MICROSAR Socket Adaptor

Technical Reference

Version 10.1.0

Authors	Alex Lunkenheimer, Marc Weber, Michael Dangelmaier, Philipp Christmann, Michael Seidenspinner
Status	Released

Document Information

History

Author	Date	Version	Remarks
Alex Lunkenheimer	2008-11-20	1.0	Creation of document
Alex Lunkenheimer	2009-10-05	2.0	Tool based configuration
Marc Weber	2012-01-16	2.1	Call-back for Ethernet State Manager and minor changes
Marc Weber	2012-07-24	2.2	DoIP extensions; dynamic UDP port usage
Michael Dangelmaier	2012-10-04	2.3	DaVinci Configurator Pro support; AUTOSAR 4 support
Michael Dangelmaier	2013-07-10	2.4	Customer specific extensions
Michael Dangelmaier	2014-05-23	3.0	Updated to AUTOSAR 4.1
Michael Dangelmaier	2015-03-26	4.0	Updated to AUTOSAR 4.2
Michael Dangelmaier	2015-05-16	5.0	Optimized UDP retry behavior, Added BSD Socket API
Philipp Christmann	2015-11-16	6.0	Support of post-build loadable
Michael Dangelmaier	2016-03-23	6.1	Support of TLS client Trigger Transmit API with SduLength In/Out Description for shutdown mechanism
Michael Dangelmaier	2016-04-28	6.2	Release of BSD-Socket API
Michael Dangelmaier	2016-05-31	6.3	Extension of BSD Socket API to support SOME/IP-SD under Linux
Michael Dangelmaier	2016-11-11	7.0	MainFunction splitting Optimized TP transmission Trigger Transmit API for SoAd_IfTransmit Optimized buffer handling for PDU fan-out
Michael Dangelmaier	2017-01-23	7.1	Event Queues and Timeout Lists
Michael Dangelmaier	2017-02-22	7.2	Support Buffer Size up to 128kB
Michael Dangelmaier	2017-05-08	8.0	Updated component history
Michael Dangelmaier	2017-05-30	8.1	Updated service IDs
Michael Dangelmaier	2017-06-19	8.2	PDU reception verification Transmission on specific socket connection Forward socket connection on reception
Michael Dangelmaier	2017-08-01	8.3	Updated API description
Michael Dangelmaier	2017-08-28	8.4	Reworked critical section chapter
Michael Dangelmaier	2018-03-19	9.0	Reworked TP-API description
Michael Seidenspinner	2018-07-31	10.0	Support INTEGRITY
Michael Dangelmaier	2018-08-22	10.1	Support VLAN priorities for Linux

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_SocketAdaptor.pdf	4.2.2
[2]	AUTOSAR	AUTOSAR_SWS_SocketAdaptor.pdf	4.3.0
[3]	AUTOSAR	AUTOSAR_SWS_DevelopmentErrorTracer.pdf	4.1.2
[4]	AUTOSAR	AUTOSAR_SWS_DiagnosticEventManager.pdf	4.1.2
[5]	AUTOSAR	AUTOSAR_BasicSoftwareModules.pdf	V1.0.0
[6]	AUTOSAR	AUTOSAR_SWS_TcpIp.pdf	4.2.1

Scope of the Document

This technical reference describes the general use of the Socket Adaptor basis software. Please refer to your Release Notes to get a detailed description of the platform (host, compiler) your Vector Ethernet Bundle has been configured for.

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
1.00	Created
1.01	BSD socket support (WinSock)
1.02	RAM variables moved into configuration
1.03	Spread buffer support (PbufType)
1.04	PduR integration
3.00	Extended functionality for IPv6
3.01	Support for UdpNm and other upper layer protocols DoIP: New header format for Routing Activation Request Call-back for EthSM
3.02	MISRA conformance
3.03	Changed IP address assignment call-back
3.04	DoIP extensions; dynamic UDP port usage

3.05	DaVinci Configurator Pro support;
	IPv6 support (DaVinci Configurator Pro only);
	Support of PDUR API according to AUTOSAR 4 for DoIP (DaVinci Configurator Pro only);
	Module optimization and clean-up

3.06	Possibility to choose various ISO13400 stages (DIS, FDIS, IS); Stubs for Cancel Transmit/-Receive and Change-/Read Parameter; Support of multiple additional include files for DoIP; RTE Service Ports to configure for VIN, GID, PowerMode Support of additional features XCP-Routing and Mirroring
3.07	Support of an additional customer specific feature IPv6 support for GENy configuration tool
4.00	Support of AUTOSAR 4.1 only with limited feature support
4.01	Extended functionality according to AUTOSAR 4.1
4.02	Support of streaming-based TCP TxConfirmation
4.03	Support of interfaces according to AUTOSAR 4.1.3
4.05	Service Discovery extensions; Support of Alive Supervision Timeout
4.07	Service Discovery extensions; Reworked Shutdown mechanism for upper layers (e.g. DoIP)
5.00	Complete extraction of DoIP
6.00	Support of major features of AUTOSAR 4.2.1
7.00	Optimized UDP retry behavior,

Component Version	New Features
	Added BSD Socket API
8.00	Support of configuration variant post-build loadable
8.01	Support of TLS client; Trigger Transmit API with SduLength In/Out
8.02	Release of BSD-Socket API
8.03	Extension of BSD Socket API to support SOME/IP-SD under Linux

9.00	MainFunction splitting;
	Optimized TP transmission;
	Trigger Transmit API for SoAd_IfTransmit;
	Optimized buffer handling for PDU fan-out

9.01	Event Queues and Timeout Lists
9.02	Support Buffer Size up to 128kB
10.00	Reworked header includes (P3 CAD)
10.01	Adapted API pattern (P3 Implementation API Pattern)
10.02	PDU reception verification (Callout for Diagnostic Firewall Use Case); “Transmission on specific socket connection” and “Forward socket connection on reception” (Support optimized PDU handling for C/S calls)
10.03	P3 Code Refactoring
10.04	P3 Code Refactoring / CDD Step 2
11.00	P3 Code Refactoring / CDD Step 3; SAFE Code Refactoring Support QNX
12.00	Support INTEGRITY
12.01	Support VLAN priorities for Linux

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module Socket Adaptor as specified in [1].

Supported AUTOSAR Release*:	4.2	
Supported Configuration Variants:	pre-compile, post-build	
Vendor ID:	SOAD_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	SOAD_MODULE_ID	56 decimal (according to ref. [5])

* For the precise AUTOSAR Release 4.1.x please see the release specific documentation.

The Socket Adaptor provides communication between PDU based communication and socket based communication via TcpIp. Following key features are offered by the Socket Adaptor:

- Support of TCP and UDP sockets over lower module TcpIp
- Supports multiple socket connections per local socket to support multiple communication partners on the same local socket
- Control API for socket connections or automated socket connection handling by Socket Adaptor
- Independent reception (Socket Route) and transmission path (Pdu Route) on a socket connection
- Support of Interface (IF) and Transport Protocol (TP) PDUs for upper layers
- Generic upper layer configuration

Figure 2-1 provides a functional overview over Socket Adaptor and some examples of possible configuration variants.

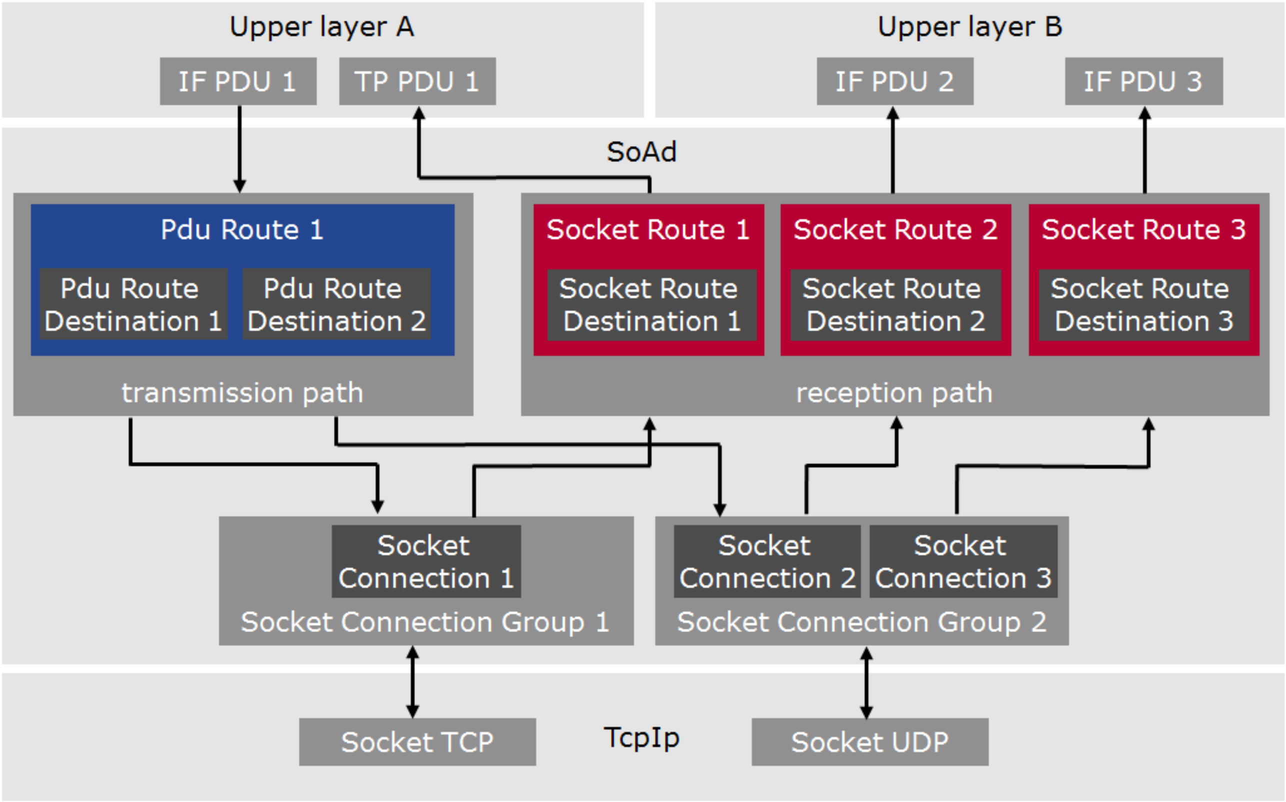
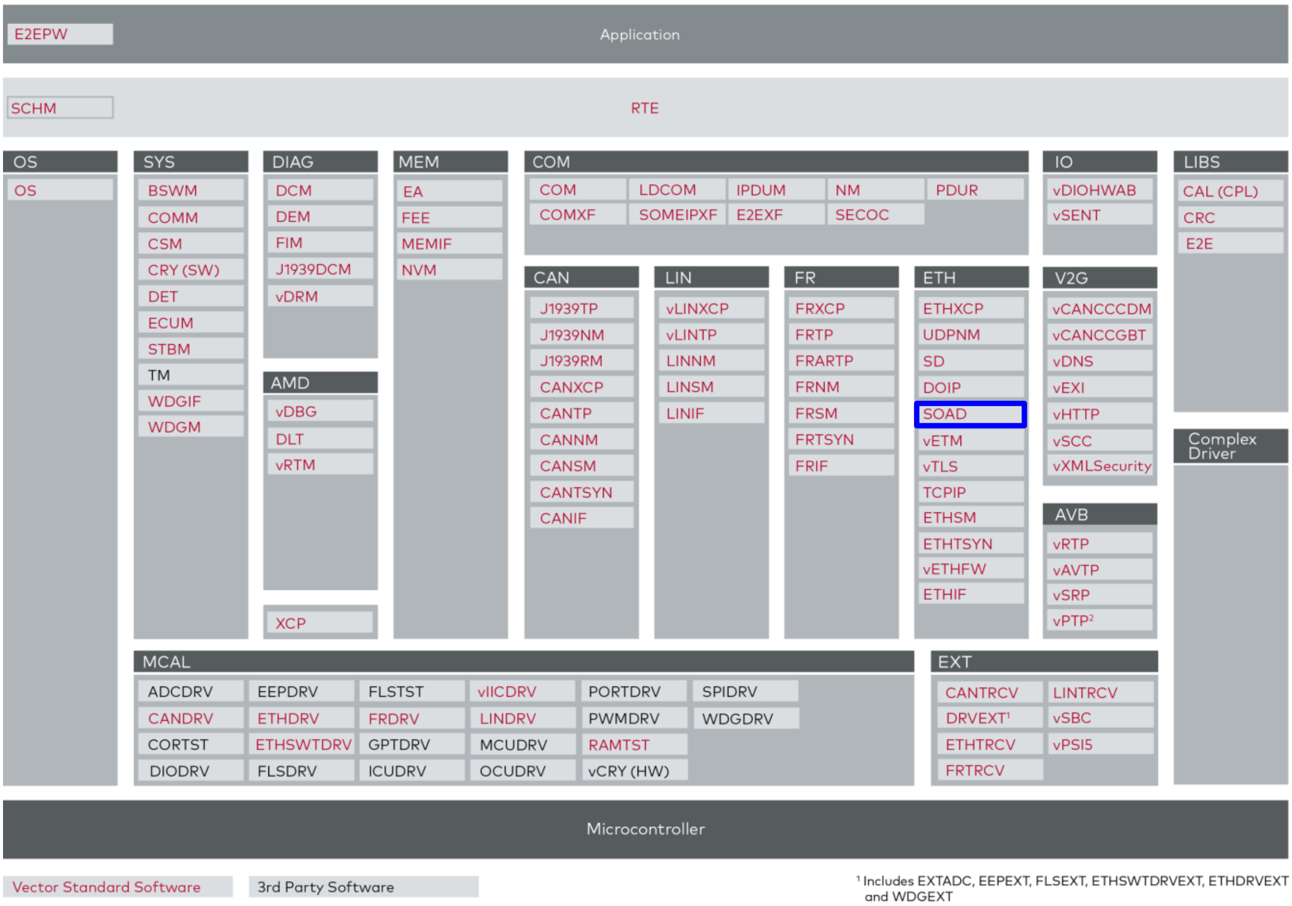


Figure 2-1 Functional Overview

2.1 Architecture Overview

The following figure shows where the Socket Adaptor is located in the AUTOSAR architecture.



[1] Includes EXTADC, EEPEXT, FLSEXT, ETHSWTDRVEXT, ETHDRVEXT and WDGEXT
[2] Functionality represented in ETHTSYN and STBM

Figure 2-2 AUTOSAR 4.2 Architecture Overview

The next figure shows the interfaces of the Socket Adaptor. These interfaces are described in chapter 5.

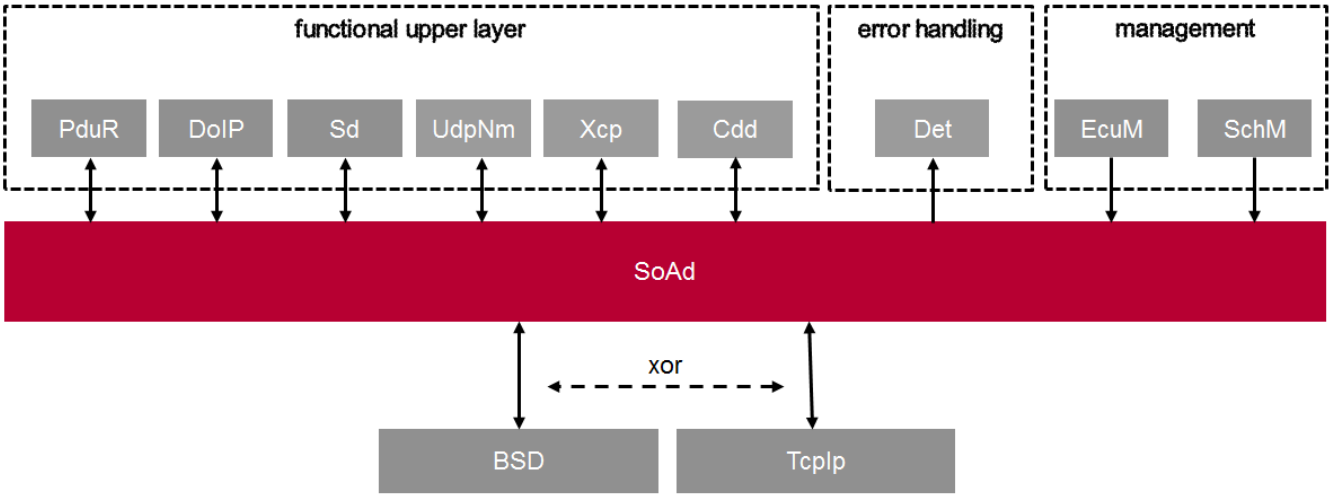


Figure 2-3 Interfaces to adjacent modules of the Socket Adaptor

Applications do not access the services of the BSW modules directly. They use the service ports provided by the BSW modules via the RTE. The Socket Adaptor does not support any service ports.

3 Functional Description

The features listed in the following tables cover the complete functionality specified for the Socket Adaptor.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported AUTOSAR standard conform feature

> Table 3-2 Not supported AUTOSAR standard conform features

> Table 3-3 Not supported AUTOSAR optional features

Vector Informatik provides further Socket Adaptor functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 3-4 Features provided beyond the AUTOSAR standard

3.1 Features

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Socket Connections and Socket Connection Groups
PDU Transmission
PDU Reception
PDU Header option
Best Match Algorithm
Message Acceptance Policy
TP PDU Cancelation
Disconnection and recovery
Routing Groups
PDU fan-out
Buffer handling (e.g. nPduUdpTxBuffer)
Error handling
Version check
Address assignment services
Support of post-build loadable

Table 3-1 Supported AUTOSAR standard conform feature

3.2 Deviations

The following features specified in [1] are not supported:

3.2.1 Not supported standard conform features

Not Supported AUTOSAR Standard Conform Features
Socket Routes (TCP/UDP) with multiple TP upper layers and disabled PDU Header option
Change Parameter service

Table 3-2 Not supported AUTOSAR standard conform features

3.2.2 Not supported optional features

Not Supported AUTOSAR Standard Conform Features
Ressource Management Option

Table 3-3 Not supported AUTOSAR optional features

3.2.3 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard.

Features Provided Beyond The AUTOSAR Standard
Best Match Algorithm with PDU Header validation
Best Match Algorithm with Socket Route validation
UDP immediate IF transmission confirmation (TxConfirmation)
API extension to get the remote address of received data (SoAd_GetRcvRemoteAddr)
Additional SoConModeChg notification
BSD Socket API for Linux, QNX and INTEGRITY (only for configuration variant pre-compile)
Shutdown mechanism
TLS client
MainFunction splitting
Optimized TP transmission
Trigger Transmit API for SoAd_IfTransmit
Optimized buffer handling for PDU fan-out
Event Queues and Timeout Lists
PDU reception verification

Table 3-4 Features provided beyond the AUTOSAR standard

Caution
There may be also some other deviations which are not documented here.

3.2.4 Known Issues (low priority)

3.2.4.1 ESCAN00087305

Restricted functionality of compiler abstraction

The compiler abstraction for pointers does always use the identical 'ptrclass', independently from the memory location of the target. (It is not differentiated between variables stored in the pre-compile or post-build memory sections.)

Hence, the compiler abstraction cannot be used to specify and optimize pointers (would lead to compiler errors).

Workaround: Do not use special optimizations in compiler abstraction.

3.2.5 Hints

3.2.5.1 CDD Contribution Type

Socket Adaptor supports the “CddSoAdUpperLayerContribution” with schema according to AUTOSAR 4.0.3. Older versions support “CddComIfUpperLayerContribution” instead of “CddSoAdUpperLayerContribution”.

3.2.5.2 API deviation

The API to upper layer modules is implemented according to AUTOSAR 4.1.3 and partly to 4.2.1.

Please refer to chapter 5 for details.

3.2.5.3 UDP socket ressources bound at startup

If a UDP socket connection remote address contains wildcards, socket connection can be opened on reception according to [1]. To support this feature corresponding socket connection must bind a TcpIp socket at ECU startup (i.e. first MainFunction cycle).

3.2.5.4 SoAd_SetUniqueRemoteAddress() disables alive supervision timeout

If `SoAd_SetUniqueRemoteAddress()` is called for a UDP socket connection group and a corresponding socket connection in state online is found, alive supervision timeout will be disabled for this socket connection.

3.2.5.5 SoAd_CloseSoCon() if open/close counter is 0

If `SoAd_CloseSoCon()` is called with parameter `abort` set to `TRUE` and open/close counter is 0, caused by socket connection open in reception of data, the corresponding socket connection will be closed anyway. If parameter `abort` is set to `FALSE`, socket connection is not closed.

This behavior was implemented to close socket connections by user in all cases and to prevent always open socket connections that blocks communication with other remote entities.

3.3 Initialization

The Socket Adaptor is initialized via a `SoAd_InitMemory()` call followed by call of `SoAd_Init()`.

