VECTOR

Safety Manual

CBD1400351 D04

ID	SPECDOC22553
Status	Released
Generated on	2016-09-16 11:23

Contents

Safety Manual

Version	Date	Author	Remarks
1.00.00	2015-11-13	Jonas Wolf	Initial creation
1.01.00	2015-12-18	Jonas Wolf	Information about ASIL added.
1.02.00	2016-01-29	Jonas Wolf	Improvements in formulation.
1.03.00	2016-02-25	Jonas Wolf	Improvements in formulation.
1.03.01	2016-03-30	Jonas Wolf	Review findings incorporated.
1.03.02	2016-05-13	Jonas Wolf	Added hint on hardware-software integration (SMI-4).
1.03.03	2016-07-20	Hartmut Hoerner	Added SMI related to interrupt handling.

1 General Part

1.1 Introduction

1.1.1 Purpose

This document describes the assumptions made by Vector during the development of MICROSAR Safe as Software Safety Element out of Context (SEooC). This document provides information on how to integrate MICROSAR Safe into your safety-related project.

This document is intended for the user of MICROSAR Safe. It shall be read by project managers, safety managers, and engineers to allow proper integration of MICROSAR Safe.

1.1.2 Scope

This document adds additional information to the components that are marked with an ASIL in the delivery description provided by Vector. Neither QM Vector components, nor components by other vendors are in the scope of this document.

Vector assumes that hardware and compiler manuals are correct and complete.
Vector uses the hardware reference manuals and compiler manuals for the development of MICROSAR Safe. Vector has no means to verify correctness or completeness of the hardware and compiler manuals.
Example information that may be critical from these manuals is the register assignment by compiler. This information is used to built up the context that is saved and restored by the operating system.
The compiler manual from the compiler version specified for the project is considered. The considered hardware manuals are documented in the Technical Reference of the hardware-specific component.

A general description of Vector's approach to ISO 26262 is described in [2]. This document is available on request.

1.1.3 Definitions

The words *shall*, *shall not*, *should*, *can* in this document are to be interpreted as described here:

Shall means that the definition is an absolute requirement of the specification.

Shall not means that the definition is an absolute prohibition of the specification.

Should means that there may exist valid reasons in particular circumstances to ignore a particular definition, but the full implications must be understood and carefully weighed before choosing a different course.

Can means that a definition is truly optional.

The user of MICROSAR Safe can deviate from all constraints and requirements in this Safety Manual in the responsibility of the user of MICROSAR Safe, if equivalent measures are used.
If a measure is equivalent can be decided in the responsibility of the user of MICROSAR Safe.

1.1.4 References

No.	Source	Title	Version
[1]	ISO	ISO 26262 Road vehicles — Functional safety (all parts)	2011/2012
[2]	Vector	ISO 26262 Compliance Document	1.2.1
[3]	Vector	MICROSAR Safe Product Information	1.1
[4]	Vector	MICROSAR Safe Silence Verifier Technical Reference	1.4

1.1.5 Overview

This document is automatically generated. The content of this document depends on the components and microcontroller of your delivery. This document is thus valid only for the delivery from Vector that it is included in.

The structure of this document comprises:

- a general section that covers all assumptions and constraints that are always applicable, and
- a microcontroller specific section that covers all aspects of the selected microcontroller (only if microcontroller specific components are part of the delivery), and
- a section for each component that covers its constraints and necessary verification steps.

Vector's assumptions on the environment of the MICROSAR Safe components as well as the integration process are described.

Vector developed MICROSAR Safe as Safety Element out of Context for projects demanding ASIL D software. All requirements in this document apply independently from the actual highest ASIL of the project.

1.2 Concept

MICROSAR Safe comprises a set of components developed according to ISO 26262. These components can be combined - together with other measures - to build a safe system according to ISO 26262.

Please read the Product Information MICROSAR Safe [3] first.

1.2.1 Technical Safety Requirements

These are the assumed technical safety requirements on the Safety Element out of Context MICROSAR Safe. These requirements are expected to match the requirements in the actual item development.

All technical safety requirements are assigned an ASIL D to service as many projects as possible.

No fault tolerant time intervals are given. Timing depends on the used hardware and its configuration. It is assumed that the user configures MICROSAR Safe adequately for the intended use.

No safe state is defined since MICROSAR Safe allows the user to define the desired behavior in case of a detected fault.

1.2.1.1 Initialization

TSR-1 The system shall initialize the CPU, MPU, watchdog, and operating system.
Rationale: Initialization of the hardware, e.g. clocks, memory protection, scheduling etc. is necessary to enable the other safety requirements.
MICROSAR Safe Feature: The ECU State Manager (EcuM) is responsible for performing the configured driver initialization. After initialization the EcuM starts the operating system. The EcuM distributes the post-built loadable configuration information within the ECU. The documentation of the operating system describes which parts of the CPU it initializes. The startup code and main function is in the responsibility of the user of MICROSAR Safe.

1.2.1.2 Self-test

TSR-2 The system shall perform self-tests based on the requirements of the system.
Rationale: It may be necessary to periodically test individual components of the system to detect latent faults.
MICROSAR Safe Feature :The operating system provides a function to self-test the effectiveness of the MPU settings.
The ECU State Manager services callouts to user code that can check RAM consistency after wakeup.
Other hardware self-tests are usually performed by MCAL components not developed by Vector.
End-to-end protection of communication provides its own fault detection mechanisms.

1.2.1.3 Reset of ECU

TSR-3 The system shall reset itself in case of a detected fault.
Rationale: Resetting the CPU of the ECU is in most cases an appropriate measure to achieve a safe state.
It is assumed that the reset state of a microcontroller leads to the safe state of the ECU and system, since a reset may occur at any time due to e.g. EMC.
MICROSAR Safe Feature: MICROSAR Safe does not reset the ECU on its own (there may be exceptions for the operating system).

