

MICROSAR Classic SOME/IP Transformer

Technical Reference

Version 4.37.0

Authors	visrco, visso, vispah, vissi, vislsa, visdes, jkugler
Status	Released

Document Information

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	Specification of SOME/IP Transformer	R20-11
[2]	AUTOSAR	List of Basic Software Modules	R20-11

Scope of the Document

This technical reference describes the general use of the SOME/IP Transformer.



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

1	Introduction.....	6
1.1	Architecture Overview	6
2	Functional Description	7
2.1	Features	7
2.1.1	Deviations	7
2.2	Initialization	8
2.3	States	9
2.4	Main Functions	9
2.5	Error Handling.....	9
2.5.1	Development Error Reporting.....	9
2.5.2	Production Code Error Reporting	9
3	Integration.....	10
3.1	Embedded Implementation	10
4	API Description.....	11
4.1	Services provided by SomelpXf	11
4.1.1	SomelpXf_InitMemory.....	11
4.1.2	SomelpXf_Init	11
4.1.3	SomelpXf_DeInit.....	12
4.1.4	SomelpXf_GetVersionInfo.....	13
4.1.5	Sender / Receiver communication.....	14
4.1.5.1	SomelpXf_<transformerId>	14
4.1.5.2	SomelpXf_Inv_<transformerId>	15
4.1.6	Client / Server communication	16
4.1.6.1	SomelpXf_<transformerId>	16
4.1.6.2	SomelpXf_Inv_<transformerId>	17
5	Configuration.....	18
5.1	Configuration Variants.....	18
5.2	Enabling / Disabling of data transformation	18
5.3	Configuration of Sender / Receiver Communication	18
6	Glossary and Abbreviations	19
6.1	Glossary	19
7	Additional Copyrights	20
7.1	Abbreviations	20

8 Contact..... 21

Illustrations

Figure 1-1	AUTOSAR 4.2 Architecture Overview	6
------------	---	---

Tables

Table 2-1	Supported AUTOSAR standard conform features	7
Table 2-2	Not supported AUTOSAR standard conform features	8
Table 3-1	Implementation files	10
Table 4-1	SomelpXf_InitMemory	11
Table 4-2	SomelpXf_Init	12
Table 4-3	SomelpXf_DeInit	12
Table 4-4	SomelpXf_GetVersionInfo	13
Table 4-5	SomelpXf_<transformerId>	14
Table 4-6	SomelpXf_Inv_<transformerId>	15
Table 4-7	SomelpXf_<transformerId>	16
Table 4-8	SomelpXf_Inv_<transformerId>	17
Table 6-1	Glossary	19
Table 7-1	Free and Open Source Software Licenses	20
Table 7-2	Abbreviations	20

1 Introduction

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

Supported Configuration Variants:	pre-compile	
Vendor ID:	SOMEIPXF_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	SOMEIPXF_MODULE_ID	174 decimal (according to ref. [2])

The SomeIpXf module provides the functionality to serialize data in the SOME/IP on-the-wire format.

