Document Title	Requirements on Libraries
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	314
Document Classification	Auxiliary

Document Version	2.1.0
Document Status	Final
Part of Release	4.0
Revision	2

Document Change History

Date	Version	Changed by	Change Description
13.10.2010	2.1.0	AUTOSAR Administration	• Typo's correction: E2E instead of E2e (Chapter1, Page 6)
23.11.2009	2.0.0	AUTOSAR Administration	• Enlargement of the scope of the initial document to general library requirement • Introduction of Error handling • Introduction of E2E profiles • Legal disclaimer revised
23.06.2008	1.0.1	AUTOSAR Administration	• Legal disclaimer revised • Rename from "Requirements on CRC Routines" to "Requirements on Libraries"
13.12.2007	1.0.0	AUTOSAR Administration	Initial Release: Split from Requirements on Memory Services

Disclaimer

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Scope of Document

This document specifies requirements on the **AUTOSAR Libraries**. It applies to all the libraries specified by AUTOSAR:

Library short name	Description
Mfx	Library of **M**athematical **F**i**X**ed point calculations
Mfl	Library of **M**athematical **FL**oating point point calculations
Ifx	Library of **I**nterpolation functions of **F**i**X**ed point
Ifl	Library of **I**nterpolation functions of **FL**oating point
Bfx	Library of **B**it handling
Efx	Library of **E**xtended functions on **F**ixed point
Crc	Library of CRC routines
E2E	SW-C End-to-End Communication Protection Library

Each of these libraries has its own independent SWS but this SRS document applies to all libraries.

Conformance to all requirements is mandatory for all implementations. “Configurable” also means, the requirement must be met, but such functionality can be disabled, if not needed in an ECU (BSW or SW-C).

This document was initially dedicated to CRC routines. In order to keep traceability, the CRC requirements are still present but in a dedicated chapter.

2 Conventions to be used

- In requirements, the following specific semantics shall be used (based on the Internet Engineering Task Force IETF).

  The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as:

  - SHALL: This word means that the definition is an absolute requirement of the specification.
  - SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
  - MUST: This word means that the definition is an absolute requirement of the specification due to legal issues.
  - MUST NOT: This phrase means that the definition is an absolute prohibition of the specification due to legal constraints.
  - SHOULD: This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
  - SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
  - MAY: This word, or the adjective „OPTIONAL“, means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation, which does not include a particular option, MUST be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, MUST be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)

3 Acronyms and abbreviations

All technical terms used in this document, except the ones listed in the table below, can be found in the official AUTOSAR glossary [].

Acronyms and abbreviations that have a local scope and therefore are not contained in the AUTOSAR glossary, appear in the glossary below.

Abbreviation / Acronym:	***Description:***
API	Application Programming Interface. Preferred term is "function".
AR AR_	Abbreviation used in place of AUTOSAR
BFX	Library of Bit handling
CRC	Cyclic Redundancy check
DET	Development Error Tracer
E2E	End to End
EcuM	ECU Manager
EFX	Extended function on Fixed point
IFL	Interpolation function s of FLoating point
IFX	Interpolation function s of FiXed point
Library	Set of APIs (i.e. functions) that may be called from any modules (BSW modules or SW-C)
MFL	Mathematical FLoating point calculation
MFX	Mathematical FiXed point calculation
OS	Operating System

4 Requirements Specification

4.1 Functional Overview

The AUTOSAR libraries provide other BSW modules and application SW-Cs with mathematical services.

The libraries offer C functions that can be called from source code, i.e. from BSW modules, from SW-C, from RTE or from Complex Device Drivers.