Example

```
SoAd_Init(SoAd_Config_Ptr);
```

3.3.1 Configuration Variants 1, 2 (Pre-Compile and Link-Time)

At configuration Variant 1 (Pre-compile) and Variant 2 (Link-Time) the SoAd module has to be initialized using the `SoAd_Init()` function with the address of the pre-compile configuration data passed as parameter. The declaration of the pre-compile configuration data is contained in the files SoAd_Lcfg.h and SoAd_Lcfg.c.

3.3.2 Configuration Variant 3 (Post-build)

In this configuration Variant, the SoAd module has to be initialized using the `SoAd_Init()` function with the address of the post-build configuration data passed as parameter. The declaration of the post-build configuration data is contained in the files SoAd_PBcfg.h and SoAd_PBcfg.c.

Please refer to chapter 6.1 to get information about supported configuration variants.

3.4 States

The Socket Adaptor has an extended state handling after calling the initialization functions (described in chapter before). Figure 3-1 shows the states of Socket Adaptor when using the shutdown feature described in 6.2.4.6.

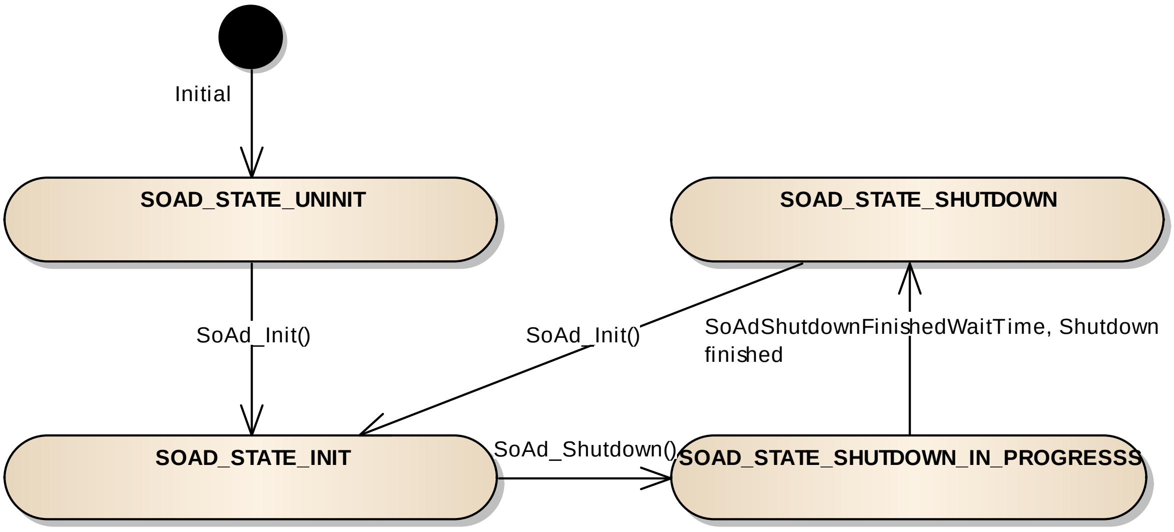


Figure 3-1 Module states

3.5 Main Functions

The Socket Adaptor has one main function (except when 6.2.4.8 MainFunction splitting is enabled) which handles

- Transmission/reception and socket handling on BSD Socket API
- Socket connection state handling

- TP transmission/reception
- TP transmission/reception cancellation
- UDP nPduUdpTxBuffer
- TriggerTransmit transmission
- Handle pending transmission confirmation

3.6 Error Handling

3.6.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [3], if development error reporting is enabled (i.e. pre-compile parameter `SOAD_DEV_ERROR_DETECT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported Socket Adaptor ID is 56.

The reported service IDs identify the services which are described in 5.2. The following table presents the service IDs and the related services:

Service ID	Service
0x01	SOAD_SID_INIT
0x02	SOAD_SID_GET_VERSION_INFO
0x03	SOAD_SID_IF_TRANSMIT
0x04	SOAD_SID_TP_TRANSMIT
0x05	SOAD_SID_TP_CANCEL_TRANSMIT
0x06	SOAD_SID_TP_CANCEL_RECEIVE
0x07	SOAD_SID_GET_SO_CON_ID
0x08	SOAD_SID_OPEN_SO_CON
0x09	SOAD_SID_CLOSE_SO_CON
0x0A	SOAD_SID_REQ_IP_ADDR_ASSIGN
0x0B	SOAD_SID_RLS_IP_ADDR_ASSIGN
0x0C	SOAD_SID_GET_LOCAL_ADDR
0x0D	SOAD_SID_GET_PHYS_ADDR
0x0E	SOAD_SID_ENABLE_ROUTING
0x0F	SOAD_SID_DISABLE_ROUTING
0x10	SOAD_SID_SET_REMOTE_ADDR
0x11	SOAD_SID_TP_CHANGE_PARAMETER

Service ID	Service
0x12	SOAD_SID_RX_INDICATION
0x13	SOAD_SID_COPY_TX_DATA
0x14	SOAD_SID_TX_CONFIRMATION
0x15	SOAD_SID_TCP_ACCEPTED
0x16	SOAD_SID_TCP_CONNECTED
0x17	SOAD_SID_TCPIP_EVENT
0x18	SOAD_SID_LOCAL_IP_ADDR_ASSIGNMENT_CHG
0x19	SOAD_SID_MAIN_FUNCTION
0x1A	SOAD_SID_READ_DHCP_HOST_NAME_OPT
0x1B	SOAD_SID_WRITE_DHCP_HOST_NAME_OPT
0x1C	SOAD_SID_GET_REMOTE_ADDR
0x1D	SOAD_SID_IF_ROUT_GROUP_TRANSMIT
0x1E	SOAD_SID_SET_UNI_REMOTE_ADDR
0x1F	SOAD_SID_IF_SPEC_ROUT_GROUP_TRANSMIT
0x20	SOAD_SID_ENABLE_SPECIFIC_ROUTING
0x21	SOAD_SID_DISABLE_SPECIFIC_ROUTING
0xD0	SOAD_SID_MAIN_FUNCTION_RX
0xD1	SOAD_SID_MAIN_FUNCTION_STATE
0xD2	SOAD_SID_MAIN_FUNCTION_TX
0xD3	SOAD_SID_SHUTDOWN
0xD4	SOAD_SID_GET_RCV_REMOTE_ADDR
0xD5	SOAD_SID_GET_REMOTE_ADDR_STATE

Table 3-5 Service IDs

The errors reported to DET are described in the following table:

Error Code	Description
0x01	SOAD_E_NOTINIT
0x02	SOAD_E_PARAM_POINTER
0x03	SOAD_E_INV_ARG
0x04	SOAD_E_NOBUFS
0x05	SOAD_E_INV_PDUHEADER_ID
0x06	SOAD_E_INV_PDUID
0x07	SOAD_E_INV_SOCKETID

Error Code	Description
0x08	SOAD_E_INIT_FAILED

Table 3-6 Errors reported to DET

3.6.2 Production Code Error Reporting

By default, production code related errors are reported to the DEM using the service `Dem_ReportErrorStatus()` as specified in [4].

The Socket Adaptor does not support DEM errors.

4 Integration

This chapter gives necessary information for the integration of the MICROSAR Socket Adaptor into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the Socket Adaptor consists out of these files:

File Name	Description
SoAd.c	Static source file
SoAd.h	Static header file
SoAd_Cbk.h	Static header file for callback functions
SoAd_EventQueue.c	Static source file for sub-module which handles event queues
SoAd_EventQueue.h	Static header file for sub-module which handles event queues
SoAd_Priv.h	Static header file for Socket Adaptor internal usage
SoAd_RouteGrp.c	Static source file for sub-module which handles routing groups
SoAd_RouteGrp.h	Static header file for sub-module which handles routing groups
SoAd_Rx.c	Static source file for sub-module which handles reception
SoAd_Rx.h	Static header file for sub-module which handles reception
SoAd_SoCon.c	Static source file for sub-module which handles socket connections
SoAd_SoCon.h	Static header file for sub-module which handles socket connections
SoAd_TcpIp.c	Static source file for sub-module which handles TcpIp-Stack specifics
SoAd_TcpIp.h	Static header file for sub-module which handles TcpIp-Stack specifics
SoAd_TimeoutList.c	Static source file for sub-module which handles timeout lists
SoAd_TimeoutList.h	Static header file for sub-module which handles timeout lists
SoAd_Tx.c	Static source file for sub-module which handles transmission
SoAd_Tx.h	Static header file for sub-module which handles transmission
SoAd_Util.c	Static source file for sub-module which provides general operations
SoAd_Util.h	Static header file for sub-module which provides general operations
SoAd_Types.h	Static header file containing types
SoAd_Lcfg.c	Generated source file (e.g. RAM/ROM mapping tables)
SoAd_Lcfg.h	Generated header file
SoAd_Cfg.h	Generated header file for configuration parameter (e.g. feature switches)
SoAd_PBcfg.c	Generated source file (Post-build configuration)
SoAd_PBcfg.h	Generated header file (Post-build configuration)
SoAd.lib	Library if Socket Adaptor is not delivered with source code

Table 4-1 Implementation files

4.2 Critical Sections

All services and callbacks for transmission, reception and state changes of Socket Adaptor may be called in interrupt or task level. Thus a synchronization mechanism is implemented to guarantee data consistency.

The synchronization mechanism defined by AUTOSAR covers the entering and leaving of so called critical sections.

The implementation of the critical sections must avoid that multiple relevant tasks or interrupt service routines can enter each of the critical sections more than once at the same time.

Relevant interrupt services in the Socket Adaptor context are interrupt services originated from physical bus events (Ethernet, CAN, LIN, FlexRay etc.).

Relevant tasks in the Socket Adaptor context are all tasks which call Socket Adaptor API functions. Usually these tasks are limited to tasks on which other BSW modules (DoIP, Sd, Xcp etc.) are mapped to.

A critical section can be handled by using the so called “Exclusive Areas”. The Socket Adaptor defines the following exclusive area:

- SOAD_EXCLUSIVE_AREA_0 is used whenever memory accesses must be protected from accesses of interrupting calls to services and callbacks of Socket Adaptor. This exclusive area may be entered in interrupt or task context. The frequency of entering and leaving this area will be very high. The average length of stay in the area is medium.

For an implementation of the critical section it could be sufficient to

- Disable all bus relevant interrupts of all buses related to calls to Socket Adaptor API functions (e.g. gateway use-case).
- Disable all Ethernet bus relevant interrupts if all modules calling Socket Adaptor API functions are mapped to one task (e.g. SchM task) or a non-preemptive OS is used.
- Not implement the critical section in case BSD Socket API is used and the calls to Socket Adaptor API do not interrupt the task of Socket Adaptor.

Please note that these are only examples and that the actual implementation of the critical sections is highly dependent on the platform architecture and the system configuration.

4.3 Main Function cycle

In case BSD Socket API is used, some operations might take longer than with the AUTOSAR TCP/IP. This can influence the execution time of the SoAd main function. Therefore, an increased main function cycle might be required to avoid multiple task activation and to ensure a stable periodic call of the SoAd main function.

5 API Description

For an interfaces overview please see Figure 2-3.

5.1 Type Definitions

Socket Adaptor uses types which are defined by [1] .

Type Name	C-Type	Description	Value Range
SoAd_RemAddrStateType	uint8	Describes remote IP address and port state.	SOAD_SOCON_IP_SET_PORT_SET
			SOAD_SOCON_IP_SET_PORT_ANY
			SOAD_SOCON_IP_SET_PORT_NOT
			SOAD_SOCON_IP_ANY_PORT_SET
			SOAD_SOCON_IP_ANY_PORT_ANY
			SOAD_SOCON_IP_ANY_PORT_NOT
			SOAD_SOCON_IP_NOT_PORT_SET
			SOAD_SOCON_IP_NOT_PORT_ANY
			SOAD_SOCON_IP_NOT_PORT_NOT

Table 5-1 SoAd_RemAddrStateType

5.2 Services provided by Socket Adaptor

This chapter describes the service functions that are implemented by the Socket Adaptor and can be invoked by other modules. The prototypes of the service functions are provided in the header file `SoAd.h` by the Socket Adaptor.

5.2.1 SoAd_InitMemory

Prototype	
void **SoAd_InitMemory** (void)	
Parameter	
void	none
Return code	
void	none
Functional Description	
Initializes *_INIT_*-variables.	
Particularities and Limitations	
Module is uninitialized. Service to initialize module global variables at power up. This function initializes the variables in *_INIT_* sections. Used in case they are not initialized by the startup code.	
Call context	
> TASK	

> This function is Synchronous > This function is Non-Reentrant

Table 5-2 SoAd_InitMemory

5.2.2 SoAd_Init

Prototype	
void **SoAd_Init** (const SoAd_ConfigType *SoAdConfigPtr)	
Parameter	
SoAdConfigPtr [in]	Configuration structure for initializing the module.
Return code	
void	none
Functional Description	
Initializes module.	
Particularities and Limitations	
> Interrupts are disabled.SoAd_InitMemory has been called unless SoAd_ModuleInitialized is initialized by start-up code. This function initializes the module SoAd. It initializes all variables and sets the module state to initialized.	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-3 SoAd_Init

5.2.3 SoAd_IfTransmit

Prototype	
Std_ReturnType **SoAd_IfTransmit** (PduIdType SoAdSrcPduId, const PduInfoType *SoAdSrcPduInfoPtr)	
Parameter	
SoAdSrcPduId [in]	Tx PDU identifier.
SoAdSrcPduInfoPtr [in]	Pointer to PDU.
Return code	
Std_ReturnType	E_OK Transmit request was accepted.
Std_ReturnType	E_NOT_OK Transmit request was not accepted.
Functional Description	
Transmits an IF-PDU.	

Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-4 SoAd_IfTransmit

5.2.4 SoAd_IfRoutingGroupTransmit

Prototype	
`Std_ReturnType` **`SoAd_IfRoutingGroupTransmit`** `(SoAd_RoutingGroupIdType id)`	
Parameter	
id [in]	Routing group identifier.
Return code	
Std_ReturnType	E_OK Transmit request was accepted.
Std_ReturnType	E_NOT_OK Transmit request was not accepted.
Functional Description	
Triggers transmission of all IF-PDUs realted to a routing group.	
Particularities and Limitations	
- Triggers transmission via trigger transmit in main function context.	
Call context	
> TASK|ISR2 > This function is Reentrant	

Table 5-5 SoAd_IfRoutingGroupTransmit

5.2.5 SoAd_IfSpecificRoutingGroupTransmit

Prototype	
`Std_ReturnType` **`SoAd_IfSpecificRoutingGroupTransmit`** `(SoAd_RoutingGroupIdType id, SoAd_SoConIdType SoConId)`	
Parameter	
id [in]	Routing group identifier.
SoConId [in]	Socket connection identifier.
Return code	
Std_ReturnType	E_OK Transmit request was accepted.

Std_ReturnType	E_NOT_OK Transmit request was not accepted.
Functional Description	
Triggers transmission of all IF-PDUs realted to a routing group and socket connection.	
Particularities and Limitations	
- Triggers transmission via trigger transmit in main function context.	
Call context	
> TASK|ISR2 > This function is Reentrant	

Table 5-6 SoAd_IfSpecificRoutingGroupTransmit

5.2.6 SoAd_TpTransmit

Prototype	
`Std_ReturnType` **`SoAd_TpTransmit`** `(PduIdType SoAdSrcPduId, const PduInfoType *SoAdSrcPduInfoPtr)`	
Parameter	
SoAdSrcPduId [in]	Tx PDU identifier.
SoAdSrcPduInfoPtr [in]	Pointer to PDU (length is evaluated only).
Return code	
Std_ReturnType	E_OK Transmit request was accepted.
Std_ReturnType	E_NOT_OK Transmit request was not accepted.
Functional Description	
Transmits a TP-PDU.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Reentrant	

Table 5-7 SoAd_TpTransmit

5.2.7 SoAd_Shutdown

Prototype	
`Std_ReturnType` **`SoAd_Shutdown`** `(void)`	
Parameter	
void	none

Return code	
Std_ReturnType	E_OK Shutdown request was accepted.
	SOAD_E_INPROGRESS Shutdown is in progress.
	E_NOT_OK Shutdown request was not accepted.
Functional Description	
Shuts down SoAd module.	
Particularities and Limitations	
- Closes all open socket connections and disables transmission and reception.	
Call context	
> TASK|ISR2 > This function is Non-Reentrant	

Table 5-8 SoAd_Shutdown

5.2.8 SoAd_TpCancelTransmit

Prototype	
`Std_ReturnType` **`SoAd_TpCancelTransmit`** `(PduIdType PduId)`	
Parameter	
PduId [in]	Tx PDU identifier.
Return code	
Std_ReturnType	E_OK Transmit cancellation request was accepted.
Std_ReturnType	E_NOT_OK Transmit cancellation request was not accepted.
Functional Description	
Requests transmission cancellation of a specific TP-PDU.	
Particularities and Limitations	
Transmission of PDU is requested via SoAd_TpTransmit. -	
Call context	
> TASK|ISR2 > This function is Reentrant	

Table 5-9 SoAd_TpCancelTransmit

5.2.9 SoAd_TpCancelReceive

Prototype
`Std_ReturnType` **`SoAd_TpCancelReceive`** `(PduIdType PduId)`

<table>
<tr><th colspan="2">Parameter</th></tr>
<tr><td>PduId [in]</td><td>Rx PDU identifier.</td></tr>
<tr><th colspan="2">Return code</th></tr>
<tr><td>Std_ReturnType</td><td>E_OK Receive cancellation request was accepted.</td></tr>
<tr><td>Std_ReturnType</td><td>E_NOT_OK Receive cancellation request was not accepted.</td></tr>
<tr><th colspan="2">Functional Description</th></tr>
<tr><td colspan="2">Requests reception cancellation of a specific TP-PDU.</td></tr>
<tr><th colspan="2">Particularities and Limitations</th></tr>
<tr><td colspan="2">Reception of PDU is initiated via <Up>_[SoAd][Tp]StartOfReception.
-</td></tr>
<tr><td colspan="2">Call context</td></tr>
<tr><td colspan="2">> TASK|ISR2
> This function is Reentrant</td></tr>
</table>

Table 5-10 SoAd_TpCancelReceive

5.2.10 SoAd_GetSoConId

<table>
<tr><th colspan="2">Prototype</th></tr>
<tr><td colspan="2"><code>Std_ReturnType <b>SoAd_GetSoConId</b> (PduIdType TxPduId, SoAd_SoConIdType *SoConIdPtr)</code></td></tr>
<tr><th colspan="2">Parameter</th></tr>
<tr><td>TxPduId [in]</td><td>Tx PDU identifier.</td></tr>
<tr><td>SoConIdPtr [out]</td><td>Pointer to the socket connection identifier.</td></tr>
<tr><th colspan="2">Return code</th></tr>
<tr><td>Std_ReturnType</td><td>E_OK Socket connection identifier was found.</td></tr>
<tr><td>Std_ReturnType</td><td>E_NOT_OK Socket connection identifier was not found.</td></tr>
<tr><th colspan="2">Functional Description</th></tr>
<tr><td colspan="2">Returns the socket connection identifier of a specific Tx PDU identifier.</td></tr>
<tr><th colspan="2">Particularities and Limitations</th></tr>
<tr><td colspan="2">-
-</td></tr>
<tr><td colspan="2">Call context</td></tr>
<tr><td colspan="2">> TASK|ISR2
> This function is Synchronous
> This function is Reentrant</td></tr>
</table>

Table 5-11 SoAd_GetSoConId

5.2.11 SoAd_OpenSoCon

Prototype	
Std_ReturnType **SoAd_OpenSoCon** (SoAd_SoConIdType SoConId)	
Parameter	
SoConId [in]	Socket connection identifier.
Return code	
Std_ReturnType	E_OK Open request was accepted.
Std_ReturnType	E_NOT_OK Open request was not accepted.
Functional Description	
Opens a socket connection.	
Particularities and Limitations	
- Opens the socket connection in context of main function.	
Call context	
> TASK|ISR2 > This function is Reentrant	

Table 5-12 SoAd_OpenSoCon

5.2.12 SoAd_CloseSoCon

Prototype	
Std_ReturnType **SoAd_CloseSoCon** (SoAd_SoConIdType SoConId, boolean Abort)	
Parameter	
SoConId [in]	Socket connection identifier.
Abort [in]	Flag to close socket connection immediately. [range: TRUE close immediately, FALSE close when open close sequence is 0]
Return code	
Std_ReturnType	E_OK Close request was accepted.
Std_ReturnType	E_NOT_OK Close request was not accepted.
Functional Description	
Closes a socket connection.	
Particularities and Limitations	
- Closes the socket connection in context of main function.	
Call context	
> TASK|ISR2 > This function is Reentrant	

Table 5-13 SoAd_CloseSoCon

5.2.13 SoAd_RequestIpAddrAssignment

Prototype	
`Std_ReturnType **SoAd_RequestIpAddrAssignment** (SoAd_SoConIdType SoConId, SoAd_IpAddrAssignmentType Type, SoAd_SockAddrType *LocalIpAddrPtr, uint8 Netmask, SoAd_SockAddrType *DefaultRouterPtr)`	
Parameter	
SoConId [in]	Socket connection identifier.
Type [in]	IP address type.
LocalIpAddrPtr [in]	Pointer to IP address which shall be assigned.
Netmask [in]	Netmask in CIDR.
DefaultRouterPtr [in]	Pointer to default router (gateway) address.
Return code	
Std_ReturnType	E_OK Assignment request was accepted.
Std_ReturnType	E_NOT_OK Assignment request was not accepted.
Functional Description	
Requests IP address assignment on a local address identified by a socket connection.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-14 SoAd_RequestIpAddrAssignment

5.2.14 SoAd_ReleaseIpAddrAssignment

Prototype	
`Std_ReturnType **SoAd_ReleaseIpAddrAssignment** (SoAd_SoConIdType SoConId)`	
Parameter	
SoConId [in]	Socket connection identifier.
Return code	
Std_ReturnType	E_OK Release request was accepted.
Std_ReturnType	E_NOT_OK Release request was not accepted.
Functional Description	
Releases IP address assignment on a local address identified by a socket connection.	