Functionality from ECU State Manager (setting the shutdown target) can be used to reset the ECU instead, if this is the intended fault reaction.

1.2.1.4 Non-volatile memory

The system must be designed in a way that in case of the absence of nonâ€‘volatile data it is still safe (e.g. safe state or degradation).
It cannot be assured that data is saved completely or at all because a reset or loss of energy might happen at any time, e.g. brown-out, black-out.
This also implies that it is in general impossible to guarantee that the latest information is available in the non-volatile memory, e.g. the system is reset before memory stack is even notified to write data to non-volatile memory.
Thus, safety-related functionality may not rely on the availability of data in non-volatile memory.
Since the availability of data in non-volatile memory cannot be guaranteed in any case, the only sensible use-case is reading safety-related calibration data.
Writing of data into non-volatile memory must be verified to assure that the information is available in non-volatile memory. Verification can only be done manually in a protected environment, e.g. at end of line, in a workshop, etc.
ECU software cannot verify if data was written, since at any time a reset could occur and the information that had to be written is lost immediately.
Reading of data does not modify data stored in non-volatile memory. Thus, reading can be used by safety-related functionality. The memory stack has to assure that the read data is identical to the data stored in non-volatile memory.
The absence of data still has to be handled by the application.
The availability may be increased by e.g. redundant storage.

1.2.1.4.1 Saving data

TSR-4 The system shall save information in non-volatile memory.
Rationale: see text above.
MICROSAR Safe Feature: The intended block is written with the information provided by the application.

1.2.1.4.2 Loading data

TSR-5 The system shall retrieve the last stored information from non-volatile memory.
Rationale: see text in above.
MICROSAR Safe Feature: The intended block is read and the information is provided to the application.
The last or any previous completely stored block in non-volatile memory is returned by memory stack.
If CRC or ECC protections do not match block data an error is returned.
If data is stored redundantly, the redundant information is returned.
If no completely stored block is found, an error is returned.

1.2.1.5 Scheduling

1.2.1.5.1 Deterministic, hard real-time scheduling

TSR-6 The system shall execute the specified functions within their respective hard timing limits.
Rationale: Hard real-time scheduling may be used for scheduling safety mechanisms implemented in software.
This requirement is even more important for fail-operational systems, where one function may have to work, while another blocks the processor.
MICROSAR Safe Feature: The immediate priority ceiling protocol specified by AUTOSAR is capable of performing this task. The schedule tables specified by AUTOSAR may also be used on top of the scheduling algorithm.
The operating system of MICROSAR Safe implements the scheduling algorithm according to the methods for ASIL D required by ISO 26262.

1.2.1.6 Partitioning

1.2.1.6.1 Memory partitioning

TSR-7 The system shall protect software applications from unspecified memory access.
Rationale: Partitioning in software is often introduced because of different quality levels of software and different responsibilities of software development on one ECU.
Memory partitioning relies on the available MPU in hardware for the effectiveness of the mechanism.
MICROSAR Safe Feature: Memory partitioning and context switching using AUTOSAR Operating Feature System SC3 mechanisms is implemented according to the methods for ASIL D required by ISO 26262.
Adequate configuration of the memory partitions is in the responsibility of the user of MICROSAR Safe.

1.2.1.6.2 Time partitioning

1.2.1.6.2.1 Timing protection

TSR-8 The system shall detect timing faults in the software.
Rationale: Relying on a watchdog for timing protection is sometimes not sufficiently robust or efficient.
MICROSAR Safe Feature: The operating system of MICROSAR Safe implements the timing protection functionality of SC4 according to the methods for ASIL D required by ISO 26262.

1.2.1.6.2.2 Killing of applications

TSR-9 The system shall terminate software applications.
Rationale: In combination with the timing protection this allows the software to continue operation in case of a software fault.
MICROSAR Safe Feature: The operating system of MICROSAR Safe implements the termination of applications according to the methods for ASIL D required by ISO 26262.

1.2.1.7 Communication protection

1.2.1.7.1 Inter ECU communication

1.2.1.7.1.1 End-to-end protection

TSR-10 The system shall protect communication between its elements.
Rationale: Communication has to be protected against corruption, unintended replay and masquerading. The loss of a message must be detected.
This can be achieved using the end-to-end (E2E) protection
mechanism defined by AUTOSAR.
MICROSAR Safe Feature: MICROSAR Safe implements the E2E functionality according to
ISO 26262. This also includes the CRC library functionality.

1.2.1.7.1.2 Protection by cryptographic algorithms

TSR-11 The system shall protect communication between its elements using cryptographic hash algorithms to detect accidental corruption of the communication.
Rationale: Cyclic redundancy codes (CRC) provide a specified hamming distance given a polynomial and data block size.
Cryptographic hash functions provide a probabilistic statement on data corruption detection depending on the hash function, data block size and hash value size.
MICROSAR Safe Feature: The Cryptographic Service Manager of MICROSAR Safe is implemented according to the methods for ASIL D required by ISO 26262.
The Cryptographic Service Manager services the main function and functions to calculate a cryptographic hash function according to AUTOSAR specification.

1.2.1.7.2 Intra ECU communication

1.2.1.7.2.1 Intra OS application communication

TSR-16 The microcontroller software shall communicate within its applications.
Rationale: Software components need to communicate.
Protection of the memory against random hardware faults is expected by the system (e.g. via ECC RAM and lock-step mode).
MICROSAR Safe Feature: The RTE provides services to allow communication of software components within OS applications (intra-partition communication).

1.2.1.7.2.2 Inter OS application communication

TSR-12 The microcontroller software shall communicate between its applications.
Rationale: Multi-core systems may need to exchange safety-related information between applications.
Protection of the memory against random hardware faults is expected by the system (e.g. via ECC RAM and lock-step mode).
MICROSAR Safe Feature: The operating system of MICROSAR Safe implements the inter-OS application (IOS) functionality according to the methods for ASIL D required by ISO 26262.
The RTE provides services to allow communication between OS applications (inter-partition communication).