4.2 Functional Requirements

4.2.1 Configuration

4.2.1.1 [BSW31400001] The functional behavior of each library functions shall not be configurable

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	The functional behavior of each library functions shall not be configurable
Type:	New
Importance:	High
Description:	The functional behavior of library functions shall not be configurable. For some given inputs (c function parameters), a function shall return always the same outputs, as defined with the function specification. However, the internal behavior can be configurable. For examples: choose resource consumption strategy e.g. priority to CPU/RAM/ROM. But this is implementation specific and not standardized by AUTOSAR.
Rationale:	A SW-C that uses a library function expects a deterministic and standardized behavior. It the SW integrator has the possibility to configure and change the behavior, the SW-C may react unexpectedly.
Use Case:	Division function. In case of division by zero, the function shall always return the same value. If the SW integrator has the possibility to configure this return value, it may be right for some SW-Cs but it may also be catastrophic for other SW-Cs.
Dependencies:	None
Conflicts:	None
Supporting Material:	None

4.2.1.2 [BSW31400015] Shared library code

Initiator:	WP LIBRARIES
Date:	09-october-2008
Short Description:	Shared library code
Type:	New
Importance:	High

Description:	In case memory partitioning is enabled on a given microcontroller, then it shall be possible to configure the microcontroller so that the library code is shared between all callers of the shared library.
Rationale:	This enables to reduce the Flash memory consumption.
Use Case:	In a partitioned system, SW-Cs in different partitions access the same library.
Dependencies:	None
Conflicts:	None
Supporting Material:	None

4.2.2 Initialisation

4.2.2.1 [BSW31400002] A library shall be operational before all BSW modules and application SW-Cs

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	A library shall be operational before all BSW modules and application SW-Cs
Type:	New
Importance:	High
Description:	A library shall not require initialization phase.
Rationale:	A Library function may be called at the very first step of ECU initialization, e.g. even by the OS or EcuM, thus the library shall be ready.
Use Case:	AUTOSAR OS initialization may call bit handling function
Dependencies:	None
Conflicts:	None
Supporting Material:	None

4.2.3 Normal Operation

4.2.3.1 [BSW31400004] Using libraries shall not pass through a port interface

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	Using libraries shall not pass through a port interface
Type:	New
Importance:	High
Description:	SW-Cs shall directly invoke library functions, without passing through port RTE interface. To access the library API, the SW-Cs shall directly include the library header file.
Rationale:	The SW developer should be free to use libraries without having to change the SW-C interface description: reduce effort and inconsistencies The port+RTE mechanism is not required for library calls: no consistency check, no queuing, no communication outside ECU, etc. - Using library functions is a software designer decision. It is related to implementation, not to SW-C interface. - Calling a library function is an elemental operation. It is not like a client/server operation - because, they are often used, library function shall be called in an efficient

	way Thus, the function can be directly called from the source code, e.g. runnables, without using RTE API.
Use Case:	Application includes floating point arithmetic library header, and calls floating point routines in control loop calculations, without crossing RTE.
Dependencies:	Re-entrancy of libraries is a prerequisite
Conflicts:	None
Supporting Material:	None

4.2.3.2 [BSW31400005] Library header file

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	Library header file
Type:	New
Importance:	medium
Description:	Each library shall provide one header file with its public interface. This header shall declare all the public function prototypes and types defined by the library specification. The header file shall be named like: <library short name>.h
Rationale:	Access to function prototypes and types Standardization of header file name
Use Case:	#include "AR_MFX.h"
Dependencies:	BSW31400004
Conflicts:	None
Supporting Material:	None

4.2.3.3 [BSW31400006] The header #include is placed by the SW-C developer

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	The header #include is placed by the SW-C developer
Type:	New
Importance:	low
Description:	The statement "<library short name>.h" should be placed by the developer or an application code generator but not by the RTE generator
Rationale:	The DependencyOnLibrary update might be forgotten by the developer. The use of libraries should be visible while reading a source file Putting a #include is a simple operation and does not require the support of RTE
Use Case:	none
Dependencies:	None
Conflicts:	BSW31400007
Supporting Material:	None