1.1 Architecture Overview

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

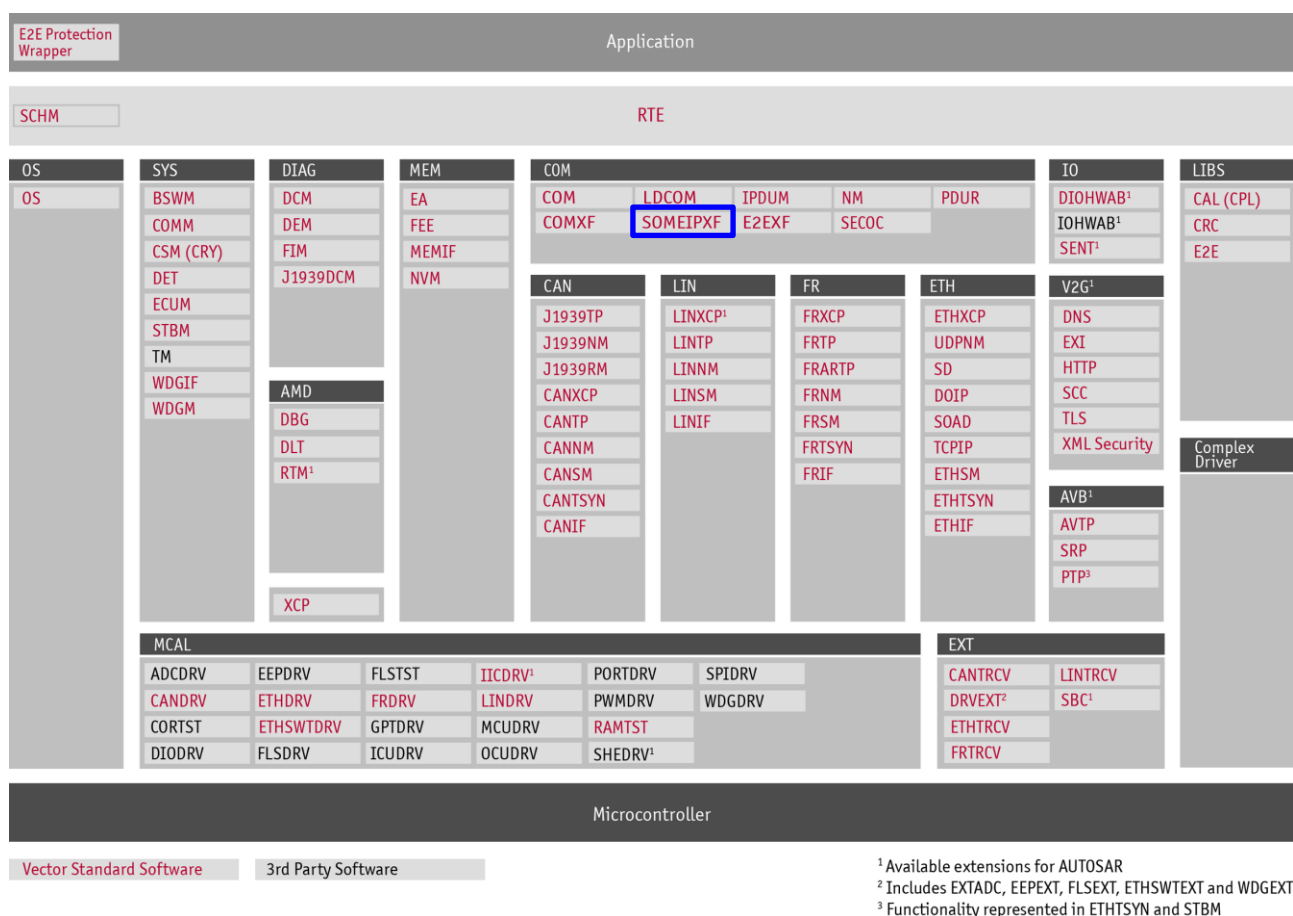


Figure 1-1 AUTOSAR 4.2 Architecture Overview

2 Functional Description

2.1 Features

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

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

- > Table 2-1 Supported AUTOSAR standard conform features
- > Table 2-2 Not supported AUTOSAR standard conform features

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features

Serialization / Deserialization of complex data for S/R communication.

Serialization / Deserialization of complex data for C/S communication.

Table 2-1 Supported AUTOSAR standard conform features

2.1.1 Deviations

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

Category	Description
Functional	The serialization / deserialization of the following data types: <ul style="list-style-type: none">- Bitfields- Extensible structs
Functional	Development error detection.
Functional	Message type for an autonomous error response is still set to "ERROR (0x81)" as specified in AR 4.2.2 and before.
Functional	If message type of transformed Sender/Receiver communication is not set, "REQUEST_NO_RETURN (0x01)" is used as default.
Functional	If less data than expected are handed over the SOME/IP transformer during deserialization of data, initial value is not considered and deserialization will always abort with E_SER_MALFORMED_MESSAGE.
Functional	If the length is greater than the expected length of a struct (as specified in the data type definition), the unexpected data is not skipped and deserialization aborts with E_SER_MALFORMED_MESSAGE.
Functional	Endianness for UTF-16 strings is not evaluated at the base type. Performing a conversion if platform endianness and SOMEIPTransformerDescription.byteOrder have different values.
Config	DataPrototypeTransformationProps
Functional	TLV with optional unions
Functional	Optional acknowledgment message types

