDP

Author:

Bernd Loehr, NewTec GmbH, 2015-09-11

Remarks:

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

Id

[iec61375-2-3.h](#) 1522 2016-03-01 10:17:09Z bloehr

5.1.2 Define Documentation

5.1.2.1 #define ETBN_STATUS_COMID 122

ETBN STATUS Telegram PD.

tbd!

5.1.2.2 #define TTDB_NET_DIR_REQ_COMID 106

TTDB manager telegram MD: Get the NETWORK_TRAIN_DIRECTORY.

MD request

5.1.2.3 #define TTDB_OP_DIR_INFO_COMID 101

TTDB manager telegram MD: Push the OP_TRAIN_DIRECTORY.

MD notification

5.1.2.4 #define TTDB_STAT_CST_REQ_COMID 104

TTDB manager telegram MD: Get the static consist information.

MD request

5.1.2.5 #define TTDB_TRN_DIR_REQ_COMID 102

TTDB manager telegram MD: Get the TRAIN_DIRECTORY.

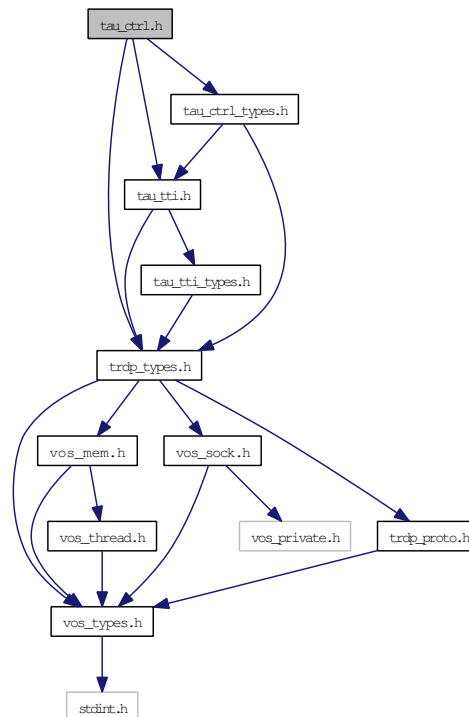
MD request

5.2 tau_ctrl.h File Reference

TRDP utility interface definitions.

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

Include dependency graph for tau_ctrl.h:



Functions

- EXT_DECL [TRDP_ERR_T tau_initEcspCtrl](#) (TRDP_APP_SESSION_T appHandle, [TRDP_IP_ADDR_T](#) ecspIpAddr)
Function to init ECSP control interface.
- EXT_DECL [TRDP_ERR_T tau_terminateEcspCtrl](#) (TRDP_APP_SESSION_T appHandle)
Function to close ECSP control interface.
- EXT_DECL [TRDP_ERR_T tau_setEcspCtrl](#) (TRDP_APP_SESSION_T appHandle, TRDP_ECSP_CTRL_T *pEcspCtrl)
Function to set ECSP control information.
- EXT_DECL [TRDP_ERR_T tau_getEcspStat](#) (TRDP_APP_SESSION_T appHandle, TRDP_ECSP_STAT_T *pEcspStat, [TRDP_PD_INFO_T](#) *pPdInfo)
Function to get ECSP status information.

- EXT_DECL [TRDP_ERR_T tau_requestEcsConf](#) (TRDP_APP_SESSION_T appHandle, const void *pUserRef, [TRDP_MD_CALLBACK_T](#) pfCbFunction, TRDP_ECSP_CONF_REQUEST_T *pEcsConfRequest)

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

5.2.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- ETB control

Note:

Project: TCNOpen TRDP prototype stack

Author:

Armin-H. Weiss (initial version)

Remarks:

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

Id

[tau_ctrl.h](#) 1483 2015-12-16 14:43:30Z bloehr

5.2.2 Function Documentation

5.2.2.1 EXT_DECL TRDP_ERR_T tau_getEcsStat (TRDP_APP_SESSION_T appHandle, TRDP_ECSP_STAT_T *pEcsStat, TRDP_PD_INFO_T *pPdInfo)

Function to get ECSP status information.

Parameters:

- ← *appHandle* Application Handle
- ↔ *pEcsStat* Pointer to the ECSP status structure
- ↔ *pPdInfo* Pointer to PD status information

Return values:

- TRDP_NO_ERR* no error
- TRDP_NOINIT_ERR* module not initialised
- TRDP_PARAM_ERR* Parameter error

5.2.2.2 EXT_DECL TRDP_ERR_T tau_initEcsCtrl (TRDP_APP_SESSION_T *appHandle*, TRDP_IP_ADDR_T *ecspIpAddr*)

Function to init ECSP control interface.

Parameters:

- ← *appHandle* Application handle
- ← *ecspIpAddr* ECSP address

Return values:

- TRDP_NO_ERR* no error
- TRDP_INIT_ERR* initialisation error

5.2.2.3 EXT_DECL TRDP_ERR_T tau_requestEcsConfirm (TRDP_APP_SESSION_T *appHandle*, const void * *pUserRef*, TRDP_MD_CALLBACK_T *pfCbFunction*, TRDP_ECSP_CONF_REQUEST_T * *pEcsConfRequest*)

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

Parameters:

- ← *appHandle* Application Handle
- ← *pUserRef* user reference returned with reply
- ← *pfCbFunction* Pointer to callback function, NULL for default
- ← *pEcsConfRequest* Pointer to confirmation data

Return values:

- TRDP_NO_ERR* no error
- TRDP_NOINIT_ERR* module not initialised
- TRDP_PARAM_ERR* Parameter error

5.2.2.4 EXT_DECL TRDP_ERR_T tau_setEcsCtrl (TRDP_APP_SESSION_T *appHandle*, TRDP_ECSP_CTRL_T * *pEcsCtrl*)

Function to set ECSP control information.

Parameters:

- ← *appHandle* Application handle
- ← *pEcsCtrl* Pointer to the ECSP control structure

Return values:

- TRDP_NO_ERR* no error
- TRDP_NOINIT_ERR* module not initialised
- TRDP_PARAM_ERR* Parameter error

5.2.2.5 EXT_DECL TRDP_ERR_T tau_terminateEcspCtrl (TRDP_APP_SESSION_T *appHandle*)

Function to close ECSP control interface.

Parameters:

← *appHandle* Application handle

Return values:

TRDP_NO_ERR no error

TRDP_UNKNOWN_ERR undefined error

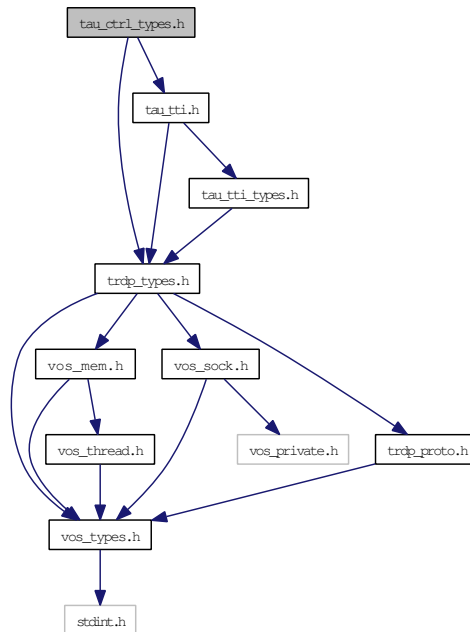
5.3 tau_ctrl_types.h File Reference

TRDP utility interface definitions.

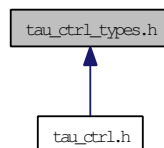
```
#include "trdp_types.h"
```

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

Include dependency graph for tau_ctrl_types.h:



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



Data Structures

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

Types for ETB control.

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

5.3.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following

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

Note:

Project: TCNOpen TRDP prototype stack

Author:

Armin-H. Weiss (initial version)

Remarks:

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

Id

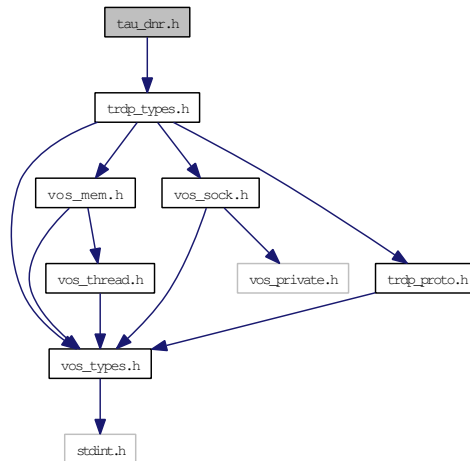
[tau_ctrl_types.h](#) 1510 2016-02-17 14:03:45Z bloehr

5.4 tau_dnr.h File Reference

TRDP utility interface definitions.

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

Include dependency graph for tau_dnr.h:



Functions

- EXT_DECL [TRDP_ERR_T](#) [tau_initDnr](#) (TRDP_APP_SESSION_T appHandle, [TRDP_IP_ADDR_T](#) dnsIpAddr, UINT16 dnsPort, const CHAR8 *hostsFileName)
Function to init DNR.
- EXT_DECL void [tau_deInitDnr](#) (TRDP_APP_SESSION_T appHandle)
Release any resources allocated by DNR.
- EXT_DECL TRDP_DNR_STATE_T [tau_DNRstatus](#) (TRDP_APP_SESSION_T appHandle)
Function to get the status of DNR.
- EXT_DECL [TRDP_ERR_T](#) [tau_getOwnIds](#) (TRDP_APP_SESSION_T appHandle, TRDP_LABEL_T devId, TRDP_LABEL_T vehId, TRDP_LABEL_T cstId)
Who am I ?.
- EXT_DECL [TRDP_IP_ADDR_T](#) [tau_getOwnAddr](#) (TRDP_APP_SESSION_T appHandle)
Function to get the own IP address.
- EXT_DECL [TRDP_ERR_T](#) [tau_uri2Addr](#) (TRDP_APP_SESSION_T appHandle, [TRDP_IP_ADDR_T](#) *pAddr, const TRDP_URI_T pUri)
Function to convert a URI to an IP address.
- EXT_DECL [TRDP_ERR_T](#) [tau_addr2Uri](#) (TRDP_APP_SESSION_T appHandle, TRDP_URI_HOST_T pUri, [TRDP_IP_ADDR_T](#) addr)
Function to convert an IP address to a URI.

5.4.1 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:

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

Id

[tau_dnr.h](#) 1522 2016-03-01 10:17:09Z bloehr

BL 2015-12-14: Ticket #8: DNR client

5.4.2 Function Documentation

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

Function to convert an IP address to a URI.

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

Parameters:

- ← *appHandle* Handle returned by [tlc_openSession\(\)](#).
- *pUri* Pointer to a string to return the URI host part
- ← *addr* IP address, 0==own address

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* Parameter error

5.4.2.2 EXT_DECL void tau_deInitDnr (TRDP_APP_SESSION_T *appHandle*)

Release any resources allocated by DNR.

Parameters:

← *appHandle* Handle returned by [tlc_openSession\(\)](#).

Return values:

none

5.4.2.3 EXT_DECL TRDP_DNR_STATE_T tau_DNRstatus (TRDP_APP_SESSION_T *appHandle*)

Function to get the status of DNR.

Parameters:

← *appHandle* Handle returned by [tlc_openSession\(\)](#)

Return values:

TRDP_DNR_NOT_AVAILABLE no error

TRDP_DNR_UNKNOWN enabled, but cache is empty

TRDP_DNR_ACTIVE enabled, cache has values

TRDP_DNR_HOSTSFILE enabled, hostsfile used (static mode)

5.4.2.4 EXT_DECL TRDP_IP_ADDR_T tau_getOwnAddr (TRDP_APP_SESSION_T *appHandle*)

Function to get the own IP address.

Parameters:

← *appHandle* Handle returned by [tlc_openSession\(\)](#).

Return values:

own IP address

5.4.2.5 EXT_DECL TRDP_ERR_T tau_getOwnIds (TRDP_APP_SESSION_T *appHandle*, TRDP_LABEL_T *devId*, TRDP_LABEL_T *vehId*, 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:

← *appHandle* Handle returned by [tlc_openSession\(\)](#).

→ *devId* Returns the device label (host name)

→ *vehId* Returns the vehicle label

→ *cstId* Returns the consist label

Return values:*TRDP_NO_ERR* no error*TRDP_PARAM_ERR* Parameter error**5.4.2.6 EXT_DECL TRDP_ERR_T tau_initDnr (TRDP_APP_SESSION_T *appHandle*,
TRDP_IP_ADDR_T *dnsIpAddr*, UINT16 *dnsPort*, const CHAR8 * *hostsFileName*)**

Function to init DNR.

Parameters:← *appHandle* Handle returned by [tlc_openSession\(\)](#).← *dnsIpAddr* DNS/ECSP IP address.← *dnsPort* DNS port number.← *hostsFileName* Optional host file name as ECSP replacement/addition.**Return values:***TRDP_NO_ERR* no error*TRDP_INIT_ERR* initialisation error**5.4.2.7 EXT_DECL TRDP_ERR_T tau_uri2Addr (TRDP_APP_SESSION_T *appHandle*,
TRDP_IP_ADDR_T * *pAddr*, const TRDP_URI_T *pUri*)**

Function to convert a URI to an IP address.

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

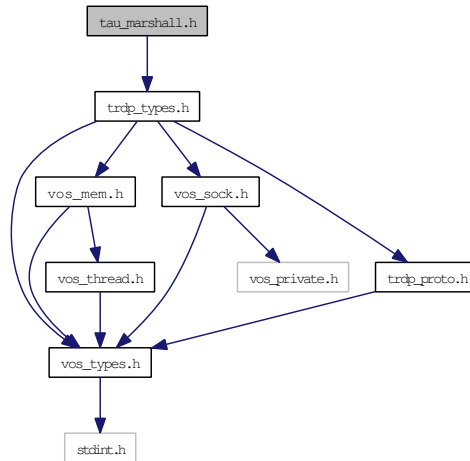
Parameters:← *appHandle* Handle returned by [tlc_openSession\(\)](#).→ *pAddr* Pointer to return the IP address← *pUri* Pointer to a URI or an IP Address string, NULL==own URI**Return values:***TRDP_NO_ERR* no error*TRDP_PARAM_ERR* Parameter error

5.5 tau_marshall.h File Reference

TRDP utility interface definitions.

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

Include dependency graph for tau_marshall.h:



Functions

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

Types for marshalling / unmarshalling.

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

marshall function.

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

marshall data set function.

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

unmarshall function.

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

unmarshall data set function.

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

Calculate data set size by given data set id.

- EXT_DECL [TRDP_ERR_T](#) [tau_calcDatasetSizeByComId](#) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT32 *pDestSize, [TRDP_DATASET_T](#) **ppDSPointer)
Calculate data set size by given ComId.

5.5.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- marshall/unmarshall

Note:

Project: TCNOpen TRDP prototype stack

Author:

Armin-H. Weiss

Remarks:

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

Id

[tau_marshall.h](#) 1479 2015-12-14 14:53:45Z bloehr

BL 2015-12-14: Ticket #33: source size check for marshall

5.5.2 Function Documentation

5.5.2.1 EXT_DECL TRDP_ERR_T tau_calcDatasetSize (void *pRefCon, UINT32 dsId, UINT8 *pSrc, UINT32 srcSize, UINT32 *pDestSize, TRDP_DATASET_T **ppDSPointer)

Calculate data set size by given data set id.

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_INIT_ERR* marshall not initialised
- TRDP_PARAM_ERR* data set id not existing

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

Calculate data set size by given ComId.

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_INIT_ERR* marshalling not initialised
- TRDP_PARAM_ERR* data set id not existing

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

Types for marshalling / unmarshalling.

Function to initialise the marshalling/unmarshalling.

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_MEM_ERR* provided buffer to small
- TRDP_PARAM_ERR* Parameter error

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

marshall function.

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_MEM_ERR* provided buffer to small
- TRDP_INIT_ERR* marshalling not initialised
- TRDP_COMID_ERR* comid not existing
- TRDP_PARAM_ERR* Parameter error

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

marshall data set function.

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_MEM_ERR* provided buffer to small
- TRDP_INIT_ERR* marshalling not initialised
- TRDP_COMID_ERR* comid not existing
- TRDP_PARAM_ERR* Parameter error

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

unmarshall function.

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_MEM_ERR* provided buffer to small
- TRDP_INIT_ERR* marshalling not initialised
- TRDP_COMID_ERR* comid not existing

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

unmarshall data set function.

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_MEM_ERR* provided buffer to small
- TRDP_INIT_ERR* marshalling not initialised
- TRDP_COMID_ERR* comid not existing

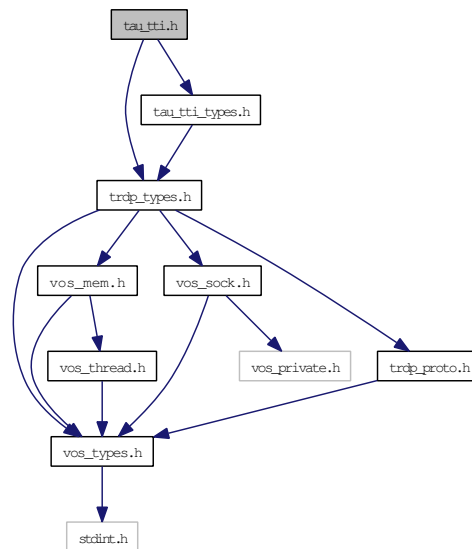
5.6 tau_tti.h File Reference

TRDP utility interface definitions.

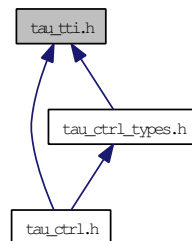
```
#include "trdp_types.h"
```

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

Include dependency graph for tau_tti.h:



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



Functions

- EXT_DECL [TRDP_ERR_T tau_initTTIaccess](#) (TRDP_APP_SESSION_T appHandle, [VOS_SEMA_T](#) userAction, [TRDP_IP_ADDR_T](#) ecspIpAddr, CHAR8 *hostsFileName)
Function to init TTI access.
- EXT_DECL void [tau_deInitTTI](#) (TRDP_APP_SESSION_T appHandle)
Function to terminate TTI access.
- EXT_DECL [TRDP_ERR_T tau_getOpTrDirectory](#) (TRDP_APP_SESSION_T appHandle, TRDP_OP_TRAIN_DIR_STATE_T *pOpTrDirState, TRDP_OP_TRAIN_DIR_T *pOpTrDir)
Function to retrieve the operational train directory state.

- EXT_DECL [TRDP_ERR_T](#) [tau_getTrDirectory](#) (TRDP_APP_SESSION_T appHandle, TRDP_TRAIN_DIR_T *pTrDir)

Function to retrieve the operational train directory.

- EXT_DECL [TRDP_ERR_T](#) [tau_getStaticCstInfo](#) (TRDP_APP_SESSION_T appHandle, [TRDP_CONSIST_INFO_T](#) *pCstInfo, TRDP_UUID_T const cstUUID)

Function to retrieve the operational train directory.

- EXT_DECL [TRDP_ERR_T](#) [tau_getTTI](#) (TRDP_APP_SESSION_T appHandle, TRDP_OP_TRAIN_DIR_STATE_T *pOpTrDirState, TRDP_OP_TRAIN_DIR_T *pOpTrDir, TRDP_TRAIN_DIR_T *pTrDir, TRDP_TRAIN_NET_DIR_T *pTrNetDir)

Function to retrieve the operational train directory.

- EXT_DECL [TRDP_ERR_T](#) [tau_getTrnCstCnt](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pTrnCstCnt)

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

- EXT_DECL [TRDP_ERR_T](#) [tau_getTrnVehCnt](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pTrnVehCnt)

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

- EXT_DECL [TRDP_ERR_T](#) [tau_getCstVehCnt](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pCstVehCnt, const TRDP_LABEL_T pCstLabel)

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

- EXT_DECL [TRDP_ERR_T](#) [tau_getCstFctCnt](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pCstFctCnt, const TRDP_LABEL_T pCstLabel)

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

- EXT_DECL [TRDP_ERR_T](#) [tau_getCstFctInfo](#) (TRDP_APP_SESSION_T appHandle, [TRDP_FUNCTION_INFO_T](#) *pFctInfo, const TRDP_LABEL_T pCstLabel, UINT16 maxFctCnt)

Function to retrieve the function information of the consist.

- EXT_DECL [TRDP_ERR_T](#) [tau_getVehInfo](#) (TRDP_APP_SESSION_T appHandle, [TRDP_VEHICLE_INFO_T](#) *pVehInfo, const TRDP_LABEL_T pVehLabel, const TRDP_LABEL_T pCstLabel)

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

- EXT_DECL [TRDP_ERR_T](#) [tau_getCstInfo](#) (TRDP_APP_SESSION_T appHandle, [TRDP_CONSIST_INFO_T](#) *pCstInfo, const TRDP_LABEL_T pCstLabel)

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

- EXT_DECL [TRDP_ERR_T](#) [tau_getVehOrient](#) (TRDP_APP_SESSION_T appHandle, UINT8 *pVehOrient, UINT8 *pCstOrient, TRDP_LABEL_T pVehLabel, TRDP_LABEL_T pCstLabel)

Function to retrieve the orientation of the given vehicle.

5.6.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- train topology information access

Note:

Project: TCNOpen TRDP prototype stack

Author:

Armin-H. Weiss (initial version)

Remarks:

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

Id

[tau_tti.h](#) 1512 2016-02-18 11:05:20Z bloehr

BL 2016-02-18: Ticket #7: Add train topology information support

5.6.2 Function Documentation

5.6.2.1 EXT_DECL void tau_deInitTTI (TRDP_APP_SESSION_T *appHandle*)

Function to terminate TTI access.

Parameters:

← *appHandle* Handle returned by [tlc_openSession\(\)](#).

Return values:

none

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

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

Parameters:

← *appHandle* Handle returned by [tlc_openSession\(\)](#).

→ *pCstFctCnt* Pointer to the number of functions to be returned

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

Return values:

TRDP_NO_ERR no error

TRDP_PARAM_ERR Parameter error

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

Function to retrieve the function information of the consist.

Parameters:

- ← *appHandle* Handle returned by [tlc_openSession\(\)](#).
- *pFctInfo* Pointer to function info list to be returned. Memory needs to be provided by application. Set NULL if not used.
- ← *pCstLabel* 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

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

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

Parameters:

- ← *appHandle* Handle returned by [tlc_openSession\(\)](#).
- *pCstInfo* Pointer to the consist info to be returned.
- ← *pCstLabel* Pointer to a consist label. NULL means own consist.

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* Parameter error

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

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

Parameters:

- ← *appHandle* Handle returned by [tlc_openSession\(\)](#).
- *pCstVehCnt* Pointer to the number of vehicles to be returned
- ← *pCstLabel* Pointer to a consist label. NULL means own consist.

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* Parameter error

5.6.2.6 `EXT_DECL TRDP_ERR_T tau_getOpTrDirectory (TRDP_APP_SESSION_T appHandle,
TRDP_OP_TRAIN_DIR_STATE_T * pOpTrDirState, TRDP_OP_TRAIN_DIR_T *
pOpTrDir)`

Function to retrieve the operational train directory state.

Parameters:

- ← *appHandle* Handle returned by `tlc_openSession()`.
- *pOpTrDirState* Pointer to an operational train directory state structure to be returned.
- *pOpTrDir* Pointer to an operational train directory structure to be returned.

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* Parameter error

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

Function to retrieve the operational train directory.

Parameters:

- ← *appHandle* Handle returned by `tlc_openSession()`.
- *pCstInfo* Pointer to a consist info structure to be returned.
- ← *cstUUID* UUID of the consist the consist info is requested for.

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* Parameter error

5.6.2.8 `EXT_DECL TRDP_ERR_T tau_getTrDirectory (TRDP_APP_SESSION_T appHandle,
TRDP_TRAIN_DIR_T * pTrDir)`

Function to retrieve the operational train directory.

Parameters:

- ← *appHandle* Handle returned by `tlc_openSession()`.
- *pTrDir* Pointer to a train directory structure to be returned.

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* Parameter error

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

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

Parameters:

- ← *appHandle* Handle returned by [tlc_openSession\(\)](#).
- *pTrnCstCnt* Pointer to the number of consists to be returned

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* Parameter error

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

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

Parameters:

- ← *appHandle* Handle returned by [tlc_openSession\(\)](#).
- *pTrnVehCnt* Pointer to the number of vehicles to be returned

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* Parameter error

5.6.2.11 EXT_DECL TRDP_ERR_T tau_getTTI (TRDP_APP_SESSION_T *appHandle*, TRDP_OP_TRAIN_DIR_STATE_T * *pOpTrDirState*, TRDP_OP_TRAIN_DIR_T * *pOpTrDir*, TRDP_TRAIN_DIR_T * *pTrDir*, TRDP_TRAIN_NET_DIR_T * *pTrNetDir*)

Function to retrieve the operational train directory.

Parameters:

- ← *appHandle* Handle returned by [tlc_openSession\(\)](#).
- *pOpTrDirState* Pointer to an operational train directory state structure to be returned.
- *pOpTrDir* Pointer to an operational train directory structure to be returned.
- *pTrDir* Pointer to a train directory structure to be returned.
- *pTrNetDir* Pointer to a train network directory structure to be returned.

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* Parameter error

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

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

Parameters:

- ← *appHandle* Handle returned by [tlc_openSession\(\)](#).
- *pVehInfo* Pointer to the vehicle info to be returned.
- ← *pVehLabel* Pointer to a vehicle label. NULL means own vehicle if cstLabel refers to own consist.
- ← *pCstLabel* Pointer to a consist label. NULL means own consist.

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* Parameter error

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

Function to retrieve the orientation of the given vehicle.

Parameters:

- ← *appHandle* Handle returned by [tlc_openSession\(\)](#).
- *pVehOrient* Pointer to the vehicle orientation to be returned '00'B = not known (corrected vehicle)
'01'B = same as operational train direction '10'B = inverse to operational train direction
- *pCstOrient* Pointer to the consist orientation to be returned '00'B = not known (corrected vehicle)
'01'B = same as operational train direction '10'B = inverse to operational train direction
- ← *pVehLabel* vehLabel = NULL means own vehicle if cstLabel == NULL
- ← *pCstLabel* cstLabel = NULL means own consist

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* Parameter error

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

Function to init TTI access.

Parameters:

- ← *appHandle* Handle returned by [tlc_openSession\(\)](#).
- ← *userAction* Semaphore to fire if inauguration took place.
- ← *ecspIpAddr* ECSP IP address.

← *hostsFileName* Optional host file name as ECSP replacement.

Return values:

TRDP_NO_ERR no error

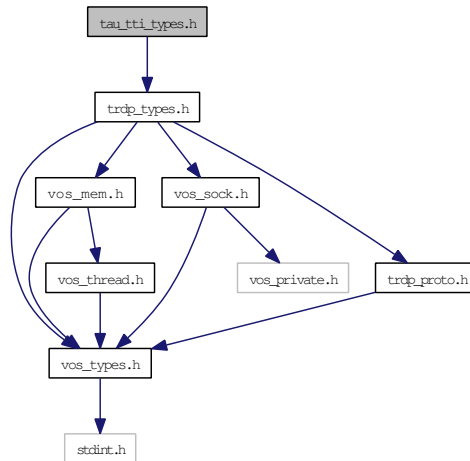
TRDP_INIT_ERR initialisation error

5.7 tau_tti_types.h File Reference

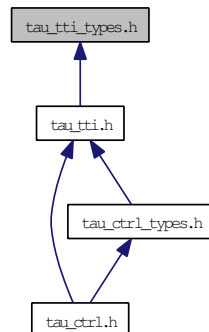
TRDP utility interface definitions.