4.2.3.4 [BSW31400009] Re-entrancy

Initiator:	WP LIBRARIES
Date:	27-june-2008

Short Description:	Re-entrancy
Type:	New
Importance:	High
Description:	All library functions shall be re-entrant, which means that they shall be able to handle several simultaneous, interleaved or/and concurrent requests.. A function, in order to be re-entrant, it (1) shall not call any non-re-entrant functions, (2) shall not write any global nor static variables. If some kind of data shall be handled, the callers have to create (define) them and pass them as function parameters. A library function only runs in the context of the caller, on the core where it is called, in the same protection environment. A library function can only call library functions. A library function is synchronous, e.g. it does not have wait points.
Rationale:	Avoid consistency mechanisms which bring to weak efficiency
Use Case:	Multitasking environments Every BSW modules and SW-C will use the same functions in different tasks
Dependencies:	[BSW00312]
Conflicts:	See Req “Error notification”
Supporting Material:	None

4.2.3.5 [BSW31400010] Specific types

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	Specific types
Type:	New
Importance:	Medium
Description:	A library shall define (typedef) its own specific types in the library header file if and only if they are not yet defined by AUTOSAR in std_types.h and platform_types.h. These types shall be identified in the corresponding SWS by the name and the description, but not the actual implementation. No new implementation-specific types are allowed in the implementation of the public library interface, i.e. other types that are not specified in the SWS shall not be present. The implementation (typedef) can be different, e.g. according the platform. The caller shall not rely on any implementation. Using C operators on these specific types is forbidden.
Rationale:	A library may handle some specific types not defined by AUTOSAR Only SWS specified type are allowed in order to insure code portability independent from any specific AUTOSAR library implementation
Use Case:	Types u64, S64 for the 64bits data math library
Dependencies:	None
Conflicts:	None
Supporting Material:	None

4.2.3.6 [BSW31400011] Naming convention for functions and types

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	Naming convention for functions and types
Type:	New
Importance:	low
Description:	All function names and type names shall start with "Library short name_"
Rationale:	Avoid collision with already existing library Quickly identified AUTOSAR library calls in the code

Use Case:	None
Dependencies:	None
Conflicts:	None
Supporting Material:	None

4.2.3.7 [BSW31400012] Passing parameters with structure is allowed

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	Passing parameters with structure is allowed
Type:	New
Importance:	High
Description:	Library functions are allowed to have as parameters the (1) elementary data types (e.g. sint32), (2) pointers to C structs, (3) C structs Comment about efficiency: C does not use structure filling if a structure is given as parameter. C works with pointers. For e.g. in case of interpolation functions curve/map structure is a part of data set. It is more runtime overhead to set a lot of parameters than to set one addressing register for the curve/map structure. In some processors Structure elements are accessed in single clock cycle.
Rationale:	Function calls become simpler. If a set of fixed parameters is required. They can be defined once in a structure and may be used several times. In some circumstances, it makes sense to group several function parameters into one of few structures: - when there are many parameters, - when some parameters can be grouped by functionality

Use Case:	`sint16 EFX_PGOV_WIN ( sint32 X,`
	`sint32 Kp,`
	`sint32 KpPos,`
	`sint32 KpNeg,`
	`sint32 WinPos,`
	`sint32 WinNeg )`
	
	sint16 EFX_PGOV_WIN (sint32 X, const PWin_Type * Struct)

Dependencies:	[BSW31400008]: if a parameter is added to the structure, a new structure name and a new function name shall be defined. So there is no risk that a new structure field would be forgotten in case of library evolution.
Conflicts:	None
Supporting Material:	None

4.2.4 Shutdown Operation

4.2.4.1 [BSW31400003] A library shall be operational until the shutdown

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	A library shall be operational until the shutdown
Type:	New
Importance:	High

Description:	A library should not require a shutdown operation phase. If so, it shall occur after all AUTOSAR BSW modules shutdown operations
Rationale:	A Library function may be called at the very latest step of ECU shutdown, e.g. even by the OS, thus the library shall be ready until the end
Use Case:	AUTOSAR OS shutdown operation may call bit handling function
Dependencies:	None
Conflicts:	None
Supporting Material:	None