Particularities and Limitations
-
-
Call context
> TASK|ISR2
> This function is Synchronous
> This function is Reentrant

Table 5-15 SoAd_ReleaseIpAddrAssignment

5.2.15 SoAd_GetLocalAddr

Prototype	
`Std_ReturnType` **`SoAd_GetLocalAddr`** `(SoAd_SoConIdType SoConId, SoAd_SockAddrType *LocalAddrPtr, uint8 *NetmaskPtr, SoAd_SockAddrType *DefaultRouterPtr)`	
Parameter	
SoConId [in]	Socket connection identifier.
LocalAddrPtr [out]	Pointer to local address (IP and Port).
NetmaskPtr [out]	Pointer to network mask (CIDR Notation).
DefaultRouterPtr [out]	Pointer to default router (gateway).
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Returns a local IP address identified by a socket connection.	
Particularities and Limitations	
-	
-	
Call context	
> TASK|ISR2	
> This function is Synchronous	
> This function is Reentrant	

Table 5-16 SoAd_GetLocalAddr

5.2.16 SoAd_GetPhysAddr

Prototype
`Std_ReturnType` **`SoAd_GetPhysAddr`** `(SoAd_SoConIdType SoConId, uint8 *PhysAddrPtr)`

Parameter	
SoConId [in]	Socket connection identifier.
PhysAddrPtr [out]	Pointer to physical address.
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Returns the physical address (MAC address) of a local interface identified by a socket connection.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-17 SoAd_GetPhysAddr

5.2.17 SoAd_GetRemoteAddr

Prototype	
`Std_ReturnType` **`SoAd_GetRemoteAddr`** `(SoAd_SoConIdType SoConId, SoAd_SockAddrType *IpAddrPtr)`	
Parameter	
SoConId [in]	Socket connection identifier.
IpAddrPtr [out]	Pointer to remote address.
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Returns the remote address of a socket connection.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-18 SoAd_GetRemoteAddr

5.2.18 SoAd_GetRemoteAddrState

Prototype	
`Std_ReturnType **SoAd_GetRemoteAddrState** (SoAd_SoConIdType SoConId, SoAd_SockAddrType *IpAddrPtr, SoAd_RemAddrStateType *RemAddrState)`	
Parameter	
SoConId [in]	Socket connection identifier.
IpAddrPtr [out]	Pointer to remote address.
RemAddrState [out]	Pointer to remote address state.
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Returns the remote address and remote address state of a socket connection.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-19 SoAd_GetRemoteAddrState

5.2.19 SoAd_GetRcvRemoteAddr

Prototype	
`Std_ReturnType **SoAd_GetRcvRemoteAddr** (SoAd_SoConIdType SoConId, SoAd_SockAddrType *IpAddrPtr)`	
Parameter	
SoConId [in]	Socket connection identifier.
IpAddrPtr [out]	Pointer to remote address.
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Returns the remote address of the last received message on a socket connection.	
Particularities and Limitations	
-	

- Configuration Variant(s): SOAD_VGET_RCV_REMOTE_ADDR_ENABLED
Call context
> TASK|ISR2 > This function is Synchronous > This function is Reentrant

Table 5-20 SoAd_GetRcvRemoteAddr

5.2.20 SoAd_EnableRouting

Prototype	
`Std_ReturnType **SoAd_EnableRouting** (SoAd_RoutingGroupIdType id)`	
Parameter	
id [in]	Routing group identifier.
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Enables a routing group.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-21 SoAd_EnableRouting

5.2.21 SoAd_EnableSpecificRouting

Prototype	
`Std_ReturnType **SoAd_EnableSpecificRouting** (SoAd_RoutingGroupIdType id, SoAd_SoConIdType SoConId)`	
Parameter	
id [in]	Routing group identifier.
SoConId [in]	Socket connection identifier.
Return code	
Std_ReturnType	E_OK Request was accepted.

Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Enables a routing group on a specific socket connection.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-22 SoAd_EnableSpecificRouting

5.2.22 SoAd_DisableRouting

Prototype	
`Std_ReturnType SoAd_DisableRouting (SoAd_RoutingGroupIdType id)`	
Parameter	
id [in]	Routing group identifier.
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Disables a routing group.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-23 SoAd_DisableRouting

5.2.23 SoAd_DisableSpecificRouting

Prototype

`Std_ReturnType SoAd_DisableSpecificRouting (SoAd_RoutingGroupIdType id,`
`SoAd_SoConIdType SoConId)`

Parameter	
id [in]	Routing group identifier.
SoConId [in]	Socket connection identifier.
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Disables a routing group on a specific socket connection.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-24 SoAd_DisableSpecificRouting

5.2.24 SoAd_SetRemoteAddr

Prototype	
`Std_ReturnType` **`SoAd_SetRemoteAddr`** `(SoAd_SoConIdType SoConId, SoAd_SockAddrType *RemoteAddrPtr)`	
Parameter	
SoConId [in]	Socket connection identifier.
RemoteAddrPtr [in]	Pointer to remote address.
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Sets the remote address of a socket connection.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-25 SoAd_SetRemoteAddr

5.2.25 SoAd_SetUniqueRemoteAddr

Prototype	
Std_ReturnType **SoAd_SetUniqueRemoteAddr** (SoAd_SoConIdType SoConId, SoAd_SockAddrType *RemoteAddrPtr, SoAd_SoConIdType *AssignedSoConIdPtr)	
Parameter	
SoConId [in]	Socket connection identifier (any socket connection in socket connection group).
RemoteAddrPtr [in]	Pointer to remote address.
AssignedSoConIdPtr [out]	Pointer to assigned socket connection identifier.
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Sets the remote address of a suitable socket connection in a socket connection group.	
Particularities and Limitations	
- Considers the best match algorithm to select the socket connection.	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-26 SoAd_SetUniqueRemoteAddr

5.2.26 SoAd_TpChangeParameter

Prototype	
Std_ReturnType **SoAd_TpChangeParameter** (PduIdType id, TPParameterType parameter, uint16 value)	
Parameter	
id [in]	PDU identifier.
parameter [in]	Parameter type.
value [in]	Parameter value.
Return code	
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Changes socket connection specific parameter.	

Particularities and Limitations
- Has no functionality but is required by AUTOSAR.
Call context
> TASK|ISR2 > This function is Synchronous > This function is Reentrant

Table 5-27 SoAd_TpChangeParameter

5.2.27 SoAd_ReadDhcpHostNameOption

Prototype	
`Std_ReturnType` **`SoAd_ReadDhcpHostNameOption`** `(SoAd_SoConIdType SoConId, uint8 *length, uint8 *data)`	
Parameter	
SoConId [in]	Socket connection identifier.
length [in,out]	Length of buffer for hostname (length of provided buffer, updated to length of hostname).
data [out]	Pointer to buffer for hostname.
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Returns the DHCP hostname option currently configured on a local interface identified by a socket connection.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-28 SoAd_ReadDhcpHostNameOption

5.2.28 SoAd_WriteDhcpHostNameOption

Prototype
`Std_ReturnType` **`SoAd_WriteDhcpHostNameOption`** `(SoAd_SoConIdType SoConId, uint8 length, const uint8 *data)`

Parameter	
SoConId [in]	Socket connection identifier.
length [in]	Length of buffer for hostname.
data [in]	Pointer to buffer for hostname.
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Sets the DHCP hostname option on a local interface identified by a socket connection.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-29 SoAd_WriteDhcpHostNameOption

5.2.29 SoAd_GetVersionInfo

Prototype	
`void SoAd_GetVersionInfo (Std_VersionInfoType *versioninfo)`	
Parameter	
versioninfo [out]	Pointer to where to store the version information. Parameter must not be NULL.
Return code	
void	none
Functional Description	
Returns the version information.	
Particularities and Limitations	
- Returns version information, vendor ID and AUTOSAR module ID of the component. Configuration Variant(s): SOAD_VERSION_INFO_API	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-30 SoAd_GetVersionInfo

5.2.30 SoAd_MainFunctionRx

Prototype	
void **SoAd_MainFunctionRx** (void)	
Parameter	
void	none
Return code	
void	none
Functional Description	
Schedules the Socket Adaptor (Entry point for scheduling) and handles asynchronous reception.	
Particularities and Limitations	
- -	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-31 SoAd_MainFunctionRx

5.2.31 SoAd_MainFunctionState

Prototype	
void **SoAd_MainFunctionState** (void)	
Parameter	
void	none
Return code	
void	none
Functional Description	
Schedules the Socket Adaptor (Entry point for scheduling) and handles states.	
Particularities and Limitations	
- -	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-32 SoAd_MainFunctionState

5.2.32 SoAd_MainFunctionTx

Prototype	
`void SoAd_MainFunctionTx (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Schedules the Socket Adaptor (Entry point for scheduling) and handles asynchronous transmission.	
Particularities and Limitations	
- -	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-33 SoAd_MainFunctionTx

5.2.33 SoAd_MainFunction

Prototype	
`void SoAd_MainFunction (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Schedules the Socket Adaptor (Entry point for scheduling) and handles asynchronous reception and transmission and states.	
Particularities and Limitations	
- -	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-34 SoAd_MainFunction

5.3 Services used by Socket Adaptor

In the following table services provided by other components, which are used by the Socket Adaptor are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET	Det_ReportError
TcpIp	TcpIp_SoAdGetSocket
TcpIp	TcpIp_ChangeParameter
TcpIp	TcpIp_Bind
TcpIp	TcpIp_TcpListen
TcpIp	TcpIp_TcpConnect
TcpIp	TcpIp_TcpTransmit
TcpIp	TcpIp_UdpTransmit
TcpIp	TcpIp_TcpReceived
TcpIp	TcpIp_Close
TcpIp	TcpIp_RequestIpAddrAssignment
TcpIp	TcpIp_ReleaseIpAddrAssignment
TcpIp	TcpIp_GetIpAddr
TcpIp	TcpIp_GetCtrlIdx
TcpIp	TcpIp_GetRemotePhysAddr
TcpIp	TcpIp_DhcpReadOption
TcpIp	TcpIp_DhcpWriteOption
TcpIp	TcpIp_DhcpV6ReadOption
TcpIp	TcpIp_DhcpV6WriteOption
IpBase	IpBase_Copy
Linux/QNX/INTEGRITY	accept
Linux/QNX/INTEGRITY	bind
Linux/QNX/INTEGRITY	connect
Linux/QNX/INTEGRITY	fcntl
Linux/QNX/INTEGRITY	freeifaddrs
Linux/QNX/INTEGRITY	gethostname
Linux/QNX/INTEGRITY	getifaddrs
Linux/QNX/INTEGRITY	if_nametoindex
Linux/QNX/INTEGRITY	If_indextoname

Component	API
`Linux/QNX/INTEGRITY`	`listen`
`Linux/QNX/INTEGRITY`	`select`
`Linux/QNX`	`recv`
`Linux/QNX`	`recvfrom`
`Linux/QNX/INTEGRITY`	`send`
`Linux/QNX/INTEGRITY`	`sendto`
`Linux/QNX/INTEGRITY`	`sethostname`
`Linux/QNX/INTEGRITY`	`setsockopt`
`Linux/QNX/INTEGRITY`	`close`
`Linux/QNX/INTEGRITY`	`socket`
`Linux/INTEGRITY`	`ioctl`
`QNX/INTEGRITY`	`recvmsg`

Table 5-35 Services used by the Socket Adaptor

5.4 Callback Functions

This chapter describes the callback functions that are implemented by the Socket Adaptor and can be invoked by other modules. The prototypes of the callback functions are provided in the header file `SoAd_Cbk.h` by the Socket Adaptor.

5.4.1 SoAd_RxIndication

Prototype	
`void` **`SoAd_RxIndication`** `(SoAd_SocketIdType SocketId, SoAd_SockAddrType *RemoteAddrPtr, uint8 *BufPtr, uint16 Length)`	
Parameter	
SocketId [in]	Socket identifier.
RemoteAddrPtr [in]	Pointer to remote address.
BufPtr [in]	Pointer to buffer of received data.
Length [in]	Length of received data.
Return code	
void	none
Functional Description	
Receives data from sockets.	
Particularities and Limitations	
- -	

Call context
> TASK|ISR2 > This function is Synchronous > This function is Reentrant

Table 5-36 SoAd_RxIndication

5.4.2 SoAd_CopyTxData

Prototype	
`BufReq_ReturnType SoAd_CopyTxData (SoAd_SocketIdType SocketId, uint8 *BufPtr, uint16 *BufLengthPtr)`	
Parameter	
SocketId [in]	Socket identifier.
BufPtr [in]	Pointer to buffer of provided transmission buffer.
BufLength|BufLengthPtr [in,out]	Pointer to length|Length of provided transmission buffer.
Return code	
BufReq_ReturnType	BUFREQ_OK Copy request accepted.
BufReq_ReturnType	BUFREQ_E_NOT_OK Copy request not accepted.
Functional Description	
Copies data to provided transmission buffer.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-37 SoAd_CopyTxData

5.4.3 SoAd_TxConfirmation

Prototype	
`void SoAd_TxConfirmation (SoAd_SocketIdType SocketId, uint16 Length)`	
Parameter	
SocketId [in]	Socket identifier.
Length [in]	Length of confirmed data.

Return code	
void	none
Functional Description	
Confirms transmission of data.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-38 SoAd_TxConfirmation

5.4.4 SoAd_LocalIpAddrAssignmentChg

Prototype	
void **SoAd_LocalIpAddrAssignmentChg** (SoAd_LocalAddrIdType IpAddrId, SoAd_IpAddrStateType State)	
Parameter	
IpAddrId [in]	IP address identifier.
State [in]	State of IP address assignment.
Return code	
void	none
Functional Description	
Receives local IP address assignment state changes.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-39 SoAd_LocalIpAddrAssignmentChg

5.4.5 SoAd_TcpAccepted

Prototype	
Std_ReturnType **SoAd_TcpAccepted** (SoAd_SocketIdType SocketId, SoAd_SocketIdType	

`SocketIdConnected, SoAd_SockAddrType *RemoteAddrPtr)`	
Parameter	
SocketId [in]	Listen socket identifier.
SocketIdConnected [in]	Connected socket identifier.
RemoteAddrPtr [in]	Pointer to remote addres.
Return code	
Std_ReturnType	E_OK Connection was accepted.
Std_ReturnType	E_NOT_OK Connection was not accepted.
Functional Description	
Accepts TCP connections on a listen socket.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-40 SoAd_TcpAccepted

5.4.6 SoAd_TcpConnected

Prototype	
`void SoAd_TcpConnected (SoAd_SocketIdType SocketId)`	
Parameter	
SocketId [in]	Socket identifier.
Return code	
void	none
Functional Description	
Handles TCP connections which have been initiated locally and are now successfully connected.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-41 SoAd_TcpConnected

5.4.7 SoAd_TcpIpEvent

Prototype	
void **SoAd_TcpIpEvent** (SoAd_SocketIdType SocketId, SoAd_EventType Event)	
Parameter	
SocketId [in]	Socket identifier.
Event [in]	Event type. [TCPIP_TCP_RESET, TCPIP_TCP_CLOSED, TCPIP_TCP_FIN_RECEIVED, TCPIP_UDP_CLOSED]
Return code	
void	none
Functional Description	
Handles events on sockets.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-42 SoAd_TcpIpEvent

5.5 Configurable Interfaces

At its configurable interfaces the Socket Adaptor expects notification and callout functions which must be provided by the specific upper layer <Up> (e.g. PduR). The expected interface depends on configuration of each upper layer.

Availability, configuration dependencies and function prototypes are described in the following sub-chapters for each function.

5.5.1 <Up>_[SoAd][If]RxIndication

Prototype	
void **<Up>_[SoAd][If]RxIndication** (PduIdType RxPduId, const PduInfoType* PduInfoPtr)	
Parameter	
RxPduId [in]	Rx PDU identifier
PduInfoPtr [in]	Pointer to PDU
Return code	
void	none

Functional Description
Receives IF-PDU.
Particularities and Limitations
- -
Call context
> TASK|ISR2 > This function is Synchronous > This function is Reentrant

Table 5-43 <Up>_[SoAd][If]RxIndication

5.5.2 <Up>_[SoAd][If]TriggerTransmit

Prototype	
`Std_ReturnType` **`<Up>_[SoAd][If]TriggerTransmit`** `(PduIdType TxPduId, PduInfoType* PduInfoPtr)`	
Parameter	
TxPduId [in]	Tx PDU identifier
PduInfoPtr [in/out]	Pointer to PDU
Return code	
Std_ReturnType	E_OK Request was accepted.
Std_ReturnType	E_NOT_OK Request was not accepted.
Functional Description	
Copies data for a previously requested PDU via SoAd_IfTransmit if trigger transmit is used for the PDU.	
Particularities and Limitations	
- Available for upper layer with IF-API and enabled 'SoAdRoutingGroupTxTriggerable' for at least one routing group referenced in any 'SoAdPduRoute' or if 'SoAdTxIfTriggerTransmit' is enabled. PduInfoPtr->SduLength is set to provided buffer size and upper layer has to consider this value before copying to buffer.	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-44 <Up>_[SoAd][If]TriggerTransmit

5.5.3 <Up>_[SoAd][If]TxConfirmation

Prototype	
void **<Up>_[SoAd][If]TxConfirmation** (PduIdType TxPduId)	
Parameter	
TxPduId [in]	Tx PDU identifier
Return code	
void	none
Functional Description	
Confirms transmission of an IF-PDU.	
Particularities and Limitations	
- Available for upper layer with IF-API and ‘SoAdIfTxConfirmation’ in ‘SoAdBswModules’ is enabled.	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-45 <Up>_[SoAd][If]TxConfirmation

5.5.4 <Up>_[SoAd][Tp]StartOfReception

Prototype	
BufReq_ReturnType **<Up>_[SoAd][Tp]StartOfReception** (PduIdType RxPduId, PduInfoType* info, PduLengthType TpSduLength, PduLengthType* bufferSizePtr)	
Parameter	
RxPduId [in]	Rx PDU identifier
info [in]	No used [range: NULL_PTR]
TpSduLength [in]	Total length of PDU to be received
bufferSizePtr [out]	Available receive buffer
Return code	
BufReq_ReturnType	BUFREQ_OK Reception request was accepted.
BufReq_ReturnType	BUFREQ_E_NOT_OK Reception request was not accepted.
Functional Description	
Starts reception of a TP-PDU.	
Particularities and Limitations	
- ‘BUFREQ_E_OVFL’ as return value is treated like ‘BUFREQ_E_NOT_OK’. Parameter ‘info’ can be configured to constant pointer by enabling ‘SoAdTpStartOfReceptionWithConstPointer’.	