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

Include dependency graph for tau_tti_types.h:



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



Data Structures

- struct [GNU_PACKED](#)
Types for ETB control.
- struct [TRDP_ETB_INFO_T](#)
Types for train configuration information.
- struct [TRDP_CLTR_CST_INFO_T](#)
Closed train consists information.
- struct [TRDP_PROP_T](#)
Application defined properties.

- struct [TRDP_FUNCTION_INFO_T](#)
function/device information structure
- struct [TRDP_VEHICLE_INFO_T](#)
vehicle information structure
- struct [TRDP_CONSIST_INFO_T](#)
consist information structure
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.
- struct [GNU_PACKED](#)
Types for ETB control.

Defines

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

5.7.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

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

Note:

Project: TCNOpen TRDP prototype stack

Author:

Armin-H. Weiss (initial version)

Remarks:

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

Id

[tau_tti_types.h](#) 1511 2016-02-17 17:30:14Z bloehr

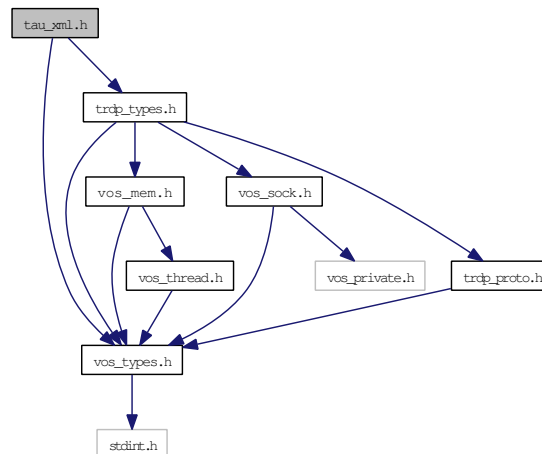
5.8 tau_xml.h File Reference

TRDP utility interface definitions.

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

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

Include dependency graph for tau_xml.h:



Data Structures

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

Enumerations

- enum [TRDP_EXCHG_OPTION_T](#) {
[TRDP_EXCHG_UNSET](#) = 0,
[TRDP_EXCHG_SOURCE](#) = 1,
[TRDP_EXCHG_SINK](#) = 2,
[TRDP_EXCHG_SOURCESINK](#) = 3 }
Type attribute for telegrams.
- enum [TRDP_DBG_OPTION_T](#) {
[TRDP_DBG_DEFAULT](#) = 0,
[TRDP_DBG_OFF](#) = 0x01,

```

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

```

Control for debug output format on application level.

Functions

- EXT_DECL [TRDP_ERR_T tau_prepareXmlDoc](#) (const CHAR8 *pFileName, [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd)

Load XML file into DOM tree, prepare XPath context.

- EXT_DECL void [tau_freeXmlDoc](#) ([TRDP_XML_DOC_HANDLE_T](#) *pDocHnd)

Free all the memory allocated by tau_prepareXmlDoc.

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

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

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

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

- EXT_DECL [TRDP_ERR_T tau_readXmlDatasetConfig](#) (const [TRDP_XML_DOC_HANDLE_T](#) *pDocHnd, UINT32 *pNumComId, [TRDP_COMID_DSID_MAP_T](#) **ppComIdDsIdMap, UINT32 *pNumDataset, [TRDP_DATASET_T](#) *pDataset)

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

- EXT_DECL void [tau_freeXmlDatasetConfig](#) (UINT32 numComId, [TRDP_COMID_DSID_MAP_T](#) *pComIdDsIdMap, UINT32 numDataset, [TRDP_DATASET_T](#) **ppNumDataset)

Function to free the memory for the DataSet configuration.

- EXT_DECL void [tau_freeTelegrams](#) (UINT32 numExchgPar, [TRDP_EXCHG_PAR_T](#) *pExchgPar)

Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.

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:

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

Id

[tau_xml.h](#) 1509 2016-02-11 14:29:05Z bloehr

BL 2016-02-11: Ticket #102: Custom XML parser, libxml2 not needed anymore

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).

5.8.2.2 enum TRDP_EXCHG_OPTION_T

Type attribute for telegrams.

Enumerator:

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

5.8.3 Function Documentation**5.8.3.1 EXT_DECL void tau_freeTelegrams (UINT32 numExchgPar, TRDP_EXCHG_PAR_T * pExchgPar)**

Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.

Parameters:

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

5.8.3.2 EXT_DECL void tau_freeXmlDatasetConfig (UINT32 numComId, TRDP_COMID_DSID_MAP_T * pComIdDsIdMap, UINT32 numDataset, TRDP_DATASET_T ** pNumDataset)

Function to free the memory for the DataSet configuration.

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

Parameters:

← **numComId** The number of entries in the ComId DataSetId mapping list
 ← **pComIdDsIdMap** Pointer to an array of structures of type [TRDP_COMID_DSID_MAP_T](#)
 ← **numDataset** The number of datasets found in the configuration
 ← **pNumDataset** Pointer to an array of pointers to a structures of type [TRDP_DATASET_T](#)

Return values:

none

5.8.3.3 EXT_DECL void tau_freeXmlDoc (TRDP_XML_DOC_HANDLE_T * pDocHnd)

Free all the memory allocated by tau_prepareXmlDoc.

Parameters:

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

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

Load XML file into DOM tree, prepare XPath context.

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* File does not exist

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

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

Parameters:

- ← *pDocHnd* Handle of the XML document prepared by tau_prepareXmlDoc
- *pNumComId* Pointer to the number of entries in the ComId DatasetId mapping list
- *ppComIdDsIdMap* Pointer to an array of a structures of type [TRDP_COMID_DSID_MAP_T](#)
- *pNumDataset* Pointer to the number of datasets found in the configuration
- *papDataset* Pointer to an array of pointers to a structures of type TRDP_DATASET_T

Return values:

- TRDP_NO_ERR* no error
- TRDP_MEM_ERR* provided buffer to small
- TRDP_PARAM_ERR* File not existing

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

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

Parameters:

- ← *pDocHnd* Handle of the XML document prepared by tau_prepareXmlDoc
- *pMemConfig* Memory configuration
- *pDbgConfig* Debug printout configuration for application use
- *pNumComPar* Number of configured com parameters
- *ppComPar* Pointer to array of com parameters

- *pNumIfConfig* Number of configured interfaces
- *ppIfConfig* Pointer to an array of interface parameter sets

Return values:

- TRDP_NO_ERR* no error
- TRDP_MEM_ERR* provided buffer too small
- TRDP_PARAM_ERR* File not existing

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

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

Parameters:

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

Return values:

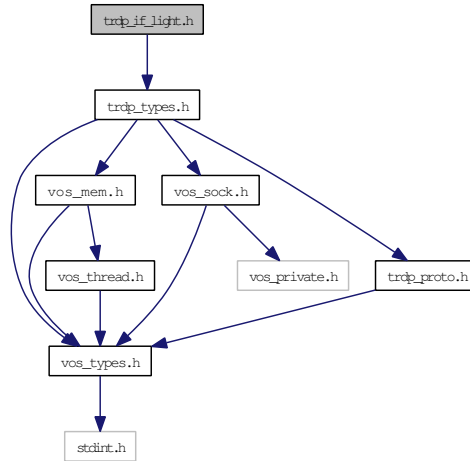
- TRDP_NO_ERR* no error
- TRDP_MEM_ERR* provided buffer too small
- TRDP_PARAM_ERR* File not existing

5.9 trdp_if_light.h File Reference

TRDP Light interface functions (API).

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

Include dependency graph for trdp_if_light.h:



Functions

- EXT_DECL [TRDP_ERR_T](#) [tlc_init](#) (const [TRDP_PRINT_DBG_T](#) pPrintDebugString, void *pRefCon, const [TRDP_MEM_CONFIG_T](#) *pMemConfig)
Support for message data can only be excluded during compile time!
- EXT_DECL [TRDP_ERR_T](#) [tlc_openSession](#) (TRDP_APP_SESSION_T *pAppHandle, [TRDP_IP_ADDR_T](#) ownIpAddr, [TRDP_IP_ADDR_T](#) leaderIpAddr, const [TRDP_MARSHALL_CONFIG_T](#) *pMarshall, const [TRDP_PD_CONFIG_T](#) *pPdDefault, const [TRDP_MD_CONFIG_T](#) *pMdDefault, const [TRDP_PROCESS_CONFIG_T](#) *pProcessConfig)
Open a session with the TRDP stack.
- EXT_DECL [TRDP_ERR_T](#) [tlc_reinitSession](#) (TRDP_APP_SESSION_T appHandle)
Re-Initialize.
- EXT_DECL [TRDP_ERR_T](#) [tlc_configSession](#) (TRDP_APP_SESSION_T appHandle, const [TRDP_MARSHALL_CONFIG_T](#) *pMarshall, const [TRDP_PD_CONFIG_T](#) *pPdDefault, const [TRDP_MD_CONFIG_T](#) *pMdDefault, const [TRDP_PROCESS_CONFIG_T](#) *pProcessConfig)
(Re-)configure a session.
- EXT_DECL [TRDP_ERR_T](#) [tlc_closeSession](#) (TRDP_APP_SESSION_T appHandle)
Close a session.
- EXT_DECL [TRDP_ERR_T](#) [tlc_terminate](#) (void)
Un-Initialize.
- EXT_DECL [TRDP_ERR_T](#) [tlc_setETBTopoCount](#) (TRDP_APP_SESSION_T appHandle, UINT32 etbTopoCnt)

Set new topocount for trainwide communication.

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

Set new operational train topocount for direction/orientation sensitive 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_IP_ADDR_T tlc_getOwnIpAddress](#) (TRDP_APP_SESSION_T appHandle)

Get the interface address.

- EXT_DECL [TRDP_ERR_T tlp_publish](#) (TRDP_APP_SESSION_T appHandle, TRDP_PUB_T *pPubHandle, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr, UINT32 interval, UINT32 redId, [TRDP_FLAGS_T](#) pktFlags, const [TRDP_SEND_PARAM_T](#) *pSendParam, const UINT8 *pData, UINT32 dataSize)

Prepare for sending PD messages.

- EXT_DECL [TRDP_ERR_T tlp_republish](#) (TRDP_APP_SESSION_T appHandle, TRDP_PUB_T pubHandle, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr)

Prepare for sending PD messages.

- EXT_DECL [TRDP_ERR_T tlp_unpublish](#) (TRDP_APP_SESSION_T appHandle, TRDP_PUB_T pubHandle)

Stop sending PD messages.

- EXT_DECL [TRDP_ERR_T tlp_put](#) (TRDP_APP_SESSION_T appHandle, TRDP_PUB_T pubHandle, const UINT8 *pData, UINT32 dataSize)

Update the process data to send.

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

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

- EXT_DECL [TRDP_ERR_T tlp_getRedundant](#) (TRDP_APP_SESSION_T appHandle, UINT32 redId, BOOL8 *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 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr, UINT32 redId, [TRDP_FLAGS_T](#) pktFlags, const [TRDP_SEND_PARAM_T](#) *pSendParam, const UINT8 *pData, UINT32 dataSize, UINT32 replyComId, [TRDP_IP_ADDR_T](#) replyIpAddr)

Initiate sending PD messages (PULL).

- EXT_DECL [TRDP_ERR_T tlp_subscribe](#) (TRDP_APP_SESSION_T appHandle, TRDP_SUB_T *pSubHandle, const void *pUserRef, [TRDP_PD_CALLBACK_T](#) pfCbFunction, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr, [TRDP_FLAGS_T](#) pktFlags, UINT32 timeout, [TRDP_TO_BEHAVIOR_T](#) toBehavior)

Prepare for receiving PD messages.

- EXT_DECL [TRDP_ERR_T tlp_resubscribe](#) (TRDP_APP_SESSION_T appHandle, TRDP_SUB_T subHandle, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr)

Reprepare for receiving PD messages.

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

Stop receiving PD messages.

- EXT_DECL [TRDP_ERR_T tlp_get](#) (TRDP_APP_SESSION_T appHandle, TRDP_SUB_T subHandle, [TRDP_PD_INFO_T](#) *pPdInfo, UINT8 *pData, UINT32 *pDataSize)

Get the last valid PD message.

- EXT_DECL [TRDP_ERR_T tlm_notify](#) (TRDP_APP_SESSION_T appHandle, const void *pUserRef, [TRDP_MD_CALLBACK_T](#) pfCbFunction, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr, [TRDP_FLAGS_T](#) pktFlags, const [TRDP_SEND_PARAM_T](#) *pSendParam, const UINT8 *pData, UINT32 dataSize, const TRDP_URI_USER_T sourceURI, const TRDP_URI_USER_T destURI)

Initiate sending MD notification message.

- EXT_DECL [TRDP_ERR_T tlm_request](#) (TRDP_APP_SESSION_T appHandle, const void *pUserRef, [TRDP_MD_CALLBACK_T](#) pfCbFunction, [TRDP_UUID_T](#) *pSessionId, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) srcIpAddr, [TRDP_IP_ADDR_T](#) destIpAddr, [TRDP_FLAGS_T](#) pktFlags, UINT32 numReplies, UINT32 replyTimeout, UINT32 maxNumRetries, const [TRDP_SEND_PARAM_T](#) *pSendParam, const UINT8 *pData, UINT32 dataSize, const TRDP_URI_USER_T sourceURI, const TRDP_URI_USER_T destURI)

Initiate sending MD request message.

- EXT_DECL [TRDP_ERR_T tlm_confirm](#) (TRDP_APP_SESSION_T appHandle, const [TRDP_UUID_T](#) *pSessionId, UINT16 userStatus, const [TRDP_SEND_PARAM_T](#) *pSendParam)

Initiate sending MD confirm message.

- EXT_DECL [TRDP_ERR_T tlm_abortSession](#) (TRDP_APP_SESSION_T appHandle, const [TRDP_UUID_T](#) *pSessionId)

Cancel an open session.

- EXT_DECL [TRDP_ERR_T tlm_addListener](#) (TRDP_APP_SESSION_T appHandle, TRDP_LIST_T *pListenHandle, const void *pUserRef, [TRDP_MD_CALLBACK_T](#) pfCbFunction, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) mcDestIpAddr, [TRDP_FLAGS_T](#) pktFlags, const TRDP_URI_USER_T destURI)

Subscribe to MD messages.

- EXT_DECL [TRDP_ERR_T tlm_readListener](#) (TRDP_APP_SESSION_T appHandle, TRDP_LIS_T listenHandle, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, [TRDP_IP_ADDR_T](#) mcDestIpAddr)
Resubscribe to MD messages.
- EXT_DECL [TRDP_ERR_T tlm_delListener](#) (TRDP_APP_SESSION_T appHandle, TRDP_LIS_T listenHandle)
Remove Listener.
- [TRDP_ERR_T tlm_reply](#) (TRDP_APP_SESSION_T appHandle, const [TRDP_UUID_T](#) *pSessionId, UINT32 comId, UINT16 userStatus, const [TRDP_SEND_PARAM_T](#) *pSendParam, const UINT8 *pData, UINT32 dataSize)
Send a MD reply message.
- [TRDP_ERR_T tlm_replyQuery](#) (TRDP_APP_SESSION_T appHandle, const [TRDP_UUID_T](#) *pSessionId, UINT32 comId, UINT16 userStatus, UINT32 confirmTimeout, const [TRDP_SEND_PARAM_T](#) *pSendParam, const UINT8 *pData, UINT32 dataSize)
Send a MD reply query message.
- [TRDP_ERR_T tlm_replyErr](#) (TRDP_APP_SESSION_T appHandle, const [TRDP_UUID_T](#) *pSessionId, UINT32 comId, [TRDP_REPLY_STATUS_T](#) replyStatus, const [TRDP_SEND_PARAM_T](#) *pSendParam)
Send a MD reply message.
- EXT_DECL const CHAR8 * [tlc_getVersionString](#) (void)
Return a human readable version representation.
- EXT_DECL const [TRDP_VERSION_T](#) * [tlc_getVersion](#) (void)
Return version.
- EXT_DECL [TRDP_ERR_T tlc_getStatistics](#) (TRDP_APP_SESSION_T appHandle, [TRDP_STATISTICS_T](#) *pStatistics)
Return statistics.
- EXT_DECL [TRDP_ERR_T tlc_getSubsStatistics](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pNumSubs, [TRDP_SUBS_STATISTICS_T](#) *pStatistics)
Return PD subscription statistics.
- EXT_DECL [TRDP_ERR_T tlc_getPubStatistics](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pNumPub, [TRDP_PUB_STATISTICS_T](#) *pStatistics)
Return PD publish statistics.
- EXT_DECL [TRDP_ERR_T tlc_getUdpListStatistics](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pNumList, [TRDP_LIST_STATISTICS_T](#) *pStatistics)
Return UDP MD listener statistics.
- EXT_DECL [TRDP_ERR_T tlc_getTcpListStatistics](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pNumList, [TRDP_LIST_STATISTICS_T](#) *pStatistics)
Return TCP MD listener statistics.

- EXT_DECL [TRDP_ERR_T tlc_getRedStatistics](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pNumRed, [TRDP_RED_STATISTICS_T](#) *pStatistics)
Return redundancy group statistics.
- EXT_DECL [TRDP_ERR_T tlc_getJoinStatistics](#) (TRDP_APP_SESSION_T appHandle, UINT16 *pNumJoin, UINT32 *pIpAddr)
Return join statistics.
- EXT_DECL [TRDP_ERR_T tlc_resetStatistics](#) (TRDP_APP_SESSION_T appHandle)
Reset statistics.

5.9.1 Detailed Description

TRDP Light interface functions (API).

Low level functions for communicating using the TRDP protocol

Note:

Project: TCNOpen TRDP prototype stack

Author:

Bernd Loehr, NewTec GmbH

Remarks:

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

Id

[trdp_if_light.h](#) 1526 2016-03-02 13:45:31Z newtecbosse

BL 2015-11-24: Accessor for IP address of session BL 2015-09-04: Ticket #99: refCon for [tlc_init\(\)](#)

BL 2014-07-14: Ticket #46: Protocol change: operational topocount needed

5.9.2 Function Documentation

5.9.2.1 EXT_DECL TRDP_ERR_T tlc_closeSession (TRDP_APP_SESSION_T appHandle)

Close a session.

Clean up and release all resources of that session

Parameters:

← *appHandle* The handle returned by [tlc_openSession](#)

Return values:

TRDP_NO_ERR no error

TRDP_NOINIT_ERR handle invalid

TRDP_PARAM_ERR handle NULL

5.9.2.2 EXT_DECL TRDP_ERR_T tlc_configSession (TRDP_APP_SESSION_T *appHandle*, const TRDP_MARSHALL_CONFIG_T * *pMarshall*, const TRDP_PD_CONFIG_T * *pPdDefault*, const TRDP_MD_CONFIG_T * *pMdDefault*, const TRDP_PROCESS_CONFIG_T * *pProcessConfig*)

(Re-)configure a session.

tlc_configSession is called by openSession, but may also be called later on to change the defaults.

Parameters:

- ← *appHandle* A handle for further calls to the trdp stack
- ← *pMarshall* Pointer to marshalling configuration
- ← *pPdDefault* Pointer to default PD configuration
- ← *pMdDefault* Pointer to default MD configuration
- ← *pProcessConfig* Pointer to process configuration only option parameter is used here to define session behavior all other parameters are only used to feed statistics

Return values:

- TRDP_NO_ERR* no error
- TRDP_INIT_ERR* not yet initied
- TRDP_PARAM_ERR* parameter error

5.9.2.3 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_openSession
- ← *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

5.9.2.4 EXT_DECL TRDP_ERR_T tlc_getInterval (TRDP_APP_SESSION_T *appHandle*, TRDP_TIME_T * *pInterval*, TRDP_FDS_T * *pFileDesc*, INT32 * *pNoDesc*)

Get the lowest time interval for PDs.

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

Parameters:

- ← *appHandle* The handle returned by tlc_openSession

- *pInterval* pointer to needed interval
- ↔ *pFileDesc* pointer to file descriptor set
- *pNoDesc* pointer to put no of used descriptors (for select())

Return values:

- TRDP_NO_ERR* no error
- TRDP_NOINIT_ERR* handle invalid

5.9.2.5 EXT_DECL TRDP_ERR_T tlc_getJoinStatistics (TRDP_APP_SESSION_T *appHandle*, UINT16 * *pNumJoin*, UINT32 * *pIpAddr*)

Return join statistics.

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

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_NOINIT_ERR* handle invalid
- TRDP_PARAM_ERR* parameter error
- TRDP_MEM_ERR* there are more items than requested

5.9.2.6 EXT_DECL TRDP_IP_ADDR_T tlc_getOwnIpAddress (TRDP_APP_SESSION_T *appHandle*)

Get the interface address.

Parameters:

- *appHandle* A handle for further calls to the trdp stack

Return values:

- realIP*

5.9.2.7 EXT_DECL TRDP_ERR_T tlc_getPubStatistics (TRDP_APP_SESSION_T *appHandle*, UINT16 * *pNumPub*, TRDP_PUB_STATISTICS_T * *pStatistics*)

Return PD publish statistics.

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

Parameters:

- ← *appHandle* the handle returned by `tlc_openSession`
- ↔ *pNumPub* Pointer to the number of publishers
- *pStatistics* pointer to a list with the publish statistics information

Return values:

- TRDP_NO_ERR* no error
- TRDP_NOINIT_ERR* handle invalid
- TRDP_PARAM_ERR* parameter error
- TRDP_MEM_ERR* there are more subscriptions than requested

5.9.2.8 EXT_DECL TRDP_ERR_T tlc_getRedStatistics (TRDP_APP_SESSION_T *appHandle*, UINT16 * *pNumRed*, TRDP_RED_STATISTICS_T * *pStatistics*)

Return redundancy group statistics.

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

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_NOINIT_ERR* handle invalid
- TRDP_PARAM_ERR* parameter error
- TRDP_MEM_ERR* there are more subscriptions than requested

5.9.2.9 EXT_DECL TRDP_ERR_T tlc_getStatistics (TRDP_APP_SESSION_T *appHandle*, TRDP_STATISTICS_T * *pStatistics*)

Return statistics.

Memory for statistics information must be preserved by the user.

Parameters:

- ← *appHandle* the handle returned by `tlc_openSession`
- *pStatistics* Pointer to statistics for this application session

Return values:

- TRDP_NO_ERR* no error
- TRDP_NOINIT_ERR* handle invalid
- TRDP_PARAM_ERR* parameter error

5.9.2.10 EXT_DECL TRDP_ERR_T tlc_getSubsStatistics (TRDP_APP_SESSION_T *appHandle*, UINT16 **pNumSubs*, TRDP_SUBS_STATISTICS_T **pStatistics*)

Return PD subscription statistics.

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

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_NOINIT_ERR* handle invalid
- TRDP_PARAM_ERR* parameter error
- TRDP_MEM_ERR* there are more subscriptions than requested

5.9.2.11 EXT_DECL TRDP_ERR_T tlc_getTcpListStatistics (TRDP_APP_SESSION_T *appHandle*, UINT16 **pNumList*, TRDP_LIST_STATISTICS_T **pStatistics*)

Return TCP MD listener statistics.

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

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_NOINIT_ERR* handle invalid
- TRDP_PARAM_ERR* parameter error
- TRDP_MEM_ERR* there are more subscriptions than requested

5.9.2.12 EXT_DECL TRDP_ERR_T tlc_getUdpListStatistics (TRDP_APP_SESSION_T *appHandle*, UINT16 **pNumList*, TRDP_LIST_STATISTICS_T **pStatistics*)

Return UDP MD listener statistics.

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

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_NOINIT_ERR* handle invalid
- TRDP_PARAM_ERR* parameter error
- TRDP_MEM_ERR* there are more subscriptions than requested

5.9.2.13 EXT_DECL const TRDP_VERSION_T* tlc_getVersion (void)

Return version.

Return pointer to version structure

Return values:

- const* TRDP_VERSION_T

5.9.2.14 EXT_DECL const CHAR8* tlc_getVersionString (void)

Return a human readable version representation.

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

Return values:

- const* string

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

Support for message data can only be excluded during compile time!

Initialize the TRDP stack.

`tlc_init` initializes the memory subsystem and takes a function pointer to an output function for logging.

Parameters:

- ← *pPrintDebugString* Pointer to debug print function
- ← *pRefCon* user context
- ← *pMemConfig* Pointer to memory configuration

Return values:

- TRDP_NO_ERR* no error
- TRDP_MEM_ERR* memory allocation failed
- TRDP_PARAM_ERR* initialization error

5.9.2.16 `EXT_DECL TRDP_ERR_T tlc_openSession (TRDP_APP_SESSION_T *pAppHandle, TRDP_IP_ADDR_T ownIpAddr, TRDP_IP_ADDR_T leaderIpAddr, const TRDP_MARSHALL_CONFIG_T *pMarshall, const TRDP_PD_CONFIG_T *pPdDefault, const TRDP_MD_CONFIG_T *pMdDefault, const TRDP_PROCESS_CONFIG_T *pProcessConfig)`

Open a session with the TRDP stack.

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

Parameters:

- *pAppHandle* A handle for further calls to the trdp stack
- ← *ownIpAddr* Own IP address, can be different for each process in multihoming systems, if zero, the default interface / IP will be used.
- ← *leaderIpAddr* IP address of redundancy leader
- ← *pMarshall* Pointer to marshalling configuration
- ← *pPdDefault* Pointer to default PD configuration
- ← *pMdDefault* Pointer to default MD configuration
- ← *pProcessConfig* Pointer to process configuration only option parameter is used here to define session behavior all other parameters are only used to feed statistics

Return values:

- TRDP_NO_ERR* no error
- TRDP_INIT_ERR* not yet initied
- TRDP_PARAM_ERR* parameter error
- TRDP SOCK_ERR* socket error

5.9.2.17 `EXT_DECL TRDP_ERR_T tlc_process (TRDP_APP_SESSION_T appHandle, TRDP_FDS_T *pRfds, INT32 *pCount)`

Work loop of the TRDP handler.

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

Parameters:

- ← *appHandle* The handle returned by `tlc_openSession`
- ← *pRfds* pointer to set of ready descriptors
- ↔ *pCount* pointer to number of ready descriptors

Return values:

- TRDP_NO_ERR* no error
- TRDP_NOINIT_ERR* handle invalid

5.9.2.18 EXT_DECL TRDP_ERR_T tlc_reinitSession (TRDP_APP_SESSION_T *appHandle*)

Re-Initialize.

Should be called by the application when a link-down/link-up event has occurred during normal operation. We need to re-join the multicast groups...

Parameters:

← *appHandle* The handle returned by tlc_openSession

Return values:

TRDP_NO_ERR no error

TRDP_NOINIT_ERR handle invalid

TRDP_PARAM_ERR handle NULL

5.9.2.19 EXT_DECL TRDP_ERR_T tlc_resetStatistics (TRDP_APP_SESSION_T *appHandle*)

Reset statistics.

Parameters:

← *appHandle* the handle returned by tlc_openSession

Return values:

TRDP_NO_ERR no error

TRDP_NOINIT_ERR handle invalid

TRDP_PARAM_ERR parameter error

5.9.2.20 EXT_DECL TRDP_ERR_T tlc_setETBTopoCount (TRDP_APP_SESSION_T *appHandle*, UINT32 *etbTopoCnt*)

Set new topocount for trainwide communication.