1.2.1.8 Watchdog services

1.2.1.8.1 Program flow monitoring

TSR-13 The system shall provide a mechanism to detect faults in program flow.
Rationale. Program flow can be corrupted by random hardware faults or software faults.
MICROSAR Safe Feature: MICROSAR Safe watchdog stack implements program flow monitoring functionality according to the methods for ASIL D required by ISO 26262.

1.2.1.8.2 Alive monitoring

TSR-14 The system shall provide a mechanism to detect stuck software.
Rationale: Alive monitoring is used to reset the software or controller in case it is unresponsive.
MICROSAR Safe Feature: MICROSAR Safe watchdog stack implements alive monitoring functionality according to the methods for ASIL D required by ISO 26262.

1.2.1.8.3 Deadline monitoring

TSR-15 The system shall provide a mechanism to detect deadline violations.
Rationale: Deadline monitoring using the watchdog stack can be used to implement timing monitoring.
MICROSAR Safe Feature: MICROSAR Safe watchdog stack implements deadline monitoring functionality according to the methods for ASIL D required by ISO 26262.

1.2.1.9 Peripheral in- and output

1.2.1.9.1 Peripheral input

TSR-17 The system shall read input values from peripheral devices.
Rationale: In- and output is the most common use case for safety mechanisms.
Some MCAL manufacturers call this feature safe acquisition.
MICROSAR Safe Feature: MCAL components used for peripheral access are not developed by Vector.

1.2.1.9.2 Peripheral output

TSR-18 The system shall write output values to peripheral devices.
Rationale: In- and output is the most common use case for safety mechanisms.
Some MCAL manufacturers call this feature safe actuation.
MICROSAR Safe Feature: MCAL components used for peripheral access are not developed by Vector.

1.2.2 Environment

1.2.2.1 Safety Concept

SMI-14
The user of MICROSAR Safe shall be responsible for the functional safety concept.

The overall functional safety concept is in the responsibility of the user of MICROSAR Safe. MICROSAR Safe can only provide parts that can be used to implement the

functional safety concept of the item.
It is also the responsibility of the user of MICROSAR Safe to configure MICROSAR Safe as intended by the user's safety concept.
The safety concept shall only rely on safety features explicitly described in this safety manual. If a component from MICROSAR Safe does not explicitly describe safety features in this safety manual, this component has been developed according to the methods for ASIL D to provide coexistence with other ASIL components.

- Example: NvM provides safety features for writing and reading of data, the lower layers, i.e. MemIf, Ea, Fee and drivers, only provide the ASIL for coexistence.

The safety concept shall **not** rely on functionality that is **not** explicitly described as safety feature in this safety manual. This functionality may fail silently in case of a detected fault.

- Example: If a potential out-of-bounds memory access, e.g. due to invalid input or misconfiguration, is detected the requested function will not be performed. An error via DET is only reported if error reporting is enabled.

SMI-1
The user of MICROSAR Safe shall adequately address hardware faults.

The components of MICROSAR Safe can support in the detection and handling of some hardware faults (e.g. using watchdog).
MICROSAR Safe does not provide redundant data storage.
The user of MICROSAR Safe especially has to address faults in volatile random access memory, non-volatile memory, e.g. flash or EEPROM, and the CPU.
MICROSAR Safe relies on the adequate detection of faults in memory and the CPU by other means, e.g. hardware. Thus, Vector recommends using lock-step CPUs together with ECC memory.
See also SMI-14.

SMI-10
The user of MICROSAR Safe shall ensure that the reset or powerless state is a safe state of the system.

This assumption is added to this Safety Manual, because it is used in Vector's safety analyses and development process.

SMI-20
The user of MICROSAR Safe shall implement a timing monitoring using e.g. a watchdog.

The components of MICROSAR Safe do not provide mechanisms to monitor their own timing behavior.
The watchdog stack that is part of MICROSAR Safe can be used to fulfill this assumption.
If the functional safety concept also requires a logic monitoring, The watchdog stack that is part of MICROSAR Safe can be used to implement it.
The watchdog is one way to perform timing monitoring. Today the watchdog is the most common approach. In future there may be different approaches e.g. by monitoring using a

different ECU.
See also SMI-14.

SMI-98
The user of MICROSAR Safe shall ensure an end-to-end protection for safety-relevant communication between ECUs.

The communication components of MICROSAR Safe do not assume sending or receiving as a safety requirement, because considered faults can only be detected using additional information like a cycle counter. Vector always assumes that an end-to-end protection or equivalent mechanism is implemented on application level.
Considered faults in communication are:

- Failure of communication peer
- Message masquerading
- Message corruption
- Unintended message repetition
- Insertion of messages
- Re-sequencing
- Message loss
- Message delay

This requirement can be fulfilled by e.g. using the end-to-end protection wrapper for safety related communication.

SMI-11
The user of MICROSAR Safe shall ensure data consistency for its application.

Data consistency is not automatically provided when using MICROSAR Safe. MICROSAR Safe only provides services to support enforcement of data consistency. Their application is in the responsibility of the user of MICROSAR Safe.
To ensure data consistency in an application, critical sections need to be identified and protected.
To identify critical sections in the code, e.g. review or static code analysis can be used.
To protect critical sections, e.g. the services to disable and enable interrupts provided by the MICROSAR Safe operating system can be used.
To verify correctly implemented protection, e.g. stress testing or review can be used.
Note the AUTOSAR specification with respect to nesting and sequence of calls to interrupt enabling and disabling functions.

1.2.2.2 Use of MICROSAR Safe Components

SMI-2

The user of MICROSAR Safe shall adequately select the type definitions to reflect the hardware platform and compiler environment.