Call context
> TASK|ISR2 > This function is Reentrant

Table 5-46 <Up>_[SoAd][Tp]StartOfReception

5.5.5 <Up>_[SoAd][Tp]CopyRxData

Prototype	
BufReq_ReturnType **<Up>_[SoAd][Tp]CopyRxData** (PduIdType RxPduId, const PduInfoType* PduInfoPtr, PduLengthType* bufferSizePtr)	
Parameter	
RxPduId [in]	Rx PDU identifier
PduInfoPtr [in]	Pointer to PDU
bufferSizePtr [out]	Available receive buffer
Return code	
BufReq_ReturnType	BUFREQ_OK Copy request was accepted.
BufReq_ReturnType	BUFREQ_E_NOT_OK Copy request was not accepted.
Functional Description	
Copies received data of a TP-PDU.	
Particularities and Limitations	
- 'BUFREQ_E_OVFL' as return value is treated like 'BUFREQ_E_NOT_OK'. Parameter 'PduInfoPtr' can be configured to constant pointer by enabling 'SoAdTpCopyRxDataWithConstPointer'.	
Call context	
> TASK|ISR2 > This function is Reentrant	

Table 5-47 <Up>_[SoAd][Tp]CopyRxData

5.5.6 <Up>_[SoAd][Tp]RxIndication

Prototype	
void **<Up>_[SoAd][Tp]RxIndication** (PduIdType RxPduId, Std_ReturnType result)	
Parameter	
RxPduId [in]	Rx PDU identifier
result [in]	Reception result
Return code	
void	none

<table>
<tr><th>Functional Description</th></tr>
<tr><td>Indicates that a TP-PDU reception is finished.</td></tr>
<tr><th>Particularities and Limitations</th></tr>
<tr><td>-
-</td></tr>
<tr><th>Call context</th></tr>
<tr><td>> TASK|ISR2
> This function is Reentrant</td></tr>
</table>

Table 5-48 <Up>_[SoAd][Tp]RxIndication

5.5.7 <Up>_[SoAd][Tp]CopyTxData

<table>
<tr><th colspan="2">Prototype</th></tr>
<tr><td colspan="2">BufReq_ReturnType **<Up>_[SoAd][Tp]CopyTxData** (PduIdType TxPduId, const PduInfoType* PduInfoPtr, RetryInfoType* retry, PduLengthType* availableDataPtr)</td></tr>
<tr><th colspan="2">Parameter</th></tr>
<tr><td>TxPduId [in]</td><td>Tx PDU identifier</td></tr>
<tr><td>PduInfoPtr [in/out]</td><td>Pointer to PDU</td></tr>
<tr><td>retry [in]</td><td>Not used [range: NULL_PTR]</td></tr>
<tr><td>availableDataPtr [out]</td><td>Available transmission buffer</td></tr>
<tr><th colspan="2">Return code</th></tr>
<tr><td>BufReq_ReturnType</td><td>BUFREQ_OK Copy request was accepted.</td></tr>
<tr><td>BufReq_ReturnType</td><td>BUFREQ_E_NOT_OK Copy request was not accepted.</td></tr>
<tr><th colspan="2">Functional Description</th></tr>
<tr><td colspan="2">Copies data to be transmitted of a TP-PDU.</td></tr>
<tr><th colspan="2">Particularities and Limitations</th></tr>
<tr><td colspan="2">-
‘BUFREQ_E_OVFL’ as return value is treated like ‘BUFREQ_E_NOT_OK’.
Parameter ‘PduInfoPtr’ can be configured to constant pointer by enabling ‘SoAdTpCopyTxDataWithConstPointer’.</td></tr>
<tr><th colspan="2">Call context</th></tr>
<tr><td colspan="2">> TASK|ISR2
> This function is Reentrant</td></tr>
</table>

Table 5-49 <Up>_[SoAd][Tp]CopyTxData

5.5.8 <Up>_[SoAd][Tp]TxConfirmation

Prototype	
`void <Up>_[SoAd][Tp]TxConfirmation (PduIdType TxPduId, Std_ReturnType result)`	
Parameter	
TxPduId [in]	Tx PDU identifier
result [in]	Transmission result
Return code	
void	none
Functional Description	
Confirms transmission of a TP-PDU.	
Particularities and Limitations	
- -	
Call context	
> TASK|ISR2 > This function is Reentrant	

Table 5-50 <Up>_[SoAd][Tp]TxConfirmation

5.5.9 <Up>_SoConModeChg

Prototype	
`void <Up>_SoConModeChg (SoAd_SoConIdType SoConId, SoAd_SoConModeType Mode)`	
Parameter	
SoConId [in]	Socket connection identifier
Mode [in]	New socket connection mode
Return code	
void	None
Functional Description	
Notifies about a socket connection mode change	
Particularities and Limitations	
- Available if ‘SoAdBswModules/SoAdSoConModeChg’ is enabled for an upper layer or if a ‘SoAdGeneral/SoAdAdditionalSoConModeChgCallback’ is configured.	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-51 <Up>_SoConModeChg

5.5.10 <Up>_LocalIpAddrAssignmentChg

Prototype	
void **<Up>_LocalIpAddrAssignmentChg** (SoAd_SoConIdType SoConId, SoAd_IpAddrStateType State)	
Parameter	
SoConId [in]	Socket connection identifier
State [in]	New socket connection local IP address state
Return code	
void	None
Functional Description	
Notifies about IP assignment state for a specific socket connection.	
Particularities and Limitations	
- Available if ‘SoAdBswModules/SoAdLocalIpAddrAssigmentChg’ is enabled for an upper layer or if a ‘SoAdGeneral/SoAdAdditionalLocalIpAddrAssignmentChgCallback’ is configured.	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-52 <Up>_LocalIpAddrAssignmentChg

5.5.11 <Up>_ShutdownFinished

Prototype	
void **<Up>_ShutdownFinished** (void)	
Parameter	
void	none
Return code	
void	None
Functional Description	
Notifies about finished shutdown.	
Particularities and Limitations	
- > Available if ‘SoAdBswModules/SoAdShutdownFinishedCbk’ is enabled for an upper layer.	
Call context	
> TASK|ISR2 > This function is Synchronous	

> This function is Reentrant

Table 5-53 <Up>_ShutdownFinished

5.5.12 <Up_TcpTlsSocketCreatedNotification>

Prototype	
`void <Up_TcpTlsSocketCreatedNotification> (SoAd_SoConIdType SoConId, SoAd_SocketIdType SocketId)`	
Parameter	
SoConId [in]	Socket connection identifier
SocketId [in]	Socket identifier
Return code	
void	None
Functional Description	
Notifies about created TCP TLS socket.	
Particularities and Limitations	
- Available if ‘SoAdTcpTlsCallback/SoAdTcpTlsSocketCreatedNotification’ is configured.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 5-54 <Up_TcpTlsSocketCreatedNotification>

5.5.13 <Up_VerifyRxPdu>

Prototype	
`Std_ReturnType <Up_VerifyRxPdu> (const SoAd_SockAddrType * LocalAddrPtr, const SoAd_SockAddrType * RemoteAddrPtr, SoAd_PduHdrIdType PduHdrId, const PduInfoType * PduInfoPtr)`	
Parameter	
LocalAddrPtr [in]	Pointer to local socket address
RemoteAddrPtr [in]	Pointer to remote socket address
PduHdrId [in]	PDU Header ID
PduInfoPtr [in]	Pointer to PDU data [range: NULL_PTR if SoAdVerifyRxPduMaxDataLength == 0]
Return code	
Std_ReturnType	E_OK PDU reception verification succeeded
Std_ReturnType	E_NOT_OK PDU reception verification failed

Functional Description
Verifies a PDU reception and indicates if a reception shall be continued or dropped.
Particularities and Limitations
- Configurable in ‘SoAdVerifyRxPduCallback/SoAdVerifyRxPdu’
Call context
> TASK > This function is Synchronous > This function is Reentrant

Table 5-55 <Up_VerifyRxPdu>

6 Configuration

There is one configuration tool to configure and generate the Socket Adaptor:

- DaVinci Configurator Pro

6.1 Configuration Variants

The Socket Adaptor supports the configuration variants

- `VARIANT-PRE-COMPILE`
- `VARIANT-POST-BUILD-LOADABLE`

The configuration classes of the Socket Adaptor parameters depend on the supported configuration variants. For their definitions please see the SoAd_bswmd.arxml file.

6.2 Configuration with DaVinci Configurator Pro

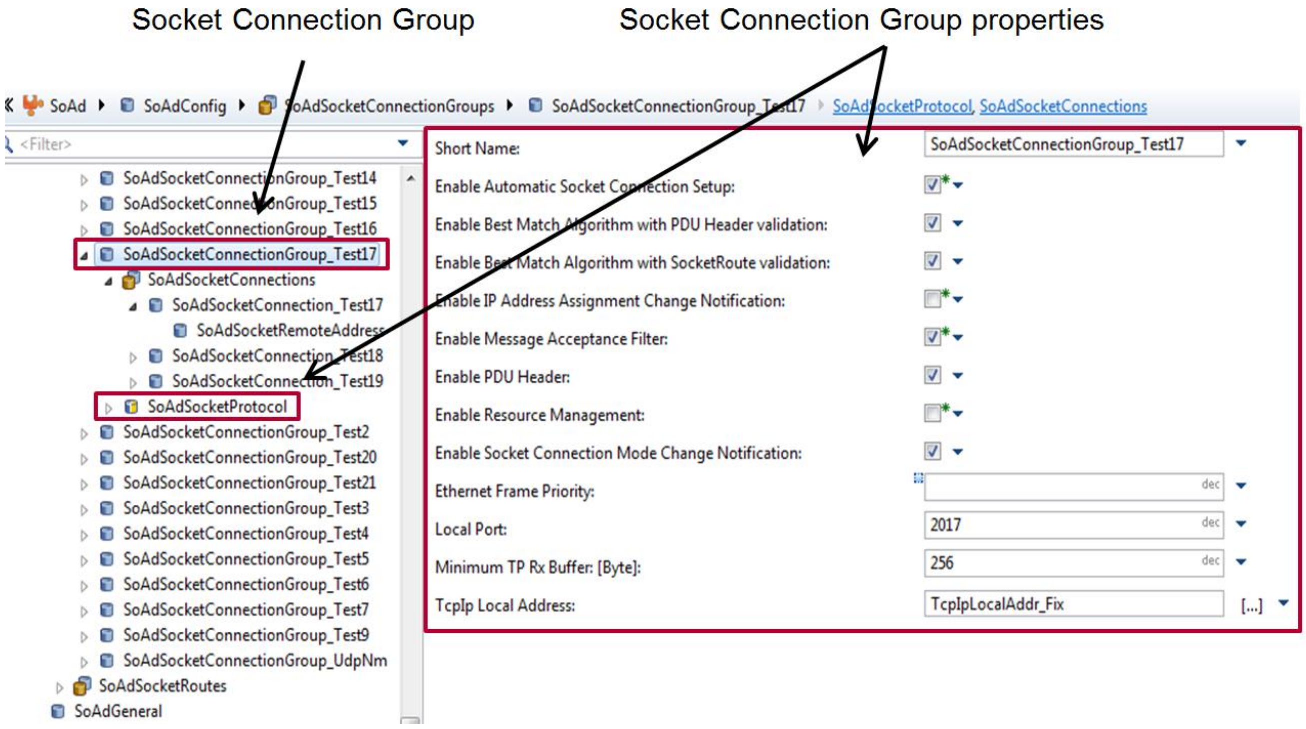
The Socket Adaptor is configured with the help of the configuration tool DaVinci Configurator Pro.

In the following sub-chapters some configuration hints are given to understand how to configure the Socket Adaptor correctly.

6.2.1 Socket Connection handling

6.2.1.1 Socket Connection Group

To support multiple communication partners on one local socket (e.g. multiple clients on one server) the Socket Adaptor can define multiple socket connections. The general difference between these socket connections is the remote address (address of communication partner). All common properties are defined in socket connection group configuration (refer to Figure 6-1 and Figure 6-2).



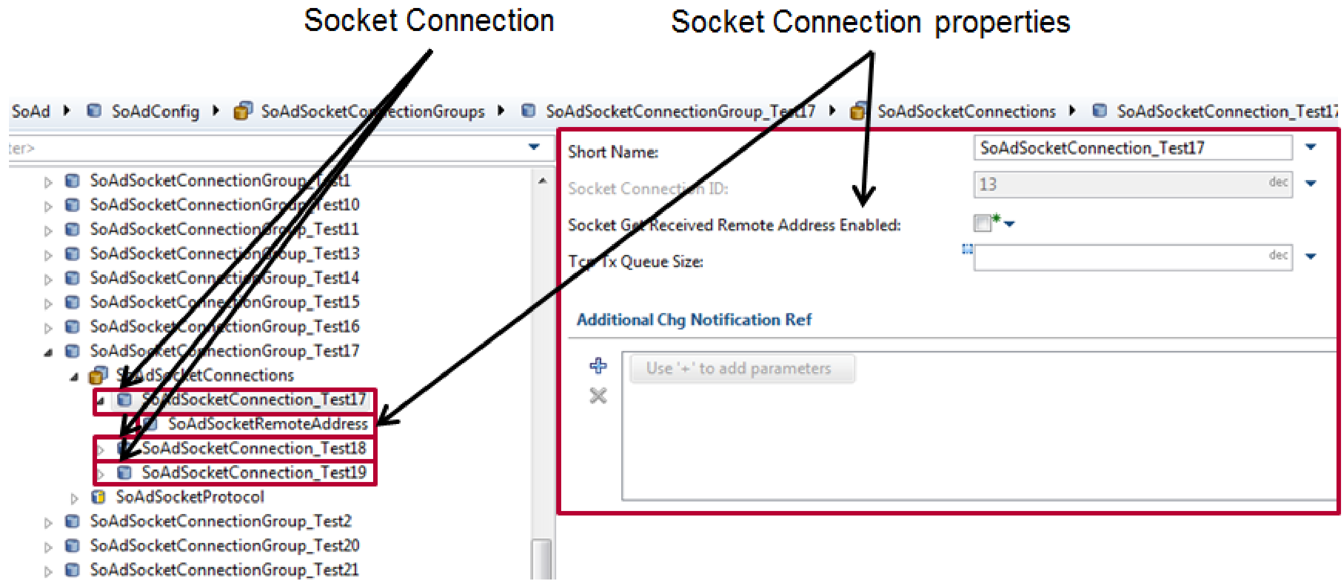
SoAd ▸ SoAdConfig ▸ SoAdSocketConnectionGroups ▸ SoAdSocketConnectionGroup_Test17 ▸ SoAdSocketProtocol, SoAdSocketConnections

<Filter>

- SoAdSocketConnectionGroup_Test14
- SoAdSocketConnectionGroup_Test15
- SoAdSocketConnectionGroup_Test16
- SoAdSocketConnectionGroup_Test17
  - SoAdSocketConnections
    - SoAdSocketConnection_Test17
      - SoAdSocketRemoteAddress
    - SoAdSocketConnection_Test18
    - SoAdSocketConnection_Test19
  - SoAdSocketProtocol
- SoAdSocketConnectionGroup_Test2
- SoAdSocketConnectionGroup_Test20
- SoAdSocketConnectionGroup_Test21
- SoAdSocketConnectionGroup_Test3
- SoAdSocketConnectionGroup_Test4
- SoAdSocketConnectionGroup_Test5
- SoAdSocketConnectionGroup_Test6
- SoAdSocketConnectionGroup_Test7
- SoAdSocketConnectionGroup_Test9
- SoAdSocketConnectionGroup_UdpNm
- SoAdSocketRoutes
- SoAdGeneral

Short Name:	SoAdSocketConnectionGroup_Test17
Enable Automatic Socket Connection Setup:	☑
Enable Best Match Algorithm with PDU Header validation:	☑
Enable Best Match Algorithm with SocketRoute validation:	☑
Enable IP Address Assignment Change Notification:	□
Enable Message Acceptance Filter:	☑
Enable PDU Header:	☑
Enable Resource Management:	□
Enable Socket Connection Mode Change Notification:	☑
Ethernet Frame Priority:	dec
Local Port:	2017 dec
Minimum TP Rx Buffer: [Byte]:	256 dec
TcpIp Local Address:	TcpIpLocalAddr_Fix [...]

Figure 6-1 Socket Connection Group configuration



SoAd ▸ SoAdConfig ▸ SoAdSocketConnectionGroups ▸ SoAdSocketConnectionGroup_Test17 ▸ SoAdSocketConnections ▸ SoAdSocketConnection_Test17

ter>

- SoAdSocketConnectionGroup_Test1
- SoAdSocketConnectionGroup_Test10
- SoAdSocketConnectionGroup_Test11
- SoAdSocketConnectionGroup_Test13
- SoAdSocketConnectionGroup_Test14
- SoAdSocketConnectionGroup_Test15
- SoAdSocketConnectionGroup_Test16
- SoAdSocketConnectionGroup_Test17
  - SoAdSocketConnections
    - SoAdSocketConnection_Test17
      - SoAdSocketRemoteAddress
    - SoAdSocketConnection_Test18
    - SoAdSocketConnection_Test19
  - SoAdSocketProtocol
- SoAdSocketConnectionGroup_Test2
- SoAdSocketConnectionGroup_Test20
- SoAdSocketConnectionGroup_Test21

Short Name:	SoAdSocketConnection_Test17
Socket Connection ID:	13 dec
Socket Get Received Remote Address Enabled:	□
Tcp Tx Queue Size:	dec

Additional Chg Notification Ref

Use '+' to add parameters

Figure 6-2 Socket Connection configuration

In case of TCP and special cases of UDP buffers must be configured in the TcpIp module. The tool provides solving actions to add and optimize buffers.

6.2.1.2 Socket Connection establishment

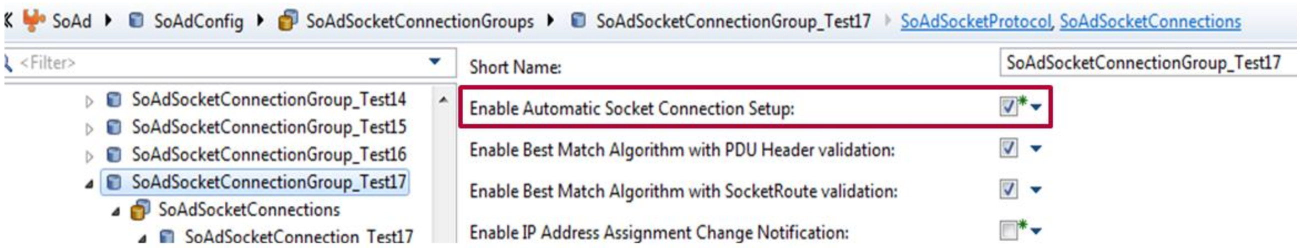
There are basically two different ways how to handle socket connections. Depending on the use case following ways can be chosen:

- Manual
- Automatic

If an upper layer needs to control the socket connection state or remote address a manual socket connection establishment is recommended (e.g. DoIP). In all other cases Socket Adaptor can handle the socket connection on itself (e.g. PduR).

6.2.1.2.1 Manual

Manual socket connection establishment is enabled if the corresponding parameter is disabled (Figure 6-3).



SoAd ▸ SoAdConfig ▸ SoAdSocketConnectionGroups ▸ SoAdSocketConnectionGroup_Test17 ▸ SoAdSocketProtocol, SoAdSocketConnections

<Filter>

- SoAdSocketConnectionGroup_Test14
- SoAdSocketConnectionGroup_Test15
- SoAdSocketConnectionGroup_Test16
- SoAdSocketConnectionGroup_Test17
  - SoAdSocketConnections
    - SoAdSocketConnection_Test17

Short Name: SoAdSocketConnectionGroup_Test17

Enable Automatic Socket Connection Setup: ☑

Enable Best Match Algorithm with PDU Header validation: ☑

Enable Best Match Algorithm with SocketRoute validation: ☑

Enable IP Address Assignment Change Notification: □

Figure 6-3 Socket Connection Setup parameter

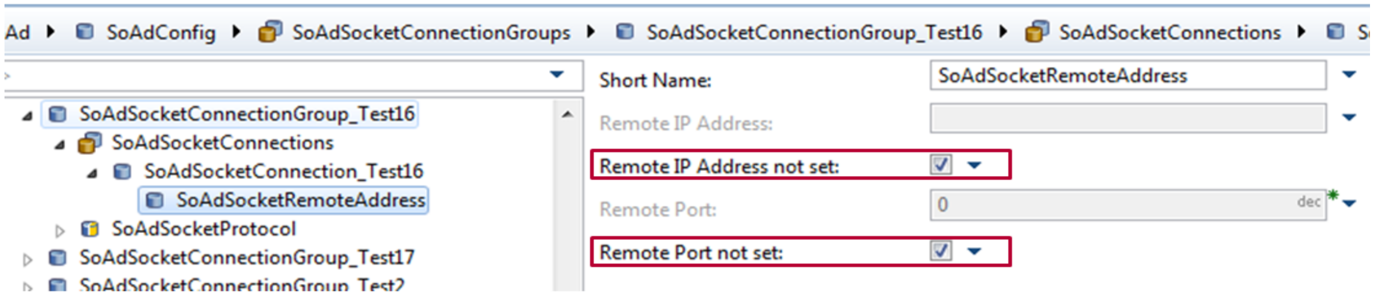
Following services are now available:

- SoAd_OpenSoCon()
- SoAd_CloseSoCon()
- SoAd_SetRemoteAddr()
- SoAd_SetUniqueRemoteAddr()

These services can be used to open and close a socket connection or set the remote address.

In case of UDP the socket connection can be opened on itself if the message acceptance filter is enabled in socket connection group and the remote address is set or wildcard.