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

Parameters:

← *appHandle* The handle returned by tlc_openSession

← *etbTopoCnt* New topocount value

5.9.2.21 EXT_DECL TRDP_ERR_T tlc_setOpTrainTopoCount (TRDP_APP_SESSION_T *appHandle*, UINT32 *opTrnTopoCnt*)

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

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

Parameters:

← *appHandle* The handle returned by tlc_openSession

← *opTrnTopoCnt* New operational topocount value

5.9.2.22 EXT_DECL TRDP_ERR_T tlc_terminate (void)

Un-Initialize.

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

Return values:

TRDP_NO_ERR no error

5.9.2.23 EXT_DECL TRDP_ERR_T tlm_abortSession (TRDP_APP_SESSION_T appHandle, const TRDP_UUID_T *pSessionId)

Cancel an open session.

Abort an open session; any pending messages will be dropped

Parameters:

← **appHandle** the handle returned by tlc_openSession

← **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

5.9.2.24 EXT_DECL TRDP_ERR_T tlm_addListener (TRDP_APP_SESSION_T appHandle, TRDP_LIS_T *pListenHandle, const void *pUserRef, TRDP_MD_CALLBACK_T pfCbFunction, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, TRDP_IP_ADDR_T mcDestIpAddr, TRDP_FLAGS_T pktFlags, const TRDP_URI_USER_T destURI)

Subscribe to MD messages.

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

Parameters:

← **appHandle** the handle returned by tlc_openSession

→ **pListenHandle** Handle for this listener returned

← **pUserRef** user supplied value returned with received message

← **pfCbFunction** Pointer to listener specific callback function, NULL to use default function

← **comId** comId to be observed

← **etbTopoCnt** ETB topocount to use, 0 if consist local communication

← **opTrnTopoCnt** operational topocount, != 0 for orientation/direction sensitive communication

← **mcDestIpAddr** multicast group to listen on

← **pktFlags** OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_TCP

← *destURI* only functional group of destination URI

Return values:

TRDP_NO_ERR no error
TRDP_PARAM_ERR parameter error
TRDP_MEM_ERR out of memory
TRDP_NOINIT_ERR handle invalid

5.9.2.25 EXT_DECL TRDP_ERR_T tlm_confirm (TRDP_APP_SESSION_T *appHandle*, const TRDP_UUID_T * *pSessionId*, UINT16 *userStatus*, const TRDP_SEND_PARAM_T * *pSendParam*)

Initiate sending MD confirm message.

Send a MD confirmation message User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

Parameters:

← *appHandle* the handle returned by tlc_openSession
 ← *pSessionId* Session ID returned by request
 ← *userStatus* Info for requester about application errors
 ← *pSendParam* Pointer to send parameters, NULL to use default send parameters

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

5.9.2.26 EXT_DECL TRDP_ERR_T tlm_delListener (TRDP_APP_SESSION_T *appHandle*, TRDP_LIS_T *listenHandle*)

Remove Listener.

Parameters:

← *appHandle* the handle returned by tlc_openSession
 → *listenHandle* Handle for this listener

Return values:

TRDP_NO_ERR no error
TRDP_PARAM_ERR parameter error
TRDP_NOINIT_ERR handle invalid

5.9.2.27 `EXT_DECL TRDP_ERR_T tlm_notify (TRDP_APP_SESSION_T appHandle, const void * pUserRef, TRDP_MD_CALLBACK_T pfCbFunction, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, TRDP_IP_ADDR_T srcIpAddr, TRDP_IP_ADDR_T destIpAddr, TRDP_FLAGS_T pktFlags, const TRDP_SEND_PARAM_T * pSendParam, const UINT8 * pData, UINT32 dataSize, const TRDP_URI_USER_T sourceURI, const TRDP_URI_USER_T destURI)`

Initiate sending MD notification message.

Send a MD notification message

Parameters:

- ← *appHandle* the handle returned by `tlc_openSession`
- ← *pUserRef* user supplied value returned with reply
- ← *pfCbFunction* Pointer to listener specific callback function, NULL to use default function
- ← *comId* comId of packet to be sent
- ← *etbTopoCnt* ETB topocount to use, 0 if consist local communication
- ← *opTrnTopoCnt* operational topocount, != 0 for orientation/direction sensitive communication
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to
- ← *pktFlags* OPTIONS: `TRDP_FLAGS_DEFAULT`, `TRDP_FLAGS_MARSHALL`, `TRDP_FLAGS_TCP`
- ← *pSendParam* optional pointer to send parameter, NULL - default parameters are used
- ← *pData* pointer to packet data / dataset
- ← *dataSize* size of packet data
- ← *sourceURI* only functional group of source URI
- ← *destURI* only functional group of destination URI

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* parameter error
- TRDP_MEM_ERR* out of memory
- TRDP_NOINIT_ERR* handle invalid

5.9.2.28 `EXT_DECL TRDP_ERR_T tlm_readdListener (TRDP_APP_SESSION_T appHandle, TRDP_LIS_T listenHandle, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, TRDP_IP_ADDR_T mcDestIpAddr)`

Resubscribe to MD messages.

Readd a listener after topoCount changes to get notified when messages are received

Parameters:

- ← *appHandle* the handle returned by `tlc_openSession`
- *listenHandle* Handle for this listener
- ← *etbTopoCnt* ETB topocount to use, 0 if consist local communication

- ← *opTrnTopoCnt* operational topocount, != 0 for orientation/direction sensitive communication
- ← *mcDestIpAddr* multicast group to listen on

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* parameter error
- TRDP_MEM_ERR* out of memory
- TRDP_NOINIT_ERR* handle invalid

5.9.2.29 *TRDP_ERR_T* tlm_reply (*TRDP_APP_SESSION_T* *appHandle*, const *TRDP_UUID_T* * *pSessionId*, *UINT32* *comId*, *UINT16* *userStatus*, const *TRDP_SEND_PARAM_T* * *pSendParam*, const *UINT8* * *pData*, *UINT32* *dataSize*)

Send a MD reply message.

Send a MD reply message after receiving an request User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

Parameters:

- ← *appHandle* the handle returned by tlc_openSession
- ← *pSessionId* Session ID returned by indication
- ← *comId* comId of packet to be sent
- ← *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

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

5.9.2.30 *TRDP_ERR_T* tlm_replyErr (*TRDP_APP_SESSION_T* *appHandle*, const *TRDP_UUID_T* * *pSessionId*, *UINT32* *comId*, *TRDP_REPLY_STATUS_T* *replyStatus*, const *TRDP_SEND_PARAM_T* * *pSendParam*)

Send a MD reply message.

Send a MD error reply message after receiving an request User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

Parameters:

- ← *appHandle* the handle returned by tlc_openSession

- ← *pSessionId* Session ID returned by indication
- ← *comId* ComId for reply
- ← *replyStatus* Info for requester about stack errors
- ← *pSendParam* Pointer to send parameters, NULL to use default send parameters

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

5.9.2.31 TRDP_ERR_T tlm_replyQuery (TRDP_APP_SESSION_T appHandle, const TRDP_UUID_T * pSessionId, UINT32 comId, UINT16 userStatus, UINT32 confirmTimeout, const TRDP_SEND_PARAM_T * pSendParam, const UINT8 * pData, UINT32 dataSize)

Send a MD reply query message.

Send a MD reply query message after receiving a request and ask for confirmation. User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

Parameters:

- ← *appHandle* the handle returned by tlc_openSession
- ← *pSessionId* Session ID returned by indication
- ← *comId* comId of packet to be sent
- ← *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

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

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

Initiate sending MD request message.

Send a MD request message

Parameters:

- ← *appHandle* the handle returned by `tlc_openSession`
- ← *pUserRef* user supplied value returned with reply
- ← *pfCbFunction* Pointer to listener specific callback function, NULL to use default function
- *pSessionId* return session ID
- ← *comId* comId of packet to be sent
- ← *etbTopoCnt* ETB topocount to use, 0 if consist local communication
- ← *opTrnTopoCnt* operational topocount, != 0 for orientation/direction sensitive communication
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to
- ← *pktFlags* OPTIONS: `TRDP_FLAGS_DEFAULT`, `TRDP_FLAGS_MARSHALL`, `TRDP_FLAGS_TCP`
- ← *numReplies* number of expected replies, 0 if unknown
- ← *replyTimeout* timeout for reply
- ← *maxNumRetries* maximum number of retries (0 ... 2)
- ← *pSendParam* Pointer to send parameters, NULL to use default send parameters
- ← *pData* pointer to packet data / dataset
- ← *dataSize* size of packet data
- ← *sourceURI* only functional group of source URI
- ← *destURI* only functional group of destination URI

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* parameter error
- TRDP_MEM_ERR* out of memory
- TRDP_NOINIT_ERR* handle invalid

5.9.2.33 `EXT_DECL TRDP_ERR_T tlp_get (TRDP_APP_SESSION_T appHandle, TRDP_SUB_T subHandle, TRDP_PD_INFO_T * pPdInfo, UINT8 * pData, UINT32 * pDataSize)`

Get the last valid PD message.

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

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* parameter error
- TRDP_SUB_ERR* not subscribed
- TRDP_TIMEOUT_ERR* packet timed out
- TRDP_NOINIT_ERR* handle invalid
- TRDP_COMID_ERR* ComID not found when marshalling

5.9.2.34 EXT_DECL TRDP_ERR_T tlp_getRedundant (TRDP_APP_SESSION_T *appHandle*, UINT32 *redId*, BOOL8 * *pLeader*)

Get status of redundant ComIds.

Parameters:

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

5.9.2.35 EXT_DECL TRDP_ERR_T tlp_publish (TRDP_APP_SESSION_T *appHandle*, TRDP_PUB_T * *pPubHandle*, UINT32 *comId*, UINT32 *etbTopoCnt*, UINT32 *opTrnTopoCnt*, TRDP_IP_ADDR_T *srcIpAddr*, TRDP_IP_ADDR_T *destIpAddr*, UINT32 *interval*, UINT32 *redId*, TRDP_FLAGS_T *pktFlags*, const TRDP_SEND_PARAM_T * *pSendParam*, const UINT8 * *pData*, UINT32 *dataSize*)

Prepare for sending PD messages.

Queue a PD message, it will be send when `tlc_publish` has been called

Parameters:

- ← *appHandle* the handle returned by `tlc_openSession`
- *pPubHandle* returned handle for related re/unpublish
- ← *comId* comId of packet to send

- ← *etbTopoCnt* ETB topocount to use, 0 if consist local communication
- ← *opTrnTopoCnt* operational topocount, != 0 for orientation/direction sensitive communication
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to
- ← *interval* frequency of PD packet (>= 10ms) in usec
- ← *redId* 0 - Non-redundant, > 0 valid redundancy group
- ← *pktFlags* OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
- ← *pSendParam* optional pointer to send parameter, NULL - default parameters are used
- ← *pData* pointer to data packet / dataset, NULL if sending starts later with [tlp_put\(\)](#)
- ← *dataSize* size of data packet >= 0 and <= TRDP_MAX_PD_DATA_SIZE

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

5.9.2.36 EXT_DECL TRDP_ERR_T tlp_put (TRDP_APP_SESSION_T *appHandle*, TRDP_PUB_T *pubHandle*, const UINT8 * *pData*, UINT32 *dataSize*)

Update the process data to send.

Update previously published data. The new telegram will be sent earliest when `tlc_process` is called.

Parameters:

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

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* parameter error on uninitialized parameter or changed `dataSize` compared to published one
- TRDP_PUB_ERR* not published
- TRDP_NOINIT_ERR* handle invalid
- TRDP_COMID_ERR* ComID not found when marshalling

5.9.2.37 EXT_DECL TRDP_ERR_T tlp_republish (TRDP_APP_SESSION_T *appHandle*, TRDP_PUB_T *pubHandle*, UINT32 *etbTopoCnt*, UINT32 *opTrnTopoCnt*, TRDP_IP_ADDR_T *srcIpAddr*, TRDP_IP_ADDR_T *destIpAddr*)

Prepare for sending PD messages.

Reinitialize and queue a PD message, it will be send when tlc_publish has been called

Parameters:

- ← *appHandle* the handle returned by tlc_openSession
- ← *pubHandle* handle for related unpublish
- ← *etbTopoCnt* ETB topocount to use, 0 if consist local communication
- ← *opTrnTopoCnt* operational topocount, != 0 for orientation/direction sensitive communication
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to

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

5.9.2.38 EXT_DECL TRDP_ERR_T tlp_request (TRDP_APP_SESSION_T *appHandle*, TRDP_SUB_T *subHandle*, UINT32 *comId*, UINT32 *etbTopoCnt*, UINT32 *opTrnTopoCnt*, TRDP_IP_ADDR_T *srcIpAddr*, TRDP_IP_ADDR_T *destIpAddr*, UINT32 *redId*, TRDP_FLAGS_T *pktFlags*, const TRDP_SEND_PARAM_T **pSendParam*, const UINT8 **pData*, UINT32 *dataSize*, UINT32 *replyComId*, TRDP_IP_ADDR_T *replyIpAddr*)

Initiate sending PD messages (PULL).

Send a PD request message

Parameters:

- ← *appHandle* the handle returned by tlc_openSession
- ← *subHandle* handle from related subscribe
- ← *comId* comId of packet to be sent
- ← *etbTopoCnt* ETB topocount to use, 0 if consist local communication
- ← *opTrnTopoCnt* operational topocount, != 0 for orientation/direction sensitive communication
- ← *srcIpAddr* own IP address, 0 - srcIP will be set by the stack
- ← *destIpAddr* where to send the packet to
- ← *redId* 0 - Non-redundant, > 0 valid redundancy group
- ← *pktFlags* OPTIONS: TTRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
- ← *pSendParam* optional pointer to send parameter, NULL - default parameters are used
- ← *pData* pointer to packet data / dataset

- ← *dataSize* size of packet data
- ← *replyComId* comId of reply
- ← *replyIpAddr* IP for reply

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* parameter error
- TRDP_MEM_ERR* could not insert (out of memory)
- TRDP_NOINIT_ERR* handle invalid

5.9.2.39 EXT_DECL TRDP_ERR_T tlp_resubscribe (TRDP_APP_SESSION_T *appHandle*, TRDP_SUB_T *subHandle*, UINT32 *etbTopoCnt*, UINT32 *opTrnTopoCnt*, TRDP_IP_ADDR_T *srcIpAddr*, TRDP_IP_ADDR_T *destIpAddr*)

Reprepare for receiving PD messages.

Resubscribe to a specific PD ComID and source IP

Parameters:

- ← *appHandle* the handle returned by tlc_openSession
- ← *subHandle* handle for this subscription
- ← *etbTopoCnt* ETB topocount to use, 0 if consist local communication
- ← *opTrnTopoCnt* operational topocount, != 0 for orientation/direction sensitive communication
- ← *srcIpAddr* IP for source filtering, set 0 if not used
- ← *destIpAddr* IP address to join

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

5.9.2.40 EXT_DECL TRDP_ERR_T tlp_setRedundant (TRDP_APP_SESSION_T *appHandle*, UINT32 *redId*, BOOL8 *leader*)

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

Parameters:

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

5.9.2.41 `EXT_DECL TRDP_ERR_T tlp_subscribe (TRDP_APP_SESSION_T appHandle,
TRDP_SUB_T * pSubHandle, const void * pUserRef, TRDP_PD_CALLBACK_T
pfCbFunction, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt,
TRDP_IP_ADDR_T srcIpAddr, TRDP_IP_ADDR_T destIpAddr, TRDP_FLAGS_T
pktFlags, UINT32 timeout, TRDP_TO_BEHAVIOR_T toBehavior)`

Prepare for receiving PD messages.

Subscribe to a specific PD ComID and source IP

Parameters:

- ← *appHandle* the handle returned by `tlc_openSession`
- *pSubHandle* return a handle for this subscription
- ← *pUserRef* user supplied value returned within the info structure
- ← *pfCbFunction* Pointer to subscriber specific callback function, NULL to use default function
- ← *comId* comId of packet to receive
- ← *etbTopoCnt* ETB topocount to use, 0 if consist local communication
- ← *opTrnTopoCnt* operational topocount, != 0 for orientation/direction sensitive communication
- ← *srcIpAddr* IP for source filtering, set 0 if not used Used e.g. for source filtering of redundant devices.
- ← *destIpAddr* IP address to join
- ← *pktFlags* OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_-MARSHALL, TRDP_FLAGS_CALLBACK
- ← *timeout* timeout (>= 10ms) in usec
- ← *toBehavior* OPTION: TRDP_TO_DEFAULT, TRDP_TO_SET_TO_ZERO, TRDP_TO_KEEP_-LAST_VALUE

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

5.9.2.42 `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_openSession`
- ← *pubHandle* the handle returned by `publish`

Return values:

- TRDP_NO_ERR* no error
- TRDP_PARAM_ERR* parameter error
- TRDP_NOPUB_ERR* not published
- TRDP_NOINIT_ERR* handle invalid

5.9.2.43 EXT_DECL TRDP_ERR_T tlp_unsubscribe (TRDP_APP_SESSION_T *appHandle*, TRDP_SUB_T *subHandle*)

Stop receiving PD messages.

Unsubscribe to a specific PD ComID

Parameters:

← *appHandle* the handle returned by tlc_openSession

← *subHandle* the handle for this subscription

Return values:

TRDP_NO_ERR no error

TRDP_PARAM_ERR parameter error

TRDP_SUB_ERR not subscribed

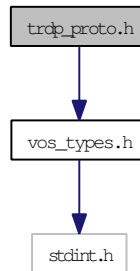
TRDP_NOINIT_ERR handle invalid

5.10 trdp_proto.h File Reference

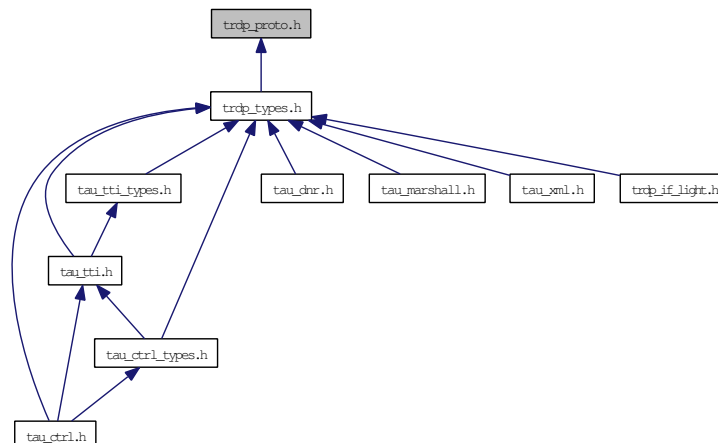
Definitions for the TRDP protocol.

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

Include dependency graph for trdp_proto.h:



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



Data Structures

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

Defines

- #define [TRDP_PD_UDP_PORT](#) 17224
process data UDP port
- #define [TRDP_MD_UDP_PORT](#) 17225

message data UDP port

- #define [TRDP_MD_TCP_PORT](#) 17225

message data TCP port

- #define [TRDP_PROTO_VER](#) 0x0100

Protocol version.

- #define [TRDP_PROTOCOL_VERSION_CHECK_MASK](#) 0xFF00

Version check, two digits are relevant.

- #define [TRDP_SESS_ID_SIZE](#) 16

Session ID (UUID) size in MD header.

- #define [TRDP_DEST_URI_SIZE](#) 32

max.

- #define [TRDP_MIN_PD_HEADER_SIZE](#) sizeof(PD_HEADER_T)

PD header size with FCS.

- #define [TRDP_MAX_PD_DATA_SIZE](#) 1432

PD data.

- #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 [TRDP_VAR_SIZE](#) 0

Variable size dataset.

- #define [TRDP_ETBCTRL_COMID](#) 1

TRDP reserved COMIDs in the range 1 .

- #define [TRDP_ETBCTRL_DSID](#) 1

TRDP reserved data set ids in the range 1 .

Enumerations

- enum `TRDP_MSG_T` {
`TRDP_MSG_PD` = 0x5064,
`TRDP_MSG_PP` = 0x5070,
`TRDP_MSG_PR` = 0x5072,
`TRDP_MSG_PE` = 0x5065,
`TRDP_MSG_MN` = 0x4D6E,
`TRDP_MSG_MR` = 0x4D72,
`TRDP_MSG_MP` = 0x4D70,
`TRDP_MSG_MQ` = 0x4D71,
`TRDP_MSG_MC` = 0x4D63,
`TRDP_MSG_ME` = 0x4D65 }

Message Types.

5.10.1 Detailed Description

Definitions for the TRDP protocol.

TRDP internal type definitions

Note:

Project: TCNOpen TRDP prototype stack

Author:

Bernd Loehr, NewTec GmbH

Remarks:

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

Id

[trdp_proto.h](#) 1454 2015-10-16 16:14:02Z bloehr

BL 2014-07-14: Ticket #46: Protocol change: operational topocount needed

5.10.2 Define Documentation

5.10.2.1 #define TRDP_DEST_URI_SIZE 32

max.

Dest URI size in MD header

5.10.2.2 `#define TRDP_ETBCTRL_COMID 1`

TRDP reserved COMIDs in the range 1 .

.. 1000

5.10.2.3 `#define TRDP_ETBCTRL_DSID 1`

TRDP reserved data set ids in the range 1 .

.. 1000

5.10.2.4 `#define TRDP_MAX_FILE_NAME_LEN 128`

path and file name length incl.

terminating '0'

5.10.2.5 `#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: $7 + (6 * 15) + 5 * (\text{sizeof}(\text{separator})) + 1$ (terminating 0) to facilitate alignment the size will be increased by 1 byte label length incl. terminating '0'

5.10.2.6 `#define TRDP_MAX_URI_HOST_LEN (4 * TRDP_MAX_LABEL_LEN)`

URI host part length incl.

terminating '0'

5.10.2.7 `#define TRDP_MAX_URI_LEN ((6 * TRDP_MAX_LABEL_LEN) + 8)`

URI length incl.

terminating '0' and 1 padding byte

5.10.2.8 `#define TRDP_MAX_URI_USER_LEN (2 * TRDP_MAX_LABEL_LEN)`

URI user part incl.

terminating '0'

5.10.3 Enumeration Type Documentation

5.10.3.1 `enum TRDP_MSG_T`

Message Types.

Enumerator:

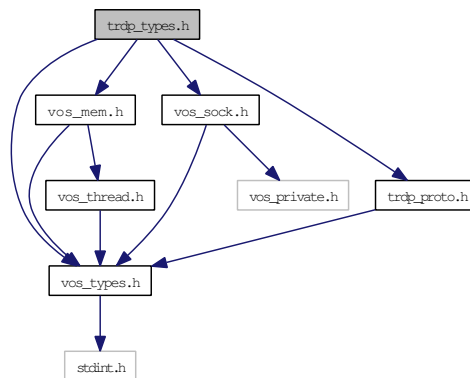
TRDP_MSG_PD 'Pd' PD Data
TRDP_MSG_PP 'Pp' PD Data (Pull Reply)
TRDP_MSG_PR 'Pr' PD Request
TRDP_MSG_PE 'Pe' PD Error
TRDP_MSG_MN 'Mn' MD Notification (Request without reply)
TRDP_MSG_MR 'Mr' MD Request with reply
TRDP_MSG_MP 'Mp' MD Reply without confirmation
TRDP_MSG_MQ 'Mq' MD Reply with confirmation
TRDP_MSG_MC 'Mc' MD Confirm
TRDP_MSG_ME 'Me' MD Error

5.11 trdp_types.h File Reference

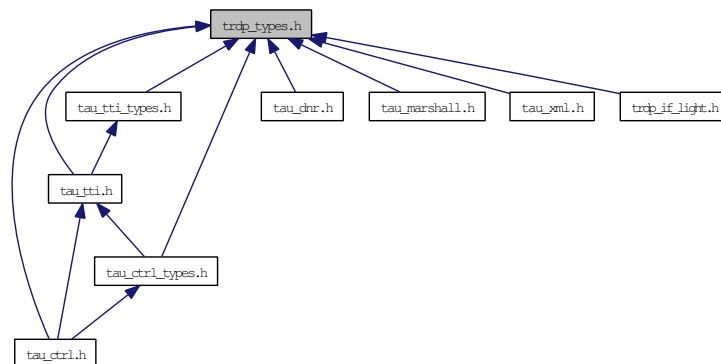
Typedefs for TRDP communication.

```
#include "vos_types.h"
#include "vos_mem.h"
#include "vos_sock.h"
#include "trdp_proto.h"
```

Include dependency graph for trdp_types.h:



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



Data Structures

- struct [TRDP_PD_INFO_T](#)
Process data info from received telegram; allows the application to generate responses.
- struct [TRDP_MD_INFO_T](#)
Message data info from received telegram; allows the application to generate responses.
- struct [TRDP_SEND_PARAM_T](#)
Quality/type of service and time to live.

- struct [TRDP_DATASET_ELEMENT_T](#)
Dataset element definition.
- struct [TRDP_DATASET](#)
Dataset definition.
- struct [TRDP_COMID_DSID_MAP_T](#)
ComId - data set mapping element definition.
- struct [TRDP_MEM_STATISTICS_T](#)
TRDP statistics type definitions.
- struct [TRDP_PD_STATISTICS_T](#)
Structure containing all general PD statistics information.
- struct [TRDP_MD_STATISTICS_T](#)
Structure containing all general MD statistics information.
- struct [TRDP_STATISTICS_T](#)
Structure containing all general memory, PD and MD statistics information.
- struct [TRDP_SUBS_STATISTICS_T](#)
Table containing particular PD subscription information.
- struct [TRDP_PUB_STATISTICS_T](#)
Table containing particular PD publishing information.
- struct [TRDP_LIST_STATISTICS_T](#)
Information about a particular MD listener.
- struct [TRDP_RED_STATISTICS_T](#)
A table containing PD redundant group information.
- struct [TRDP_MARSHALL_CONFIG_T](#)
Marshaling/unmarshalling configuration.
- struct [TRDP_PD_CONFIG_T](#)
Default PD configuration.
- struct [TRDP_MD_CONFIG_T](#)
Default MD configuration.
- struct [TRDP_MEM_CONFIG_T](#)
Enumeration type for memory pre-fragmentation, reuse of VOS definition.
- struct [TRDP_PROCESS_CONFIG_T](#)
Various flags/general TRDP options for library initialization.

Defines

- `#define USE_HEAP 0`
If this is set, we can allocate dynamically memory.
- `#define TRDP_BOOL8 TRDP_BITSET8`
1 bit relevant (equal to zero = false, not equal to zero = true)
- `#define TRDP_ANTIVALENT8 TRDP_BITSET8`
2 bit relevant (0x0 = error; 0x01 = false, 0x02 = true, 0x03 undefined)

Typedefs

- `typedef VOS_IP4_ADDR_T TRDP_IP_ADDR_T`
TRDP general type definitions.
- `typedef VOS_VERSION_T TRDP_VERSION_T`
Version information.
- `typedef VOS_TIME_T TRDP_TIME_T`
Timer value compatible with timeval / select.
- `typedef VOS_FDS_T TRDP_FDS_T`
File descriptor set compatible with fd_set / select.
- `typedef VOS_UUID_T TRDP_UUID_T`
UUID definition reuses the VOS definition.
- `typedef struct TRDP_DATASET TRDP_DATASET_T`
Dataset definition.
- `typedef TRDP_DATASET_T * pTRDP_DATASET_T`
Array of pointers to dataset.
- `typedef VOS_PRINT_DBG_T TRDP_PRINT_DBG_T`
TRDP configuration type definitions.
- `typedef VOS_LOG_T TRDP_LOG_T`
Categories for logging, reuse of the VOS definition.
- `typedef TRDP_ERR_T(* TRDP_MARSHALL_T)(void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDst, UINT32 *pDstSize, TRDP_DATASET_T **ppCachedDS)`
Function type for marshalling .
- `typedef TRDP_ERR_T(* TRDP_UNMARSHALL_T)(void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDst, UINT32 *pDstSize, TRDP_DATASET_T **ppCachedDS)`
Function type for unmarshalling.