Category	Description
Functional	Dynamic array size profile "VSA_SQUARE" and "VSA_RECTANGULAR"
Config	SomeIP transformer attributes: <ul style="list-style-type: none"> - SOMEIPTransformationProps: <ul style="list-style-type: none"> o alignment o sizeofStructLengthField o sizeofArrayLengthField o sizeofUnionLengthField o sizeofStringLengthField - SOMEIPTransformationISignalProps: <ul style="list-style-type: none"> o isDynamicLengthFieldSize o sizeofStringLengthField
Functional	Data type of length field for variable size arrays still determined from the size indicator
Functional	SomeIpXf_ExtractProtocolHeaderFields
Functional	String specific serialization is triggered even if "implementsSOMEIPStringHandling" is not defined
Functional	The deserialization of a variable string terminates with "MALFORMED_MESSAGE" only if the variable string is not the last element in the payload and if the length of the dynamic-length string is greater than the specified maximum length of the string, otherwise it does not terminate with "MALFORMED_MESSAGE" (in this case the deserializer reads the data only up to the specified maximum length of the string), as specified in AR 4.3.1 and before.
Functional	Null-termination for string de-/serialization are not considered.
Functional	Discard the last byte in case of odd length utf16 string, here a "MALFORMED_MESSAGE" is send if the array length is not a multiple of two bytes.
Functional	TLV is only supported when modeled as described in AR 4.6 and above.
Functional	If SessionHandlingSR is defined and set to true or false, it is still prioritized even though AR 23-11 or older is used. If SessionHandlingSR is not defined, it is handled according to AR 23-11 or older.

Table 2-2 Not supported AUTOSAR standard conform features

2.2 Initialization

The SomeIpXf only needs to be initialized when session handling for sender-receiver communication is configured. Otherwise, calls to `SomeIpXf_InitMemory()`, `SomeIpXf_Init()` and `SomeIpXf_DeInit()` can be omitted. On platforms in which the Random Access Memory (RAM) is not initialized by the startup code, the function `SomeIpXf_InitMemory` has to be called first and then a call to `SomeIpXf_Init` can be realized.

2.3 States

The transformer internally stores the sequence counter when session handling for sender-receiver communication is configured.

2.4 Main Functions

No main function exists because all functionality is performed within the called API.

2.5 Error Handling

2.5.1 Development Error Reporting

No development error reporting is currently supported by the SomelPx.

2.5.2 Production Code Error Reporting

No production errors are specified for the SomelPx.

3 Integration

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

3.1 Embedded Implementation

The delivery of the SomelpXf consists of:

File Name	Description	Integration Tasks
SomelpXf.c	Generated source file of the SomelpXf module.	-
SomelpXf.h	Generated main header file which shall be included by modules using the SomelpXf module.	-
SomelpXf_MemMap.h	Generated file with template areas that can be adapted by the user. It contains the SomelpXf specific part of the memory mapping.	Adapt the dedicated code areas within that file. See hints within that file.
SomelpXf_Compiler_Cfg.h	Generated file with template areas that can be adapted by the user. It contains the SomelpXf specific part of the compiler abstraction.	Adapt the dedicated code areas within that file. See hints within that file.

Table 3-1 Implementation files

4 API Description

4.1 Services provided by SomeIpXf

4.1.1 SomeIpXf_InitMemory

Prototype	
void SomeIpXf_InitMemory (void)	
Parameter	
void	none
Return code	
void	none
Functional Description	
This service initializes the transformer internal state variables if the compiler does not support initialized variables.	
Particularities and Limitations	
<ul style="list-style-type: none">> This function is synchronous.> This function is non-reentrant.> If this function is used it shall be called before any other SomeIpXf function after startup.	
Expected Caller Context	
This function needs to be called either from trusted context or before the MPU is activated.	