4.2.5 Fault Operation

4.2.5.1 [BSW31400013] Error detection

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	Error detection
Type:	New
Importance:	High
Description:	Function should check at runtime (both in production and development code) the value of input parameters, especially cases where erroneous value can bring to fatal error or unpredictable result, if they have the values allowed by the function specification. All the error cases shall be listed in SWS and the function should return a specified value (in SWS) that is not configurable. This value is dependant of the function and the error case so it is determined case by case.
Rationale:	Avoid fatal error, provide standardized behaviour
Use Case:	Division by zero, negative number of a square root, out of range, overflow, underflow, etc.
Dependencies:	[BSW31400001]
Conflicts:	None
Supporting Material:	None

4.3 Non-Functional Requirements (Qualities)

4.3.1 Compatibility

4.3.1.1 [BSW31400008] Backward compatibility

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	Backward compatibility
Type:	New
Importance:	medium
Description:	For a given function prototype name, the behavior and the parameters shall not evolve once it is a part of an AUTOSAR final release. For any reasons, if the specification of a behavior has to be changed in AUTOSAR release N+1,

	e.g. the WP LIBRARIES decides of a new feature, or a parameter/type has to be added/removed/modified from the specification, and if library implementations might be already in production, then a new function prototype shall be created. The previous one (AUTOSAR release N) shall be kept unchanged. In case of a type implementation is specified in SWS, then if this implementation changes, a new type name and API name shall be created and the previous one shall be retained. In case of a type implementation is not defined in SWS, then the library developer is free to change the implementation without creating new type name.
Rationale:	Avoid SW-C re-developing in case of library evolution. Avoid having to integrate new library releases if SW-C do not require new function. if an integrator has to add different SW-C which rely on different libraries(AUTOSAR specification) versions, he can (has to) purchase the latest library version since the ascending compatibility is insured.
Use Case:	On a project, the integrator uses a library compatible with AR4.0, and a SW-C that use this library. On a next V cycle, a new library release compatible with AR4.1 is integrated. With the ascending compatibility rule, it is ensured that the SW-C will react in the same way even if new functions have been added.
Dependencies:	None
Conflicts:	None
Supporting Material:	None

4.3.1.2 [BSW31400016] Non AUTOSAR library

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	Non AUTOSAR library
Type:	New
Importance:	medium
Description:	A SW-C may use a non-AUTOSAR library available on the market. In this case, the developer may not deliver this library but just mention it in the SW-C template (DependencyOnLibrary). It is the responsibility of the integrator to get this library to be able to integrate the SW-C. This non-AUTOSAR library should respect the requirements of this document. It is recommended that non-AUTOSAR functions start with a specific vendor prefix.
Rationale:	Allows freedom of implementation. Avoid name collisions.
Use Case:	Supplier specific libraries.
Dependencies:	[BSW31400011]
Conflicts:	None
Supporting Material:	None

4.3.2 Others

4.3.2.1 [BSW31400007] Using a library should be documented

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	Using a Library should be documented
Type:	New
Importance:	medium
Description:	If a BSW module or a SW-C uses a Library, the developer should add an Implementation-DependencyOnLibrary in the BSW/SW-C template. minVersion and maxVersion parameters correspond to the supplier version. In case of AUTOSAR library, these parameters may be left empty because a SW-C or BSW module may rely on a library behaviour, not on a supplier implementation. However, the SW-C or BSW modules shall be compatible with the AUTOSAR platform where they are integrated.
Rationale:	SW integrator to checks the AUTOSAR platform compatibilities while integrating a BSW module or a SW-C
Use Case:	none
Dependencies:	None
Conflicts:	BSW31400006
Supporting Material:	None

4.3.2.2 [BSW31400017] Usage of macros should be avoided

Initiator:	WP LIBRARIES
Date:	27-june-2008
Short Description:	Usage of macros should be avoided
Type:	New
Importance:	Low
Description:	The function should be declared as function or inline function. Macro #define should not be used
Rationale:	Macros do not specify argument type and return type so there is more chance of improper use
Use Case:	none
Dependencies:	None
Conflicts:	BSW31400006
Supporting Material:	None