- typedef void(* [TRDP_PD_CALLBACK_T](#))(void *pRefCon, TRDP_APP_SESSION_T appHandle, const [TRDP_PD_INFO_T](#) *pMsg, UINT8 *pData, UINT32 dataSize)
Callback for receiving indications, timeouts, releases, responses.
- typedef void(* [TRDP_MD_CALLBACK_T](#))(void *pRefCon, TRDP_APP_SESSION_T appHandle, const [TRDP_MD_INFO_T](#) *pMsg, UINT8 *pData, UINT32 dataSize)
Callback for receiving indications, timeouts, releases, responses.

Enumerations

- enum [TRDP_ERR_T](#) {
 [TRDP_NO_ERR](#) = 0,
 [TRDP_PARAM_ERR](#) = -1,
 [TRDP_INIT_ERR](#) = -2,
 [TRDP_NOINIT_ERR](#) = -3,
 [TRDP_TIMEOUT_ERR](#) = -4,
 [TRDP_NODATA_ERR](#) = -5,
 [TRDP_SOCKET_ERR](#) = -6,
 [TRDP_IO_ERR](#) = -7,
 [TRDP_MEM_ERR](#) = -8,
 [TRDP_SEMA_ERR](#) = -9,
 [TRDP_QUEUE_ERR](#) = -10,
 [TRDP_QUEUE_FULL_ERR](#) = -11,
 [TRDP_MUTEX_ERR](#) = -12,
 [TRDP_THREAD_ERR](#) = -13,
 [TRDP_BLOCK_ERR](#) = -14,
 [TRDP_INTEGRATION_ERR](#) = -15,
 [TRDP_NOCONN_ERR](#) = -16,
 [TRDP_NOSESSION_ERR](#) = -30,
 [TRDP_SESSION_ABORT_ERR](#) = -31,
 [TRDP_NOSUB_ERR](#) = -32,
 [TRDP_NOPUB_ERR](#) = -33,
 [TRDP_NOLIST_ERR](#) = -34,
 [TRDP_CRC_ERR](#) = -35,
 [TRDP_WIRE_ERR](#) = -36,
 [TRDP_TOPO_ERR](#) = -37,
 [TRDP_COMID_ERR](#) = -38,
 [TRDP_STATE_ERR](#) = -39,
 [TRDP_APP_TIMEOUT_ERR](#) = -40,
 [TRDP_APP_REPLYTO_ERR](#) = -41,
 [TRDP_APP_CONFIRMTO_ERR](#) = -42,
 [TRDP_REPLYTO_ERR](#) = -43,


```
TRDP_CONFIRMTO_ERR = -44,  
TRDP_REQCONFIRMTO_ERR = -45,  
TRDP_PACKET_ERR = -46,  
TRDP_UNRESOLVED_ERR = -47,  
TRDP_XML_PARSER_ERR = -48,  
TRDP_INUSE_ERR = -49,  
TRDP_MARSHALLING_ERR = -50,  
TRDP_UNKNOWN_ERR = -99 }
```

Return codes for all API functions, -1.

- enum `TRDP_REPLY_STATUS_T`

TRDP data transfer type definitions.

- enum `TRDP_FLAGS_T` {
 `TRDP_FLAGS_DEFAULT` = 0,
 `TRDP_FLAGS_NONE` = 0x01,
 `TRDP_FLAGS_MARSHALL` = 0x02,
 `TRDP_FLAGS_CALLBACK` = 0x04,
 `TRDP_FLAGS_TCP` = 0x08,
 `TRDP_FLAGS_FORCE_CB` = 0x10 }

Various flags for PD and MD packets.

- enum `TRDP_RED_STATE_T` {
 `TRDP_RED_FOLLOWER` = 0,
 `TRDP_RED_LEADER` = 1 }

Redundancy states.

- enum `TRDP_TO_BEHAVIOR_T` {
 `TRDP_TO_DEFAULT` = 0,
 `TRDP_TO_SET_TO_ZERO` = 1,
 `TRDP_TO_KEEP_LAST_VALUE` = 2 }

How invalid PD shall be handled.

- enum `TRDP_DATA_TYPE_T` {
 `TRDP_BITSET8` = 1,
 `TRDP_CHAR8` = 2,
 `TRDP_UTF16` = 3,
 `TRDP_INT8` = 4,
 `TRDP_INT16` = 5,
 `TRDP_INT32` = 6,
 `TRDP_INT64` = 7,
 `TRDP_UINT8` = 8,
 `TRDP_UINT16` = 9,

```

TRDP_UINT32 = 10,
TRDP_UINT64 = 11,
TRDP_REAL32 = 12,
TRDP_REAL64 = 13,
TRDP_TIMEDATE32 = 14,
TRDP_TIMEDATE48 = 15,
TRDP_TIMEDATE64 = 16,
TRDP_TYPE_MAX = 30 }

```

TRDP dataset description definitions.

- enum TRDP_OPTION_T { ,
TRDP_OPTION_BLOCK = 0x01,
TRDP_OPTION_TRAFFIC_SHAPING = 0x02,
TRDP_OPTION_NO_REUSE_ADDR = 0x04,
TRDP_OPTION_NO_MC_LOOP_BACK = 0x08,
TRDP_OPTION_NO_UDP_CHK = 0x10 }

Various flags/general TRDP options for library initialization.

5.11.1 Detailed Description

Typedefs for TRDP communication.

F

Note:

Project: TCNOpen TRDP prototype stack

Author:

Bernd Loehr, NewTec GmbH

Remarks:

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

BL 2016-02-11: Ticket #111: 'unit', 'scale', 'offset' attributes added to TRDP_DATASET_ELEMENT BL 2016-01-25: Ticket #106: User needs to be informed on every received PD packet BL 2015-12-14: Ticket #33: source size check for marshalling BL 2015-08-05: Ticket #81: Counts for packet loss BL 2014-07-14: Ticket #46: Protocol change: operational topocount needed BL 2014-02-27: Ticket #17: [tlp_subscribe\(\)](#) returns wrong *pSubHandle

5.11.2 Typedef Documentation

5.11.2.1 typedef VOS_IP4_ADDR_T TRDP_IP_ADDR_T

TRDP general type definitions.

5.11.2.2 `typedef TRDP_ERR_T(* TRDP_MARSHALL_T)(void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDst, UINT32 *pDstSize, TRDP_DATASET_T **ppCachedDS)`

Function type for marshalling .

The function must know about the dataset's alignment etc.

Parameters:

- ← **pRefCon* pointer to user context
- ← *comId* ComId to identify the structure out of a configuration
- ← **pSrc* pointer to received original message
- ← *srcSize* size of the source buffer
- ← **pDst* pointer to a buffer for the treated message
- ↔ **pDstSize* size of the provide buffer / size of the treated message
- ↔ **ppCachedDS* pointer to pointer of cached dataset

Return values:

- TRDP_NO_ERR* no error
- TRDP_MEM_ERR* provided buffer to small
- TRDP_COMID_ERR* comid not existing

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

Callback for receiving indications, timeouts, releases, responses.

Parameters:

- ← *appHandle* handle returned also by tlc_init
- ← **pRefCon* pointer to user context
- ← **pMsg* pointer to received message information
- ← **pData* pointer to received data
- ← *dataSize* size of received data pointer to received data

5.11.2.4 `typedef void(* TRDP_PD_CALLBACK_T)(void *pRefCon, TRDP_APP_SESSION_T appHandle, const TRDP_PD_INFO_T *pMsg, UINT8 *pData, UINT32 dataSize)`

Callback for receiving indications, timeouts, releases, responses.

Parameters:

- ← **pRefCon* pointer to user context
- ← *appHandle* application handle returned by tlc_openSession
- ← **pMsg* pointer to received message information
- ← **pData* pointer to received data
- ← *dataSize* size of received data pointer to received data

5.11.2.5 typedef VOS_PRINT_DBG_T TRDP_PRINT_DBG_T

TRDP configuration type definitions.

Callback function definition for error/debug output, reuse of the VOS defined function.

5.11.2.6 typedef VOS_TIME_T TRDP_TIME_T

Timer value compatible with timeval / select.

Relative or absolute date, depending on usage

5.11.2.7 typedef TRDP_ERR_T(* TRDP_UNMARSHALL_T)(void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDst, UINT32 *pDstSize, TRDP_DATASET_T **ppCachedDS)

Function type for unmarshalling.

The function must know about the dataset's alignment etc.

Parameters:

- ← **pRefCon* pointer to user context
- ← *comId* ComId to identify the structure out of a configuration
- ← **pSrc* pointer to received original message
- ← *srcSize* data length from TRDP packet header
- ← **pDst* pointer to a buffer for the treated message
- ↔ **pDstSize* size of the provide buffer / size of the treated message
- ↔ **ppCachedDS* pointer to pointer of cached dataset

Return values:

- TRDP_NO_ERR* no error
- TRDP_MEM_ERR* provide buffer to small
- TRDP_COMID_ERR* comid not existing

5.11.3 Enumeration Type Documentation

5.11.3.1 enum TRDP_DATA_TYPE_T

TRDP dataset description definitions.

Dataset element definition

Enumerator:

- TRDP_BITSET8* =UINT8
- TRDP_CHAR8* char, can be used also as UTF8
- TRDP_UTF16* Unicode UTF-16 character.
- TRDP_INT8* Signed integer, 8 bit.
- TRDP_INT16* Signed integer, 16 bit.

TRDP_INT32 Signed integer, 32 bit.
TRDP_INT64 Signed integer, 64 bit.
TRDP_UINT8 Unsigned integer, 8 bit.
TRDP_UINT16 Unsigned integer, 16 bit.
TRDP_UINT32 Unsigned integer, 32 bit.
TRDP_UINT64 Unsigned integer, 64 bit.
TRDP_REAL32 Floating point real, 32 bit.
TRDP_REAL64 Floating point real, 64 bit.
TRDP_TIMEDATE32 32 bit UNIX time
TRDP_TIMEDATE48 48 bit TCN time (32 bit UNIX time and 16 bit ticks)
TRDP_TIMEDATE64 32 bit UNIX time + 32 bit microseconds (== struct timeval)
TRDP_TYPE_MAX Values greater are considered nested datasets.

5.11.3.2 enum TRDP_ERR_T

Return codes for all API functions, -1.

.-29 taken over from vos

Enumerator:

TRDP_NO_ERR No error.
TRDP_PARAM_ERR Parameter missing or out of range.
TRDP_INIT_ERR Call without valid initialization.
TRDP_NOINIT_ERR Call with invalid handle.
TRDP_TIMEOUT_ERR Timeout.
TRDP_NODATA_ERR Non blocking mode: no data received.
TRDP SOCK_ERR Socket error / option not supported.
TRDP_IO_ERR Socket IO error, data can't be received/sent.
TRDP_MEM_ERR No more memory available.
TRDP_SEMA_ERR Semaphore not available.
TRDP_QUEUE_ERR Queue empty.
TRDP_QUEUE_FULL_ERR Queue full.
TRDP_MUTEX_ERR Mutex not available.
TRDP_THREAD_ERR Thread error.
TRDP_BLOCK_ERR System call would have blocked in blocking mode.
TRDP_INTEGRATION_ERR Alignment or endianness for selected target wrong.
TRDP_NOCONN_ERR No TCP connection.
TRDP_NOSESSION_ERR No such session.
TRDP_SESSION_ABORT_ERR Session aborted.
TRDP_NOSUB_ERR No subscriber.
TRDP_NOPUB_ERR No publisher.
TRDP_NOLIST_ERR No listener.

TRDP_CRC_ERR Wrong CRC.
TRDP_WIRE_ERR Wire.
TRDP_TOPO_ERR Invalid topo count.
TRDP_COMID_ERR Unknown ComId.
TRDP_STATE_ERR Call in wrong state.
TRDP_APP_TIMEOUT_ERR Application Timeout.
TRDP_APP_REPLYTO_ERR Application Reply Sent Timeout.
TRDP_APP_CONFIRMTO_ERR Application Confirm Sent Timeout.
TRDP_REPLYTO_ERR Protocol Reply Timeout.
TRDP_CONFIRMTO_ERR Protocol Confirm Timeout.
TRDP_REQCONFIRMTO_ERR Protocol Confirm Timeout (Request sender).
TRDP_PACKET_ERR Incomplete message data packet.
TRDP_UNRESOLVED_ERR DNR: address could not be resolved.
TRDP_XML_PARSER_ERR Returned by the tau_xml subsystem.
TRDP_INUSE_ERR Resource is still in use.
TRDP_MARSHALLING_ERR Source size exceeded, dataset mismatch.
TRDP_UNKNOWN_ERR Unspecified error.

5.11.3.3 enum TRDP_FLAGS_T

Various flags for PD and MD packets.

Enumerator:

TRDP_FLAGS_DEFAULT Default value defined in tlc_openDession will be taken.
TRDP_FLAGS_NONE No flags set.
TRDP_FLAGS_MARSHALL Optional marshalling/unmarshalling in TRDP stack.
TRDP_FLAGS_CALLBACK Use of callback function.
TRDP_FLAGS_TCP Use TCP for message data.
TRDP_FLAGS_FORCE_CB Force a callback for every received packet.

5.11.3.4 enum TRDP_OPTION_T

Various flags/general TRDP options for library initialization.

Enumerator:

TRDP_OPTION_BLOCK Default: Use nonblocking I/O calls, polling necessary Set: Read calls will block, use select().
TRDP_OPTION_TRAFFIC_SHAPING Use traffic shaping - distribute packet sending Default: OFF.
TRDP_OPTION_NO_REUSE_ADDR Do not allow re-use of address/port (-> no multihoming) Default: Allow.
TRDP_OPTION_NO_MC_LOOP_BACK Do not allow loop back of multicast traffic Default: Allow.
TRDP_OPTION_NO_UDP_CHK Suppress UDP CRC generation Default: Compute UDP CRC.

5.11.3.5 enum TRDP_RED_STATE_T

Redundancy states.

Enumerator:

TRDP_RED_FOLLOWER Redundancy follower - redundant PD will be not sent out.

TRDP_RED_LEADER Redundancy leader - redundant PD will be sent out.

5.11.3.6 enum TRDP_REPLY_STATUS_T

TRDP data transfer type definitions.

Reply status messages

5.11.3.7 enum TRDP_TO_BEHAVIOR_T

How invalid PD shall be handled.

Enumerator:

TRDP_TO_DEFAULT Default value defined in tlc_openDession will be taken.

TRDP_TO_SET_TO_ZERO If set, data will be reset to zero on time out.

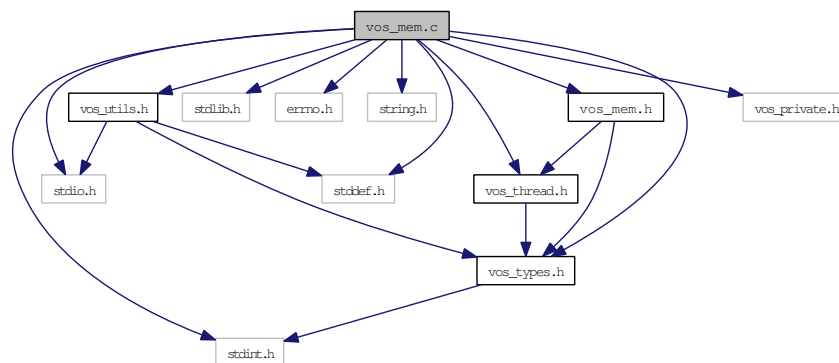
TRDP_TO_KEEP_LAST_VALUE If set, last received values will be returned.

5.12 vos_mem.c File Reference

Memory functions.

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

Include dependency graph for vos_mem.c:



Functions

- EXT_DECL [VOS_ERR_T vos_memInit](#) (UINT8 *pMemoryArea, UINT32 size, const UINT32 fragMem[VOS_MEM_NBLOCKSIZES])
Initialize the memory unit.
- EXT_DECL void [vos_memDelete](#) (UINT8 *pMemoryArea)
Delete the memory area.
- EXT_DECL UINT8 * [vos_memAlloc](#) (UINT32 size)
Allocate a block of memory (from memory area above).
- EXT_DECL void [vos_memFree](#) (void *pMemBlock)
Deallocate a block of memory (from memory area above).

- EXT_DECL `VOS_ERR_T vos_memCount` (UINT32 *pAllocatedMemory, UINT32 *pFreeMemory, UINT32 *pMinFree, UINT32 *pNumAllocBlocks, UINT32 *pNumAllocErr, UINT32 *pNumFreeErr, UINT32 blockSize[VOS_MEM_NBLOCKSIZES], UINT32 usedBlockSize[VOS_MEM_NBLOCKSIZES])
Return used and available memory (of memory area above).
- EXT_DECL void `vos_qsort` (void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))
Sort an array.
- EXT_DECL void * `vos_bsearch` (const void *pKey, const void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))
Binary search in a sorted array.
- EXT_DECL INT32 `vos_strncmp` (const CHAR8 *pStr1, const CHAR8 *pStr2, UINT32 count)
Case insensitive string compare.
- EXT_DECL void `vos_strncpy` (CHAR8 *pStrDst, const CHAR8 *pStrSrc, UINT32 count)
String copy with length limitation.
- EXT_DECL void `vos_strncat` (CHAR8 *pStrDst, UINT32 count, const CHAR8 *pStrSrc)
String concatenation with length limitation.
- EXT_DECL `VOS_ERR_T vos_queueCreate` (`VOS_QUEUE_POLICY_T` queueType, UINT32 maxNoOfMsg, `VOS_QUEUE_T` *pQueueHandle)
Initialize a message queue.
- EXT_DECL `VOS_ERR_T vos_queueSend` (`VOS_QUEUE_T` queueHandle, UINT8 *pData, UINT32 size)
Send a message.
- EXT_DECL `VOS_ERR_T vos_queueReceive` (`VOS_QUEUE_T` queueHandle, UINT8 **ppData, UINT32 *pSize, UINT32 usTimeout)
Get a message.
- EXT_DECL `VOS_ERR_T vos_queueDestroy` (`VOS_QUEUE_T` queueHandle)
Destroy a message queue.

5.12.1 Detailed Description

Memory functions.

OS abstraction of memory access and control

Note:

Project: TCNOpen TRDP prototype stack

Author:

Bernd Loehr, NewTec GmbH

Remarks:

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

Id

[vos_mem.c](#) 1519 2016-02-26 16:18:39Z bloehr

Changes: BL 2016-02-10: Debug print: tabs before size output BL 2012-12-03: ID 1: "using uninitialized PD_ELE_T.pullIpAddress variable" ID 2: "uninitialized PD_ELE_T newPD → pNext in tlp_subscribe()"

5.12.2 Function Documentation**5.12.2.1 EXT_DECL void* vos_bsearch (const void * *pKey*, const void * *pBuf*, UINT32 *num*, UINT32 *size*, int(*)(const void *, const void *) *compare*)**

Binary search in a sorted array.

This is just a wrapper for the standard bsearch function.

Parameters:

- ← *pKey* Key to search for
- ← *pBuf* Pointer to the array to search
- ← *num* number of elements
- ← *size* size of one element
- ← *compare* Pointer to compare function return -n if arg1 < arg2, return 0 if arg1 == arg2, return +n if arg1 > arg2 where n is an integer != 0

Return values:

Pointer to found element or NULL

5.12.2.2 EXT_DECL UINT8* vos_memAlloc (UINT32 *size*)

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

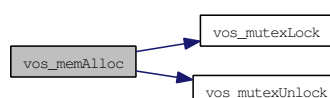
Parameters:

- ← *size* Size of requested block

Return values:

Pointer to memory area
NULL if no memory available

Here is the call graph for this function:



5.12.2.3 EXT_DECL VOS_ERR_T vos_memCount (UINT32 * *pAllocatedMemory*, UINT32 * *pFreeMemory*, UINT32 * *pMinFree*, UINT32 * *pNumAllocBlocks*, UINT32 * *pNumAllocErr*, UINT32 * *pNumFreeErr*, UINT32 *blockSize*[VOS_MEM_NBLOCKSIZES], UINT32 *usedBlockSize*[VOS_MEM_NBLOCKSIZES])

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

Parameters:

- *pAllocatedMemory* Pointer to allocated memory size
- *pFreeMemory* Pointer to free memory size
- *pMinFree* Pointer to minimal free memory size in statistics interval
- *pNumAllocBlocks* Pointer to number of allocated memory blocks
- *pNumAllocErr* Pointer to number of allocation errors
- *pNumFreeErr* Pointer to number of free errors
- *blockSize* Pointer to list of memory block sizes
- *usedBlockSize* Pointer to list of used memoryblocks

Return values:

- VOS_NO_ERR* no error
- VOS_INIT_ERR* module not initialised

5.12.2.4 EXT_DECL void vos_memDelete (UINT8 * *pMemoryArea*)

Delete the memory area.

This will eventually invalidate any previously allocated memory blocks! It should be called last before the application quits. No further access to the memory blocks is allowed after this call.

Parameters:

- ← *pMemoryArea* Pointer to memory area used

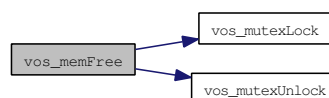
5.12.2.5 EXT_DECL void vos_memFree (void * *pMemBlock*)

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

Parameters:

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

Here is the call graph for this function:



5.12.2.6 EXT_DECL VOS_ERR_T vos_memInit (UINT8 * *pMemoryArea*, UINT32 *size*, const UINT32 *fragMem*[VOS_MEM_NBLOCKSIZES])

Initialize the memory unit.

Init a supplied block of memory and prepare it for use with `vos_memAlloc` and `vos_memFree`. The used block sizes can be supplied and will be preallocated. If half of the overall size of the requested memory area would be pre-allocated, either by the default pre-allocation table or a provided one, no pre-allocation takes place.

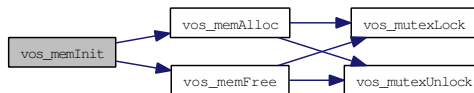
Parameters:

- ← *pMemoryArea* Pointer to memory area to use
- ← *size* Size of provided memory area
- ← *fragMem* Pointer to list of preallocated block sizes, used to fragment memory for large blocks

Return values:

- VOS_NO_ERR** no error
- VOS_PARAM_ERR** parameter out of range/invalid
- VOS_MEM_ERR** no memory available
- VOS_MUTEX_ERR** no mutex available

Here is the call graph for this function:



5.12.2.7 EXT_DECL void vos_qsort (void * *pBuf*, UINT32 *num*, UINT32 *size*, int(*) (const void *, const void *) *compare*)

Sort an array.

This is just a wrapper for the standard `qsort` function.

Parameters:

- ↔ *pBuf* Pointer to the array to sort
- ← *num* number of elements
- ← *size* size of one element
- ← *compare* Pointer to compare function return -n if `arg1 < arg2`, return 0 if `arg1 == arg2`, return +n if `arg1 > arg2` where n is an integer != 0

Return values:

none

5.12.2.8 EXT_DECL VOS_ERR_T vos_queueCreate (VOS_QUEUE_POLICY_T *queueType*, UINT32 *maxNoOfMsg*, VOS_QUEUE_T * *pQueueHandle*)

Initialize a message queue.

Returns a handle for further calls

Parameters:

← *queueType* Define queue type (1 = FIFO, 2 = LIFO, 3 = PRIO)

← *maxNoOfMsg* Maximum number of messages

→ *pQueueHandle* Handle of created queue

Return values:

VOS_NO_ERR no error

VOS_INIT_ERR module not initialised

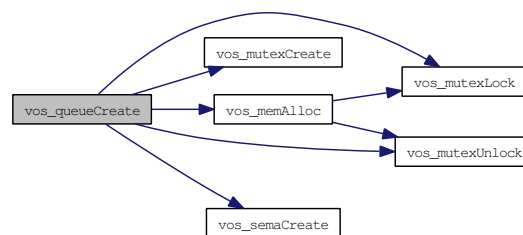
VOS_NOINIT_ERR invalid handle

VOS_PARAM_ERR parameter out of range/invalid

VOS_INIT_ERR not supported

VOS_QUEUE_ERR error creating queue

Here is the call graph for this function:



5.12.2.9 EXT_DECL VOS_ERR_T vos_queueDestroy (VOS_QUEUE_T *queueHandle*)

Destroy a message queue.

Free all resources used by this queue

Parameters:

← *queueHandle* Queue handle

Return values:

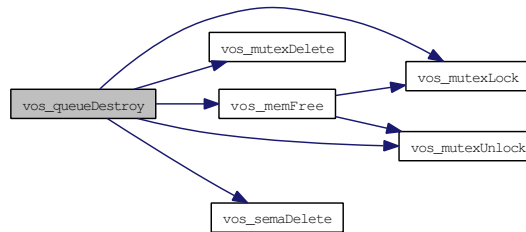
VOS_NO_ERR no error

VOS_INIT_ERR module not initialised

VOS_NOINIT_ERR invalid handle

VOS_PARAM_ERR parameter out of range/invalid

Here is the call graph for this function:



5.12.2.10 EXT_DECL VOS_ERR_T vos_queueReceive (VOS_QUEUE_T *queueHandle*, UINT8 ***ppData*, UINT32 **pSize*, UINT32 *usTimeout*)

Get a message.

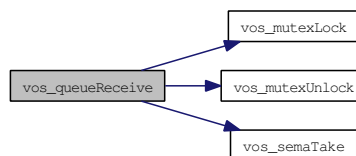
Parameters:

- ← *queueHandle* Queue handle
- *ppData* Pointer to data pointer to be received
- *pSize* Size of receive data
- ← *usTimeout* Maximum time to wait for a message (in usec)

Return values:

- VOSNO_ERR** no error
- VOS_INIT_ERR** module not initialised
- VOS_NOINIT_ERR** invalid handle
- VOS_PARAM_ERR** parameter out of range/invalid
- VOS_QUEUE_ERR** queue is empty

Here is the call graph for this function:



5.12.2.11 EXT_DECL VOS_ERR_T vos_queueSend (VOS_QUEUE_T *queueHandle*, UINT8 **pData*, UINT32 *size*)

Send a message.

Parameters:

- ← *queueHandle* Queue handle

← *pData* Pointer to data to be sent

← *size* Size of data to be sent

Return values:

VOS_NO_ERR no error

VOS_INIT_ERR module not initialised

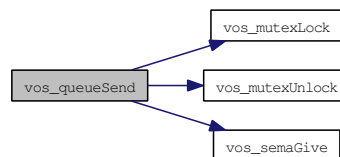
VOS_NOINIT_ERR invalid handle

VOS_PARAM_ERR parameter out of range/invalid

VOS_INIT_ERR not supported

VOS_QUEUE_ERR error creating queue

Here is the call graph for this function:



5.12.2.12 EXT_DECL void vos_strncat (CHAR8 * *pStrDst*, UINT32 *count*, const CHAR8 * *pStrSrc*)

String concatenation with length limitation.