The user of MICROSAR Safe is responsible for selecting the correct platform types (PlatformTypes.h) and compiler abstraction (Compiler.h). Especially the size of the predefined types must match the target environment.
Example: A uint32 must be mapped to an unsigned integer type with a size of 32 bits.
The user of MICROSAR Safe can use the platform types provided by Vector. Vector has created and verified the platform types mapping according to the information provided by the user of MICROSAR Safe.

SMI-12
The user of MICROSAR Safe shall initialize all components of MICROSAR Safe prior to using them.

This constraint is required by AUTOSAR anyway. It is added to this Safety Manual, because Vector assumes initialized components in its safety analyses and development process.
Correct initialization can be verified, e.g. during integration testing.

SMI-16
The user of MICROSAR Safe shall only pass valid pointers at all interfaces to MICROSAR Safe components.

Plausibility checks on pointers are performed by MICROSAR Safe (see also SMI-18), but they are limited. MICROSAR Safe components potentially use provided pointers to write to the location in memory.
Also the length and pointer of a buffer provided to a MICROSAR Safe component need to be consistent.
This assumption also applies to QM as well as ASIL components.
This can e.g. be verified using static code analysis tools, reviews and integration testing.

SMI-18
The user of MICROSAR Safe shall enable plausibility checks for the MICROSAR Safe components.

This setting is necessary to introduce defensive programming and increase robustness at the interfaces as required by ISO 26262.
This setting can be found at /MICROSAR/EcuC/EcucGeneral/EcuCSafeBswChecks in the DaVinci Configurator.
This setting is enforced by an MSSV plug-in.
This setting does not enable error reporting to the DET component.

SMI-1725
The user of MICROSAR Safe shall configure and use the interrupt system correctly.

The user of MICROSAR Safe is responsible for a correct and consistent configuration and usage of the interrupt system.
Especially the following topics shall be verified:

- Consistent configuration of interrupt category, level and priority in OS and MCAL modules
- Correct assignment of logical channels/instances to interrupt vectors in case of MCAL modules with multiple channels/instances
- The interrupt controller is configured in a mode which processes interrupts of the same level sequentially to avoid unbounded interrupt nesting

1.2.2.3 Partitioning

SMI-9
The user of MICROSAR Safe shall ensure that for one AUTOSAR functional cluster (e.g. System Services, Operating System, CAN, COM, etc.) only components from Vector are used.

This assumption is required because of dependencies within the development process of Vector.
This assumption does not apply to the MCAL or the EXT cluster.
Vector may have requirements on MCAL or EXT components depending on the upper layers that are used and provided by Vector. For example, the watchdog driver is considered to have safety requirements allocated to its initialization and triggering services. Details are described in the component specific parts of this safety manual.
This assumption does not apply to components that are not provided by Vector.
In case the partitioning solution is used, this assumption only partially applies to the System Services cluster. Only the Watchdog Manager and Watchdog Interface need to be used from Vector then, because the Watchdog Manager and Watchdog Interface will be placed in separate memory partitions apart from the other System Services components.

SMI-32
The user of MICROSAR Safe shall provide an argument for coexistence for software that resides in the same partition as components from MICROSAR Safe.

Vector considers an ISO 26262-compliant development process for the software as an argument for coexistence (see [1] Part 9 Clause 6). Vector assumes that especially freedom from interference with respect to memory is provided by an ISO 26262-compliant development process.
Redundant data storage as the only measure by the other software is not considered a sufficient measure.
If ASIL components provided by Vector are used, this requirement is fulfilled.

SMI-99
The user of MICROSAR Safe shall verify that the memory mapping is consistent with the partitioning concept.
The volatile data of every component shall be placed in the associated memory partition. This can be verified e.g. by review of the linker map file.
The memory sections for each component placed in RAM can be identified <MIP>_START_SEC_VAR[_<xxx>], where <MIP> is the Module Implementation Prefix of the component.

1.2.2.4 Resources

SMI-33
The user of MICROSAR Safe shall provide sufficient resources in RAM, ROM, stack and CPU runtime for MICROSAR Safe.

Selection of the microcontroller and memory capacities as well as dimensioning of the stacks is in the responsibility of the user of MICROSAR Safe.
If MICROSAR Safe components have specific requirements, these are documented in the respective Technical Reference document.

1.2.3 Process

SMI-15
The user of MICROSAR Safe shall follow the instructions of the corresponding Technical Reference of the components.

Especially deviations from AUTOSAR specifications are described in the Technical References.
If there are constraints for the implementation of an exclusive area, these are described in the Technical References.

SMI-5
The user of MICROSAR Safe shall verify all code that is modified during integration of MICROSAR Safe.

Code that is typically modified by the user of MICROSAR Safe during integration comprises generated templates, hooks, callouts, or similar.
This assumption also applies if interfaces between components are looped through user-defined functions.
Vector assumes that this verification also covers ISO 26262:6-9. Vector assumes that modified code that belongs to a Vector component, e.g. EcuM callouts or OS trace hooks can at least coexist with this component, because no separation in memory or time is implemented.
Example: Callouts of the EcuM are executed in the context of the EcuM.
Non-trusted functions provided by the Vector operating system can be used to implement a separation in memory in code modified by the user of MICROSAR Safe.
Support by Vector can be requested on a per-project basis.

SMI-30
The user of MICROSAR Safe shall only modify source code of MICRSAR Safe that is explicitly allowed to be changed.

Usually no source code of MICROSAR Safe is allowed to be changed by the user of MICROSAR Safe.
The user of MICROSAR Safe can check if the source code was modified by e.g, comparing it to the original delivery.

SMI-8

The user of MICROSAR Safe shall verify generated functions according to ISO 26262:6-9.

Generated functions can be identified when searching through the generated code.
Support by Vector can be requested on a per-project basis.
An example of generated functions is the configured rules of the Basic Software Manager (BSWM). Their correctness can only be verified by the user of MICROSAR Safe. Please note, however, that BSWM does not provide safety features.
This requirement does not apply to MICROSAR SafeRTE.