4.3.2.3 [BSW31400018] A library function can only call library functions

Initiator:	WP LIBRARIES, WP Software Architecture and OS
Date:	08-december-2008
Short Description:	A library function can only call library functions
Type:	New
Importance:	high
Description:	A library function shall not call any BSW modules functions, e.g. the DET. A library function can call other library functions.
Rationale:	A library function shall be re-entrant. Other BSW modules functions may not be re-entrant.
Use Case:	Multi-core architecture

	Memory protection scheme
Dependencies:	BSW31400009
Conflicts:	None
Supporting Material:	None

5 Requirements Specification of CRC library

5.1 Functional Overview

The CRC Library provides functions for 8 bit, 16 bit and 32 bit CRC (cyclic redundancy check) calculations. The CRC library can be scaled in terms of

- Table based calculation (fast, but higher code size)
- Runtime calculation (slow, but smaller code size)
- Different standard CRC generator polynomials

Hardware supported CRC calculation may be supported in the future.

5.2 Functional Requirements

5.2.1 General

5.2.1.1 [BSW08525] Support of standard polynomials

Initiator:	BMW
Date:	10.03.2005
Short Description:	Support of standard polynomials
Type:	New
Importance:	High
Description:	The CRC library shall support the following generator polynomials: CRC 8-bit SAE-J850 (0x1D) CRC 16-bit CRC-CCITT CRC 32-bit Ethernet IEEE-802 (0x04C11DB7)
Rationale:	These polynomials are considered to be standard.
Use Case:	CRC 8-bit : Detect errors/inconsistent data in Communication CRC16 and 32-bit : Detect errors/inconsistent data in NVRAM/RAM
Dependencies:	None
Conflicts:	None

5.3 Non-Functional Requirements (Qualities)

5.3.1 General

5.3.1.1 [BSW08518] CRC calculation method complexity

Initiator:	BMW
Date:	18.02.2005
Short Description:	CRC calculation method complexity

Type:	New
Importance:	High
Description:	The CRC Library shall provide different calculation methods (algorithm), optimizing either performance or memory usage (e.g. for runtime calculation).
Rationale:	Allow for optimization of code size or execution time depending on specific ECU requirements.
Use Case:	None
Dependencies:	None
Conflicts:	None
Supporting Material:	None

5.3.1.2 [BSW08526] CRC standard calculation methods

Initiator:	BMW
Date:	18.02.2005
Short Description:	CRC standard calculation methods
Type:	New
Importance:	High
Description:	The CRC Library shall support current standards of CRC calculation: • table based • runtime calculated • hardware based (may be supported in the future)
Rationale:	Allow for optimization of code size or execution time depending on specific ECU requirements.
Use Case:	None
Dependencies:	None
Conflicts:	None
Supporting Material:	None

5.3.1.3 [BSW08521] CRC routine schedulability

Initiator:	BMW
Date:	18.02.2005
Short Description:	CRC routine schedulability
Type:	New
Importance:	High
Description:	All CRC routines shall allow step-by-step-wise calculation of a large data block that is passed by start address, length and start value.
Rationale:	CRC calculation of large data blocks without blocking the whole system.
Use Case:	Example: The CRC calculation of a 4k ROM data block shall be performed. If the CRC routine would calculate the WHOLE block within one call, the watchdog would not be triggered anymore and cause a reset. Thus, the calculation has to be done in several steps (e.g. 16 byte wise)
Dependencies:	General requirement: [BSW00313] Deadlock prevention
Conflicts:	None
Supporting Material:	None

6 Requirements Specification of SW-C End-to-End Communication Protection Library

6.1 Functional Overview

The SW-C End-to-End Communication Protection Library (in short: E2E library) provides functions for detecting errors in (safety-related) communication between safety-related SW-Cs. The protection is done by means of protecting of safety-related data elements exchanged between SW-Cs, and the responsibility to protect/check the signal is given to the SW-Cs (application), which call directly the E2E library. The library is supposed to work over a priori any communication stack used for inter-SW-C communication, which are currently FlexRay, CAN and LIN. In future, when further communication stacks are added, there might be a need to add further E2E profiles.