Table 4-1 SomeIpXf_InitMemory

4.1.2 SomeIpXf_Init

Prototype	
void SomeIpXf_Init (const SomeIpXf_ConfigType *config)	
Parameter	
config	Pointer to the transformer's configuration data.
Return code	
void	none
Functional Description	
Initialization function.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 4-2 SomelpXf_Init

4.1.3 SomelpXf_DeInit

Prototype	
void SomeIpXf_DeInit (void)	
Parameter	
void	none
Return code	
void	none
Functional Description	
Deinitialization function.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 4-3 SomelpXf_DeInit

4.1.4 SomelpXf_GetVersionInfo

Prototype	
void SomeIpXf_GetVersionInfo (Std_VersionInfoType *versioninfo)	
Parameter	
versioninfo	Pointer to where to store the version information of this module.
Return code	
void	none
Functional Description	
This API returns version information, vendor ID and AUTOSAR module ID of the called transformer module.	
Particularities and Limitations	
This API is only available if enabled by the configuration parameter XfrmVersionInfoApi.	
Expected Caller Context	
This function can be called in any context.	

Table 4-4 SomelpXf_GetVersionInfo

4.1.5 Sender / Receiver communication

4.1.5.1 SomIpXf_<transformerId>

Prototype	
Std_ReturnType SomIpXf_<transformerId> (uint8 *buffer, uint32 *bufferLength, const <type> *dataElement)	
Parameter	
buffer	Buffer allocated by the RTE, where the transformed data has to be stored by the transformer.
bufferLength	Used length of the buffer.
dataElement	Data element which shall be transformed.
Return code	
E_OK	Serialization successful.
SOMEIPXF_E_SER_GENERIC_ERROR	A generic error occurred.
Functional Description	
Serialization of data element based on the SOME/IP on the wire format for S/R communication.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 4-5 SomIpXf_<transformerId>

4.1.5.2 SomelpXf_Inv_<transformerId>

Prototype	
Std_ReturnType SomelpXf_Inv_<transformerId> (const uint8 *buffer, uint32 bufferLength, <type> *dataElement)	
Parameter	
buffer	Buffer allocated by the RTE, where the serialized data is stored by the Rte.
bufferLength	Used length of the buffer.
dataElement	Data element which is the result of the transformation and contains the deserialized data element.
Return code	
E_OK	Deserialization successful.
SOMEIPXF_E_SER_GENERIC_ERROR	A generic error occurred.
SOMEIPXF_E_SER_WRONG_PROTOCOL_VERSION	The version of the receiving transformer did not match to the version of the sending transformer.
SOMEIPXF_E_SER_WRONG_INTERFACE_VERSION	Interface version of serialized data is not supported.
SOMEIPXF_E_SER_MALFORMED_MESSAGE	The received data was malformed. No valid output could be produced.
SOMEIPXF_E_SER_WRONG_MESSAGE_TYPE	The received message type was not expected.
Functional Description	
Deserialization of data element based on the SOME/IP on the wire format for S/R communication.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 4-6 SomelpXf_Inv_<transformerId>

4.1.6 Client / Server communication

4.1.6.1 SomelpXf_<transformerId>

Prototype	
<pre>Std_ReturnType SomeIpXf_<transformerId> (const Rte_Cs_TransactionHandleType *transactionHandle, uint8 *buffer, uint32 *bufferLength, [Std_ReturnType returnValue], [<type> data_1] ... [<type> data_n])</pre>	
Parameter	
transactionHandle	Transaction handle (clientId and sequenceCounter) needed to differentiate between multiple requests.
buffer	Buffer allocated by the RTE, where the transformed data has to be stored by the transformer.
bufferLength	Used length of the buffer.
returnValue	Return value of the server runnable which needs to be serialized on server side for transmission to the calling client. This argument is only available for serializers of the response of a Client/Server communication.
data_1	Client/Server operation argument which shall be transformed (in the same order as in the corresponding interface).
data_n	Client/Server operation argument which shall be transformed (in the same order as in the corresponding interface).
Return code	
E_OK	Serialization successful.
SOMEIPXF_E_SER_GENERIC_ERROR	A generic error occurred.
Functional Description	
Serialization of data element based on the SOME/IP on the wire format for C/S communication.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 4-7 SomelpXf_<transformerId>