SMI-19
The user of MICROSAR Safe shall execute the MICROSAR Safe Silence Verifier (MSSV).

Details on the required command line arguments and integration into the tool chain can be found in [4].
If the report shows “Overall Check Result: Fail", please contact the Safety Manager at Vector. See the Product Information MICROSAR Safe for contact details.

SMI-4
The user of MICROSAR Safe shall perform the integration (ISO 26262:6-10) and verification (ISO 26262:6-11) processes as required by ISO 26262.

Especially the safety mechanisms must be verified in the final target ECU.
Vector assumes that by performing the integration and verification processes as required by ISO 26262 the generated configuration data, e.g. data tables, task priorities or PDU handles, are sufficiently checked. An additional review of the configuration data is then considered not necessary.
Integration does not apply to a MICROSAR Safe component that consists of several subcomponents. This integration is already performed by Vector. The integration of subcomponents is validated during creation of the safety case by Vector based on the configuration handed in by the user of MICROSAR Safe.
However, integration of all MICROSAR Safe components in the specific use-case of the user of MICROSAR Safe is the responsibility of the user of MICROSAR Safe. This also includes the hardware-software integration in the context of the target ECU.
Support by Vector can be requested on a per-project basis.

SMI-100
The user of MICROSAR Safe shall ensure that a consistent set of generated configuration is used for verification and production.
Make sure that the same generated files are used for testing and production code, i.e. be aware that configuration can be changed without generating the code again.
Make sure that all generated files have the same configuration basis, i.e. always generate the MICROSAR Safe configuration for all components for a relevant release of the ECU software.

SMI-176
The user of MICROSAR Safe shall verify the integrity of the delivery by Vector.
Run the SIPModificationChecker.exe and verify that the source code, BSWMD and safety manual files are unchanged.

SMI-31
The user of MICROSAR Safe shall verify the consistency of the binary downloaded into the ECU's flash memory.

This also includes re-programming of flash memory via a diagnostics service. The consistency of the downloaded binary can be checked by the bootloader or the application. MICROSAR Safe assumes a correct program image.

SMI-3
The user of MICROSAR Safe shall evaluate all tools (incl. compiler) that are used by the user of MICROSAR Safe according to ISO 26262:8-11.

Evaluation especially has to be performed for the compiler, linker, debugging and test tools.
Vector provides a guideline for the evaluation of the Tool Confidence Level (TCL) for the tools provided by Vector (e.g. DaVinci Configurator).
Vector has evaluated the tools exclusively used by Vector during the development of MICROSAR Safe.

2 Safety Manual BswM

2.1 Safety features

This component does not provide safety features.

2.2 Configuration constraints

This component does not have configuration constraints.

2.3 Additional Verification measures

This component does not require additional verification measures.

2.4 Dependencies to other components

2.4.1 Safety features required from other components

This component does not require safety features from other components.

2.4.2 Coexistence with other components

SMI-55
This component requires coexistence with CanNm, EthSM, Com, Det, Dem, Nm, RTE, NvM, PduR, FrSM, Dcm, LinIf, LinTp, LinSM, EcuM, CanSM, J1939Dcm, J1939Nm, J1939Rm, Sd, WdgM and ComM components if the interface for those components is configured.

2.5 Dependencies to hardware

This component does not use a direct hardware interface.

3 Safety Manual Crc

3.1 Safety features

SMI-344
Crc provides the following safety features:

ID	Safety feature
CREQ-858	Crc shall provide a service to calculate 8-bit SAE-J1850 CRC.
CREQ-859	Crc shall provide a service to calculate 8-bit 0x2F CRC.
CREQ-860	Crc shall provide a service to calculate 16-bit CCITT CRC.
CREQ-861	Crc shall provide a service to calculate 32-bit IEEE-802.3 CRC.
CREQ-862	Crc shall provide a service to calculate 32-bit E2E Profile 4 CRC.

3.2 Configuration constraints

This component has no configuration constraints.

3.3 Additional verification measures

SMI-49
The user of MICROSAR Safe shall verify that the CRC is calculated for the intended data.

This includes the intended buffer and its size (see also SMI-16), start value and if it is the first call to the service.
Verification can be performed by the "magic check" (see AUTOSAR SWS Crc).

If Crc is used by a MICROSAR Safe component (e.g. E2E, NvM), this requirement is fulfilled for the MICROSAR Safe component.

3.4 Dependencies to other components

3.4.1 Safety features required from other components

This component does not require safety features from other components.

3.4.2 Coexistence with other components

This component does not require coexistence with other components.

It is assumed that the user of Crc has the adequate ASIL.

3.5 Dependencies to hardware

This component does not use a direct hardware interface.

4 Safety Manual Dem

SMI-2056
Please note: This version is not fully developed according to ISO 26262 ASIL D. Vector provides an alternative argument for freedom from interference with respect to memory though. The Silent methodology developed by Vector has been completely implemented for this component.

4.1 Safety features

This component does not provide safety features.

4.2 Configuration constraints

SMI-2054
The user of MICROSAR Safe shall configure the following attributes:

- Set */MICROSAR/Dem/DemGeneral/DemOBDSupport* to *DEM_OBD_NO_OBD_SUPPORT*.
- Set */MICROSAR/Dem/DemGeneral/DemJ1939Support* to *FALSE*.

These settings are enforced by an MSSV plugin.

4.3 Additional verification measures

SMI-2055
The user of MICROSAR Safe shall verify that tables *Dem_Cfg_*`<CallbackType>` only contain function pointers matching the signature listed for `<CallbackType>` in the table below.
If all events use the same callback, some of the tables might be optimized into a constant access macro *Dem_Cfg_Get*`<CallbackType>`. In that case the respective table does not exist. Instead, the user of MICROSAR Safe shall verify the validity of the function pointer returned by the macro *Dem_Cfg_Get*`<CallbackType>`.
The value *NULL_PTR* is valid for all callback types listed in below table.