Parameters:

← *pStrDst* Destination string

← *count* Size of destination buffer

← *pStrSrc* Null terminated string to append

Return values:

none

5.12.2.13 EXT_DECL void vos_strncpy (CHAR8 * *pStrDst*, const CHAR8 * *pStrSrc*, UINT32 *count*)

String copy with length limitation.

Parameters:

← *pStrDst* Destination string

← *pStrSrc* Null terminated string to copy

← *count* Maximum number of characters to copy

Return values:

none

**5.12.2.14 EXT_DECL INT32 vos_strnicmp (const CHAR8 * *pStr1*, const CHAR8 * *pStr2*,
UINT32 *count*)**

Case insensitive string compare.

Parameters:

- ← *pStr1* Null terminated string to compare
- ← *pStr2* Null terminated string to compare
- ← *count* Maximum number of characters to compare

Return values:

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

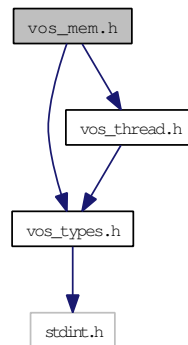
5.13 vos_mem.h File Reference

Memory and queue functions for OS abstraction.

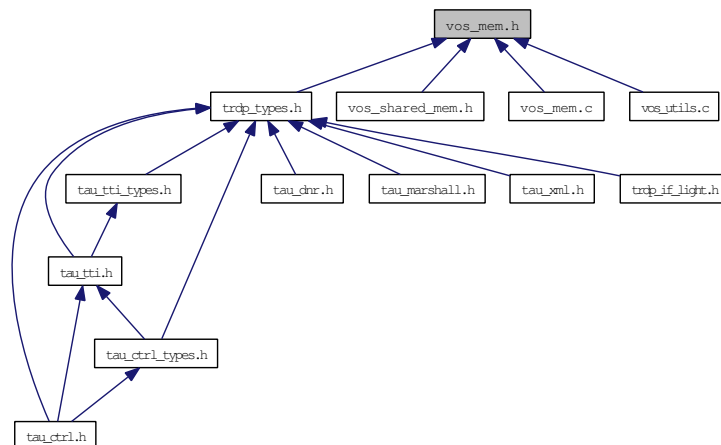
```
#include "vos_types.h"
```

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

Include dependency graph for vos_mem.h:



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



Defines

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

Typedefs

- `typedef struct VOS_QUEUE * VOS_QUEUE_T`

Opaque queue define.

Enumerations

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

Functions

- EXT_DECL [VOS_ERR_T](#) [vos_memInit](#) (UINT8 *pMemoryArea, UINT32 size, const UINT32 fragMem[VOS_MEM_NBLOCKSIZES])
Initialize the memory unit.
- EXT_DECL void [vos_memDelete](#) (UINT8 *pMemoryArea)
Delete the memory area.
- EXT_DECL UINT8 * [vos_memAlloc](#) (UINT32 size)
Allocate a block of memory (from memory area above).
- EXT_DECL void [vos_memFree](#) (void *pMemBlock)
Deallocate a block of memory (from memory area above).
- EXT_DECL [VOS_ERR_T](#) [vos_memCount](#) (UINT32 *pAllocatedMemory, UINT32 *pFreeMemory, UINT32 *pMinFree, UINT32 *pNumAllocBlocks, UINT32 *pNumAllocErr, UINT32 *pNumFreeErr, UINT32 blockSize[VOS_MEM_NBLOCKSIZES], UINT32 usedBlockSize[VOS_MEM_NBLOCKSIZES])
Return used and available memory (of memory area above).
- EXT_DECL void [vos_qsort](#) (void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))
Sort an array.
- EXT_DECL void * [vos_bsearch](#) (const void *pKey, const void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))
Binary search in a sorted array.
- EXT_DECL INT32 [vos_strncmp](#) (const CHAR8 *pStr1, const CHAR8 *pStr2, UINT32 count)
Case insensitive string compare.
- EXT_DECL void [vos_strncpy](#) (CHAR8 *pStr1, const CHAR8 *pStr2, UINT32 count)
String copy with length limitation.
- EXT_DECL void [vos_strncat](#) (CHAR8 *pStrDst, UINT32 count, const CHAR8 *pStrSrc)
String concatenation with length limitation.
- EXT_DECL [VOS_ERR_T](#) [vos_queueCreate](#) ([VOS_QUEUE_POLICY_T](#) queueType, UINT32 maxNoOfMsg, [VOS_QUEUE_T](#) *pQueueHandle)
Initialize a message queue.

- EXT_DECL `VOS_ERR_T vos_queueSend` (`VOS_QUEUE_T` queueHandle, `UINT8 *pData`, `UINT32` size)
Send a message.
- EXT_DECL `VOS_ERR_T vos_queueReceive` (`VOS_QUEUE_T` queueHandle, `UINT8 **ppData`, `UINT32 *pSize`, `UINT32 usTimeout`)
Get a message.
- EXT_DECL `VOS_ERR_T vos_queueDestroy` (`VOS_QUEUE_T` queueHandle)
Destroy a message queue.

5.13.1 Detailed Description

Memory and queue functions for OS abstraction.

This module provides memory control supervision

Note:

Project: TCNOpen TRDP prototype stack

Author:

Bernd Loehr, NewTec GmbH Peter Brander (Memory scheme)

Remarks:

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

Id

`vos_mem.h` 1519 2016-02-26 16:18:39Z bloehr

5.13.2 Define Documentation

5.13.2.1 #define VOS_MEM_BLOCKSIZEs

Value:

```
{32, 48, 128, 180, 256, 512, 1024, 1480, 2048, \
    4096, 11520, 16384, 32768, 65536, 131072}
```

We internally allocate memory always by these block sizes.

The largest available block is 524288 Bytes, provided the overall size of the used memory allocation area is larger.

5.13.2.2 #define VOS_MEM_PREALLOCATE {0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0}

Default pre-allocation of free memory blocks.

To avoid problems with too many small blocks and no large one. Specify how many of each block size that should be pre-allocated (and freed!) to pre-segment the memory area.

5.13.3 Function Documentation

5.13.3.1 EXT_DECL void* vos_bsearch (const void * *pKey*, const void * *pBuf*, UINT32 *num*, UINT32 *size*, int(*)(const void *, const void *) *compare*)

Binary search in a sorted array.

This is just a wrapper for the standard bsearch function.

Parameters:

- ← *pKey* Key to search for
- ← *pBuf* Pointer to the array to search
- ← *num* number of elements
- ← *size* size of one element
- ← *compare* Pointer to compare function return -n if arg1 < arg2, return 0 if arg1 == arg2, return +n if arg1 > arg2 where n is an integer != 0

Return values:

Pointer to found element or NULL

5.13.3.2 EXT_DECL UINT8* vos_memAlloc (UINT32 *size*)

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

Parameters:

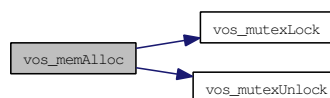
- ← *size* Size of requested block

Return values:

Pointer to memory area

NULL if no memory available

Here is the call graph for this function:



5.13.3.3 `EXT_DECL VOS_ERR_T vos_memCount (UINT32 * pAllocatedMemory, UINT32 * pFreeMemory, UINT32 * pMinFree, UINT32 * pNumAllocBlocks, UINT32 * pNumAllocErr, UINT32 * pNumFreeErr, UINT32 blockSize[VOS_MEM_NBLOCKSIZES], UINT32 usedBlockSize[VOS_MEM_NBLOCKSIZES])`

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

Parameters:

- *pAllocatedMemory* Pointer to allocated memory size
- *pFreeMemory* Pointer to free memory size
- *pMinFree* Pointer to minimal free memory size in statistics interval
- *pNumAllocBlocks* Pointer to number of allocated memory blocks
- *pNumAllocErr* Pointer to number of allocation errors
- *pNumFreeErr* Pointer to number of free errors
- *blockSize* Pointer to list of memory block sizes
- *usedBlockSize* Pointer to list of used memory blocks

Return values:

- VOS_NO_ERR* no error
- VOS_INIT_ERR* module not initialised

5.13.3.4 `EXT_DECL void vos_memDelete (UINT8 * pMemoryArea)`

Delete the memory area.

This will eventually invalidate any previously allocated memory blocks! It should be called last before the application quits. No further access to the memory blocks is allowed after this call.

Parameters:

- ← *pMemoryArea* Pointer to memory area to use

This will eventually invalidate any previously allocated memory blocks! It should be called last before the application quits. No further access to the memory blocks is allowed after this call.

Parameters:

- ← *pMemoryArea* Pointer to memory area used

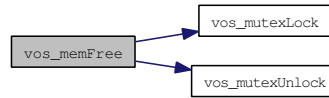
5.13.3.5 `EXT_DECL void vos_memFree (void * pMemBlock)`

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

Parameters:

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

Here is the call graph for this function:



5.13.3.6 EXT_DECL VOS_ERR_T vos_memInit (UINT8 * *pMemoryArea*, UINT32 *size*, const UINT32 *fragMem*[VOS_MEM_NBLOCKSIZES])

Initialize the memory unit.

Init a supplied block of memory and prepare it for use with vos_alloc and vos_dealloc. The used block sizes can be supplied and will be preallocated.

Parameters:

- ← *pMemoryArea* Pointer to memory area to use
- ← *size* Size of provided memory area
- ← *fragMem* Pointer to list of preallocate block sizes, used to fragment memory for large blocks

Return values:

- VOS_NO_ERR** no error
- VOS_PARAM_ERR** parameter out of range/invalid
- VOS_MEM_ERR** no memory available

Init a supplied block of memory and prepare it for use with vos_memAlloc and vos_memFree. The used block sizes can be supplied and will be preallocated. If half of the overall size of the requested memory area would be pre-allocated, either by the default pre-allocation table or a provided one, no pre-allocation takes place.

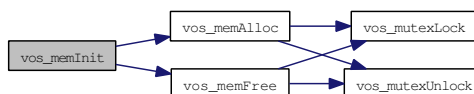
Parameters:

- ← *pMemoryArea* Pointer to memory area to use
- ← *size* Size of provided memory area
- ← *fragMem* Pointer to list of preallocated block sizes, used to fragment memory for large blocks

Return values:

- VOS_NO_ERR** no error
- VOS_PARAM_ERR** parameter out of range/invalid
- VOS_MEM_ERR** no memory available
- VOS_MUTEX_ERR** no mutex available

Here is the call graph for this function:



5.13.3.7 EXT_DECL void vos_qsort (void * *pBuf*, UINT32 *num*, UINT32 *size*, int(*)(const void *, const void *) *compare*)

Sort an array.

This is just a wrapper for the standard qsort function.

Parameters:

- ↔ *pBuf* Pointer to the array to sort
- ← *num* number of elements
- ← *size* size of one element
- ← *compare* Pointer to compare function return -n if arg1 < arg2, return 0 if arg1 == arg2, return +n if arg1 > arg2 where n is an integer != 0

Return values:

none

5.13.3.8 EXT_DECL VOS_ERR_T vos_queueCreate (VOS_QUEUE_POLICY_T *queueType*, UINT32 *maxNoOfMsg*, VOS_QUEUE_T * *pQueueHandle*)

Initialize a message queue.

Returns a handle for further calls

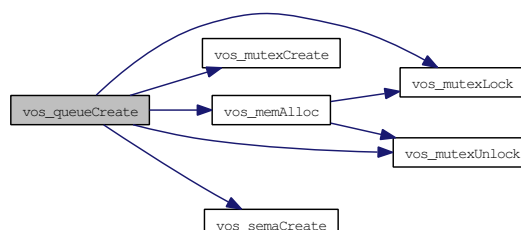
Parameters:

- ← *queueType* Define queue type (1 = FIFO, 2 = LIFO, 3 = PRIO)
- ← *maxNoOfMsg* Maximum number of messages
- *pQueueHandle* Handle of created queue

Return values:

- VOS_NO_ERR* no error
- VOS_INIT_ERR* module not initialised
- VOS_NOINIT_ERR* invalid handle
- VOS_PARAM_ERR* parameter out of range/invalid
- VOS_INIT_ERR* not supported
- VOS_QUEUE_ERR* error creating queue

Here is the call graph for this function:



5.13.3.9 EXT_DECL VOS_ERR_T vos_queueDestroy (VOS_QUEUE_T *queueHandle*)

Destroy a message queue.

Free all resources used by this queue

Parameters:

← *queueHandle* Queue handle

Return values:

VOS_NO_ERR no error

VOS_INIT_ERR module not initialised

VOS_NOINIT_ERR invalid handle

VOS_PARAM_ERR parameter out of range/invalid

Free all resources used by this queue

Parameters:

← *queueHandle* Queue handle

Return values:

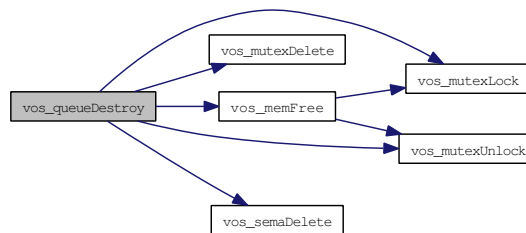
VOS_NO_ERR no error

VOS_INIT_ERR module not initialised

VOS_NOINIT_ERR invalid handle

VOS_PARAM_ERR parameter out of range/invalid

Here is the call graph for this function:



5.13.3.10 EXT_DECL VOS_ERR_T vos_queueReceive (VOS_QUEUE_T *queueHandle*, UINT8 ** *ppData*, UINT32 * *pSize*, UINT32 *usTimeout*)

Get a message.

Parameters:

← *queueHandle* Queue handle

→ *ppData* Pointer to data pointer to be received

→ *pSize* Size of receive data

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

Return values:

VOSNO_ERR no error
VOS_INIT_ERR module not initialised
VOS_NOINIT_ERR invalid handle
VOS_PARAM_ERR parameter out of range/invalid
VOS_QUEUE_ERR queue is empty

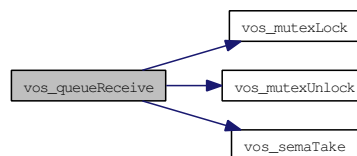
Parameters:

← *queueHandle* Queue handle
→ *ppData* Pointer to data pointer to be received
→ *pSize* Size of receive data
← *usTimeout* Maximum time to wait for a message (in usec)

Return values:

VOSNO_ERR no error
VOS_INIT_ERR module not initialised
VOS_NOINIT_ERR invalid handle
VOS_PARAM_ERR parameter out of range/invalid
VOS_QUEUE_ERR queue is empty

Here is the call graph for this function:



5.13.3.11 EXT_DECL VOS_ERR_T vos_queueSend (VOS_QUEUE_T *queueHandle*, UINT8 * *pData*, UINT32 *size*)

Send a message.

Parameters:

← *queueHandle* Queue handle
← *pData* Pointer to data to be sent
← *size* Size of data to be sent

Return values:

VOS_NO_ERR no error
VOS_INIT_ERR module not initialised

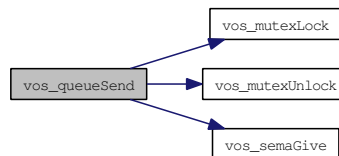
VOS_NOINIT_ERR invalid handle

VOS_PARAM_ERR parameter out of range/invalid

VOS_INIT_ERR not supported

VOS_QUEUE_ERR error creating queue

Here is the call graph for this function:



5.13.3.12 EXT_DECL void vos_strncat (CHAR8 * *pStrDst*, UINT32 *count*, const CHAR8 * *pStrSrc*)

String concatenation with length limitation.

Parameters:

- ← *pStrDst* Destination string
- ← *count* Size of destination buffer
- ← *pStrSrc* Null terminated string to append

Return values:

none

5.13.3.13 EXT_DECL void vos_strncpy (CHAR8 * *pStrDst*, const CHAR8 * *pStrSrc*, UINT32 *count*)

String copy with length limitation.

Parameters:

- ← *pStrDst* Destination string
- ← *pStrSrc* Null terminated string to copy
- ← *count* Maximum number of characters to copy

Return values:

none

5.13.3.14 EXT_DECL INT32 vos_strnicmp (const CHAR8 * *pStr1*, const CHAR8 * *pStr2*,
UINT32 *count*)

Case insensitive string compare.

Parameters:

- ← *pStr1* Null terminated string to compare
- ← *pStr2* Null terminated string to compare
- ← *count* Maximum number of characters to compare

Return values:

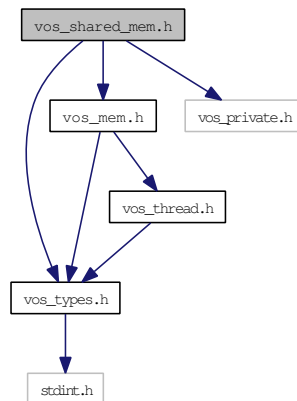
- 0* - equal
- <*0* - string1 less than string 2
- >*0* - string 1 greater than string 2

5.14 vos_shared_mem.h File Reference

Shared Memory functions for OS abstraction.

```
#include "vos_types.h"
#include "vos_mem.h"
#include "vos_private.h"
```

Include dependency graph for vos_shared_mem.h:



Functions

- EXT_DECL [VOS_ERR_T vos_sharedOpen](#) (const CHAR8 *pKey, VOS_SHRD_T *pHandle, UINT8 **ppMemoryArea, UINT32 *pSize)
Create a shared memory area or attach to existing one.
- EXT_DECL [VOS_ERR_T vos_sharedClose](#) (VOS_SHRD_T handle, const UINT8 *pMemoryArea)
Close connection to the shared memory area.

5.14.1 Detailed Description

Shared Memory functions for OS abstraction.

This module provides shared memory control supervision

Note:

Project: TCNOpen TRDP prototype stack

Author:

Kazumasa Aiba, TOSHIBA

Remarks:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>. Copyright TOSHIBA, Japan, 2013.

Id

[vos_mem.h](#) 282 2013-01-11 07:08:44Z 97029

5.14.2 Function Documentation

5.14.2.1 EXT_DECL VOS_ERR_T vos_sharedClose (VOS_SHRD_T *handle*, const UINT8 * *pMemoryArea*)

Close connection to the shared memory area.

If the area was created by the calling process, the area will be closed (freed). If the area was attached, it will be detached. This function is not available in each target implementation.

Parameters:

- ← *handle* Returned handle
- ← *pMemoryArea* Pointer to memory area

Return values:

- VOS_NO_ERR* no error
- VOS_MEM_ERR* no memory available

5.14.2.2 EXT_DECL VOS_ERR_T vos_sharedOpen (const CHAR8 * *pKey*, VOS_SHRD_T * *pHandle*, UINT8 ** *ppMemoryArea*, UINT32 * *pSize*)

Create a shared memory area or attach to existing one.

The first call with the a specified key will create a shared memory area with the supplied size and will return a handle and a pointer to that area. If the area already exists, the area will be opened. This function is not available in each target implementation.

Parameters:

- ← *pKey* Unique identifier (file name)
- *pHandle* Pointer to returned handle
- *ppMemoryArea* Pointer to pointer to memory area
- ↔ *pSize* Pointer to size of area to allocate, on return actual size after attach

Return values:

- VOS_NO_ERR* no error
- VOS_MEM_ERR* no memory available

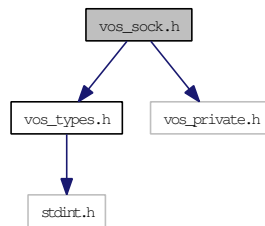
5.15 vos_sock.h File Reference

Typedefs for OS abstraction.

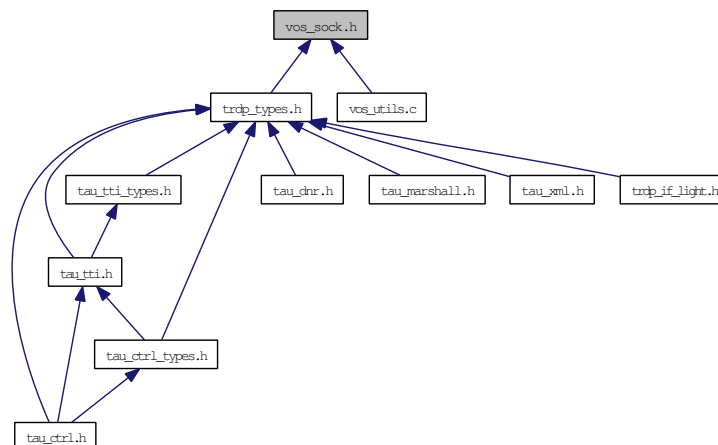
```
#include "vos_types.h"
```

```
#include "vos_private.h"
```

Include dependency graph for vos_sock.h:



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



Data Structures

- struct [VOS_SOCKET_OPT_T](#)
Common socket options.

Defines

- #define [VOS_MAX_SOCKET_CNT](#) 4
The maximum number of sockets influences memory usage; for small systems we should define a smaller set.
- #define [VOS_MAX_MULTICAST_CNT](#) 5
The maximum number of multicast groups one socket can join.

- #define [VOS_TTL_MULTICAST](#) 64
The maximum number of hops a multicast packet can take.
- #define [VOS_MAX_IF_NAME_SIZE](#) 16
The maximum number of IP interface adapters that can be handled by VOS.
- #define [VOS_MAX_NUM_IF](#) 8
The maximum number of unicast addresses that can be handled by VOS.
- #define [VOS_MAX_NUM_UNICAST](#) 10
The MAC size supported by VOS.
- #define [VOS_MAC_SIZE](#) 6
Size of socket send and receive buffer.
- #define [VOS_INVALID_SOCKET](#) -1
Invalid socket number.

Functions

- EXT_DECL UINT16 [vos_htons](#) (UINT16 val)
Byte swapping 2 Bytes.
- EXT_DECL UINT16 [vos_ntohs](#) (UINT16 val)
Byte swapping 2 Bytes.
- EXT_DECL UINT32 [vos_htonl](#) (UINT32 val)
Byte swapping 4 Bytes.
- EXT_DECL UINT32 [vos_ntohl](#) (UINT32 val)
Byte swapping 4 Bytes.
- EXT_DECL UINT32 [vos_dottedIP](#) (const CHAR8 *pDottedIP)
Convert IP address from dotted dec.
- EXT_DECL const CHAR8 * [vos_ipDotted](#) (UINT32 ipAddress)
Convert IP address to dotted dec.
- EXT_DECL BOOL8 [vos_isMulticast](#) (UINT32 ipAddress)
Check if the supplied address is a multicast group address.
- EXT_DECL [VOS_ERR_T](#) [vos_getInterfaces](#) (UINT32 *pAddrCnt, [VOS_IF_REC_T](#) ifAddrs[])
 - Get a list of interface addresses The caller has to provide an array of interface records to be filled.*
- EXT_DECL BOOL8 [vos_netIfUp](#) ([VOS_IP4_ADDR_T](#) ifAddress)
Get the state of an interface.
- EXT_DECL INT32 [vos_select](#) (INT32 highDesc, [VOS_FDS_T](#) *pReadableFD, [VOS_FDS_T](#) *pWriteableFD, [VOS_FDS_T](#) *pErrorFD, [VOS_TIME_T](#) *pTimeout)

select function.

- EXT_DECL [VOS_ERR_T](#) [vos_sockInit](#) (void)
Initialize the socket library.
- EXT_DECL void [vos_sockTerm](#) (void)
De-Initialize the socket library.
- EXT_DECL [VOS_ERR_T](#) [vos_sockGetMAC](#) (UINT8 pMAC[VOS_MAC_SIZE])
Return the MAC address of the default adapter.
- EXT_DECL [VOS_ERR_T](#) [vos_sockOpenUDP](#) (INT32 *pSock, const [VOS_SOCKET_OPT_T](#) *pOptions)
Create an UDP socket.
- EXT_DECL [VOS_ERR_T](#) [vos_sockOpenTCP](#) (INT32 *pSock, const [VOS_SOCKET_OPT_T](#) *pOptions)
Create a TCP socket.
- EXT_DECL [VOS_ERR_T](#) [vos_sockClose](#) (INT32 sock)
Close a socket.
- EXT_DECL [VOS_ERR_T](#) [vos_sockSetOptions](#) (INT32 sock, const [VOS_SOCKET_OPT_T](#) *pOptions)
Set socket options.
- EXT_DECL [VOS_ERR_T](#) [vos_sockJoinMC](#) (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)
Join a multicast group.
- EXT_DECL [VOS_ERR_T](#) [vos_sockLeaveMC](#) (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)
Leave a multicast group.
- EXT_DECL [VOS_ERR_T](#) [vos_sockSendUDP](#) (INT32 sock, const UINT8 *pBuffer, UINT32 *pSize, UINT32 ipAddress, UINT16 port)
Send UDP data.
- EXT_DECL [VOS_ERR_T](#) [vos_sockReceiveUDP](#) (INT32 sock, UINT8 *pBuffer, UINT32 *pSize, UINT32 *pSrcIPAddr, UINT16 *pSrcIPPort, UINT32 *pDstIPAddr, BOOL8 peek)
Receive UDP data.
- EXT_DECL [VOS_ERR_T](#) [vos_sockBind](#) (INT32 sock, UINT32 ipAddress, UINT16 port)
Bind a socket to an address and port.
- EXT_DECL [VOS_ERR_T](#) [vos_sockListen](#) (INT32 sock, UINT32 backlog)
Listen for incoming TCP connections.
- EXT_DECL [VOS_ERR_T](#) [vos_sockAccept](#) (INT32 sock, INT32 *pSock, UINT32 *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 *pSize)

Send TCP data.

- EXT_DECL [VOS_ERR_T vos_sockReceiveTCP](#) (INT32 sock, UINT8 *pBuffer, UINT32 *pSize)

Receive TCP data.

- EXT_DECL [VOS_ERR_T vos_sockSetMulticastIf](#) (INT32 sock, UINT32 mcIfAddress)

Set Using Multicast I/F.

- EXT_DECL [VOS_IP4_ADDR_T vos_determineBindAddr](#) (VOS_IP4_ADDR_T srcIP, VOS_IP4_ADDR_T mcGroup, VOS_IP4_ADDR_T rcvMostly)

Determines the address to bind to since the behaviour in the different OS is different.

5.15.1 Detailed Description

Typedefs for OS abstraction.

This is the declaration for the OS independend socket interface

Note:

Project: TCNOpen TRDP prototype stack

Author:

Bernd Loehr, NewTec GmbH

Remarks:

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

Id

[vos_sock.h](#) 1477 2015-12-11 17:36:53Z bloehr

5.15.2 Define Documentation

5.15.2.1 #define VOS_MAX_SOCKET_CNT 4

The maximum number of sockets influences memory usage; for small systems we should define a smaller set.

The maximum number of concurrent usable sockets per application session

5.15.2.2 #define VOS_TTL_MULTICAST 64

The maximum number of hops a multicast packet can take.

The maximum size for the interface name

5.15.3 Function Documentation

5.15.3.1 EXT_DECL VOS_IP4_ADDR_T vos_determineBindAddr (VOS_IP4_ADDR_T *srcIP*, VOS_IP4_ADDR_T *mcGroup*, VOS_IP4_ADDR_T *rcvMostly*)

Determines the address to bind to since the behaviour in the different OS is different.