4.1.6.2 SomelpXf_Inv_<transformerId>

Prototype	
<pre>Std_ReturnType SomeIpXf_Inv_<transformerId> (Rte-Cs_TransactionHandleType *transactionHandle, const uint8 *buffer, uint32 bufferLength, [Std_ReturnType *returnValue], [<type> *data_1] ... [<type> *data_n])</pre>	
Parameter	
transactionHandle	Transaction handle (clientId and sequenceCounter) needed to differentiate between multiple requests.
buffer	Buffer allocated by the RTE, where the serialized data is stored by the Rte.
bufferLength	Used length of the buffer.
returnValue	Return value of the server runnable which needs to be serialized on server side for transmission to the calling client. This argument is only available for deserializers of the response of a Client/Server communication.
data_1	Client/Server operation argument which shall be transformed (in the same order as in the corresponding interface).
data_n	Client/Server operation argument which shall be transformed (in the same order as in the corresponding interface).
Return code	
E_OK	Deserialization successful.
SOMEIPXF_E_SER_GENERIC_ERROR	A generic error occurred.
SOMEIPXF_E_SER_WRONG_PROTOCOL_VERSION	The version of the receiving transformer did not match to the version of the sending transformer.
SOMEIPXF_E_SER_WRONG_INTERFACE_VERSION	Interface version of serialized data is not supported.
SOMEIPXF_E_SER_MALFORMED_MESSAGE	The received data was malformed. No valid output could be produced.
SOMEIPXF_E_SER_WRONG_MESSAGE_TYPE	The received message type was not expected.
Functional Description	
Deserialization of data element based on the SOME/IP on the wire format for C/S communication.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 4-8 SomelpXf_Inv_<transformerId>

5 Configuration

In the SomeIpXf the attributes can be configured with the following tools:

- > Configuration in DaVinci Configuration

Currently, only the GetVersionInfo API can be enabled / disabled in the SomeIpXf Ecu configuration.

5.1 Configuration Variants

The SomeIpXf supports the configuration variants

- > VARIANT-PRE-COMPILE

The configuration classes of the SomeIpXf parameters depend on the supported configuration variants. For their definitions please see the `SomeIpXf_bswmd.arxml` file.

5.2 Enabling / Disabling of data transformation

It is possible to enable the SomeIpXf through the configuration of a transformer chain in the system description according to AUTOSAR.

5.3 Configuration of Sender / Receiver Communication

The message types of sender / receiver communication can be `REQUEST_NO_RETURN` (0x01) or `NOTIFICATION` (0x02) according to AUTOSAR. The specification allows this parameter to be undefined without specifying a default value. Currently, if the parameter is not set `REQUEST_NO_RETURN` is used as default.

6 Glossary and Abbreviations

6.1 Glossary

Term	Description
DaVinci Configurator	Configuration and generation tool for MICROSAR Classic components

Table 6-1 Glossary

7 Additional Copyrights

The MICROSAR Classic SOMEIPXF Generator contains *Free and Open Source Software* (FOSS). The following table lists the files which contain this software, the kind and version of the FOSS, the license under which this FOSS is distributed and a reference to a license file which contains the original text of the license terms and conditions. The referenced license files can be found in the directory of the RTE Generator.

File	FOSS	License	License Reference
MicrosarSomelpXfGen64.exe	Perl 5.30	Artistic License	License_Artistic.txt

Table 7-1 Free and Open Source Software Licenses

7.1 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

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

www.vector.com