The tables are generated into *Dem_Lcfg.c*.
The macros are generated into *Dem_Lcfg.h*.
`<CallbackType>` represents the entry in column *CallbackType* from below table.

CallbackType	Expected signature of the functions
CallbackClearEventAllowed	`Std_ReturnType <name>(boolean *IsAllowed)`
CallbackDtcStatusChanged	`Std_ReturnType <name>(uint32 DTC, Dem_DTCStatusMaskType oldStatus, Dem_DTCStatusMaskType newStatus)`
CallbackGetFdc	`Std_ReturnType <name>(sint8 *FDC)`
EventDataChanged	`Std_ReturnType <name>(void)`

EventStatusChanged	`Std_ReturnType <name>(Dem_EventStatusExtendedType oldStatus, Dem_EventStatusExtendedType newStatus)`
InitMonitorForEvent	`Std_ReturnType <name>(Dem_InitMonitorReasonType initReason)`
InitMonitorsForFunc	`Std_ReturnType <name>(void)`

SMI-2207
The user of MICROSAR Safe shall verify that the macro definitions in below table are defined to function pointers matching the listed signature.
The value *NULL_PTR* is valid for all callbacks listed in below table.

The macros are generated into *Dem_Lcfg.h*.

Configuration Macro	Expected signature of the function
DEM_CFG_GLOBALCBKDATA_FUNC	`Std_ReturnType <name>(Dem_EventIdType EventId)`
DEM_CFG_GLOBALCBKSTATUS_FUNC	`Std_ReturnType <name>(Dem_EventIdType EventId, Dem_EventStatusExtendedType oldStatus, Dem_EventStatusExtendedType newStatus)`
DEM_CFG_GLOBALCBKCONTROLDTCSETTING_FUNC	`Std_ReturnType <name>(boolean Status)`

SMI-2194
The user of MICROSAR Safe shall verify for all function pointers stored in table *Dem_Cfg_DataElementTable[]* that the function signature matches the *ElementKind* value according to the below table.
Dem_Cfg_DataElementTable[] is generated into *Dem_Lcfg.c*.

ElementKind	Expected signature of the function
DEM_CFG_DATA_FROM_CBK_STORED	`Std_ReturnType <name>(uint8 *data)`
DEM_CFG_DATA_FROM_CBK_CURRENT	`Std_ReturnType <name>(uint8 *data)`
DEM_CFG_DATA_FROM_CBK_STORED_WITH_EVENTID	`Std_ReturnType <name>(Dem_EventIdType EventId, uint8 *data)`
DEM_CFG_DATA_FROM_CBK_CURRENT_WITH_EVENTID	`Std_ReturnType <name>(Dem_EventIdType EventId, uint8 *data)`

other values	The function must be *NULL_PTR*

SMI-2195
The user of MICROSAR Safe shall verify that all function callbacks configured in table *Dem_Cfg_DataElementTable[]* write at most *ElementSize* bytes.

e.g. for a line { DEM_CFG_DATA_FROM_CBK_STORED, 4U, (Dem_ReadDataFPtrType)Rte_Call_CBReadData_DemDataElementClass_ReadData } verify that the application runnable triggered by the Rte when the Dem invokes operation *ReadData* on PortPrototype *DemDataElementClass* will at most write 4 bytes.

The table is generated into *Dem_Lcfg.c*.

SMI-2196
The user of MICROSAR Safe shall verify that all NV Block IDs listed in table *Dem_Cfg_MemoryBlockId[]* are intended to be used exclusively by Dem. This is required for SMI-22.

The table is generated into *Dem_Lcfg.c*

SMI-2197
The user of MICROSAR Safe shall verify that the configured NvM block size of the NV blocks referenced in table *Dem_Cfg_MemoryBlockId* is equal to the data size in table *Dem_Cfg_MemoryDataSize* for the same index.
The user of MICROSAR Safe shall verify that the NV block is also configured to use the RAM buffer referenced in *Dem_Cfg_MemoryDataPtr* for the same index. This is required for SMI-23.

Example: Verify that the NvM block descriptor referenced in *Dem_Cfg_MemoryBlockId[1]* is configured with data length *Dem_Cfg_MemoryDataSize[1]* and RAM buffer *Dem_Cfg_MemoryDataPtr[1]*

The tables *Dem_Cfg_MemoryBlockId*, *Dem_Cfg_MemoryDataSize* and *Dem_Cfg_MemoryDataPtr* are generated into *Dem_Lcfg.c*.
Refer to the NvM for the location of the tables generated by the NvM to verify the correct configuration.

SMI-2198
The user of MICROSAR Safe shall verify that when calling *Dem_GetEventExtendedDataRecord* or *Dem_GetEventFreezeFrameData*, either directly or by RTE operation *GetExtendedDataRecord* or *GetFreezeFrameData* of port interfaces *DiagnosticInfo*, *GeneralDiagnosticInfo* or *DiagnosticMonitor*

- The *DestBuffer* parameter shall point to a writeable memory area which can receive at least *sizeof(Dem_MaxDataValueSize)* bytes.

If an RTE is used, the type *Dem_MaxDataValueSize* is defined in *Rte_Type.h*.
If no RTE is used, the value of *DEM_CFG_SIZEOF_MAX_DATA_VALUE_TYPE* can be used to find the expected buffer size. This macro is generated into *Dem_Lcfg.h*.

SMI-2199
The user of MICROSAR Safe shall verify that table *Dem_Cfg_MemoryDataPtr[]* contains a pointer of type *Dem_Cfg_PrimaryEntryType* for the indices *DEM_CFG_MEMORY_PRIMARY_INDEX* (inclusive) through *DEM_CFG_MEMORY_PRIMARY_INDEX* + *DEM_CFG_GLOBAL_PRIMARY_SIZE* + *DEM_CFG_GLOBAL_SECONDARY_SIZE* (exclusive).