Parameters:

- ← *srcIP* IP to bind to (0 = any address)
- ← *mcGroup* MC group to join (0 = do not join)
- ← *rcvMostly* primarily used for receiving (tbd: bind on sender, too?)

Return values:

Address to bind to

5.15.3.2 EXT_DECL UINT32 vos_dottedIP (const CHAR8 * *pDottedIP*)

Convert IP address from dotted dec.

to !host! endianness

Parameters:

- ← *pDottedIP* IP address as dotted decimal.

Return values:

address in UINT32 in host endianness

5.15.3.3 EXT_DECL VOS_ERR_T vos_getInterfaces (UINT32 * *pAddrCnt*, VOS_IF_REC_T *ifAddrs*[])

Get a list of interface addresses The caller has to provide an array of interface records to be filled.

Parameters:

- ↔ *pAddrCnt* in: pointer to array size of interface record out: pointer to number of interface records read
- ↔ *ifAddrs* array of interface records

Return values:

- VOS_NO_ERR* no error
- VOS_PARAM_ERR* *pAddrCnt* and/or *ifAddrs* == NULL
- VOS_MEM_ERR* memory allocation error
- VOS SOCK_ERR* GetAdaptersInfo() error

5.15.3.4 EXT_DECL UINT32 vos_htonl (UINT32 *val*)

Byte swapping 4 Bytes.

Parameters:

← *val* Initial value.

Return values:

swapped value

5.15.3.5 EXT_DECL UINT16 vos_htons (UINT16 *val*)

Byte swapping 2 Bytes.

Parameters:

← *val* Initial value.

Return values:

swapped value

5.15.3.6 EXT_DECL const CHAR8* vos_ipDotted (UINT32 *ipAddress*)

Convert IP address to dotted dec.

from !host! endianness

Parameters:

← *ipAddress* address in UINT32 in host endianness

Return values:

IP address as dotted decimal.

5.15.3.7 EXT_DECL BOOL8 vos_isMulticast (UINT32 *ipAddress*)

Check if the supplied address is a multicast group address.

Parameters:

← *ipAddress* IP address to check.

Return values:

TRUE address is a multicast address

FALSE address is not a multicast address

5.15.3.8 EXT_DECL BOOL8 vos_netIfUp (VOS_IP4_ADDR_T *ifAddress*)

Get the state of an interface.

Parameters:

← *ifAddress* address of interface to check

Return values:

TRUE interface is up and ready *FALSE* interface is down / not ready

5.15.3.9 EXT_DECL UINT32 vos_ntohl (UINT32 *val*)

Byte swapping 4 Bytes.

Parameters:

← *val* Initial value.

Return values:

swapped value

5.15.3.10 EXT_DECL UINT16 vos_ntohs (UINT16 *val*)

Byte swapping 2 Bytes.

Parameters:

← *val* Initial value.

Return values:

swapped value

5.15.3.11 EXT_DECL INT32 vos_select (INT32 *highDesc*, VOS_FDS_T * *pReadableFD*, VOS_FDS_T * *pWriteableFD*, VOS_FDS_T * *pErrorFD*, VOS_TIME_T * *pTimeOut*)

select function.

Set the ready sockets in the supplied sets. Note: Some target systems might define this function as NOP.

Parameters:

- ← *highDesc* max. socket descriptor + 1
- ↔ *pReadableFD* pointer to readable socket set
- ↔ *pWriteableFD* pointer to writeable socket set
- ↔ *pErrorFD* pointer to error socket set
- ← *pTimeOut* pointer to time out value

Return values:

number of ready file descriptors

5.15.3.12 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, 17224 for PD

Return values:

- VOS_NO_ERR* no error
- VOS_PARAM_ERR* NULL parameter, parameter error
- VOS_UNKNOWN_ERR* sock descriptor unknown error

5.15.3.13 EXT_DECL VOS_ERR_T vos_sockBind (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Bind a socket to an address and port.

Parameters:

- ← *sock* socket descriptor
- ← *ipAddress* source IP to receive from, 0 for any
- ← *port* port to receive from

Return values:

- VOS_NO_ERR* no error
- VOS_PARAM_ERR* parameter out of range/invalid
- VOS_IO_ERR* Input/Output error
- VOS_MEM_ERR* resource error

5.15.3.14 EXT_DECL VOS_ERR_T vos_sockClose (INT32 *sock*)

Close a socket.

Release any resources acquired by this socket

Parameters:

- ← *sock* socket descriptor

Return values:

- VOS_NO_ERR* no error
- VOS_PARAM_ERR* pSock == NULL

5.15.3.15 EXT_DECL VOS_ERR_T vos_sockConnect (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Open a TCP connection.

Parameters:

- ← *sock* socket descriptor
- ← *ipAddress* destination IP
- ← *port* destination port

Return values:

- VOS_NO_ERR* no error
- VOS_PARAM_ERR* parameter out of range/invalid
- VOS_IO_ERR* Input/Output error

5.15.3.16 EXT_DECL VOS_ERR_T vos_sockGetMAC (UINT8 *pMAC*[VOS_MAC_SIZE])

Return the MAC address of the default adapter.

Parameters:

- *pMAC* return MAC address.

Return values:

- VOS_NO_ERR* no error
- VOS_PARAM_ERR* *pMAC* == NULL
- VOS SOCK_ERR* socket not available or option not supported

5.15.3.17 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

5.15.3.18 EXT_DECL VOS_ERR_T vos_sockJoinMC (INT32 *sock*, UINT32 *mcAddress*, UINT32 *ipAddress*)

Join a multicast group.

Note: Some target systems might not support this option.

Parameters:

- ← *sock* socket descriptor

- ← *mcAddress* multicast group to join
- ← *ipAddress* depicts interface on which to join, default 0 for any

Return values:

- VOS_NO_ERR* no error
- VOS_PARAM_ERR* parameter out of range/invalid
- VOS SOCK_ERR* option not supported

5.15.3.19 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

5.15.3.20 EXT_DECL VOS_ERR_T vos_sockListen (INT32 *sock*, UINT32 *backlog*)

Listen for incoming TCP connections.

Parameters:

- ← *sock* socket descriptor
- ← *backlog* maximum connection attempts if system is busy

Return values:

- VOS_NO_ERR* no error
- VOS_PARAM_ERR* parameter out of range/invalid
- VOS_IO_ERR* Input/Output error
- VOS_MEM_ERR* resource error

5.15.3.21 **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

5.15.3.22 **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

5.15.3.23 **EXT_DECL VOS_ERR_T vos_sockReceiveTCP (INT32 *sock*, UINT8 * *pBuffer*, UINT32 * *pSize*)**

Receive TCP data.

The caller must provide a sufficient sized buffer. If the supplied buffer is smaller than the bytes received, **pSize* will reflect the number of copied bytes and the call should be repeated until **pSize* is 0 (zero). If the socket was created in blocking-mode (default), then this call will block and will only return if data has been received or the socket was closed or an error occurred. If called in non-blocking mode, and no data is available, *VOS_NODATA_ERR* will be returned.

Parameters:

- ← *sock* socket descriptor
- *pBuffer* pointer to applications data buffer

↔ *pSize* pointer to the received data size

Return values:

VOS_NO_ERR no error

VOS_PARAM_ERR sock descriptor unknown, parameter error

VOS_IO_ERR data could not be read

VOS_NODATA_ERR no data in non-blocking

VOS_BLOCK_ERR call would have blocked in blocking mode

5.15.3.24 EXT_DECL VOS_ERR_T vos_sockReceiveUDP (INT32 *sock*, UINT8 * *pBuffer*, UINT32 * *pSize*, UINT32 * *pSrcIPAddr*, UINT16 * *pSrcIPPort*, UINT32 * *pDstIPAddr*, BOOL8 *peek*)

Receive UDP data.

The caller must provide a sufficient sized buffer. If the supplied buffer is smaller than the bytes received, *pSize will reflect the number of copied bytes and the call should be repeated until *pSize is 0 (zero). If the socket was created in blocking-mode (default), then this call will block and will only return if data has been received or the socket was closed or an error occurred. If called in non-blocking mode, and no data is available, VOS_NODATA_ERR will be returned. If pointers are provided, source IP, source port and destination IP will be reported on return.

Parameters:

← *sock* socket descriptor

→ *pBuffer* pointer to applications data buffer

↔ *pSize* pointer to the received data size

→ *pSrcIPAddr* pointer to source IP

→ *pSrcIPPort* pointer to source port

→ *pDstIPAddr* pointer to dest IP

← *peek* if true, leave data in queue

Return values:

VOS_NO_ERR no error

VOS_PARAM_ERR sock descriptor unknown, parameter error

VOS_IO_ERR data could not be read

VOS_NODATA_ERR no data

VOS_BLOCK_ERR Call would have blocked in blocking mode

5.15.3.25 EXT_DECL VOS_ERR_T vos_sockSendTCP (INT32 *sock*, const UINT8 * *pBuffer*, UINT32 * *pSize*)

Send TCP data.

Send data to the supplied address and port.

Parameters:

- ← *sock* socket descriptor
- ← *pBuffer* pointer to data to send
- ↔ *pSize* In: size of the data to send, Out: no of bytes sent

Return values:

- VOS_NO_ERR* no error
- VOS_PARAM_ERR* sock descriptor unknown, parameter error
- VOS_IO_ERR* data could not be sent
- VOS_NOCONN_ERR* no TCP connection
- VOS_BLOCK_ERR* call would have blocked in blocking mode, data partially sent

5.15.3.26 EXT_DECL VOS_ERR_T vos_sockSendUDP (INT32 *sock*, const UINT8 * *pBuffer*, UINT32 * *pSize*, UINT32 *ipAddress*, UINT16 *port*)

Send UDP data.

Send data to the given address and port.

Parameters:

- ← *sock* socket descriptor
- ← *pBuffer* pointer to data to send
- ↔ *pSize* In: size of the data to send, Out: no of bytes sent
- ← *ipAddress* destination IP
- ← *port* destination port

Return values:

- VOS_NO_ERR* no error
- VOS_PARAM_ERR* parameter out of range/invalid
- VOS_IO_ERR* data could not be sent
- VOS_BLOCK_ERR* Call would have blocked in blocking mode

5.15.3.27 EXT_DECL VOS_ERR_T vos_sockSetMulticastIf (INT32 *sock*, UINT32 *mcIfAddress*)

Set Using Multicast I/F.

Parameters:

- ← *sock* socket descriptor
- ← *mcIfAddress* using Multicast I/F Address

Return values:

- VOS_NO_ERR* no error
- VOS_PARAM_ERR* sock descriptor unknown, parameter error

5.15.3.28 EXT_DECL VOS_ERR_T vos_sockSetOptions (INT32 *sock*, const VOS_SOCK_OPT_T **pOptions*)

Set socket options.

Note: Some target systems might not support each option.

Parameters:

← *sock* socket descriptor

← *pOptions* pointer to socket options (optional)

Return values:

VOS_NO_ERR no error

VOS_PARAM_ERR parameter out of range/invalid

5.15.3.29 EXT_DECL void vos_sockTerm (void)

De-Initialize the socket library.

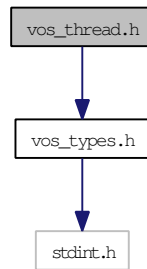
Must be called after last socket call

5.16 vos_thread.h File Reference

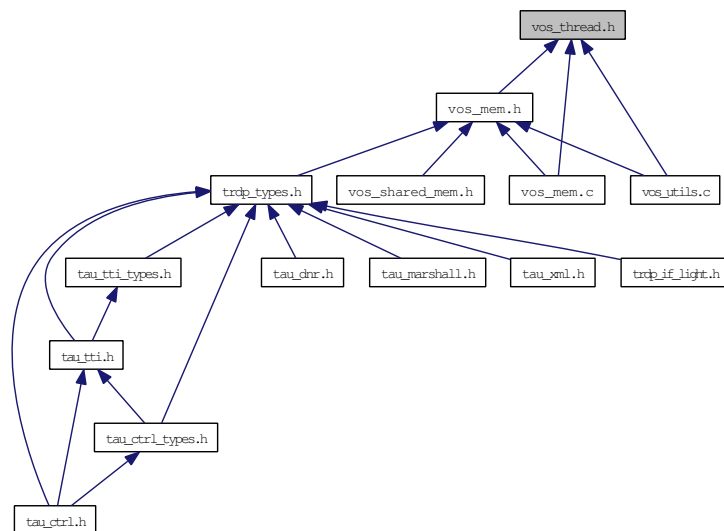
Threading functions for OS abstraction.

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

Include dependency graph for vos_thread.h:



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



Defines

- #define **VOS_MAX_THREAD_CNT** 100
The maximum number of concurrent usable threads.
- #define **VOS_SEMA_WAIT_FOREVER** 0xFFFFFFFFU
Timeout value to wait forever for a semaphore.

Typedefs

- typedef UINT8 **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 * [VOS_MUTEX_T](#)
Hidden mutex handle definition.
- typedef struct VOS_SEMA * [VOS_SEMA_T](#)
Hidden semaphore handle definition.
- typedef void * [VOS_THREAD_T](#)
Hidden thread handle definition.

Enumerations

- enum [VOS_THREAD_POLICY_T](#)
Thread policy matching pthread/Posix defines.
- enum [VOS_SEMA_STATE_T](#)
State of the semaphore.

Functions

- EXT_DECL [VOS_ERR_T](#) vos_threadInit (void)
Initialize the thread library.
- EXT_DECL void [vos_threadTerm](#) (void)
De-Initialize the thread library.
- EXT_DECL [VOS_ERR_T](#) vos_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 void [vos_cyclicThread](#) (UINT32 interval, [VOS_THREAD_FUNC_T](#) pFunction, void *pArguments)
Cyclic thread functions.
- EXT_DECL [VOS_ERR_T](#) vos_threadTerminate ([VOS_THREAD_T](#) thread)
Terminate a thread.
- EXT_DECL [VOS_ERR_T](#) vos_threadIsActive ([VOS_THREAD_T](#) thread)
Is the thread still active? This call will return VOS_NO_ERR if the thread is still active, VOS_PARAM_ERR in case it ran out.
- EXT_DECL [VOS_ERR_T](#) vos_threadDelay (UINT32 delay)

Delay the execution of the current thread by the given delay in us.

- EXT_DECL void [vos_getTime](#) (VOS_TIME_T *pTime)
Return the current time in sec and us.
- EXT_DECL const CHAR8 * [vos_getTimeStamp](#) (void)
Get a time-stamp string.
- EXT_DECL void [vos_clearTime](#) (VOS_TIME_T *pTime)
Clear the time stamp.
- EXT_DECL void [vos_addTime](#) (VOS_TIME_T *pTime, const VOS_TIME_T *pAdd)
Add the second to the first time stamp, return sum in first.
- EXT_DECL void [vos_subTime](#) (VOS_TIME_T *pTime, const VOS_TIME_T *pSub)
Subtract the second from the first time stamp, return diff in first.
- EXT_DECL INT32 [vos_cmpTime](#) (const VOS_TIME_T *pTime, const VOS_TIME_T *pCmp)
Compare the second from the first time stamp, return diff in first.
- EXT_DECL void [vos_divTime](#) (VOS_TIME_T *pTime, UINT32 divisor)
Divide the first time by the second, return quotient in first.
- EXT_DECL void [vos_mulTime](#) (VOS_TIME_T *pTime, UINT32 mul)
Multiply the first time by the second, return product in first.
- EXT_DECL void [vos_getUuid](#) (VOS_UUID_T pUUID)
Get a universal unique identifier according to RFC 4122 time based version.
- EXT_DECL VOS_ERR_T [vos_mutexCreate](#) (VOS_MUTEX_T *pMutex)
Create a mutex.
- EXT_DECL void [vos_mutexDelete](#) (VOS_MUTEX_T pMutex)
Delete a mutex.
- EXT_DECL VOS_ERR_T [vos_mutexLock](#) (VOS_MUTEX_T pMutex)
Take a mutex.
- EXT_DECL VOS_ERR_T [vos_mutexTryLock](#) (VOS_MUTEX_T pMutex)
Try to take a mutex.
- EXT_DECL VOS_ERR_T [vos_mutexUnlock](#) (VOS_MUTEX_T pMutex)
Release a mutex.
- EXT_DECL VOS_ERR_T [vos_semaCreate](#) (VOS_SEMA_T *pSema, VOS_SEMA_STATE_T initialState)
Create a semaphore.
- EXT_DECL void [vos_semaDelete](#) (VOS_SEMA_T sema)
Delete a semaphore.

- EXT_DECL `VOS_ERR_T vos_semaTake (VOS_SEMA_T sema, UINT32 timeout)`
Take a semaphore.
- EXT_DECL void `vos_semaGive (VOS_SEMA_T sema)`
Give a semaphore.

5.16.1 Detailed Description

Threading functions for OS abstraction.

Thread-, semaphore- and time-handling functions

Note:

Project: TCNOpen TRDP prototype stack

Author:

Bernd Loehr, NewTec GmbH

Remarks:

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

Id

`vos_thread.h` 1394 2015-03-27 13:58:54Z ahweiss

5.16.2 Function Documentation

5.16.2.1 EXT_DECL void vos_addTime (VOS_TIME_T * *pTime*, const VOS_TIME_T * *pAdd*)

Add the second to the first time stamp, return sum in first.

Parameters:

↔ *pTime* Pointer to time value

← *pAdd* Pointer to time value

5.16.2.2 EXT_DECL void vos_clearTime (VOS_TIME_T * *pTime*)

Clear the time stamp.

Parameters:

→ *pTime* Pointer to time value

5.16.2.3 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*

5.16.2.4 EXT_DECL void vos_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 (incl. runtime)
- ← *pFunction* Pointer to the thread function
- ← *pArguments* Pointer to the thread function parameters

Return values:

void

5.16.2.5 EXT_DECL void vos_divTime (VOS_TIME_T * *pTime*, UINT32 *divisor*)

Divide the first time by the second, return quotient in first.

Parameters:

- ↔ *pTime* Pointer to time value
- ← *divisor* Divisor

5.16.2.6 EXT_DECL void vos_getTime (VOS_TIME_T * *pTime*)

Return the current time in sec and us.

Parameters:

- *pTime* Pointer to time value

5.16.2.7 EXT_DECL const CHAR8* vos_getTimeStamp (void)

Get a time-stamp string.

Get a time-stamp string for debugging in the form "yyyymmdd-hh:mm:ss.ms" Depending on the used OS / hardware the time might not be a real-time stamp but relative from start of system.

Return values:

timestamp "yyyymmdd-hh:mm:ss.ms"

5.16.2.8 EXT_DECL void vos_getUuid (VOS_UUID_T pUUID)

Get a universal unique identifier according to RFC 4122 time based version.

Parameters:

→ *pUUID* Pointer to a universal unique identifier

5.16.2.9 EXT_DECL void vos_mulTime (VOS_TIME_T *pTime, UINT32 mul)

Multiply the first time by the second, return product in first.

Parameters:

↔ *pTime* Pointer to time value

← *mul* Factor

5.16.2.10 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

5.16.2.11 EXT_DECL void vos_mutexDelete (VOS_MUTEX_T pMutex)

Delete a mutex.

Release the resources taken by the mutex.

Parameters:

← *pMutex* mutex handle

Return values:

VOS_NO_ERR no error

5.16.2.12 EXT_DECL VOS_ERR_T vos_mutexLock (VOS_MUTEX_T pMutex)

Take a mutex.

Wait for the mutex to become available (lock).

Parameters:

← *pMutex* mutex handle

Return values:

VOS_NO_ERR no error

VOS_INIT_ERR module not initialised

VOS_NOINIT_ERR invalid handle

5.16.2.13 EXT_DECL VOS_ERR_T vos_mutexTryLock (VOS_MUTEX_T pMutex)

Try to take a mutex.

If mutex is can't be taken VOS_MUTEX_ERR is returned.

Parameters:

← *pMutex* mutex handle

Return values:

VOS_NO_ERR no error

VOS_INIT_ERR module not initialised

VOS_NOINIT_ERR invalid handle

VOS_MUTEX_ERR no mutex available

5.16.2.14 EXT_DECL VOS_ERR_T vos_mutexUnlock (VOS_MUTEX_T pMutex)

Release a mutex.

Unlock the mutex.

Parameters:

← *pMutex* mutex handle

5.16.2.15 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

5.16.2.16 EXT_DECL void vos_semaDelete (VOS_SEMA_T *sema*)

Delete a semaphore.

This will eventually release any processes waiting for the semaphore.

Parameters:

- ← *sema* semaphore handle

5.16.2.17 EXT_DECL void vos_semaGive (VOS_SEMA_T *sema*)

Give a semaphore.

Release (increase) a semaphore.

Parameters:

- ← *sema* semaphore handle

5.16.2.18 EXT_DECL VOS_ERR_T vos_semaTake (VOS_SEMA_T *sema*, UINT32 *timeout*)

Take a semaphore.

Try to get (decrease) a semaphore.

Parameters:

- ← *sema* semaphore handle
- ← *timeout* Max. time in us to wait, 0 means no wait

Return values:

VOS_NO_ERR no error
VOS_INIT_ERR module not initialised
VOS_NOINIT_ERR invalid handle
VOS_PARAM_ERR parameter out of range/invalid
VOS_SEMA_ERR could not get semaphore in time

5.16.2.19 EXT_DECL void vos_subTime (VOS_TIME_T * *pTime*, const VOS_TIME_T * *pSub*)

Subtract the second from the first time stamp, return diff in first.

Parameters:

↔ ***pTime*** Pointer to time value
 ← ***pSub*** Pointer to time value

5.16.2.20 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

5.16.2.21 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

5.16.2.22 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

5.16.2.23 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

5.16.2.24 EXT_DECL void vos_threadTerm (void)

De-Initialize the thread library.

Must be called after last thread/timer call

5.16.2.25 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

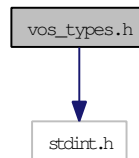
VOS_PARAM_ERR parameter out of range/invalid

5.17 vos_types.h File Reference

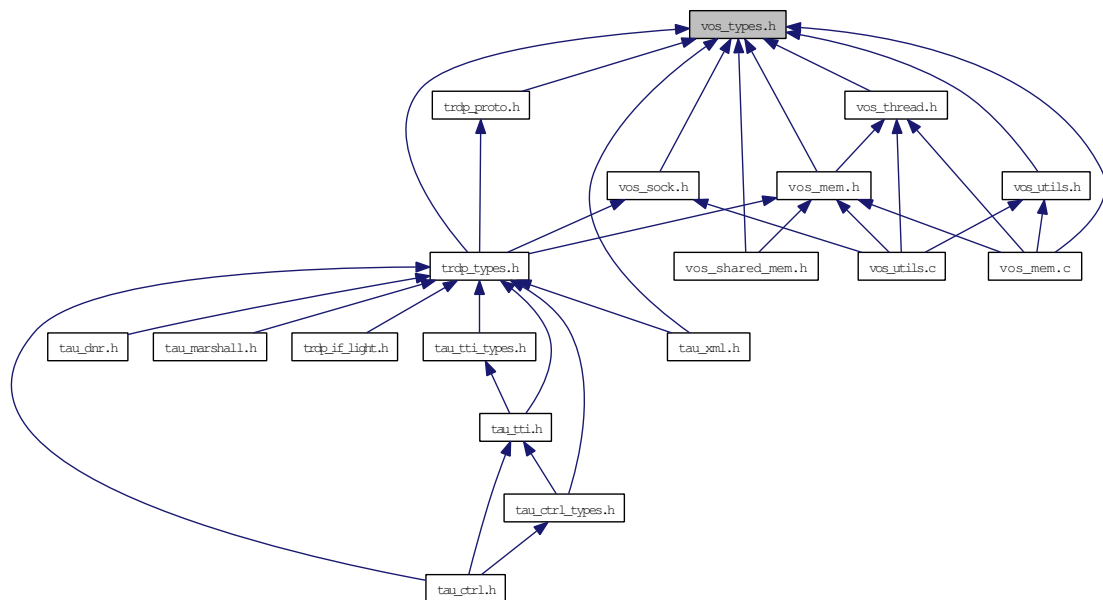
Typedefs for OS abstraction.

```
#include <stdint.h>
```

Include dependency graph for vos_types.h:



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



Data Structures

- struct **VOS_VERSION_T**
Version information.
- struct **VOS_TIME_T**
Timer value compatible with timeval / select.

Defines

- #define **INLINE** inline
- inline macros*

- #define `AV_ERROR` 0x00
ANTIVALENT8 values.
- #define `TR_DIR1` 0x01
Directions/Orientations.

Typedefs

- typedef UINT8 `VOS_UUID_T` [16]
universal unique identifier according to RFC 4122, time based version
- typedef void(* `VOS_PRINT_DBG_T`)(void *pRefCon, `VOS_LOG_T` category, const CHAR8 *pTime, const CHAR8 *pFile, UINT16 LineNumber, const CHAR8 *pMsgStr)
Function definition for error/debug output.

Enumerations

- enum `VOS_ERR_T` {
`VOS_NO_ERR` = 0,
`VOS_PARAM_ERR` = -1,
`VOS_INIT_ERR` = -2,
`VOS_NOINIT_ERR` = -3,
`VOS_TIMEOUT_ERR` = -4,
`VOS_NODATA_ERR` = -5,
`VOS_SOCKET_ERR` = -6,
`VOS_IO_ERR` = -7,
`VOS_MEM_ERR` = -8,
`VOS_SEMA_ERR` = -9,
`VOS_QUEUE_ERR` = -10,
`VOS_QUEUE_FULL_ERR` = -11,
`VOS_MUTEX_ERR` = -12,
`VOS_THREAD_ERR` = -13,
`VOS_BLOCK_ERR` = -14,
`VOS_INTEGRATION_ERR` = -15,
`VOS_NOCONN_ERR` = -16,
`VOS_UNKNOWN_ERR` = -99 }
Return codes for all VOS API functions.
- enum `VOS_LOG_T` {
`VOS_LOG_ERROR` = 0,
`VOS_LOG_WARNING` = 1,
`VOS_LOG_INFO` = 2,
`VOS_LOG_DBG` = 3 }

Categories for logging.

5.17.1 Detailed Description

Typedefs for OS abstraction.

Note:

Project: TCNOpen TRDP prototype stack

Author:

Bernd Loehr, NewTec GmbH

Remarks:

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

Id

[vos_types.h](#) 1424 2015-08-06 11:56:31Z bloehr

5.17.2 Typedef Documentation

5.17.2.1 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

5.17.3 Enumeration Type Documentation

5.17.3.1 enum VOS_ERR_T

Return codes for all VOS API functions.

Enumerator:

VOS_NO_ERR No error.

VOS_PARAM_ERR Necessary parameter missing or out of range.

VOS_INIT_ERR Call without valid initialization.

VOS_NOINIT_ERR The supplied handle/reference is not valid.

VOS_TIMEOUT_ERR Timeout.

VOS_NODATA_ERR Non blocking mode: no data received.

VOS SOCK_ERR Socket option not supported.

VOS_IO_ERR Socket IO error, data can't be received/sent.

VOS_MEM_ERR No more memory available.

VOS_SEMA_ERR Semaphore not available.

VOS_QUEUE_ERR Queue empty.

VOS_QUEUE_FULL_ERR Queue full.

VOS_MUTEX_ERR Mutex not available.

VOS_THREAD_ERR Thread creation error.

VOS_BLOCK_ERR System call would have blocked in blocking mode.