To set a port to wildcard set port to 0. To configure an IP address to wildcard empty the remote IP address field. Vector supports a special value for a not set IP address or port (Figure 6-4). If an IP address or port is not set, a socket connection will not be opened on itself.



Ad ▸ SoAdConfig ▸ SoAdSocketConnectionGroups ▸ SoAdSocketConnectionGroup_Test16 ▸ SoAdSocketConnections ▸ S

- SoAdSocketConnectionGroup_Test16
  - SoAdSocketConnections
    - SoAdSocketConnection_Test16
      - SoAdSocketRemoteAddress
  - SoAdSocketProtocol
- SoAdSocketConnectionGroup_Test17
- SoAdSocketConnectionGroup_Test2

Short Name: SoAdSocketRemoteAddress

Remote IP Address:

Remote IP Address not set: ☑

Remote Port: 0 dec

Remote Port not set: ☑

Figure 6-4 Remote address “not set” parameter

6.2.1.2.2 Automatic

Automatic socket connection establishment is enabled if the corresponding parameter is enabled (Figure 6-3).

All services described in chapter 6.2.1.2.1 are not available and corresponding functions will return E_NOT_OK on call.

A socket connection is automatically opened if the remote address is set in configuration or on reception in case of UDP and the message acceptance filter is enabled in socket connection group.

6.2.2 Transmission path

This chapter gives a short description about how configure a transmission path.

1. Configure a socket connection according to the use case (chapter 6.2.1)
2. Add a PDU Route and choose referenced PDU and upper layer API type (Figure 6-5)

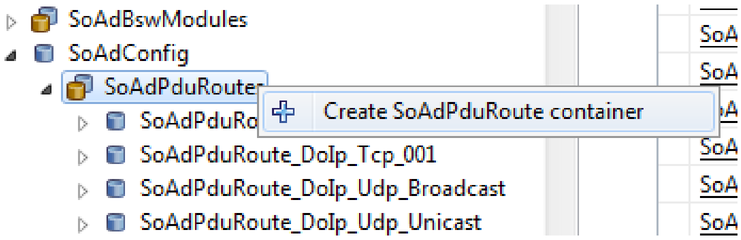

SoAdBswModules
SoAdConfig
SoAdPduRoute
Create SoAdPduRoute container
SoAdPduRo
SoAdPduRoute_DoIp_Tcp_001
SoAdPduRoute_DoIp_Udp_Broadcast
SoAdPduRoute_DoIp_Udp_Unicast
SoA
SoA
SoA
SoA
SoA

Figure 6-5 Add PDU Route

3. Add one or multiple (fan-out) PDU Route Destination (Figure 6-6) and reference the socket connection created in 1

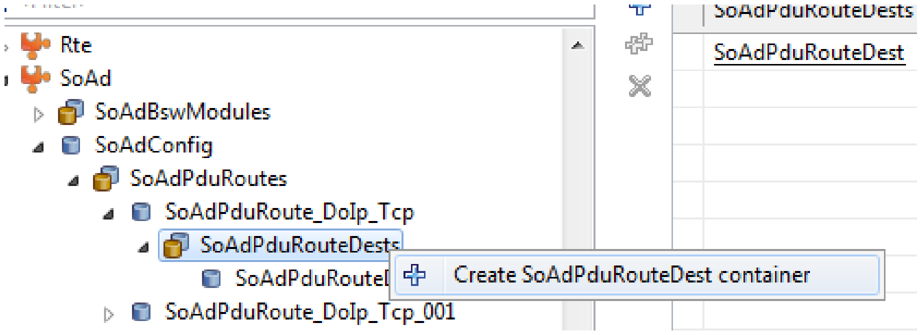


Figure 6-6 Add PDU Route Destination

6.2.3 Reception path

1. Configure a socket connection according to the use case (chapter 6.2.1)
2. Add a Socket Route and reference socket connection created in 1(Figure 6-7)

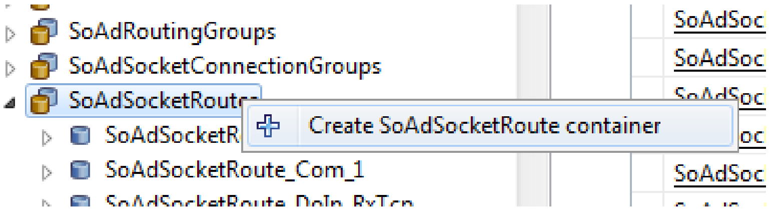


Figure 6-7 Add Socket Route

3. Add exactly one (fan-out not possible) Socket Route Destination and chose the upper layer PDU(Figure 6-8)

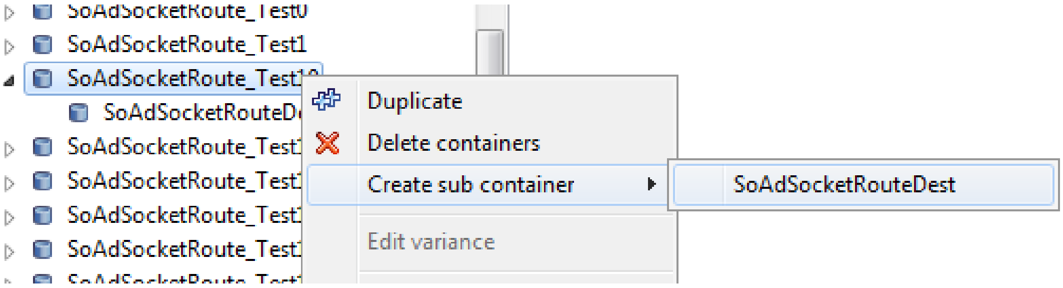


Figure 6-8 Add Socket Route Destination

6.2.4 Vector specific feature

6.2.4.1 Best Match Algorithm extensions

Best Match Algorithm according to [1] chooses a socket connection of a socket connection group considering remote addresses. In some cases it may be useful to consider more than remote address. Vector implements extensions of the Best Match Algorithm described here:

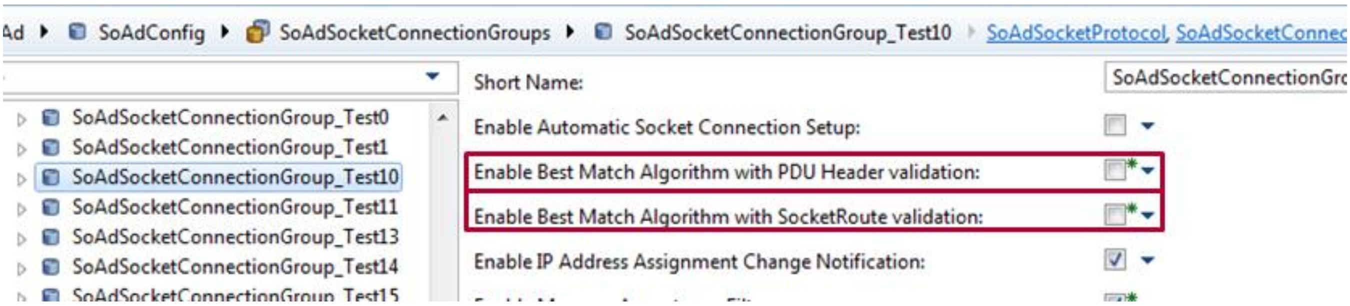
- Best Match Algorithm with PDU Header validation

  If this option is enabled PDU Header is extracted before socket connection is chosen. After PDU Header is received completely Best Match Algorithm is used to find the suitable socket connection. If the socket connection is found an additional check is performed to verify that the received PDU Header ID matches the configured value. In case of no match Best Match Algorithm will continue. If no socket connection and no socket route could be found the received PDU will be discarded.

- Best Match Algorithm with Socket Route validation

  This extension considers the existence of a socket route at a socket connection. If no socket route is configured for a socket connection the Best Match Algorithm will continue. This option can be used to prevent a socket connection setup on reception of data while other reception socket connections are used in same socket connection group.

Both described feature can be configured in the socket connection group (refer to Figure 6-9).


Ad ▸ SoAdConfig ▸ SoAdSocketConnectionGroups ▸ SoAdSocketConnectionGroup_Test10 ▸ SoAdSocketProtocol, SoAdSocketConnec
SoAdSocketConnectionGroup_Test0
SoAdSocketConnectionGroup_Test1
SoAdSocketConnectionGroup_Test10
SoAdSocketConnectionGroup_Test11
SoAdSocketConnectionGroup_Test13
SoAdSocketConnectionGroup_Test14
SoAdSocketConnectionGroup_Test15

Short Name: SoAdSocketConnectionGro

Enable Automatic Socket Connection Setup: □

Enable Best Match Algorithm with PDU Header validation: □

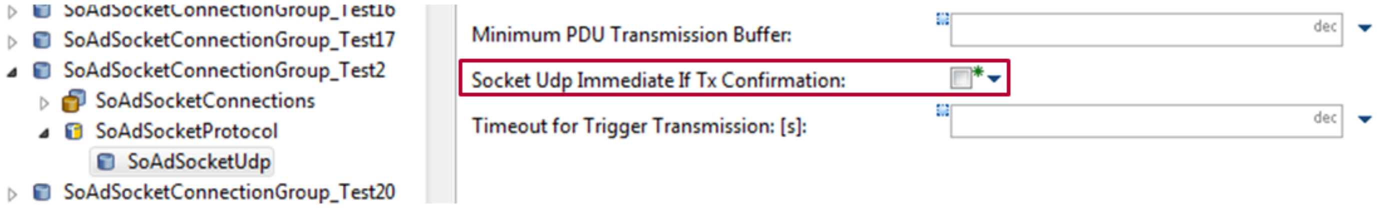
Enable Best Match Algorithm with SocketRoute validation: □

Enable IP Address Assignment Change Notification: ☑

Figure 6-9 Best Match Algorithm extensions

6.2.4.2 UDP Immediate IF TxConfirmaion

This feature is used to confirm data within interrupt level after successfully sending data. How to configure this feature is described in Figure 6-10.



SoAdSocketConnectionGroup_Test16
SoAdSocketConnectionGroup_Test17
SoAdSocketConnectionGroup_Test2
SoAdSocketConnections
SoAdSocketProtocol
SoAdSocketUdp
SoAdSocketConnectionGroup_Test20

Minimum PDU Transmission Buffer: dec
Socket Udp Immediate If Tx Confirmation: □
Timeout for Trigger Transmission: [s]: dec

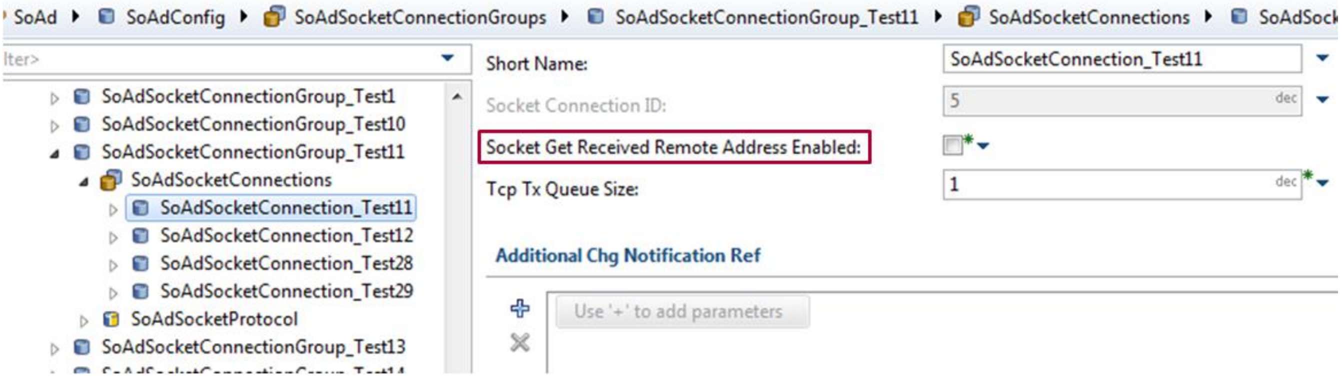
Figure 6-10 UDP Immediate IF TxConfirmation

According to [1] TxConfirmation for UDP socket connections with IF API is called within main function. It may be possible that new data are received while sent data are not confirmed yet. This may lead to incorrect behavior if upper layer is based on request and response behavior. Immediate TxConfirmaion can prevent reception before confirmation.

This feature is implemented for one PduRoute/PduRouteDestination per socket connection only.

6.2.4.3 SoAd_GetRcvRemoteAddr()

Via this API it is possible for the upper layer to retrieve the remote address of the last received message on a socket connection. To enable this feature enable the parameter marked in Figure 6-11.



SoAd ▸ SoAdConfig ▸ SoAdSocketConnectionGroups ▸ SoAdSocketConnectionGroup_Test11 ▸ SoAdSocketConnections ▸ SoAdSock

SoAdSocketConnectionGroup_Test1
SoAdSocketConnectionGroup_Test10
SoAdSocketConnectionGroup_Test11
SoAdSocketConnections
SoAdSocketConnection_Test11
SoAdSocketConnection_Test12
SoAdSocketConnection_Test28
SoAdSocketConnection_Test29
SoAdSocketProtocol
SoAdSocketConnectionGroup_Test13

Short Name: SoAdSocketConnection_Test11
Socket Connection ID: 5
Socket Get Received Remote Address Enabled: □
Tcp Tx Queue Size: 1

Additional Chg Notification Ref

Use '+' to add parameters

Figure 6-11 SoAd_GetRcvRemoteAddr

6.2.4.4 Additional SoConModeChg notification

It is possible for Socket Adaptor upper layers to get a socket connection mode change notification on a socket connection which is not referenced by a `SoAdPduRoute` or `SoAdSocketRouteDestination` to this upper layer.

To configure this additional notification add a reference to corresponding `SoAdBswModule` as described in Figure 6-12 on the `SoAdSocketConnection`.

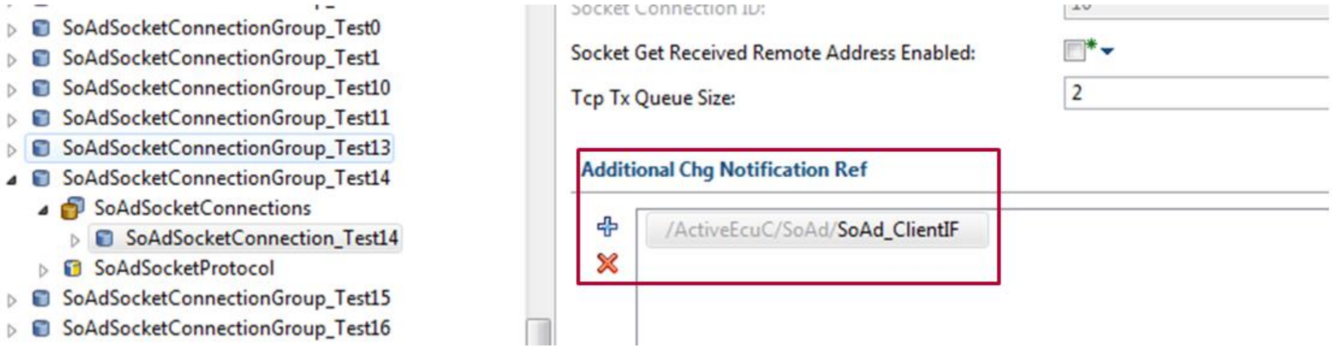

SoAdSocketConnectionGroup_Test0
SoAdSocketConnectionGroup_Test1
SoAdSocketConnectionGroup_Test10
SoAdSocketConnectionGroup_Test11
SoAdSocketConnectionGroup_Test13
SoAdSocketConnectionGroup_Test14
SoAdSocketConnections
SoAdSocketConnection_Test14
SoAdSocketProtocol
SoAdSocketConnectionGroup_Test15
SoAdSocketConnectionGroup_Test16
Socket Connection ID:
Socket Get Received Remote Address Enabled:
Tcp Tx Queue Size:
2
Additional Chg Notification Ref
/ActiveEcuC/SoAd/SoAd_ClientIF

Figure 6-12 Additional SoConModeChg notification

6.2.4.5 BSD Socket API

If this option is part of the delivery Socket Adaptor can be used with BSD Socket API of Linux, QNX or INTEGRITY. In case feature is enabled AUTOSAR TcpIp stack cannot be used anymore.

The services used by Socket Adaptor in case of BSD Socket API are listed in Table 5-35.

In order to configure the Socket Adaptor for BSD Socket API a “dummy” TcpIp module has to be configured (i.e. has different description file which is activated as described in chapter 6.2.4.5.1). The TcpIp module is used to configure address relevant information used by Socket Adaptor to bind sockets on BSD Socket API.

6.2.4.5.1 Enable/Disable BSD Socket API

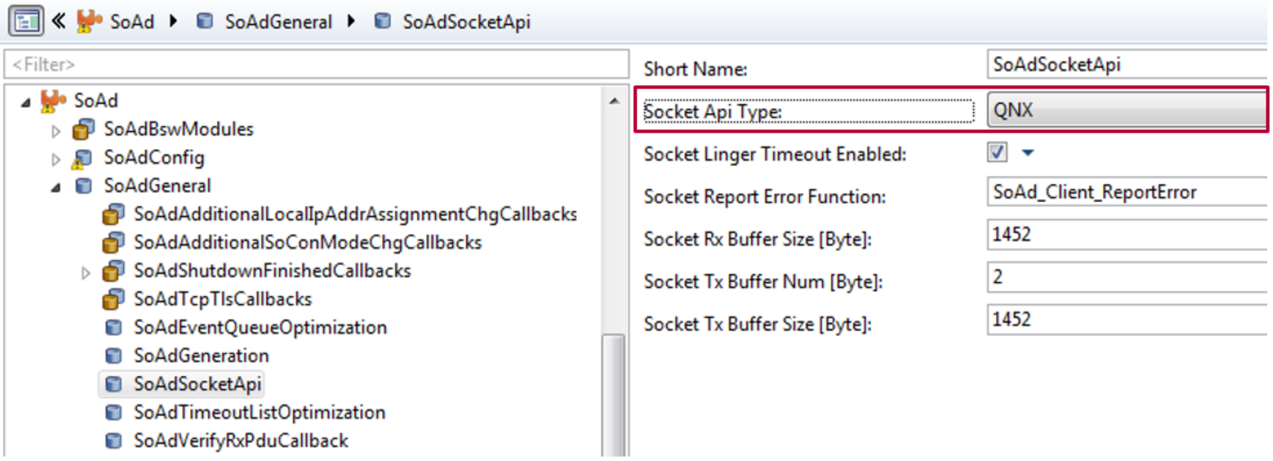
To enable this feature following steps are required. It depends on the delivery whether the following files are available. If a file is not available modification can be skipped.

1. Rename “TcpIp_bswmd.arxml” “TcpIp_bswmd.arxml_”.
   The file can be found in “…\external\BSWMD\TcpIp”
2. Rename “Tp_AsrTpTcpIp.jar” “Tp_AsrTpTcpIp.jar_”,
   rename “Tp_AsrTpTcpIp_DhcpV4Server.jar” “Tp_AsrTpTcpIp_DhcpV4Server.jar_”,
   rename “Tp_AsrTpTcpIp_IpV4.jar” “Tp_AsrTpTcpIp_IpV4.jar_” and
   rename “Tp_AsrTpTcpIp_IpV6.jar” “Tp_AsrTpTcpIp_IpV6.jar_”
   These files can be found in “…\external\DaVinciConfigurator\Generators\TcpIp”
3. Rename “TcpIp_BSD_bswmd.arxml_” “TcpIp_BSD_bswmd.arxml”
   The file can be found in “…\external\BSWMD\SoAd”
4. After starting DaVinci Configurator Pro the feature is enabled automatically by the tool. Please remove (save configuration by export before) all Ethernet modules in your configuration (“Eth”, “EthTrcv”, “EthIf”, “EthSM”)

To disable the feature again revert the steps mentioned above (e.g. import exported Ethernet modules).

6.2.4.5.2 Configure Socket API type

Besides enabling or disabling the BSD Socket API as described above the corresponding Socket API type has to be chosen as described in Figure 6-13.