The macros *DEM_CFG_MEMORY_PRIMARY_INDEX*, *DEM_CFG_GLOBAL_PRIMARY_SIZE* and *DEM_CFG_GLOBAL_SECONDARY_SIZE* are generated into in *Dem_Lcfg.h*.
The type *Dem_Cfg_PrimaryEntryType* is generated into in *Dem_Lcfg.h*.
The table *Dem_Cfg_MemoryDataPtr[]* is generated into *Dem_Lcfg.c*.

SMI-2200
Only for OEM TMC, if time series snapshots are configured. Time series snapshots are enabled by creating configuration container *DemGeneral/DemTimeSeriesSnapshot*.

The user of MICROSAR Safe shall verify that table *Dem_Cfg_MemoryDataPtr[]* contains a pointer of type *Dem_Cfg_TimeSeriesEntryType* for the indices *DEM_CFG_MEMORY_TIME_SERIES_INDEX* (inclusive) through *DEM_CFG_MEMORY_TIME_SERIES_INDEX* + *DEM_CFG_GLOBAL_TIMESERIES_SNAPSHOTS_SIZE* (exclusive).

The macros *DEM_CFG_MEMORY_TIME_SERIES_INDEX* and *DEM_CFG_GLOBAL_TIMESERIES_SNAPSHOTS_SIZE* are generated into *Dem_Lcfg.h*.
The type *Dem_Cfg_TimeSeriesEntryType* is generated into *Dem_Lcfg.h*.
The table *Dem_Cfg_MemoryDataPtr[]* is generated into *Dem_Lcfg.c*.

SMI-2201
The user of MICROSAR Safe shall verify that *sizeof(Dem_Cfg_CommitBuffer)* is as large or larger than the entries in table *Dem_Cfg_MemoryDataSize[]*.

Dem_Cfg_CommitBuffer and *Dem_Cfg_MemoryDataSize[]* are generated into *Dem_Lcfg.c*.

4.4 Safety features required from other components

This component does not require safety features from other components.

4.5 Dependencies to hardware

This component does not use a direct hardware interface.

5 Safety Manual Det

5.1 Safety features

This component does not provide safety features.

5.2 Configuration constraints

This component does not have configuration constraints.

SMI-60
The Det should not be used in series production. If it is used in series production the extended debug features shall be switched off, because they are only relevant if a debugger is attached.

The user of MICROSAR Safe shall configure and verify the following attribute:

- /MICROSAR/Det/DetGeneral/DetExtDebugSupport = False

5.3 Additional Verification measures

This component does not require additional verification measures.

5.4 Dependencies to other components

5.4.1 Safety features required from other components

This component does not require safety features from other components.

5.4.2 Coexistence with other components

SMI-54
This component requires coexistence with Dlt component if the interface for this component is configured. Callouts used during series production must also provide an argument for coexistence.

5.5 Dependencies to hardware

This component does not use a direct hardware interface.

6 Safety Manual EcuM

6.1 Safety features

SMI-34
EcuM Flex provides the following safety features:

ID	Safety feature
CREQ-470	EcuM shall provide a service to initialize the ECU management.
CREQ-454	EcuM shall provide ECU initialization on multiple cores.
CREQ-543	EcuM shall perform validation of all postbuild configurable BSW module configuration parameters.
CREQ-375	EcuM shall provide a callout to set programmable interrupts during the startup phase.
CREQ-525	EcuM shall provide a callout to initialize driver prior the start of the OS.
CREQ-488	EcuM shall provide a callout to determine the Postbuild configuration data.
CREQ-505	EcuM shall provide a callout to initialize drivers prior Postbuild data is available.
CREQ-391	EcuM shall provide a callout to reset the ECU.
CREQ-381	EcuM shall provide a callout to generate a RAM Hash.
CREQ-440	EcuM shall provide a callout to check the RAM Hash.
CREQ-509	EcuM shall provide a callout to re-initialize drivers during a restart.
CREQ-445	EcuM shall provide a service to set the current shutdown target of the ECU.
CREQ-372	EcuM shall provide a service to initiate the ECU shutdown depending on the shutdown target.
CREQ-431	EcuM shall provide a callout to notify Errors from the ECU management.
CREQ-421	EcuM shall provide a service to complete the ECU shutdown process.
CREQ-699	EcuM shall indicate mode changes to the RTE.
CREQ-691	EcuM shall provide a service to request the run state.
CREQ-692	EcuM shall provide a service to release the run state.
CREQ-693	EcuM shall provide a service to release the post run state.
CREQ-690	EcuM shall provide a service to request the post run state.

Note: RAM Hash generation and checking callouts are only available when sleep modes are configured.

SMI-286
EcuM Fix provides the following safety features:

ID	Safety feature
CREQ-470	EcuM shall provide a service to initialize the ECU management.
CREQ-454	EcuM shall provide ECU initialization on multiple cores.
CREQ-543	EcuM shall perform validation of all postbuild configurable BSW module configuration parameters.