VOS_INTEGRATION_ERR Alignment or endianness for selected target wrong.

VOS_NOCONN_ERR No TCP connection.

VOS_UNKNOWN_ERR Unknown error.

5.17.3.2 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.

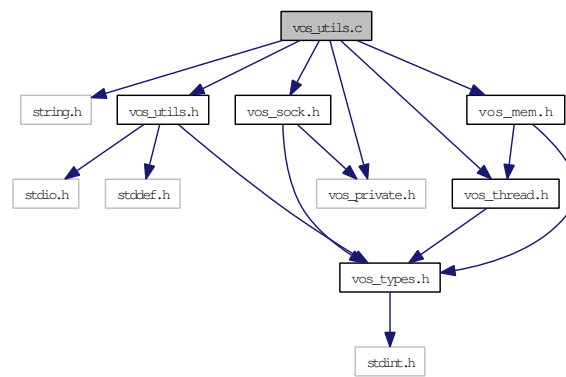
VOS_LOG_DBG This is a debug info.

5.18 vos_utils.c File Reference

Common functions for VOS.

```
#include <string.h>
#include "vos_utils.h"
#include "vos_sock.h"
#include "vos_thread.h"
#include "vos_mem.h"
#include "vos_private.h"
```

Include dependency graph for vos_utils.c:



Functions

- **VOS_ERR_T** [vos_initRuntimeConsts](#) (void)
Pre-compute alignment and endianness.
- **VOS_ERR_T** [vos_init](#) (void *pRefCon, **VOS_PRINT_DBG_T** pDebugOutput)
Initialize the virtual operating system.
- EXT_DECL void [vos_terminate](#) ()
DeInitialize the vos library.
- UINT32 [vos_crc32](#) (UINT32 crc, const UINT8 *pData, UINT32 dataLen)
Compute crc32 according to IEEE802.3.
- const char * [vos_getVersionString](#) (void)
Return a human readable version representation.
- EXT_DECL const **VOS_VERSION_T** * [vos_getVersion](#) (void)
Return version.

5.18.1 Detailed Description

Common functions for VOS.

Common functions of the abstraction layer. Mainly debugging support.

Note:

Project: TCNOpen TRDP prototype stack

Author:

Bernd Loehr, NewTec GmbH

Remarks:

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

Id

[vos_utils.c](#) 1509 2016-02-11 14:29:05Z bloehr

BL 2016-02-10: ifdef DEBUG for some functions BL 2014-02-28: Ticket #25: CRC32 calculation is not according IEEE802.3

5.18.2 Function Documentation

5.18.2.1 `UINT32 vos_crc32 (UINT32 crc, const UINT8 * pData, UINT32 dataLen)`

Compute crc32 according to IEEE802.3.

Calculate CRC for the given buffer and length.

Note: Returned CRC is inverted

Parameters:

← *crc* Initial value.

↔ *pData* Pointer to data.

← *dataLen* length in bytes of data.

Return values:

crc32 according to IEEE802.3

5.18.2.2 `EXT_DECL const VOS_VERSION_T* vos_getVersion (void)`

Return version.

Return pointer to version structure

Return values:

[VOS_VERSION_T](#)

5.18.2.3 const char* vos_getVersionString (void)

Return a human readable version representation.

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

Return values:

const string

5.18.2.4 VOS_ERR_T vos_init (void *pRefCon, VOS_PRINT_DBG_T pDebugOutput)

Initialize the virtual operating system.

Initialize the vos library.

Parameters:

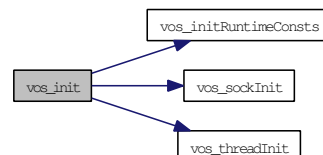
← *pRefCon* context for debug output function

← *pDebugOutput* Pointer to debug output function.

Return values:

VOS_NO_ERR no error *VOS_INTEGRATION_ERR* if endianness/alignment mismatch *VOS_SOCKET_ERR* sockets not supported *VOS_UNKNOWN_ERR* initialisation error

Here is the call graph for this function:

**5.18.2.5 VOS_ERR_T vos_initRuntimeConsts (void)**

Pre-compute alignment and endianness.

Return values:

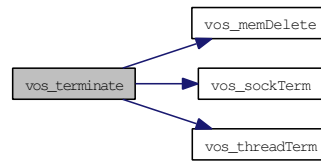
VOS_INTEGRATION_ERR or *VOS_NO_ERR*

5.18.2.6 EXT_DECL void vos_terminate ()

DeInitialize the vos library.

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

Here is the call graph for this function:

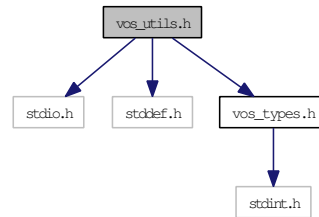


5.19 vos_utils.h File Reference

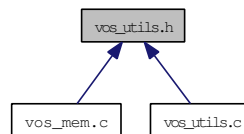
Typedefs for OS abstraction.

```
#include <stdio.h>
#include <stddef.h>
#include "vos_types.h"
```

Include dependency graph for vos_utils.h:



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



Defines

- #define **VOS_MAX_PRNT_STR_SIZE** 256
String size definitions for the debug output functions.
- #define **VOS_MAX_FRMT_SIZE** 64
Max.
- #define **VOS_MAX_ERR_STR_SIZE** (VOS_MAX_PRNT_STR_SIZE - VOS_MAX_FRMT_SIZE)
Max.
- #define **vos_snprintf**(str, size, format, args...) snprintf(str, size, format, ## args)
Safe printf function.
- #define **vos_printLogStr**(level, string)
Debug output macro without formatting options.
- #define **vos_printLog**(level, format, args...)
Debug output macro with formatting options.
- #define **ALIGNOF**(type) ((UINT32)offsetof(struct { char c; type member; }, member))
Alignment macros.

- #define `INITFCS` `0xffffffff`
CRC/FCS constants.
- #define `SIZE_OF_FCS` `4`
for better understanding of address calculations
- #define `L_ENDIAN`
Define endianness if not already done by compiler.

Functions

- EXT_DECL `UINT32 vos_crc32` (`UINT32 crc`, `const UINT8 *pData`, `UINT32 dataLen`)
Calculate CRC for the given buffer and length.
- EXT_DECL `VOS_ERR_T vos_init` (`void *pRefCon`, `VOS_PRINT_DBG_T pDebugOutput`)
Initialize the vos library.
- EXT_DECL `void vos_terminate` ()
DeInitialize the vos library.
- EXT_DECL `const CHAR8 * vos_getVersionString` (`void`)
Return a human readable version representation.
- EXT_DECL `const VOS_VERSION_T * vos_getVersion` (`void`)
Return version.

5.19.1 Detailed Description

Typedefs for OS abstraction.

Note:

Project: TCNOpen TRDP prototype stack

Author:

Bernd Loehr, NewTec GmbH

Remarks:

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

Id

[vos_utils.h](#) 1487 2015-12-21 14:33:26Z bloehr

BL 2014-02-28: Ticket #25: CRC32 calculation is not according IEEE802.3

5.19.2 Define Documentation

5.19.2.1 #define INITFCS 0xffffffff

CRC/FCS constants.

Initial FCS value

5.19.2.2 #define VOS_MAX_ERR_STR_SIZE (VOS_MAX_PRNT_STR_SIZE - VOS_MAX_FRMT_SIZE)

Max.

size of the error part

5.19.2.3 #define VOS_MAX_FRMT_SIZE 64

Max.

size of the 'format' part

5.19.2.4 #define VOS_MAX_PRNT_STR_SIZE 256

String size definitions for the debug output functions.

Max. size of the debug/error string of debug function

5.19.3 Function Documentation

5.19.3.1 EXT_DECL UINT32 vos_crc32 (UINT32 *crc*, const UINT8 * *pData*, UINT32 *dataLen*)

Calculate CRC for the given buffer and length.

For TRDP FCS CRC calculation the CRC32 according to IEEE802.3 with start value 0xffffffff is used.

Parameters:

← *crc* Initial value.

↔ *pData* Pointer to data.

← *dataLen* length in bytes of data.

Return values:

crc32 according to IEEE802.3

Calculate CRC for the given buffer and length.

Note: Returned CRC is inverted

Parameters:

← *crc* Initial value.

↔ *pData* Pointer to data.

← *dataLen* length in bytes of data.

Return values:

crc32 according to IEEE802.3

5.19.3.2 EXT_DECL const VOS_VERSION_T* vos_getVersion (void)

Return version.

Return pointer to version structure

Return values:

const [VOS_VERSION_T](#)

Return pointer to version structure

Return values:

[VOS_VERSION_T](#)

5.19.3.3 EXT_DECL const CHAR8* vos_getVersionString (void)

Return a human readable version representation.

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

Return values:

const string

5.19.3.4 EXT_DECL VOS_ERR_T vos_init (void * *pRefCon*, VOS_PRINT_DBG_T *pDebugOutput*)

Initialize the vos library.

This is used to set the output function for all VOS error and debug output.

Parameters:

← **pRefCon* user context

← **pDebugOutput* pointer to debug output function

Return values:

VOS_NO_ERR no error

VOS_INIT_ERR unsupported

Initialize the vos library.

Parameters:

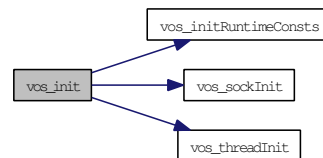
← *pRefCon* context for debug output function

← *pDebugOutput* Pointer to debug output function.

Return values:

VOS_NO_ERR no error **VOS_INTEGRATION_ERR** if endianess/alignment mismatch **VOS_SOCK_ERR** sockets not supported **VOS_UNKNOWN_ERR** initialisation error

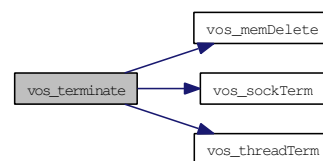
Here is the call graph for this function:

**5.19.3.5 EXT_DECL void vos_terminate ()**

DeInitialize the vos library.

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

Here is the call graph for this function:



Index

- cltrCstCnt
 - TRDP_CONSIST_INFO_T, [28](#)
- cltrCstNo
 - TRDP_CLTR_CST_INFO_T, [24](#)
- cnCnt
 - TRDP_ETB_INFO_T, [32](#)
- cnId
 - TRDP_FUNCTION_INFO_T, [34](#)
- confVehCnt
 - GNU_PACKED, [19](#), [20](#)
- confVehList
 - GNU_PACKED, [21](#)
- cstCnt
 - GNU_PACKED, [21](#)
- cstId
 - TRDP_CONSIST_INFO_T, [27](#)
- cstList
 - GNU_PACKED, [21](#)
- cstNetProp
 - GNU_PACKED, [23](#)
- cstOwner
 - TRDP_CONSIST_INFO_T, [27](#)
- cstUUID
 - GNU_PACKED, [21](#)
- cstVehNo
 - TRDP_FUNCTION_INFO_T, [33](#)
 - TRDP_VEHICLE_INFO_T, [60](#)
- datasetLength
 - GNU_PACKED, [23](#)
- destAddr
 - TRDP_PUB_STATISTICS_T, [52](#)
- deviceName
 - GNU_PACKED, [19](#)
- etbCnt
 - TRDP_CONSIST_INFO_T, [28](#)
- etbId
 - GNU_PACKED, [22](#)
 - TRDP_ETB_INFO_T, [32](#)
 - TRDP_FUNCTION_INFO_T, [33](#)
- etbInhibit
 - GNU_PACKED, [19](#)
- etbLength
 - GNU_PACKED, [19](#)
- ETBN_STATUS_COMID
 - iec61375-2-3.h, [70](#)
- etbShort
 - GNU_PACKED, [20](#)
- etbTopoCnt
 - GNU_PACKED, [21](#)
- fctCnt
 - TRDP_CONSIST_INFO_T, [28](#)
- fctId
 - TRDP_FUNCTION_INFO_T, [33](#)
- filterAddr
 - TRDP_SUBS_STATISTICS_T, [58](#)
- GNU_PACKED, [9](#)
 - confVehCnt, [19](#), [20](#)
 - confVehList, [21](#)
 - cstCnt, [21](#)
 - cstList, [21](#)
 - cstNetProp, [23](#)
 - cstUUID, [21](#)
 - datasetLength, [23](#)
 - deviceName, [19](#)
 - etbId, [22](#)
 - etbInhibit, [19](#)
 - etbLength, [19](#)
 - etbShort, [20](#)
 - etbTopoCnt, [21](#)
 - inhibit, [19](#)
 - isLead, [17](#)
 - leadDir, [17](#)
 - leadVehOfCst, [18](#)
 - lifesign, [19](#)
 - msgType, [23](#)
 - opCstCnt, [22](#)
 - opCstList, [23](#)
 - opCstNo, [22](#)
 - opTrnDirState, [20](#)
 - opTrnTopoCnt, [20](#)
 - opVehCnt, [23](#)
 - opVehList, [23](#)
 - opVehNo, [22](#)
 - ownOpCstNo, [18](#)
 - protocolVersion, [23](#)
 - reserved01, [17](#), [19](#)

- reserved02, [18](#), [20](#)
- reserved03, [18](#)
- reserved04, [18](#)
- reserved06, [18](#)
- safetyTrail, [19](#)
- sleepReqCnt, [20](#)
- trnCstNo, [18](#)
- trnDirState, [20](#)
- trnId, [22](#)
- trnNetDir, [21](#)
- trnOperator, [22](#)
- trnTopoCnt, [22](#)
- trnVehNo, [17](#)
- vehId, [22](#)
- vehOrient, [17](#)
- version, [17](#)
- iec61375-2-3.h, [67](#)
 - ETBN_STATUS_COMID, [70](#)
 - TTDB_NET_DIR_REQ_COMID, [70](#)
 - TTDB_OP_DIR_INFO_COMID, [70](#)
 - TTDB_STAT_CST_REQ_COMID, [71](#)
 - TTDB_TRN_DIR_REQ_COMID, [71](#)
- inhibit
 - GNU_PACKED, [19](#)
- INITFCS
 - vos_utils.h, [201](#)
- isLead
 - GNU_PACKED, [17](#)
- leadDir
 - GNU_PACKED, [17](#)
- leadVehOfCst
 - GNU_PACKED, [18](#)
- len
 - TRDP_PROP_T, [51](#)
- lifesign
 - GNU_PACKED, [19](#)
- msgType
 - GNU_PACKED, [23](#)
 - TRDP_MD_INFO_T, [40](#)
 - TRDP_PD_INFO_T, [47](#)
- opCstCnt
 - GNU_PACKED, [22](#)
- opCstList
 - GNU_PACKED, [23](#)
- opCstNo
 - GNU_PACKED, [22](#)
- opTrnDirState
 - GNU_PACKED, [20](#)
- opTrnTopoCnt
 - GNU_PACKED, [20](#)
- opVehCnt
 - GNU_PACKED, [23](#)
- opVehList
 - GNU_PACKED, [23](#)
- opVehNo
 - GNU_PACKED, [22](#)
- ownOpCstNo
 - GNU_PACKED, [18](#)
- protocolVersion
 - GNU_PACKED, [23](#)
- qos
 - VOS_SOCK_OPT_T, [63](#)
- reserved01
 - GNU_PACKED, [17](#), [19](#)
- reserved02
 - GNU_PACKED, [18](#), [20](#)
- reserved03
 - GNU_PACKED, [18](#)
- reserved04
 - GNU_PACKED, [18](#)
- reserved06
 - GNU_PACKED, [18](#)
- safetyTrail
 - GNU_PACKED, [19](#)
- sleepReqCnt
 - GNU_PACKED, [20](#)
- tau_xml.h
 - TRDP_DBG_CAT, [100](#)
 - TRDP_DBG_DBG, [100](#)
 - TRDP_DBG_DEFAULT, [100](#)
 - TRDP_DBG_ERR, [100](#)
 - TRDP_DBG_INFO, [100](#)
 - TRDP_DBG_LOC, [100](#)
 - TRDP_DBG_OFF, [100](#)
 - TRDP_DBG_TIME, [100](#)
 - TRDP_DBG_WARN, [100](#)
 - TRDP_EXCHG_SINK, [101](#)
 - TRDP_EXCHG_SOURCE, [101](#)
 - TRDP_EXCHG_SOURCESINK, [101](#)
 - TRDP_EXCHG_UNSET, [101](#)
- tau_addr2Uri
 - tau_dnr.h, [79](#)
- tau_calcDatasetSize
 - tau_marshall.h, [83](#)
- tau_calcDatasetSizeByComId
 - tau_marshall.h, [83](#)
- tau_ctrl.h, [72](#)
 - tau_getEcspStat, [73](#)
 - tau_initEcspCtrl, [73](#)
 - tau_requestEcspConfirm, [74](#)

- tau_setEcspCtrl, 74
- tau_terminateEcspCtrl, 74
- tau_ctrl_types.h, 76
- tau_deInitDnr
 - tau_dnr.h, 79
- tau_deInitTTI
 - tau_tti.h, 89
- tau_dnr.h, 78
 - tau_addr2Uri, 79
 - tau_deInitDnr, 79
 - tau_DNRstatus, 80
 - tau_getOwnAddr, 80
 - tau_getOwnIds, 80
 - tau_initDnr, 81
 - tau_uri2Addr, 81
- tau_DNRstatus
 - tau_dnr.h, 80
- tau_freeTelegrams
 - tau_xml.h, 101
- tau_freeXmlDatasetConfig
 - tau_xml.h, 101
- tau_freeXmlDoc
 - tau_xml.h, 101
- tau_getCstFctCnt
 - tau_tti.h, 89
- tau_getCstFctInfo
 - tau_tti.h, 89
- tau_getCstInfo
 - tau_tti.h, 90
- tau_getCstVehCnt
 - tau_tti.h, 90
- tau_getEcspStat
 - tau_ctrl.h, 73
- tau_getOpTrDirectory
 - tau_tti.h, 90
- tau_getOwnAddr
 - tau_dnr.h, 80
- tau_getOwnIds
 - tau_dnr.h, 80
- tau_getStaticCstInfo
 - tau_tti.h, 91
- tau_getTrDirectory
 - tau_tti.h, 91
- tau_getTrnCstCnt
 - tau_tti.h, 91
- tau_getTrnVehCnt
 - tau_tti.h, 92
- tau_getTTI
 - tau_tti.h, 92
- tau_getVehInfo
 - tau_tti.h, 92
- tau_getVehOrient
 - tau_tti.h, 93
- tau_initDnr
 - tau_dnr.h, 81
- tau_initEcspCtrl
 - tau_ctrl.h, 73
- tau_initMarshall
 - tau_marshall.h, 84
- tau_initTTIaccess
 - tau_tti.h, 93
- tau_marshall
 - tau_marshall.h, 84
- tau_marshall.h, 82
 - tau_calcDatasetSize, 83
 - tau_calcDatasetSizeByComId, 83
 - tau_initMarshall, 84
 - tau_marshall, 84
 - tau_marshallDs, 85
 - tau_unmarshall, 85
 - tau_unmarshallDs, 86
- tau_marshallDs
 - tau_marshall.h, 85
- tau_prepareXmlDoc
 - tau_xml.h, 101
- tau_readXmlDatasetConfig
 - tau_xml.h, 102
- tau_readXmlDeviceConfig
 - tau_xml.h, 102
- tau_readXmlInterfaceConfig
 - tau_xml.h, 103
- tau_requestEcspConfirm
 - tau_ctrl.h, 74
- tau_setEcspCtrl
 - tau_ctrl.h, 74
- tau_terminateEcspCtrl
 - tau_ctrl.h, 74
- tau_tti.h, 87
 - tau_deInitTTI, 89
 - tau_getCstFctCnt, 89
 - tau_getCstFctInfo, 89
 - tau_getCstInfo, 90
 - tau_getCstVehCnt, 90
 - tau_getOpTrDirectory, 90
 - tau_getStaticCstInfo, 91
 - tau_getTrDirectory, 91
 - tau_getTrnCstCnt, 91
 - tau_getTrnVehCnt, 92
 - tau_getTTI, 92
 - tau_getVehInfo, 92
 - tau_getVehOrient, 93
 - tau_initTTIaccess, 93
- tau_tti_types.h, 95
- tau_unmarshall
 - tau_marshall.h, 85
- tau_unmarshallDs
 - tau_marshall.h, 86
- tau_uri2Addr

- tau_dnr.h, 81
- tau_xml.h, 98
 - tau_freeTelegrams, 101
 - tau_freeXmlDatasetConfig, 101
 - tau_freeXmlDoc, 101
 - tau_prepareXmlDoc, 101
 - tau_readXmlDatasetConfig, 102
 - tau_readXmlDeviceConfig, 102
 - tau_readXmlInterfaceConfig, 103
 - TRDP_DBG_OPTION_T, 100
 - TRDP_EXCHG_OPTION_T, 100
- timeout
 - TRDP_SUBS_STATISTICS_T, 58
- tlc_closeSession
 - trdp_if_light.h, 108
- tlc_configSession
 - trdp_if_light.h, 108
- tlc_freeBuf
 - trdp_if_light.h, 109
- tlc_getInterval
 - trdp_if_light.h, 109
- tlc_getJoinStatistics
 - trdp_if_light.h, 110
- tlc_getOwnIpAddress
 - trdp_if_light.h, 110
- tlc_getPubStatistics
 - trdp_if_light.h, 110
- tlc_getRedStatistics
 - trdp_if_light.h, 111
- tlc_getStatistics
 - trdp_if_light.h, 111
- tlc_getSubsStatistics
 - trdp_if_light.h, 111
- tlc_getTcpListStatistics
 - trdp_if_light.h, 112
- tlc_getUdpListStatistics
 - trdp_if_light.h, 112
- tlc_getVersion
 - trdp_if_light.h, 113
- tlc_getVersionString
 - trdp_if_light.h, 113
- tlc_init
 - trdp_if_light.h, 113
- tlc_openSession
 - trdp_if_light.h, 113
- tlc_process
 - trdp_if_light.h, 114
- tlc_reinitSession
 - trdp_if_light.h, 114
- tlc_resetStatistics
 - trdp_if_light.h, 115
- tlc_setETBTopoCount
 - trdp_if_light.h, 115
- tlc_setOpTrainTopoCount
 - trdp_if_light.h, 115
- tlc_terminate
 - trdp_if_light.h, 115
- tlm_abortSession
 - trdp_if_light.h, 116
- tlm_addListener
 - trdp_if_light.h, 116
- tlm_confirm
 - trdp_if_light.h, 117
- tlm_delListener
 - trdp_if_light.h, 117
- tlm_notify
 - trdp_if_light.h, 117
- tlm_readdListener
 - trdp_if_light.h, 118
- tlm_reply
 - trdp_if_light.h, 119
- tlm_replyErr
 - trdp_if_light.h, 119
- tlm_replyQuery
 - trdp_if_light.h, 120
- tlm_request
 - trdp_if_light.h, 120
- tlp_get
 - trdp_if_light.h, 121
- tlp_getRedundant
 - trdp_if_light.h, 122
- tlp_publish
 - trdp_if_light.h, 122
- tlp_put
 - trdp_if_light.h, 123
- tlp_republish
 - trdp_if_light.h, 123
- tlp_request
 - trdp_if_light.h, 124
- tlp_resubscribe
 - trdp_if_light.h, 125
- tlp_setRedundant
 - trdp_if_light.h, 125
- tlp_subscribe
 - trdp_if_light.h, 125
- tlp_unpublish
 - trdp_if_light.h, 126
- tlp_unsubscribe
 - trdp_if_light.h, 126
- toBehav
 - TRDP_SUBS_STATISTICS_T, 59
- TRDP_APP_CONFIRMTO_ERR
 - trdp_types.h, 142
- TRDP_APP_REPLYTO_ERR
 - trdp_types.h, 142
- TRDP_APP_TIMEOUT_ERR
 - trdp_types.h, 142
- TRDP_BITSET8

- trdp_types.h, [140](#)
- TRDP_BLOCK_ERR
 - trdp_types.h, [141](#)
- TRDP_CHAR8
 - trdp_types.h, [140](#)
- TRDP_COMID_ERR
 - trdp_types.h, [142](#)
- TRDP_CONFIRMTO_ERR
 - trdp_types.h, [142](#)
- TRDP_CRC_ERR
 - trdp_types.h, [141](#)
- TRDP_DBG_CAT
 - tau_xml.h, [100](#)
- TRDP_DBG_DBG
 - tau_xml.h, [100](#)
- TRDP_DBG_DEFAULT
 - tau_xml.h, [100](#)
- TRDP_DBG_ERR
 - tau_xml.h, [100](#)
- TRDP_DBG_INFO
 - tau_xml.h, [100](#)
- TRDP_DBG_LOC
 - tau_xml.h, [100](#)
- TRDP_DBG_OFF
 - tau_xml.h, [100](#)
- TRDP_DBG_TIME
 - tau_xml.h, [100](#)
- TRDP_DBG_WARN
 - tau_xml.h, [100](#)
- TRDP_EXCHG_SINK
 - tau_xml.h, [101](#)
- TRDP_EXCHG_SOURCE
 - tau_xml.h, [101](#)
- TRDP_EXCHG_SOURCESINK
 - tau_xml.h, [101](#)
- TRDP_EXCHG_UNSET
 - tau_xml.h, [101](#)
- TRDP_FLAGS_CALLBACK
 - trdp_types.h, [142](#)
- TRDP_FLAGS_DEFAULT
 - trdp_types.h, [142](#)
- TRDP_FLAGS_FORCE_CB
 - trdp_types.h, [142](#)
- TRDP_FLAGS_MARSHALL
 - trdp_types.h, [142](#)
- TRDP_FLAGS_NONE
 - trdp_types.h, [142](#)
- TRDP_FLAGS_TCP
 - trdp_types.h, [142](#)
- TRDP_INIT_ERR
 - trdp_types.h, [141](#)
- TRDP_INT16
 - trdp_types.h, [140](#)
- TRDP_INT32
 - trdp_types.h, [140](#)
- TRDP_INT64
 - trdp_types.h, [141](#)
- TRDP_INT8
 - trdp_types.h, [140](#)
- TRDP_INTEGRATION_ERR
 - trdp_types.h, [141](#)
- TRDP_INUSE_ERR
 - trdp_types.h, [142](#)
- TRDP_IO_ERR
 - trdp_types.h, [141](#)
- TRDP_MARSHALLING_ERR
 - trdp_types.h, [142](#)
- TRDP_MEM_ERR
 - trdp_types.h, [141](#)
- TRDP_MSG_MC
 - trdp_proto.h, [132](#)
- TRDP_MSG_ME
 - trdp_proto.h, [132](#)
- TRDP_MSG_MN
 - trdp_proto.h, [132](#)
- TRDP_MSG_MP
 - trdp_proto.h, [132](#)
- TRDP_MSG_MQ
 - trdp_proto.h, [132](#)
- TRDP_MSG_MR
 - trdp_proto.h, [132](#)
- TRDP_MSG_PD
 - trdp_proto.h, [132](#)
- TRDP_MSG_PE
 - trdp_proto.h, [132](#)
- TRDP_MSG_PP
 - trdp_proto.h, [132](#)
- TRDP_MSG_PR
 - trdp_proto.h, [132](#)
- TRDP_Mutex_ERR
 - trdp_types.h, [141](#)
- TRDP_NO_ERR
 - trdp_types.h, [141](#)
- TRDP_NOCONN_ERR
 - trdp_types.h, [141](#)
- TRDP_NODATA_ERR
 - trdp_types.h, [141](#)
- TRDP_NOINIT_ERR
 - trdp_types.h, [141](#)
- TRDP_NOLIST_ERR
 - trdp_types.h, [141](#)
- TRDP_NOPUB_ERR
 - trdp_types.h, [141](#)
- TRDP_NOSESSION_ERR
 - trdp_types.h, [141](#)
- TRDP_NOSUB_ERR
 - trdp_types.h, [141](#)
- TRDP_OPTION_BLOCK