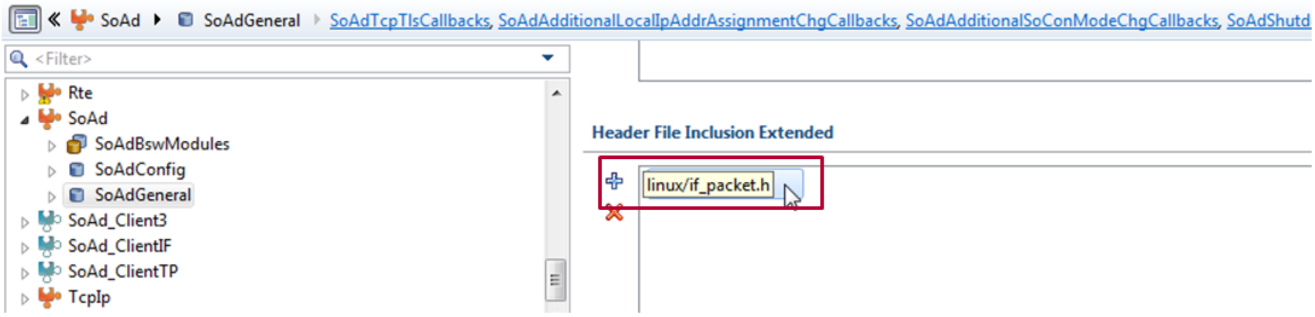

« SoAd ▸ SoAdGeneral ▸ SoAdSocketApi
<Filter>
SoAd
SoAdBswModules
SoAdConfig
SoAdGeneral
SoAdAdditionalLocalIpAddrAssignmentChgCallbacks
SoAdAdditionalSoConModeChgCallbacks
SoAdShutdownFinishedCallbacks
SoAdTcpTlsCallbacks
SoAdEventQueueOptimization
SoAdGeneration
SoAdSocketApi
SoAdTimeoutListOptimization
SoAdVerifyRxPduCallback

Short Name: SoAdSocketApi
Socket Api Type: QNX
Socket Linger Timeout Enabled: ☑
Socket Report Error Function: SoAd_Client_ReportError
Socket Rx Buffer Size [Byte]: 1452
Socket Tx Buffer Num [Byte]: 2
Socket Tx Buffer Size [Byte]: 1452

Figure 6-13 Socket API type configuration

6.2.4.5.3 Configure header file for type “sockaddr_ll”

Depending on the BSD Socket API distribution, type `sockaddr_ll` may not be defined within the already included header files. In this case add the corresponding header file via configuration as described in Figure 6-14.

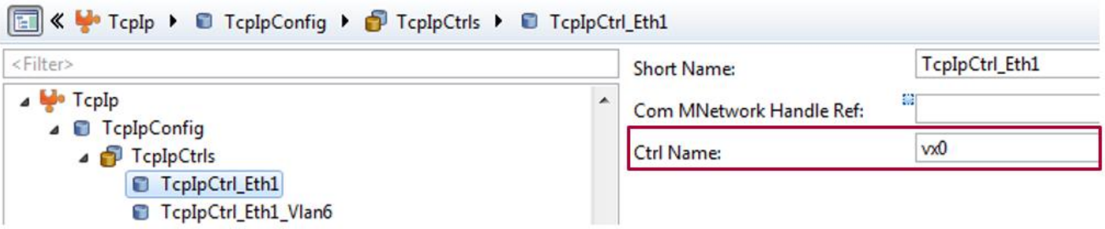

« SoAd ▸ SoAdGeneral ▸ SoAdTcpTlsCallbacks, SoAdAdditionalLocalIpAddrAssignmentChgCallbacks, SoAdAdditionalSoConModeChgCallbacks, SoAdShutd
<Filter>
Rte
SoAd
SoAdBswModules
SoAdConfig
SoAdGeneral
SoAd_Client3
SoAd_ClientIF
SoAd_ClientTP
TcpIp
Header File Inclusion Extended

linux/if_packet.h

Figure 6-14 Configure header file for type “sockaddr_ll”

6.2.4.5.4 Configure local IP address

Each local IP address belongs to a network interface. Each network interface can have multiple local IP addresses. The network interfaces and their names are configured in a configuration container as described in Figure 6-15. Parameter “Ctrl Name” has to match the name configured on the interface in Linux/QNX/INTEGRITY.

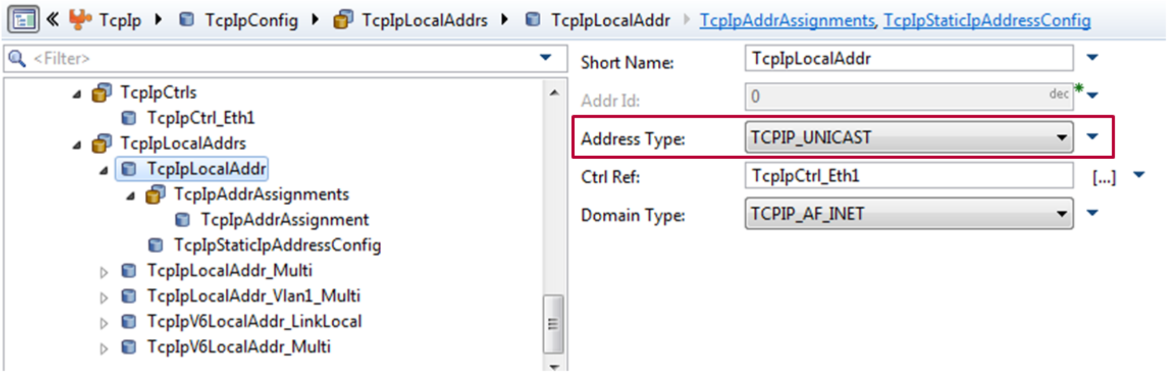


Short Name: TcpIpCtrl_Eth1
Com MNetwork Handle Ref:
Ctrl Name: vx0

Figure 6-15 Controller/Interface configuration

For a local address it has to be decided if it is used for (Figure 6-16):

1. Unicast (transmission and reception)
2. Multicast (reception only)



« TcpIp ▸ TcpIpConfig ▸ TcpIpLocalAddrs ▸ TcpIpLocalAddr ▸ TcpIpAddrAssignments, TcpIpStaticIpAddressConfig

<Filter>

- TcpIpCtrls
  - TcpIpCtrl_Eth1
- TcpIpLocalAddrs
  - TcpIpLocalAddr
    - TcpIpAddrAssignments
      - TcpIpAddrAssignment
    - TcpIpStaticIpAddressConfig
  - TcpIpLocalAddr_Multi
  - TcpIpLocalAddr_Vlan1_Multi
  - TcpIpV6LocalAddr_LinkLocal
  - TcpIpV6LocalAddr_Multi

Short Name: TcpIpLocalAddr

Addr Id: 0 dec

Address Type: TCPIP_UNICAST

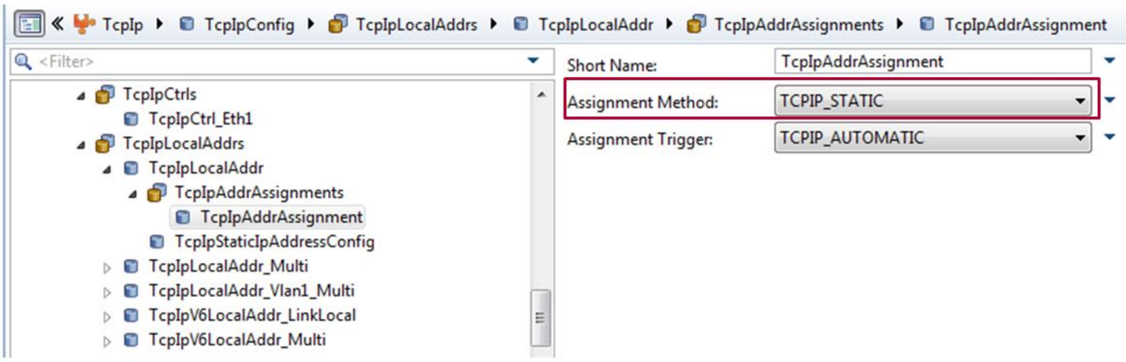
Ctrl Ref: TcpIpCtrl_Eth1 [...]

Domain Type: TCPIP_AF_INET

Figure 6-16 Address Type configuration

Afterwards it has to be decided which assignment method shall be used (Figure 6-17):

1. Static
2. DHCP



« TcpIp ▸ TcpIpConfig ▸ TcpIpLocalAddrs ▸ TcpIpLocalAddr ▸ TcpIpAddrAssignments ▸ TcpIpAddrAssignment

<Filter>

- TcpIpCtrls
  - TcpIpCtrl_Eth1
- TcpIpLocalAddrs
  - TcpIpLocalAddr
    - TcpIpAddrAssignments
      - TcpIpAddrAssignment
    - TcpIpStaticIpAddressConfig
  - TcpIpLocalAddr_Multi
  - TcpIpLocalAddr_Vlan1_Multi
  - TcpIpV6LocalAddr_LinkLocal
  - TcpIpV6LocalAddr_Multi

Short Name: TcpIpAddrAssignment

Assignment Method: TCPIP_STATIC

Assignment Trigger: TCPIP_AUTOMATIC

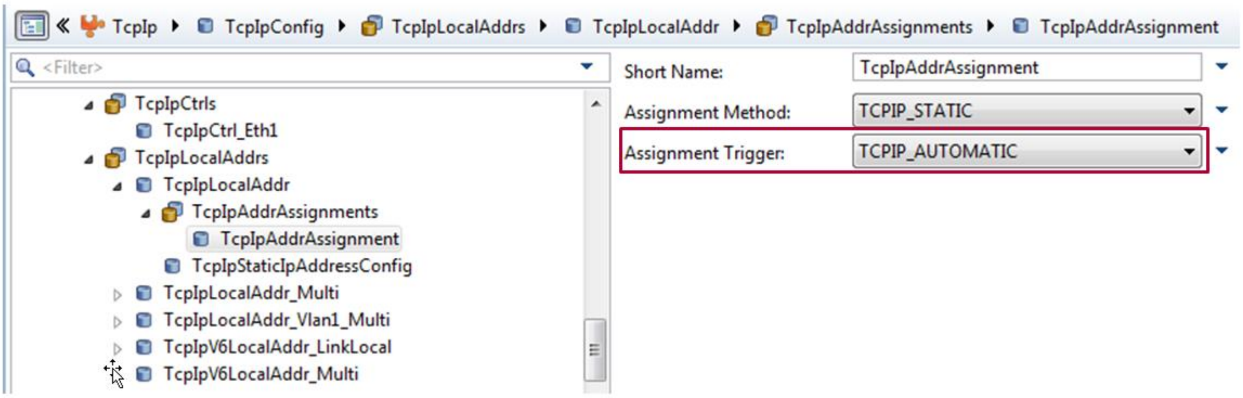
Figure 6-17 Assignment Method configuration

“Static” means that Socket Adaptor will bind corresponding sockets to this address. If the IP address is not available on the configured interface no communication is possible and Socket Adaptor will try to bind the socket in next main function again until IP address is available or IP address request has been released.

In case of “DHCP” Socket Adaptor will bind all affected sockets to the first IP address of the configured interface (except loopback address). This assignment method is valid for unicast addresses only.

For each local address an assignment trigger can be chosen (Figure 6-18):

1. Automatic
2. Manual


« TcpIp ▸ TcpIpConfig ▸ TcpIpLocalAddrs ▸ TcpIpLocalAddr ▸ TcpIpAddrAssignments ▸ TcpIpAddrAssignment
<Filter>
TcpIpCtrls
TcpIpCtrl_Eth1
TcpIpLocalAddrs
TcpIpLocalAddr
TcpIpAddrAssignments
TcpIpAddrAssignment
TcpIpStaticIpAddressConfig
TcpIpLocalAddr_Multi
TcpIpLocalAddr_Vlan1_Multi
TcpIpV6LocalAddr_LinkLocal
TcpIpV6LocalAddr_Multi

Short Name: TcpIpAddrAssignment

Assignment Method: TCPIP_STATIC

Assignment Trigger: TCPIP_AUTOMATIC

Figure 6-18 Assignment Trigger configuration

“Automatic” means that Socket Adaptor tries to assign the IP address immediately after startup (e.g. take the first IP address of interface in case of DHCP). If IP address gets lost, Socket Adaptor will try to assign this IP address again in each main function cycle until IP address is available again. In case of “DHCP” this behavior is used to handle a change of the IP address assigned by DHCP server.

For “manual” local addresses a call to `SoAd_RequestIpAddrAssignment()` is required to trigger assignment of IP address at runtime.

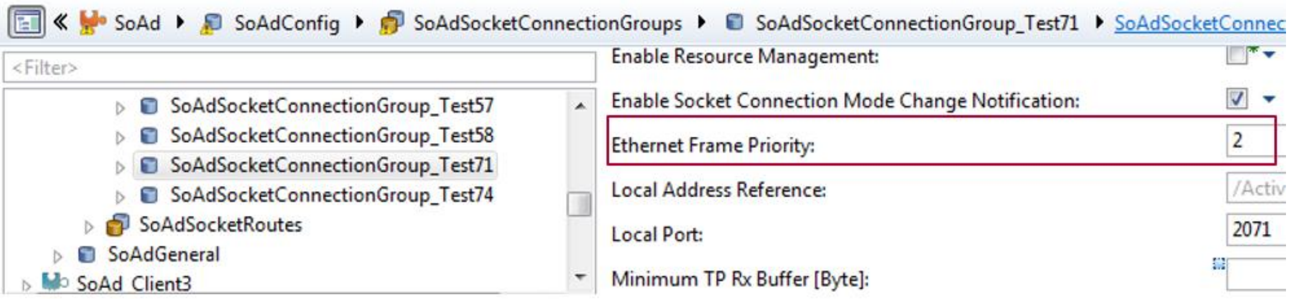
6.2.4.5.5 Request and release IP address

`SoAd_RequestIpAddrAssignment()` and `SoAd_ReleaseIpAddrAssignment()` do not support all cases which are described in AUTOSAR. As described in chapter before “static” and “DHCP” are supported only. For all other assignment methods `E_NOT_OK` is returned.

6.2.4.5.6 VLAN priority

For Linux Socket Adaptor supports to set the frame priority on sockets. The Socket Adaptor does not support this for other BSD Socket API distributions.

As defined for the AUTOSAR TcpIp stack the frame priority can be set as described in Figure 6-19.


« SoAd ▸ SoAdConfig ▸ SoAdSocketConnectionGroups ▸ SoAdSocketConnectionGroup_Test71 ▸ SoAdSocketConnec
<Filter>
SoAdSocketConnectionGroup_Test57
SoAdSocketConnectionGroup_Test58
SoAdSocketConnectionGroup_Test71
SoAdSocketConnectionGroup_Test74
SoAdSocketRoutes
SoAdGeneral
SoAd_Client3

Enable Resource Management: □

Enable Socket Connection Mode Change Notification: ☑

Ethernet Frame Priority: 2

Local Address Reference: /Activ

Local Port: 2071

Minimum TP Rx Buffer [Byte]:

Figure 6-19 Frame Priority configuration

To set the frame priority may be not sufficient since the frame priority set on a socket is handled in the queues of the Linux stack internally. To add the VLAN priority to the frame on the bus the corresponding egress map on the interface must be set.

This can be done manually (e.g. via script running at startup) or via the Socket Adaptor.

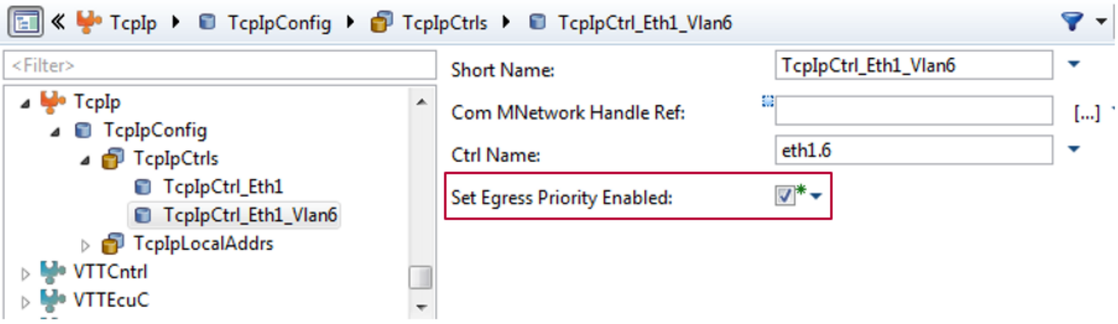
To do this manually you may use “vconfig” if available. Refer to the example below which sets the egress map so that a socket priority set to “7” is sent with VLAN priority “7”:

```
vconfig set_egress_map eth1.6 7 7
```

If a console is available you may use the “cat” command to display the ingress/egress map:

```
cat /proc/net/vlan/eth1.6
```

To let the Socket Adaptor set the egress map enable the parameter described in Figure 6-20.



Short Name: TcpIpCtrl_Eth1_Vlan6
Com MNetwork Handle Ref: [...]
Ctrl Name: eth1.6
Set Egress Priority Enabled: ☑

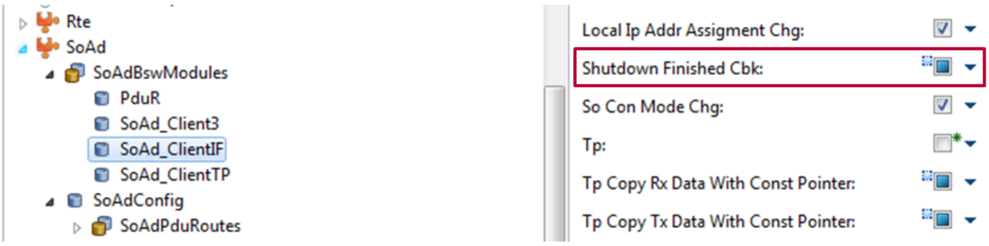
Figure 6-20 Set Egress Priority configuration

6.2.4.6 Shutdown mechanism

Service `SoAd_Shutdown()` initiates a shutdown of Socket Adaptor module. Pending transmissions will be continued, transmission and reception will be disabled and Socket Adaptor is set into a shutdown state afterwards. A call to `SoAd_Init()` is required to reinitialize Socket Adaptor again.

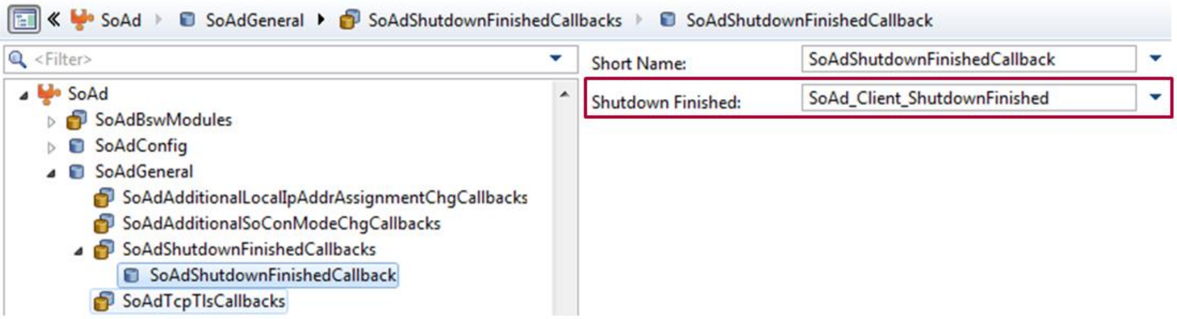
This feature is enabled always and can be used to shutdown Socket Adaptor orderly (e.g. in case of flashing the ECU).

It is possible to configure an optional callback which is called when shutdown is finished. This callback can be configured as described in Figure 6-21 for a specific upper layer or in Figure 6-22 for any other module (please consider Figure 6-23 to include additional header files).



Local Ip Addr Assigment Chg: ☑
Shutdown Finished Cbk: ■
So Con Mode Chg: ☑
Tp: □
Tp Copy Rx Data With Const Pointer: ■
Tp Copy Tx Data With Const Pointer: ■

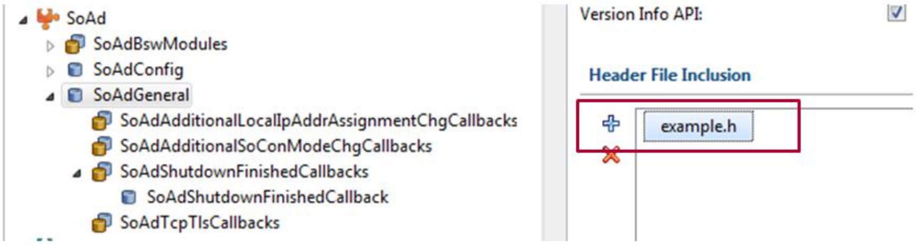
Figure 6-21 Enable upper layer specific <User>_ShutdownFinished()


« SoAd ▸ SoAdGeneral ▸ SoAdShutdownFinishedCallbacks ▸ SoAdShutdownFinishedCallback
<Filter>
SoAd
SoAdBswModules
SoAdConfig
SoAdGeneral
SoAdAdditionalLocalIpAddrAssignmentChgCallbacks
SoAdAdditionalSoConModeChgCallbacks
SoAdShutdownFinishedCallbacks
SoAdShutdownFinishedCallback
SoAdTcpTlsCallbacks

Short Name: SoAdShutdownFinishedCallback

Shutdown Finished: SoAd_Client_ShutdownFinished

Figure 6-22 Configure <User>_ShutdownFinished() for any module


SoAd
SoAdBswModules
SoAdConfig
SoAdGeneral
SoAdAdditionalLocalIpAddrAssignmentChgCallbacks
SoAdAdditionalSoConModeChgCallbacks
SoAdShutdownFinishedCallbacks
SoAdShutdownFinishedCallback
SoAdTcpTlsCallbacks

Version Info API: ☑

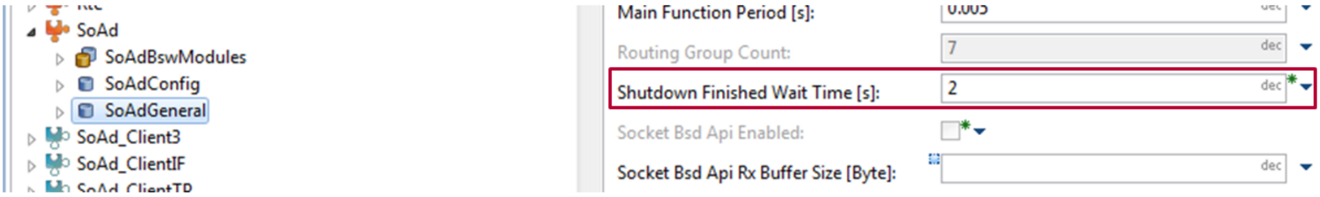
Header File Inclusion

example.h

Figure 6-23 Additional header file inclusion

It is also possible to poll module state via multiple calls to `SoAd_Shutdown()`.