6.2 Functional Requirements

6.2.1 General

6.2.1.1 [BSW08527] Protection mechanisms for inter SW-C communication up to ASIL D

Initiator:	BMW
Date:	03.09.2008
Short Description:	Protection mechanisms for inter SW-C communication up to ASIL D
Type:	New
Importance:	High
Description:	E2E library shall provide a set of safety protocols, in a form of library functions invoked by SW-Cs. The protocols shall provide the error detection rate sufficient for transmitting safety-related data up to ASIL D.
Rationale:	E2E communication protection is state-of-art in automotive safety-related series products.
Use Case:	Communication between main chassis ECU SW-C and power steering ECU SW-C
Dependencies:	None
Conflicts:	None

6.2.1.2 [BSW08528] E2E profiles

Initiator:	BMW
Date:	03.09.2008
Short Description:	E2E profiles
Type:	New
Importance:	High
Description:	E2E library shall provide E2E profiles, where each E2E profile completely defines a particular safety protocol (including header structure, behavior as state machines, error handling etc). Each E2E profile shall be an efficient

	solution for a particular communication stack used underneath (which are either FlexRay, CAN, or LIN), and the ASIL rating of the exchanged signals. Note: Each communication stack (e.g. FlexRay) has different error rates which depend on for example: - Bit error rate on channel - FIT values of HW - number of ECUs - topology (e.g. CAN->Gateway->FR) - open/closed transmission system - frequency of safety related messages The profiles, based on proven-in-use solutions, are supposed to cover typical combinations of above factors.
Rationale:	Too many standardized profiles reduce interoperability between applications. Moreover, it introduces too much specification and development efforts.
Use Case:	Protocol with 8-bit CRC for CAN, and 16-bit for long FlexRay signals.
Dependencies:	None
Conflicts:	None

6.2.1.3 [BSW08529] Available mechanisms in the E2E profiles

Initiator:	BMW
Date:	03.09.2008
Short Description:	Available mechanisms in the E2E profiles
Type:	New
Importance:	High
Description:	Each of the defined E2E profiles shall use an appropriate subset of the following mechanisms: 1. Sequence number (different sizes possible; in the state-of art it is alternatively called alive counter or consecutive number 2. Checksum: CRC8, CRC16, CRC32 3. IDs: Source ID, Destination ID, Data ID 4. Timeouts: reception timeout In other words, mechanisms not listed shall not be used. In each E2E profile, the sequence number and IDs, if used, should be all part of the transmitted data element. However, it is allowed that in a given profile, the sequence number and/or IDs are “hidden” (not transmitted), but included in the checksum.
Rationale:	These are typical measures used by safety protocols, and they can be realized by AUTOSAR.
Use Case:	Mechanisms used in an exemplary profile: 4-bit sequence counter, CRC8, Data ID, timeout
Dependencies:	None
Conflicts:	None

6.2.1.4 [BSW08530] Profile definition and interoperability

Initiator:	BMW
Date:	03.09.2008
Short Description:	Profile definition and interoperability
Type:	New
Importance:	High
Description:	Each E2Eprofile defined within the library shall: 1. Have a unique ID (IDs from E2E_01 to E2E_16 are reserved for standard AUTOSAR profiles).

	2. Define precisely a set of mechanisms (e.g. CRC of a particular polynomial) 3. Define its behavior in a semi-formal way (including state machines, error handling etc).
Rationale:	A protocol is not just a list of mechanisms (e.g CRC8 + sequence number) , but the whole logic managing the process. Standardization of header is by far not sufficient. Standardized behaviour is needed to achieve interoperability.
Use Case:	Usually one state machine per profile per communicating partner (sender, receiver, client server) is sufficient. ECU1 and ECU2 communicating. ECU1 has different implementation of E2E library than ECU2.
Dependencies:	None
Conflicts:	None

6.2.1.5 [BSW08531] Reuse of CRC routines

Initiator:	BMW
Date:	03.09.2008
Short Description:	Reuse of CRC routines
Type:	New
Importance:	High
Description:	E2E library shall not provide CRC routine implementations. Instead, it shall call the CRC routines of CRC library (document UID 016).
Rationale:	Reuse of existing AUTOSAR functionality
Use Case:	CRC8 of CRC library to be used in one of profiles for protecting CAN communication.
Dependencies:	None
Conflicts:	None