- trdp_types.h, [142](#)
- TRDP_OPTION_NO_MC_LOOP_BACK
 - trdp_types.h, [142](#)
- TRDP_OPTION_NO_REUSE_ADDR
 - trdp_types.h, [142](#)
- TRDP_OPTION_NO_UDP_CHK
 - trdp_types.h, [142](#)
- TRDP_OPTION_TRAFFIC_SHAPING
 - trdp_types.h, [142](#)
- TRDP_PACKET_ERR
 - trdp_types.h, [142](#)
- TRDP_PARAM_ERR
 - trdp_types.h, [141](#)
- trdp_proto.h
 - TRDP_MSG_MC, [132](#)
 - TRDP_MSG_ME, [132](#)
 - TRDP_MSG_MN, [132](#)
 - TRDP_MSG_MP, [132](#)
 - TRDP_MSG_MQ, [132](#)
 - TRDP_MSG_MR, [132](#)
 - TRDP_MSG_PD, [132](#)
 - TRDP_MSG_PE, [132](#)
 - TRDP_MSG_PP, [132](#)
 - TRDP_MSG_PR, [132](#)
- TRDP_QUEUE_ERR
 - trdp_types.h, [141](#)
- TRDP_QUEUE_FULL_ERR
 - trdp_types.h, [141](#)
- TRDP_REAL32
 - trdp_types.h, [141](#)
- TRDP_REAL64
 - trdp_types.h, [141](#)
- TRDP_RED_FOLLOWER
 - trdp_types.h, [143](#)
- TRDP_RED_LEADER
 - trdp_types.h, [143](#)
- TRDP_REPLYTO_ERR
 - trdp_types.h, [142](#)
- TRDP_REQCONFIRMTO_ERR
 - trdp_types.h, [142](#)
- TRDP_SEMA_ERR
 - trdp_types.h, [141](#)
- TRDP_SESSION_ABORT_ERR
 - trdp_types.h, [141](#)
- TRDP SOCK_ERR
 - trdp_types.h, [141](#)
- TRDP_STATE_ERR
 - trdp_types.h, [142](#)
- TRDP_THREAD_ERR
 - trdp_types.h, [141](#)
- TRDP_TIMEDATE32
 - trdp_types.h, [141](#)
- TRDP_TIMEDATE48
 - trdp_types.h, [141](#)
- TRDP_TIMEDATE64
 - trdp_types.h, [141](#)
- TRDP_TIMEOUT_ERR
 - trdp_types.h, [141](#)
- TRDP_TO_DEFAULT
 - trdp_types.h, [143](#)
- TRDP_TO_KEEP_LAST_VALUE
 - trdp_types.h, [143](#)
- TRDP_TO_SET_TO_ZERO
 - trdp_types.h, [143](#)
- TRDP_TOPO_ERR
 - trdp_types.h, [142](#)
- TRDP_TYPE_MAX
 - trdp_types.h, [141](#)
- trdp_types.h
 - TRDP_APP_CONFIRMTO_ERR, [142](#)
 - TRDP_APP_REPLYTO_ERR, [142](#)
 - TRDP_APP_TIMEOUT_ERR, [142](#)
 - TRDP_BITSET8, [140](#)
 - TRDP_BLOCK_ERR, [141](#)
 - TRDP_CHAR8, [140](#)
 - TRDP_COMID_ERR, [142](#)
 - TRDP_CONFIRMTO_ERR, [142](#)
 - TRDP_CRC_ERR, [141](#)
 - TRDP_FLAGS_CALLBACK, [142](#)
 - TRDP_FLAGS_DEFAULT, [142](#)
 - TRDP_FLAGS_FORCE_CB, [142](#)
 - TRDP_FLAGS_MARSHALL, [142](#)
 - TRDP_FLAGS_NONE, [142](#)
 - TRDP_FLAGS_TCP, [142](#)
 - TRDP_INIT_ERR, [141](#)
 - TRDP_INT16, [140](#)
 - TRDP_INT32, [140](#)
 - TRDP_INT64, [141](#)
 - TRDP_INT8, [140](#)
 - TRDP_INTEGRATION_ERR, [141](#)
 - TRDP_INUSE_ERR, [142](#)
 - TRDP_IO_ERR, [141](#)
 - TRDP_MARSHALLING_ERR, [142](#)
 - TRDP_MEM_ERR, [141](#)
 - TRDP_MUTEX_ERR, [141](#)
 - TRDP_NO_ERR, [141](#)
 - TRDP_NOCONN_ERR, [141](#)
 - TRDP_NODATA_ERR, [141](#)
 - TRDP_NOINIT_ERR, [141](#)
 - TRDP_NOLIST_ERR, [141](#)
 - TRDP_NOPUB_ERR, [141](#)
 - TRDP_NOSESSION_ERR, [141](#)
 - TRDP_NOSUB_ERR, [141](#)
 - TRDP_OPTION_BLOCK, [142](#)
 - TRDP_OPTION_NO_MC_LOOP_BACK, [142](#)
 - TRDP_OPTION_NO_REUSE_ADDR, [142](#)
 - TRDP_OPTION_NO_UDP_CHK, [142](#)

- TRDP_OPTION_TRAFFIC_SHAPING, 142
- TRDP_PACKET_ERR, 142
- TRDP_PARAM_ERR, 141
- TRDP_QUEUE_ERR, 141
- TRDP_QUEUE_FULL_ERR, 141
- TRDP_REAL32, 141
- TRDP_REAL64, 141
- TRDP_RED_FOLLOWER, 143
- TRDP_RED_LEADER, 143
- TRDP_REPLYTO_ERR, 142
- TRDP_REQCONFIRMTO_ERR, 142
- TRDP_SEMA_ERR, 141
- TRDP_SESSION_ABORT_ERR, 141
- TRDP SOCK_ERR, 141
- TRDP_STATE_ERR, 142
- TRDP_THREAD_ERR, 141
- TRDP_TIMEDATE32, 141
- TRDP_TIMEDATE48, 141
- TRDP_TIMEDATE64, 141
- TRDP_TIMEOUT_ERR, 141
- TRDP_TO_DEFAULT, 143
- TRDP_TO_KEEP_LAST_VALUE, 143
- TRDP_TO_SET_TO_ZERO, 143
- TRDP_TOPO_ERR, 142
- TRDP_TYPE_MAX, 141
- TRDP_UINT16, 141
- TRDP_UINT32, 141
- TRDP_UINT64, 141
- TRDP_UINT8, 141
- TRDP_UNKNOWN_ERR, 142
- TRDP_UNRESOLVED_ERR, 142
- TRDP_UTF16, 140
- TRDP_WIRE_ERR, 142
- TRDP_XML_PARSER_ERR, 142
- TRDP_UINT16
 - trdp_types.h, 141
- TRDP_UINT32
 - trdp_types.h, 141
- TRDP_UINT64
 - trdp_types.h, 141
- TRDP_UINT8
 - trdp_types.h, 141
- TRDP_UNKNOWN_ERR
 - trdp_types.h, 142
- TRDP_UNRESOLVED_ERR
 - trdp_types.h, 142
- TRDP_UTF16
 - trdp_types.h, 140
- TRDP_WIRE_ERR
 - trdp_types.h, 142
- TRDP_XML_PARSER_ERR
 - trdp_types.h, 142
- TRDP_CLTR_CST_INFO_T, 24
 - cltrCstNo, 24
- TRDP_COMID_DSID_MAP_T, 25
- TRDP_CONSIST_INFO_T, 26
 - cltrCstCnt, 28
 - cstId, 27
 - cstOwner, 27
 - etbCnt, 28
 - fctCnt, 28
 - vehCnt, 28
- TRDP_DATA_TYPE_T
 - trdp_types.h, 140
- TRDP_DATASET, 29
- TRDP_DATASET_ELEMENT_T, 30
 - type, 30
- TRDP_DBG_CONFIG_T, 31
- TRDP_DBG_OPTION_T
 - tau_xml.h, 100
- TRDP_DEST_URI_SIZE
 - trdp_proto.h, 130
- TRDP_ERR_T
 - trdp_types.h, 141
- TRDP_ETB_INFO_T, 32
 - cnCnt, 32
 - etbId, 32
- TRDP_ETBCTRL_COMID
 - trdp_proto.h, 130
- TRDP_ETBCTRL_DSID
 - trdp_proto.h, 131
- TRDP_EXCHG_OPTION_T
 - tau_xml.h, 100
- TRDP_FLAGS_T
 - trdp_types.h, 142
- TRDP_FUNCTION_INFO_T, 33
 - cnId, 34
 - cstVehNo, 33
 - etbId, 33
 - fctId, 33
- trdp_if_light.h, 104
 - tlc_closeSession, 108
 - tlc_configSession, 108
 - tlc_freeBuf, 109
 - tlc_getInterval, 109
 - tlc_getJoinStatistics, 110
 - tlc_getOwnIpAddress, 110
 - tlc_getPubStatistics, 110
 - tlc_getRedStatistics, 111
 - tlc_getStatistics, 111
 - tlc_getSubsStatistics, 111
 - tlc_getTcpListStatistics, 112
 - tlc_getUdpListStatistics, 112
 - tlc_getVersion, 113
 - tlc_getVersionString, 113
 - tlc_init, 113
 - tlc_openSession, 113
 - tlc_process, 114

- tlc_reinitSession, 114
- tlc_resetStatistics, 115
- tlc_setETBTopoCount, 115
- tlc_setOpTrainTopoCount, 115
- tlc_terminate, 115
- tlm_abortSession, 116
- tlm_addListener, 116
- tlm_confirm, 117
- tlm_delListener, 117
- tlm_notify, 117
- tlm_readdListener, 118
- tlm_reply, 119
- tlm_replyErr, 119
- tlm_replyQuery, 120
- tlm_request, 120
- tlp_get, 121
- tlp_getRedundant, 122
- tlp_publish, 122
- tlp_put, 123
- tlp_republish, 123
- tlp_request, 124
- tlp_resubscribe, 125
- tlp_setRedundant, 125
- tlp_subscribe, 125
- tlp_unpublish, 126
- tlp_unsubscribe, 126
- TRDP_IP_ADDR_T
 - trdp_types.h, 138
- TRDP_LIST_STATISTICS_T, 35
- TRDP_MARSHALL_CONFIG_T, 36
- TRDP_MARSHALL_T
 - trdp_types.h, 138
- TRDP_MAX_FILE_NAME_LEN
 - trdp_proto.h, 131
- TRDP_MAX_LABEL_LEN
 - trdp_proto.h, 131
- TRDP_MAX_URI_HOST_LEN
 - trdp_proto.h, 131
- TRDP_MAX_URI_LEN
 - trdp_proto.h, 131
- TRDP_MAX_URI_USER_LEN
 - trdp_proto.h, 131
- TRDP_MD_CALLBACK_T
 - trdp_types.h, 139
- TRDP_MD_CONFIG_T, 37
- TRDP_MD_INFO_T, 39
 - msgType, 40
- TRDP_MD_STATISTICS_T, 41
- TRDP_MEM_CONFIG_T, 43
- TRDP_MEM_STATISTICS_T, 44
- TRDP_MSG_T
 - trdp_proto.h, 131
- TRDP_OPTION_T
 - trdp_types.h, 142
- TRDP_PD_CALLBACK_T
 - trdp_types.h, 139
- TRDP_PD_CONFIG_T, 45
- TRDP_PD_INFO_T, 46
 - msgType, 47
- TRDP_PD_STATISTICS_T, 48
- TRDP_PRINT_DBG_T
 - trdp_types.h, 139
- TRDP_PROCESS_CONFIG_T, 50
- TRDP_PROP_T, 51
 - len, 51
- trdp_proto.h, 128
 - TRDP_DEST_URI_SIZE, 130
 - TRDP_ETBCTRL_COMID, 130
 - TRDP_ETBCTRL_DSID, 131
 - TRDP_MAX_FILE_NAME_LEN, 131
 - TRDP_MAX_LABEL_LEN, 131
 - TRDP_MAX_URI_HOST_LEN, 131
 - TRDP_MAX_URI_LEN, 131
 - TRDP_MAX_URI_USER_LEN, 131
 - TRDP_MSG_T, 131
- TRDP_PUB_STATISTICS_T, 52
 - destAddr, 52
- TRDP_RED_STATE_T
 - trdp_types.h, 142
- TRDP_RED_STATISTICS_T, 53
- TRDP_REPLY_STATUS_T
 - trdp_types.h, 143
- TRDP_SDT_PAR_T, 54
- TRDP_SEND_PARAM_T, 55
- TRDP_STATISTICS_T, 56
- TRDP_SUBS_STATISTICS_T, 58
 - filterAddr, 58
 - timeout, 58
 - toBehav, 59
- TRDP_TIME_T
 - trdp_types.h, 140
- TRDP_TO_BEHAVIOR_T
 - trdp_types.h, 143
- trdp_types.h, 133
 - TRDP_DATA_TYPE_T, 140
 - TRDP_ERR_T, 141
 - TRDP_FLAGS_T, 142
 - TRDP_IP_ADDR_T, 138
 - TRDP_MARSHALL_T, 138
 - TRDP_MD_CALLBACK_T, 139
 - TRDP_OPTION_T, 142
 - TRDP_PD_CALLBACK_T, 139
 - TRDP_PRINT_DBG_T, 139
 - TRDP_RED_STATE_T, 142
 - TRDP_REPLY_STATUS_T, 143
 - TRDP_TIME_T, 140
 - TRDP_TO_BEHAVIOR_T, 143
 - TRDP_UNMARSHALL_T, 140

- TRDP_UNMARSHALL_T
 - trdp_types.h, 140
- TRDP_VEHICLE_INFO_T, 60
 - cstVehNo, 60
 - vehId, 60
- TRDP_XML_DOC_HANDLE_T, 62
- trnCstNo
 - GNU_PACKED, 18
- trnDirState
 - GNU_PACKED, 20
- trnId
 - GNU_PACKED, 22
- trnNetDir
 - GNU_PACKED, 21
- trnOperator
 - GNU_PACKED, 22
- trnTopoCnt
 - GNU_PACKED, 22
- trnVehNo
 - GNU_PACKED, 17
- TTDB_NET_DIR_REQ_COMID
 - iec61375-2-3.h, 70
- TTDB_OP_DIR_INFO_COMID
 - iec61375-2-3.h, 70
- TTDB_STAT_CST_REQ_COMID
 - iec61375-2-3.h, 71
- TTDB_TRN_DIR_REQ_COMID
 - iec61375-2-3.h, 71
- tv_usec
 - VOS_TIME_T, 64
- type
 - TRDP_DATASET_ELEMENT_T, 30
- vehCnt
 - TRDP_CONSIST_INFO_T, 28
- vehId
 - GNU_PACKED, 22
 - TRDP_VEHICLE_INFO_T, 60
- vehOrient
 - GNU_PACKED, 17
- version
 - GNU_PACKED, 17
- VOS_BLOCK_ERR
 - vos_types.h, 194
- VOS_INIT_ERR
 - vos_types.h, 194
- VOS_INTEGRATION_ERR
 - vos_types.h, 194
- VOS_IO_ERR
 - vos_types.h, 194
- VOS_LOG_DBG
 - vos_types.h, 194
- VOS_LOG_ERROR
 - vos_types.h, 194
- VOS_LOG_INFO
 - vos_types.h, 194
- VOS_LOG_WARNING
 - vos_types.h, 194
- VOS_MEM_ERR
 - vos_types.h, 194
- VOS_MUTEX_ERR
 - vos_types.h, 194
- VOS_NO_ERR
 - vos_types.h, 194
- VOS_NOCONN_ERR
 - vos_types.h, 194
- VOS_NODATA_ERR
 - vos_types.h, 194
- VOS_NOINIT_ERR
 - vos_types.h, 194
- VOS_PARAM_ERR
 - vos_types.h, 194
- VOS_QUEUE_ERR
 - vos_types.h, 194
- VOS_QUEUE_FULL_ERR
 - vos_types.h, 194
- VOS_SEMA_ERR
 - vos_types.h, 194
- VOS SOCK_ERR
 - vos_types.h, 194
- VOS_THREAD_ERR
 - vos_types.h, 194
- VOS_TIMEOUT_ERR
 - vos_types.h, 194
- vos_types.h
 - VOS_BLOCK_ERR, 194
 - VOS_INIT_ERR, 194
 - VOS_INTEGRATION_ERR, 194
 - VOS_IO_ERR, 194
 - VOS_LOG_DBG, 194
 - VOS_LOG_ERROR, 194
 - VOS_LOG_INFO, 194
 - VOS_LOG_WARNING, 194
 - VOS_MEM_ERR, 194
 - VOS_MUTEX_ERR, 194
 - VOS_NO_ERR, 194
 - VOS_NOCONN_ERR, 194
 - VOS_NODATA_ERR, 194
 - VOS_NOINIT_ERR, 194
 - VOS_PARAM_ERR, 194
 - VOS_QUEUE_ERR, 194
 - VOS_QUEUE_FULL_ERR, 194
 - VOS_SEMA_ERR, 194
 - VOS SOCK_ERR, 194
 - VOS_THREAD_ERR, 194
 - VOS_TIMEOUT_ERR, 194
 - VOS_UNKNOWN_ERR, 194
- VOS_UNKNOWN_ERR

- vos_types.h, 194
- vos_addTime
 - vos_thread.h, 183
- vos_bsearch
 - vos_mem.c, 146
 - vos_mem.h, 156
- vos_clearTime
 - vos_thread.h, 183
- vos_cmpTime
 - vos_thread.h, 183
- vos_crc32
 - vos_utils.c, 196
 - vos_utils.h, 201
- vos_cyclicThread
 - vos_thread.h, 184
- vos_determineBindAddr
 - vos_sock.h, 170
- vos_divTime
 - vos_thread.h, 184
- vos_dottedIP
 - vos_sock.h, 170
- VOS_ERR_T
 - vos_types.h, 193
- vos_getInterfaces
 - vos_sock.h, 170
- vos_getTime
 - vos_thread.h, 184
- vos_getTimeStamp
 - vos_thread.h, 184
- vos_getUuid
 - vos_thread.h, 185
- vos_getVersion
 - vos_utils.c, 196
 - vos_utils.h, 202
- vos_getVersionString
 - vos_utils.c, 196
 - vos_utils.h, 202
- vos_htonl
 - vos_sock.h, 170
- vos_htons
 - vos_sock.h, 171
- vos_init
 - vos_utils.c, 197
 - vos_utils.h, 202
- vos_initRuntimeConsts
 - vos_utils.c, 197
- vos_ipDotted
 - vos_sock.h, 171
- vos_isMulticast
 - vos_sock.h, 171
- VOS_LOG_T
 - vos_types.h, 194
- VOS_MAX_ERR_STR_SIZE
 - vos_utils.h, 201
- VOS_MAX_FRMT_SIZE
 - vos_utils.h, 201
- VOS_MAX_PRNT_STR_SIZE
 - vos_utils.h, 201
- VOS_MAX_SOCKET_CNT
 - vos_sock.h, 169
- vos_mem.c, 144
 - vos_bsearch, 146
 - vos_memAlloc, 146
 - vos_memCount, 146
 - vos_memDelete, 147
 - vos_memFree, 147
 - vos_memInit, 147
 - vos_qsort, 148
 - vos_queueCreate, 148
 - vos_queueDestroy, 149
 - vos_queueReceive, 150
 - vos_queueSend, 150
 - vos_strncat, 151
 - vos_strncpy, 151
 - vos_strnicmp, 151
- vos_mem.h, 153
 - vos_bsearch, 156
 - VOS_MEM_BLOCKSIZE, 155
 - VOS_MEM_PREALLOCATE, 155
 - vos_memAlloc, 156
 - vos_memCount, 156
 - vos_memDelete, 157
 - vos_memFree, 157
 - vos_memInit, 158
 - vos_qsort, 158
 - vos_queueCreate, 159
 - vos_queueDestroy, 159
 - vos_queueReceive, 160
 - vos_queueSend, 161
 - vos_strncat, 162
 - vos_strncpy, 162
 - vos_strnicmp, 162
- VOS_MEM_BLOCKSIZE
 - vos_mem.h, 155
- VOS_MEM_PREALLOCATE
 - vos_mem.h, 155
- vos_memAlloc
 - vos_mem.c, 146
 - vos_mem.h, 156
- vos_memCount
 - vos_mem.c, 146
 - vos_mem.h, 156
- vos_memDelete
 - vos_mem.c, 147
 - vos_mem.h, 157
- vos_memFree
 - vos_mem.c, 147
 - vos_mem.h, 157

- vos_memInit
 - vos_mem.c, 147
 - vos_mem.h, 158
- vos_mulTime
 - vos_thread.h, 185
- vos_mutexCreate
 - vos_thread.h, 185
- vos_mutexDelete
 - vos_thread.h, 185
- vos_mutexLock
 - vos_thread.h, 186
- vos_mutexTryLock
 - vos_thread.h, 186
- vos_mutexUnlock
 - vos_thread.h, 186
- vos_netIfUp
 - vos_sock.h, 171
- vos_ntohl
 - vos_sock.h, 172
- vos_ntohs
 - vos_sock.h, 172
- VOS_PRINT_DBG_T
 - vos_types.h, 193
- vos_qsort
 - vos_mem.c, 148
 - vos_mem.h, 158
- vos_queueCreate
 - vos_mem.c, 148
 - vos_mem.h, 159
- vos_queueDestroy
 - vos_mem.c, 149
 - vos_mem.h, 159
- vos_queueReceive
 - vos_mem.c, 150
 - vos_mem.h, 160
- vos_queueSend
 - vos_mem.c, 150
 - vos_mem.h, 161
- vos_select
 - vos_sock.h, 172
- vos_semaCreate
 - vos_thread.h, 186
- vos_semaDelete
 - vos_thread.h, 187
- vos_semaGive
 - vos_thread.h, 187
- vos_semaTake
 - vos_thread.h, 187
- vos_shared_mem.h, 164
 - vos_sharedClose, 165
 - vos_sharedOpen, 165
- vos_sharedClose
 - vos_shared_mem.h, 165
- vos_sharedOpen
 - vos_shared_mem.h, 165
- vos_sock.h, 166
 - vos_determineBindAddr, 170
 - vos_dottedIP, 170
 - vos_getInterfaces, 170
 - vos_htonl, 170
 - vos_htons, 171
 - vos_ipDotted, 171
 - vos_isMulticast, 171
 - VOS_MAX_SOCKET_CNT, 169
 - vos_netIfUp, 171
 - vos_ntohl, 172
 - vos_ntohs, 172
 - vos_select, 172
 - vos_sockAccept, 172
 - vos_sockBind, 173
 - vos_sockClose, 173
 - vos_sockConnect, 173
 - vos_sockGetMAC, 174
 - vos_sockInit, 174
 - vos_sockJoinMC, 174
 - vos_sockLeaveMC, 175
 - vos_sockListen, 175
 - vos_sockOpenTCP, 175
 - vos_sockOpenUDP, 176
 - vos_sockReceiveTCP, 176
 - vos_sockReceiveUDP, 177
 - vos_sockSendTCP, 177
 - vos_sockSendUDP, 178
 - vos_sockSetMulticastIf, 178
 - vos_sockSetOptions, 178
 - vos_sockTerm, 179
 - VOS_TTL_MULTICAST, 169
- VOS_SOCKET_OPT_T, 63
 - qos, 63
- vos_sockAccept
 - vos_sock.h, 172
- vos_sockBind
 - vos_sock.h, 173
- vos_sockClose
 - vos_sock.h, 173
- vos_sockConnect
 - vos_sock.h, 173
- vos_sockGetMAC
 - vos_sock.h, 174
- vos_sockInit
 - vos_sock.h, 174
- vos_sockJoinMC
 - vos_sock.h, 174
- vos_sockLeaveMC
 - vos_sock.h, 175
- vos_sockListen
 - vos_sock.h, 175
- vos_sockOpenTCP

- vos_sock.h, 175
- vos_sockOpenUDP
 - vos_sock.h, 176
- vos_sockReceiveTCP
 - vos_sock.h, 176
- vos_sockReceiveUDP
 - vos_sock.h, 177
- vos_sockSendTCP
 - vos_sock.h, 177
- vos_sockSendUDP
 - vos_sock.h, 178
- vos_sockSetMulticastIf
 - vos_sock.h, 178
- vos_sockSetOptions
 - vos_sock.h, 178
- vos_sockTerm
 - vos_sock.h, 179
- vos_strncat
 - vos_mem.c, 151
 - vos_mem.h, 162
- vos_strncpy
 - vos_mem.c, 151
 - vos_mem.h, 162
- vos_strnicmp
 - vos_mem.c, 151
 - vos_mem.h, 162
- vos_subTime
 - vos_thread.h, 188
- vos_terminate
 - vos_utils.c, 197
 - vos_utils.h, 203
- vos_thread.h, 180
 - vos_addTime, 183
 - vos_clearTime, 183
 - vos_cmpTime, 183
 - vos_cyclicThread, 184
 - vos_divTime, 184
 - vos_getTime, 184
 - vos_getTimeStamp, 184
 - vos_getUuid, 185
 - vos_mulTime, 185
 - vos_mutexCreate, 185
 - vos_mutexDelete, 185
 - vos_mutexLock, 186
 - vos_mutexTryLock, 186
 - vos_mutexUnlock, 186
 - vos_semaCreate, 186
 - vos_semaDelete, 187
 - vos_semaGive, 187
 - vos_semaTake, 187
 - vos_subTime, 188
 - vos_threadCreate, 188
 - vos_threadDelay, 188
 - vos_threadInit, 189
 - vos_threadIsActive, 189
 - vos_threadTerm, 189
 - vos_threadTerminate, 189
- vos_threadCreate
 - vos_thread.h, 188
- vos_threadDelay
 - vos_thread.h, 188
- vos_threadInit
 - vos_thread.h, 189
- vos_threadIsActive
 - vos_thread.h, 189
- vos_threadTerm
 - vos_thread.h, 189
- vos_threadTerminate
 - vos_thread.h, 189
- VOS_TIME_T, 64
 - tv_usec, 64
- VOS_TTL_MULTICAST
 - vos_sock.h, 169
- vos_types.h, 191
 - VOS_ERR_T, 193
 - VOS_LOG_T, 194
 - VOS_PRINT_DBG_T, 193
- vos_utils.c, 195
 - vos_crc32, 196
 - vos_getVersion, 196
 - vos_getVersionString, 196
 - vos_init, 197
 - vos_initRuntimeConsts, 197
 - vos_terminate, 197
- vos_utils.h, 199
 - INITFCS, 201
 - vos_crc32, 201
 - vos_getVersion, 202
 - vos_getVersionString, 202
 - vos_init, 202
 - VOS_MAX_ERR_STR_SIZE, 201
 - VOS_MAX_FRMT_SIZE, 201
 - VOS_MAX_PRNT_STR_SIZE, 201
 - vos_terminate, 203
- VOS_VERSION_T, 65