If pending transmission cannot be finished or sockets cannot be closed since tester does not acknowledge closing (i.e. no “FIN” flag sent) a timeout is configured to shutdown module immediately after timeout (Figure 6-24) expired.



Main Function Period [s]: 0.005

Routing Group Count: 7

Shutdown Finished Wait Time [s]: 2

Socket Bsd Api Enabled: □

Socket Bsd Api Rx Buffer Size [Byte]:

Figure 6-24 Shutdown finished timeout

6.2.4.7 TLS client

In combination with Vector TcpIp and Tls a TLS client can be configured on each TCP client socket connection (parameter `SoAdSocketTcp/SoAdTcpSocketTcpInitate` set to true).

Note
The TLS client is not supported if BSD Socket API is used.

To enable TLS client on a socket connection, sub container `SoAdTcpTlsConfig` has to be created (Figure 6-25).

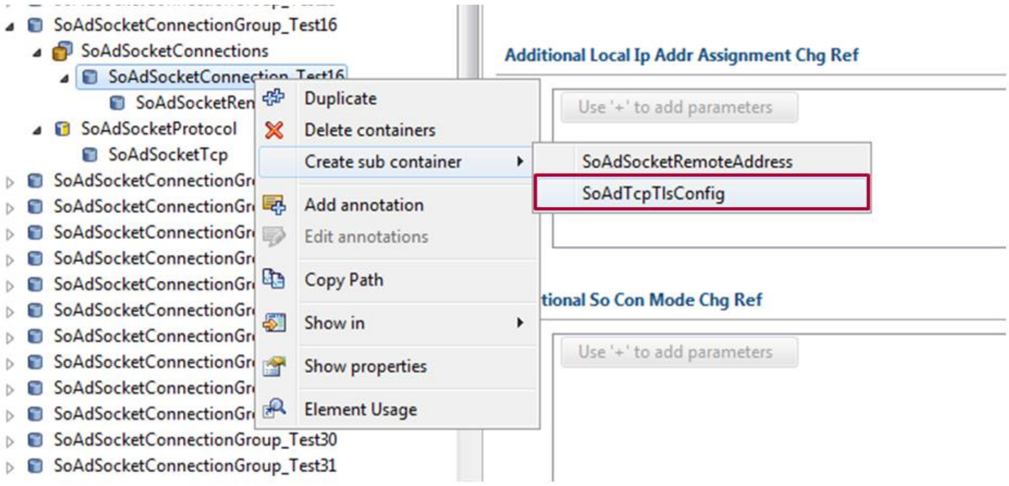
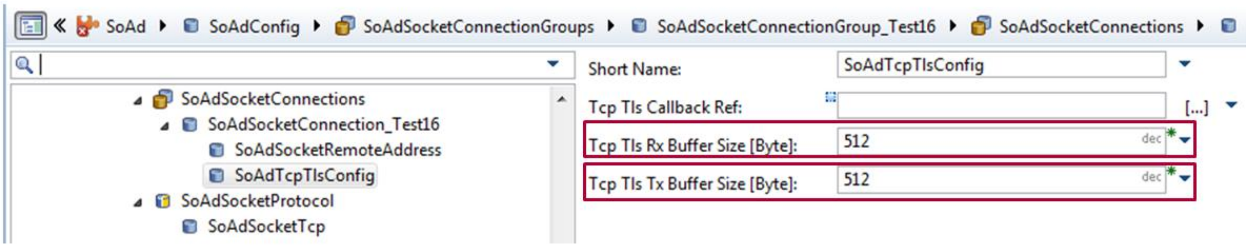


Figure 6-25 Create TLS client configuration in socket connection

Tls buffer sizes can be modified with parameters `SoAdTcpTlsTxBufferSize` and `SoAdTcpTlsRxBufferSize` (Figure 6-26). These values are validated against configuration in Tls module. Tls calculates buffers for TcpIp module and Socket Adaptor has to adapt parameters `SoAdSocketTcpTxBufferMin` and `SoAdSocketTpRxBufferMin` to match buffer sizes calculated by Tls (in a valid configuration tool provides solving actions to adapt parameters).



Short Name: SoAdTcpTlsConfig

Tcp Tls Callback Ref:

Tcp Tls Rx Buffer Size [Byte]: 512

Tcp Tls Tx Buffer Size [Byte]: 512

Figure 6-26 SoAd Tls Tx and Rx buffer configuration

Additionally a callback can be configured to notify a user about a socket creation in TcpIp module. Within this callback user can set additional parameters of socket (e.g. client certificate). These parameters are not part of Socket Adaptor therefore they are not described here.

To configure a callback create a callback container (Figure 6-27) and reference this callback container or an already existing in the corresponding socket connection (Figure 6-28). If callback prototype is declared in a separate header file add it to configuration as described in Figure 6-23.

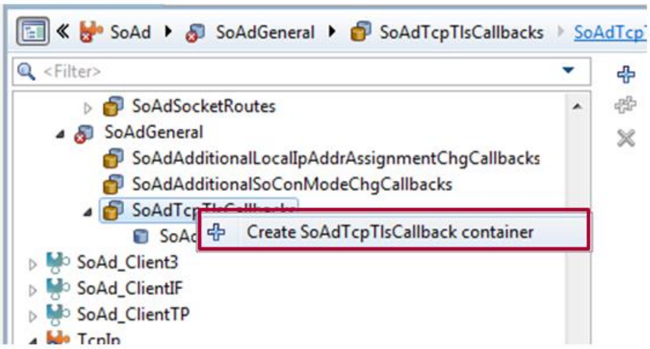
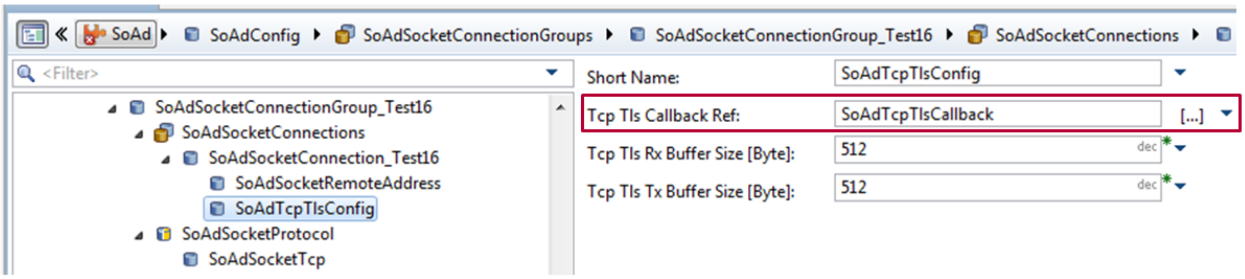


Figure 6-27 Tls callback container configuration



Short Name: SoAdTcpTlsConfig

Tcp Tls Callback Ref: SoAdTcpTlsCallback

Tcp Tls Rx Buffer Size [Byte]: 512

Tcp Tls Tx Buffer Size [Byte]: 512

Figure 6-28 Tls callback container reference configuration

6.2.4.8 MainFunction splitting

Depending on upper layer API (TP/IF) and protocol (TCP/UDP) transmission and reception are handled in MainFunction context of Socket Adaptor (refer to [1]). Also other modules of the communication stack use MainFunctions for transmission and reception (e.g. TcpIp). If there is only one MainFunction per module it has to be decided if order of calls to the MainFunctions is optimized for transmission or reception path.

Order for reception optimization:

```
TcpIp_MainFunction() -> SoAd_MainFunction()
```

Order for transmission optimization:

```
SoAd_MainFunction() -> TcpIp_MainFunction()
```

Vector supports a spitting into Rx MainFunction and Tx MainFunction to optimize transmission and reception path parallel.

Order in combination with splitting of MainFunction of TcpIp module:

```
TcpIp_MainFunctionRx() -> SoAd_MainfunctionRx() ->
SoAd_MainfunctionTx() -> TcpIp_MainFunctionTx()
```

Additionally `SoAd_MainFunctionState()` is available to handle the module states.

To enable the feature set the following parameter to true:

```
SoAd/SoAdGeneral/SoAdMainFunctionSplitEnabled
```

Caution
If splitting is enabled `SoAd_MainFunction()` must not be called anymore.

6.2.4.9 Optimized TP transmission

Socket Adaptor according to AUTOSAR can handle maximum one TP transmission per MainFunction on a PDU. But some upper layers need to send data more often.

Therefore an optimized TP transmission is supported. If this feature is enabled the entire TP transmission can be handled in transmission context as described in Figure 6-29.

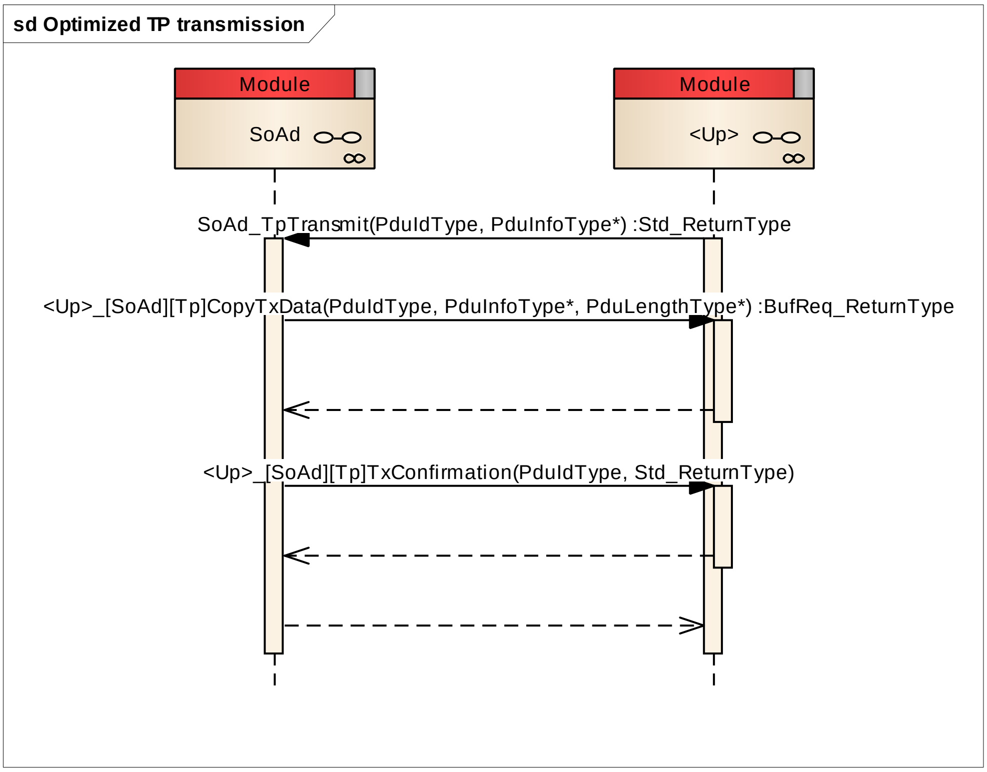
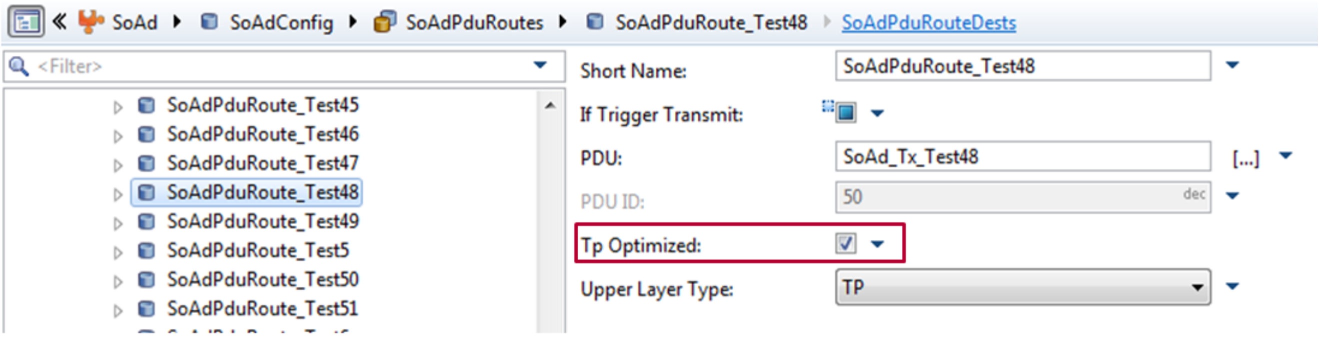

sd Optimized TP transmission
Module
SoAd
Module
<Up>
SoAd_TpTransmit(PduIdType, PduInfoType*) :Std_ReturnType
<Up>_[SoAd][Tp]CopyTxData(PduIdType, PduInfoType*, PduLengthType*) :BufReq_ReturnType
<Up>_[SoAd][Tp]TxConfirmation(PduIdType, Std_ReturnType)

Figure 6-29 Optimized TP transmission sequence

This feature can be enabled and disabled for each TP PduRoute separately (Figure 6-30).


SoAd ▸ SoAdConfig ▸ SoAdPduRoutes ▸ SoAdPduRoute_Test48 ▸ SoAdPduRouteDests

<Filter>

- SoAdPduRoute_Test45
- SoAdPduRoute_Test46
- SoAdPduRoute_Test47
- SoAdPduRoute_Test48
- SoAdPduRoute_Test49
- SoAdPduRoute_Test5
- SoAdPduRoute_Test50
- SoAdPduRoute_Test51

Short Name: SoAdPduRoute_Test48
If Trigger Transmit: □
PDU: SoAd_Tx_Test48 [...]
PDU ID: 50 dec
Tp Optimized: ☑
Upper Layer Type: TP

Figure 6-30 Optimized TP transmission configuration

Note
It is highly recommended to enable `SoAdSocketTcpImmediateTpTxConfirmation` if feature is used on TCP sockets since TP transmission is finished on reception of the TCP ACK otherwise and not in context of TP transmission request.

Note
Please consider `SoAdTcpTxQueueSize` since even if TP transmission is finished corresponding queue element is not released until reception of TCP ACK.

6.2.4.10 Trigger Transmit API for SoAd_IfTransmit

According to [1] the Trigger Transmit API is used in Socket Adaptor on transmission via `SoAd_IfRoutingGroupTransmit` or `SoAd_IfSpecificRoutingGroupTransmit`.

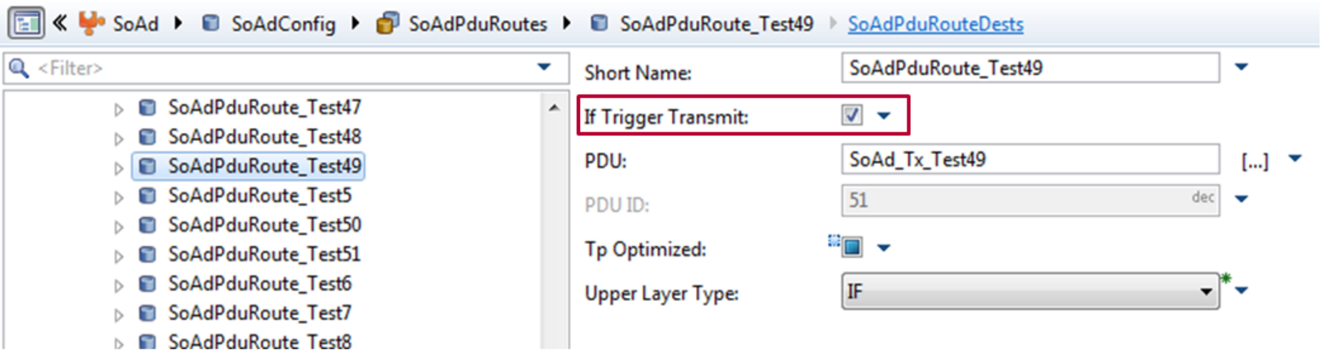
With this feature Trigger Transmit is also possible for transmissions via `SoAd_IfTransmit` like AUTOSAR specifies for other interface modules.

Caution
`SoAd_IfTransmit` has to be called with valid length (e.g. length of PDU instead of zero) since length is needed to call TcpIp transmission service before Trigger Transmit callback is called.

For some reasons length of PDU maybe unknown on call to `SoAd_IfTransmit`. In this case enable `SoAdTxDynamicLengthEnabled` in configuration tool to switch to a Vector specific transmission API between SoAd and TcpIp. With this adapted API it is possible to request a specific length but copy less data. So if PDU length is unknown maximum PDU length can be used in `SoAd_IfTransmit` and smaller length can be copied in Trigger Transmit callback.

Caution
The adapted API has to be supported by the TcpIp module. In case of BSD Socket API feature is always supported.

To enable this feature enable the corresponding parameter on the PduRoute (Figure 6-31).



SoAd ▸ SoAdConfig ▸ SoAdPduRoutes ▸ SoAdPduRoute_Test49 ▸ SoAdPduRouteDests

<Filter>

- SoAdPduRoute_Test47
- SoAdPduRoute_Test48
- SoAdPduRoute_Test49
- SoAdPduRoute_Test5
- SoAdPduRoute_Test50
- SoAdPduRoute_Test51
- SoAdPduRoute_Test6
- SoAdPduRoute_Test7
- SoAdPduRoute_Test8

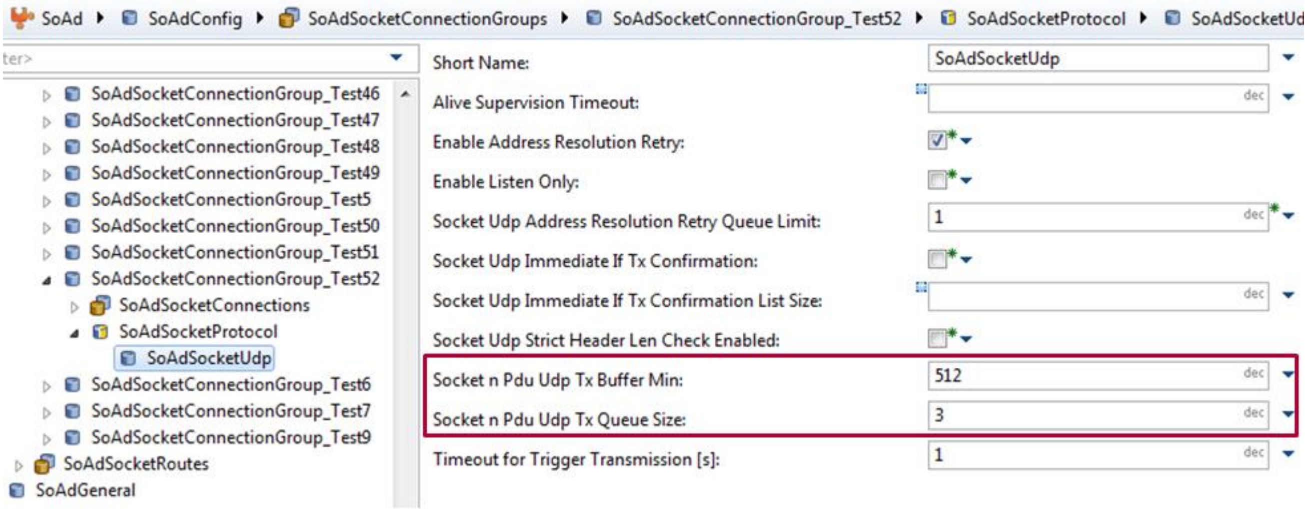
Short Name:	SoAdPduRoute_Test49
If Trigger Transmit:	☑
PDU:	SoAd_Tx_Test49
PDU ID:	51
Tp Optimized:	■
Upper Layer Type:	IF

Figure 6-31 Trigger Transmit API for SoAd_IfTransmit configuration

6.2.4.11 Optimized buffer handling for PDU fan-out

If nPdu transmission is used on UDP sockets with IF-API a buffer is generated for each socket connection (nPduUdpTxBuffer). To save RAM a PDU can be configured to use the Trigger Transmit API (refer to 6.2.4.10). In combination with nPdu a transmission request to this PDU is stored in a queue (nPduUdpTxQueue) instead of entire PDU in a buffer and if trigger condition is fulfilled for the nPdu PDU data are retrieved via the Trigger Transmit API. The PDU will be stored once in the queue even if PDU transmission is triggered multiple times. Further transmission requests to this PDU will update the length information stored in the queue.