6.2.1.6 [BSW08533] E2E CRC polynomials to be different to those used by network stack

Initiator:	Fiat / CRF
Date:	03.09.2008
Short Description:	E2E CRC polynomials to be different to those used by network stack
Type:	New
Importance:	High
Description:	CRC used in each E2E profile shall be different than the CRC used by the underlying communication protocols (FlexRay, CAN, LIN), for which the given profile is supposed to be used with.
Rationale:	Using the same polynomials twice (once in com stack and again in E2E) provides significantly lower joint detection rate than using two different polynomials. The polynomials available in AUTOSAR R3.1 are not optimal for E2E anyway.
Use Case:	If profile X is supposed to be used only for FlexRay, then its CRC shall be different than the one of FlexRay.
Dependencies:	None
Conflicts:	None

6.2.1.7 [BSW08534] Separate Error flag and error counters

Initiator:	Fiat/CRF

Date:	09.09.2008
Short Description:	Separate Error flag and error counters
Type:	New
Importance:	High
Description:	E2E library shall provide to the application layer separate errors flag and error counters for each type of detected communication failure. In other words if E2E profile X is supposed to use the sequence counter and CRC, then the following error flag shall be available to the application layer: • Data corruption • Wrong sequence • Repetition • Data loss
Rationale:	Error handling strategies are “application dependent”, and cannot be “a priory defined”
Use Case:	Enable error-dependent reaction of the SW-C using E2E library.
Dependencies:	None
Conflicts:	None

6.2.1.8 [BSW08535] Data element provided to the application layer

Initiator:	Fiat/CRF
Date:	09.09.2008
Short Description:	Data element provided to the application layer
Type:	New
Importance:	medium
Description:	E2E library should provide to the application layer the last received data element and the last correct data element.
Rationale:	Provision of means to design different error reaction strategies based upon the type of detected failure
Use Case:	Use of the last correct data in case of data corruption. Use of interpolation algorithm between last received data and last correct data in case of data loss
Dependencies:	6.2.1.8
Conflicts:	None

6.2.1.9 [BSW08536] CRC computation over E2E protection mechanism

Initiator:	Fiat / CRF
Date:	16.09.2008
Short Description:	CRC computation over E2E protection mechanism
Type:	New
Importance:	High
Description:	Either SW-C or E2E Library shall compute the intermediate CRC over application data element. E2E library shall use as initial CRC value the intermediate CRC and shall compute the CRC over the sequence counter (if it is used) and IDs (if used).
Rationale:	In case of complex data elements, the E2E library cannot compute the CRC over the data element (because the library does not know the layout of the data element – a data type may be e.g. an array of pointers to data structures, which does not occupy a consecutive address space). In such a case, the application needs to compute the CRC over the data element, and pass the computed CRC to the library. However, regardless who invokes the CRC computation (SW-C or library), the CRC used is the one of the used E2E profile.
Use Case:	

Dependencies:	None
Conflicts:	None

6.2.1.10 [BSW08537] Tolerance in SW-Cs against undetected single errors in signals

Initiator:	Exida
Date:	10.12.2008
Short Description:	Tolerance in SW-Cs against undetected single errors in signals
Type:	New
Importance:	High
Description:	SW-Cs shall tolerate at least one incorrect signal, in which error has not been detected by SW-Cs.
Rationale:	Requiring that 100% errors are detected by E2E protocol has high impact on implementation of E2E library (e.g. requiring SW or/and HW redundancy). Allowing to have a signal (in a sequence of received signals) with an error that is not detected by E2E
Use Case:	Example 1: multiple bit errors (e.g. 5 corrupted bits) that generate the same CRC as the original signal. Example 2: random HW fault or SW fault in E2E library causing that CRC Sequence Counter computation does not detect an error.
Dependencies:	The tolerance can be reached by means of an appropriate safety concept of an application.
Conflicts:	None