The queue size is configurable. If a new element exceeds queue size the nPdu is sent like specified for exceeding nPduUdpTxBuffer. So if queue is used parameters `SoAdSocketnPduUdpTxBufferMin` and `SoAdSocketnPduUdpTxQueueSize` have to be considered as trigger conditions (refer to Figure 6-32) but `SoAdSocketnPduUdpTxBufferMin` is a transmission trigger condition only and no buffer is generated.



SoAd ▸ SoAdConfig ▸ SoAdSocketConnectionGroups ▸ SoAdSocketConnectionGroup_Test52 ▸ SoAdSocketProtocol ▸ SoAdSocketUd

ter>

- SoAdSocketConnectionGroup_Test46
- SoAdSocketConnectionGroup_Test47
- SoAdSocketConnectionGroup_Test48
- SoAdSocketConnectionGroup_Test49
- SoAdSocketConnectionGroup_Test5
- SoAdSocketConnectionGroup_Test50
- SoAdSocketConnectionGroup_Test51
- SoAdSocketConnectionGroup_Test52
  - SoAdSocketConnections
  - SoAdSocketProtocol
    - SoAdSocketUdp
- SoAdSocketConnectionGroup_Test6
- SoAdSocketConnectionGroup_Test7
- SoAdSocketConnectionGroup_Test9
- SoAdSocketRoutes
- SoAdGeneral

Short Name:	SoAdSocketUdp
Alive Supervision Timeout:	
Enable Address Resolution Retry:	☑
Enable Listen Only:	□
Socket Udp Address Resolution Retry Queue Limit:	1
Socket Udp Immediate If Tx Confirmation:	□
Socket Udp Immediate If Tx Confirmation List Size:	
Socket Udp Strict Header Len Check Enabled:	□
Socket n Pdu Udp Tx Buffer Min:	512
Socket n Pdu Udp Tx Queue Size:	3
Timeout for Trigger Transmission [s]:	1

Figure 6-32 NPduUdpTxQueue configuration

Note
It is possible to have a nPduUdpTxBuffer and a nPduUdpTxQueue on one socket connection. In this case PDUs in nPduUdpTxBuffer are also represented by an element in the nPduUdpTxQueue.

6.2.4.12 Event Queues and Timeout Lists

6.2.4.12.1 Event Queues

Internally Socket Adaptor uses several event queues to store different events (e.g. socket connection states) to handle them in main function context.

It is possible to configure limitations for the event queues to reduce runtime in main function if many events occur at the same time. The limitation restricts the maximum number of executed events in one main function cycle. No events get lost if the limit is reached but they are executed in one of the next main function cycles (Figure 6-33).

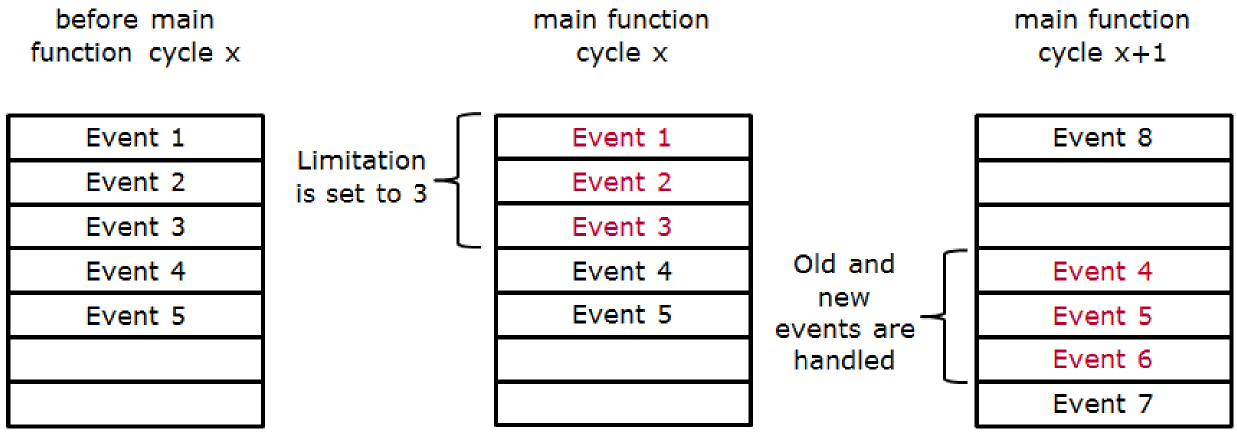
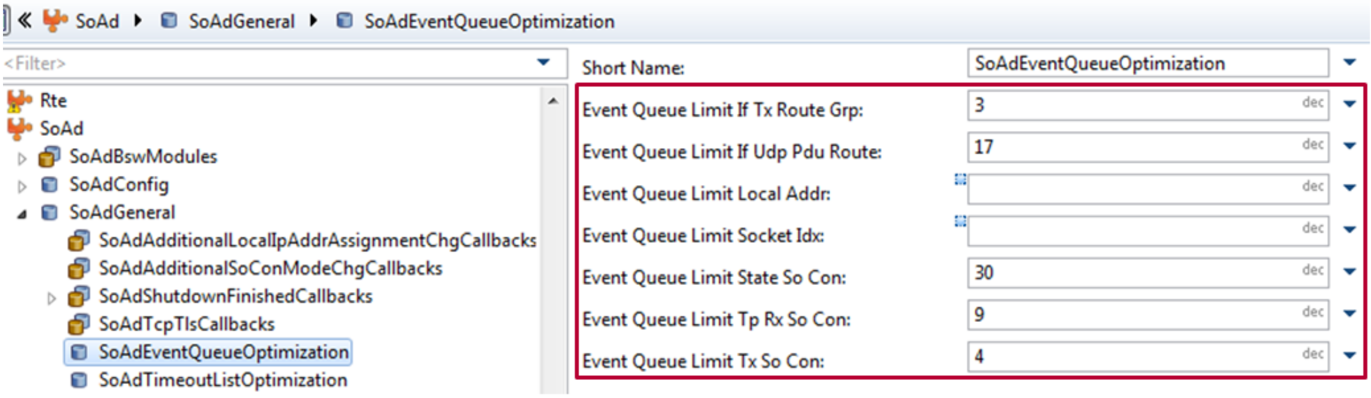


Figure 6-33 Event Queue limitation principle

To configure the event queue limitation create the corresponding configuration container and set the event queue limit (Figure 6-34). If container or parameter does not exist limitation is disabled (default) and all possible events are handled within one main function.


SoAd
SoAdGeneral
SoAdEventQueueOptimization
<Filter>
Rte
SoAd
SoAdBswModules
SoAdConfig
SoAdGeneral
SoAdAdditionalLocalIpAddrAssignmentChgCallbacks
SoAdAdditionalSoConModeChgCallbacks
SoAdShutdownFinishedCallbacks
SoAdTcpTlsCallbacks
SoAdEventQueueOptimization
SoAdTimeoutListOptimization

Short Name: SoAdEventQueueOptimization
Event Queue Limit If Tx Route Grp: 3
Event Queue Limit If Udp Pdu Route: 17
Event Queue Limit Local Addr:
Event Queue Limit Socket Idx:
Event Queue Limit State So Con: 30
Event Queue Limit Tp Rx So Con: 9
Event Queue Limit Tx So Con: 4

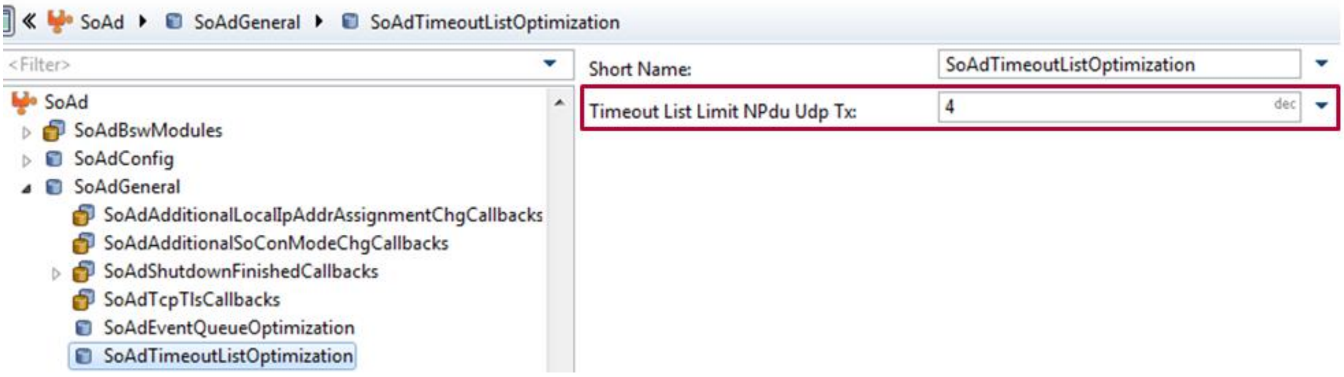
Figure 6-34 Event Queue limitation configuration

6.2.4.12.2 Timeout Lists

Beside event queues Socket Adaptor uses timeout lists to handle timeouts in main function (e.g. UDP alive supervision timeout).

It is also possible to configure limitations for the timeout lists. The limitation limits the size of the corresponding timeout list itself. So if limitation is reached a new timeout cannot be added to list and for example in case of Trigger Timeout for nPdus the corresponding transmission request is rejected.

To configure the timeout list limitation create the corresponding configuration container and set the timeout list limit (Figure 6-35). If container or parameter does not exist limitation is disabled (default) and all possible timeouts can be handled.


SoAd
SoAdGeneral
SoAdTimeoutListOptimization
<Filter>
SoAd
SoAdBswModules
SoAdConfig
SoAdGeneral
SoAdAdditionalLocalIpAddrAssignmentChgCallbacks
SoAdAdditionalSoConModeChgCallbacks
SoAdShutdownFinishedCallbacks
SoAdTcpTlsCallbacks
SoAdEventQueueOptimization
SoAdTimeoutListOptimization
dec

Short Name: SoAdTimeoutListOptimization

Timeout List Limit NPdu Udp Tx: 4

Figure 6-35 Timeout List limitation configuration

6.2.4.13 PDU reception verification

For TCP socket connections using the PDU Header option over the TP-API Socket Adaptor supports a PDU reception verification callback which can be used to filter a received PDU according to the following parameters:

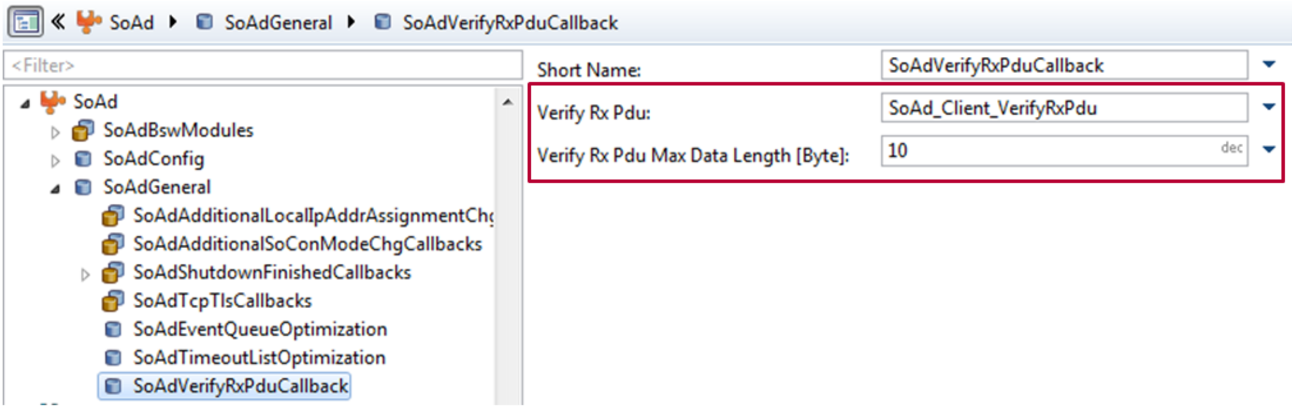
1. Local IP address and port
2. Remote IP address and port
3. PDU Header ID
4. PDU data

This feature can be used to implement a firewall on Socket Adaptor level.

In case callback `<Up_VerifyRxPdu>` is successful Socket Adaptor forwards the PDU as configured. In case callback fails Socket Adaptor drops the PDU silently and continues with the reception of PDUs received afterwards.

Figure 6-36 shows how to configure the name of the callback and the maximum amount of PDU data which are forwarded via the callback.

Figure 6-37 shows how to enable the call of the callback for a specific socket connection. If disabled on a socket connection the callback won’t be called.



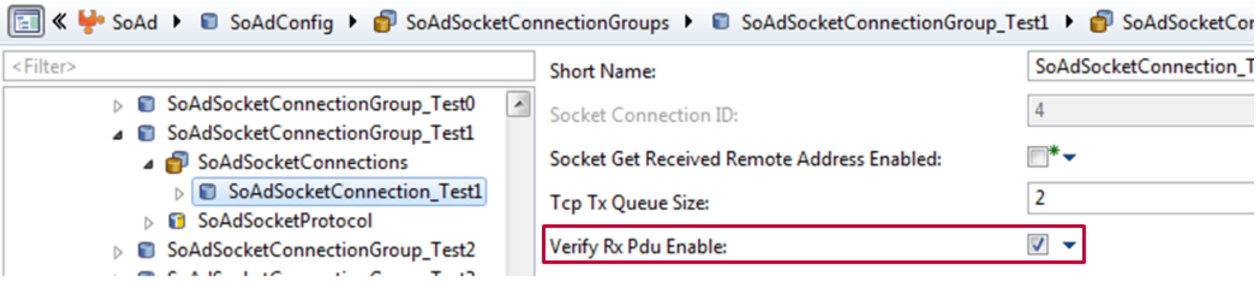
« SoAd ▸ SoAdGeneral ▸ SoAdVerifyRxPduCallback
<Filter>
SoAd
SoAdBswModules
SoAdConfig
SoAdGeneral
SoAdAdditionalLocalIpAddrAssignmentChg
SoAdAdditionalSoConModeChgCallbacks
SoAdShutdownFinishedCallbacks
SoAdTcpTlsCallbacks
SoAdEventQueueOptimization
SoAdTimeoutListOptimization
SoAdVerifyRxPduCallback

Short Name: SoAdVerifyRxPduCallback

Verify Rx Pdu: SoAd_Client_VerifyRxPdu

Verify Rx Pdu Max Data Length [Byte]: 10 dec

Figure 6-36 PDU reception verification callback configuration



« SoAd ▸ SoAdConfig ▸ SoAdSocketConnectionGroups ▸ SoAdSocketConnectionGroup_Test1 ▸ SoAdSocketCon
<Filter>
SoAdSocketConnectionGroup_Test0
SoAdSocketConnectionGroup_Test1
SoAdSocketConnections
SoAdSocketConnection_Test1
SoAdSocketProtocol
SoAdSocketConnectionGroup_Test2

Short Name: SoAdSocketConnection_T

Socket Connection ID: 4

Socket Get Received Remote Address Enabled: □

Tcp Tx Queue Size: 2

Verify Rx Pdu Enable: ☑

Figure 6-37 PDU reception verification socket connection configuration

6.2.4.14 “Transmission on specific socket connection” and “Forward socket connection on reception”

Both “Transmission on specific socket connection” and “Forward socket connection on reception” are features based on PDU meta data. These features are already described in a newer version of Socket Adaptor (see meta data requirements in [2]).

“Forward socket connection on reception” describes the mechanism to append the socket connection index as meta data to a received PDU. This feature can be used to get the socket connection index on which the PDU has been received.

“Transmission on specific socket connection” describes the mechanism to extract the socket connection index from meta data of a PDU which shall be transmitted. Socket Adaptor transmits the PDU over the corresponding socket connection. This feature can be used to restrict the PDU transmission to a specific socket connection in case a PDU fan-out (see [1]) is configured.

Both features can be used to configure an optimization for the modeling of Client/Server Calls (i.e. methods). For more details please see “RfC 64808” in the AUTOSAR Bugzilla.

To enable the usage of the meta data create the following parameter for the corresponding PDU in the EcuC module:

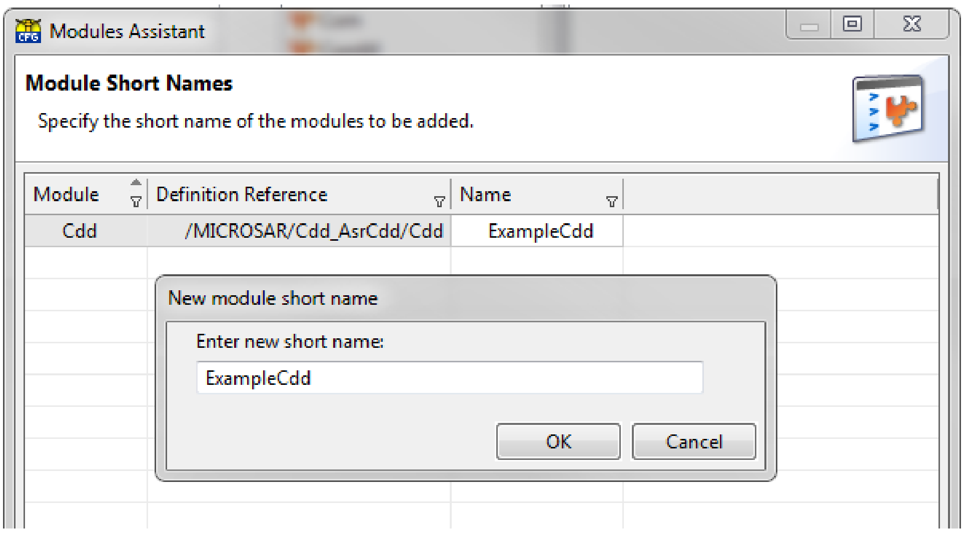
```
EcuC/EcucPduCollection/Pdu/MetaDataLength
```

Set the value to 2 since 2 bytes is the size of the socket connection index.

6.2.5 Complex Device Driver (CDD)

Chapter 3.2.5.1 describes which CDDs are supported by Socket Adaptor.

The expected API prefix of CDD within Socket Adaptor depends on the name of CDD. The name can be chosen on creation of a CDD within “Module Assistant” of DaVinci Configurator Pro (Figure 6-38).



Modules Assistant

Module Short Names

Specify the short name of the modules to be added.

Module	Definition Reference	Name
Cdd	/MICROSAR/Cdd_AsrCdd/Cdd	ExampleCdd

New module short name

Enter new short name: ExampleCdd

Figure 6-38 CDD configuration

For the example in Figure 6-38 the prefix is `ExampleCdd`. The prototype for an IF RxIndication would be `ExampleCdd_[SoAd][If]RxIndication()`.`SoAd` and `If` infixes depends on configuration of “ExampleCdd” in “SoAdBswModules”.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
BSD	Berkeley sockets used by Linux for Ethernet communication
DaVinci Configurator Pro	Generation tool for MICROSAR components
GENy	Generation tool for CANbedded and MICROSAR components

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
CDD	Complex Device Driver (Complex Driver)
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DoIP	Diagnostic over Internet Protocol
ECU	Electronic Control Unit
Eth	Ethernet Driver (MICROSAR)
EthTrcv	Ethernet Transceiver Driver (MICROSAR)
EthIf	Ethernet Interface (MICROSAR)
EthSM	Ethernet State Manager (MICROSAR)
IF	Interface API between BSW modules
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
PduR	PDU Router
RTE	Runtime Environment
SchM	Schedule Manager
SoAd	Short name of Socket Adaptor
SRS	Software Requirement Specification
SWS	Software Specification
TcpIp	Transport layer module containing TCP and UDP implementation (MICROSAR)
TCP	Transmission Control Protocol
Tls	Transport Layer Security module (MICROSAR)
TLS	Transport Layer Security
TP	Transport Protocol API between BSW modules

UDP	User Datagram Protocol
VLAN	Virtual Local Area Network

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

> News
> Products
> Demo software
> Support
> Training data
> Addresses

www.vector.com