CREQ-375	EcuM shall provide a callout to set programmable interrupts during the startup phase.
CREQ-525	EcuM shall provide a callout to initialize driver prior the start of the OS.
CREQ-488	EcuM shall provide a callout to determine the Postbuild configuration data.
CREQ-505	EcuM shall provide a callout to initialize drivers prior Postbuild data is available.
CREQ-391	EcuM shall provide a callout to reset the ECU.
CREQ-381	EcuM shall provide a callout to generate a RAM Hash.
CREQ-440	EcuM shall provide a callout to check the RAM Hash.
CREQ-509	EcuM shall provide a callout to re-initialize drivers during a restart.
CREQ-445	EcuM shall provide a service to set the current shutdown target of the ECU.
CREQ-372	EcuM shall provide a service to initiate the ECU shutdown depending on the shutdown target.
CREQ-431	EcuM shall provide a callout to notify Errors from the ECU management.
CREQ-421	EcuM shall provide a service to complete the ECU shutdown process.
CREQ-699	EcuM shall indicate mode changes to the RTE.
CREQ-703	EcuM shall provide the ECU state machine.
CREQ-707	EcuM shall trigger the NvM write job in shutdown path.
CREQ-668	EcuM shall provide a callback which is called by the NvM to notify the end of the write all job.
CREQ-691	EcuM shall provide a service to request the run state.
CREQ-692	EcuM shall provide a service to release the run state.
CREQ-693	EcuM shall provide a service to release the post run state.
CREQ-690	EcuM shall provide a service to request the post run state.
CREQ-694	EcuM shall provide a service to kill all post run requests.
CREQ-695	EcuM shall provide a service to kill all run requests.
CREQ-707	EcuM shall trigger the NvM write job in shutdown path.
CREQ-668	EcuM shall provide a callback which is called by the NvM to notify the end of the write all job.

Note: RAM Hash generation and checking callouts are only available when sleep modes are configured.

SMI-38
If EcuM service to complete the shutdown process is called prior to initiate the shutdown process, no shutdown will be performed.

6.2 Configuration constraints

SMI-36
The user of MICROSAR Safe shall configure the following attribute:

- Set /MICROSAR/EcuM/EcuMFlexGeneral/EcuMEnableDefBehaviour to FALSE.

This setting is enforced by a MSSV plugin.

6.3 Additional verification measurse

SMI-39
The user of MICROSAR Safe shall verify the intended initialization procedure during integration testing.

The user of MICROSAR Safe can verify the intended initialization procedure by performing the following tests:

- Start the ECU and verify that the intended initalization routines are called. This needs to be verified for each Postbuild-selectable configuration.
- Corrupt the Postbuild data (but not corresponding CRC) in non-volatile memory and start the ECU. Then verify that the corruption is detected by EcuM.
- Start the ECU and verify that the intended Postbuild-selectable configuration is used by the EcuM. This needs to be verified for each Postbuild-selectable configuration.

A start of the ECU includes a "cold-start", reset as well as wake-up from sleep if applicable.

This requirement only applies if TSR-1 is considered a safety requirement.

SMI-35
The user of MICROSAR Safe shall verify the intended shutdown procedure during integration testing.

The user of MICROSAR Safe can verify the intended shutdown procedure by shutting down the ECU with all configured shutdown paths. A shutdown path is a call to the service that sets the current shutdown target with a relevant (e.g. combination used to achieve safe state) combination of its parameters. For each shutdown path the intended final state of the ECU (e.g. sleep, shutdown, reset) and the method of reset (e.g. using MCU or Watchdog) is used.

The user of MICROSAR Safe shall also consider the service to initiate the ECU shutdown depending on the shutdown target as a possible shutdown path.

The user of MICROSAR Safe shall also verify the default shutdown target.

This requirement only applies if TSR-3 is considered as a safety requirement.

SMI-40
The user of MICROSAR Safe shall verify that the memory region used for RAM hash generation and verification is as intended.

Verification can be e.g. performed by review.

6.4 Dependencies to other components

6.4.1 Safety features required from other components

SMI-42

The used operating system shall provide the service to get the core ID as safety feature.

If the operating system from MICROSAR Safe is used, this dependency is fulfilled.

This requirement only applies if TSR-1 is considered a safety requirement.

6.4.2 Coexistence with other components

SMI-41
This component requires coexistence with RTE, BswM, ComM, Dem, Gpt, MCU and Det components if the interface for those components is configured.

6.5 Dependencies to hardware

This component does not use a direct hardware interface.

7 Safety Manual Fee

7.1 Safety features

This component does not provide safety features.

7.2 Configuration constraints

SMI-2553
The user of MICROSAR Safe shall verify Fee's internal buffer size.

The size is generated to macro *FEE_INTERNAL_BUFFER_SIZE*.
The buffer size shall be at least the maximum of all configured partition's *AddressAlignment* values. Each partition is generated into the array *Fee_PartitionConfig_at*. Note that the alignments store the exponents, i.e. the power to which the value 2 shall be raised.

Verification can be performed e.g. by review.

SMI-2595
The user of MICROSAR Safe shall disable data conversion.

Set */MICROSAR/Fee/FeeSpecificFeatures/FeeDataConversionApi* to *FALSE*.

This setting is enforced by an MSSV plugin.

7.3 Additional verification measures

SMI-2594
The user of MICROSAR Safe shall pass a valid pointer to a data buffer to Fee's read functions.
The size of the data buffer must match the length provided in the read request, which is passed either explicitly (*Fee_Read*), or implicitly (i.e. one element of given data type) in *Fee_GetEraseCycle* and *Fee_GetWriteCycle*.

If the NvM by Vector and in case *Fee_Read* is used, it is sufficient to verify that it is only called by the NvM (indirectly via *MemIf_Read*).

Verification can be performed e.g. by review.

SMI-2554
The user of MICROSAR Safe shall pass a valid Fee configuration pointer to *Fee_InitEx*.

If *Fee_InitEx* is used instead of *Fee_Init* to initialize the component, the user has to make sure, the pointer points to a valid structure of correct type.

Verification can be performed e.g. by review.

7.4 Safety features required from other components

This component does not require safety features from other components.

7.5 Dependencies to hardware

This component does not use a direct hardware interface.

8 Safety Manual MemIf

8.1 Safety features

This component does not provide safety features.

8.2 Configuration constraints

This component does not have configuration constraints.

8.3 Additional verification measures

This component does not require additional verification measures.

8.4 Dependencies to other components

8.4.1 Safety features required from other components

This component does not require safety features from other components.

8.4.2 Coexistence with other components

SMI-311
This component requires coexistence with NvM, Det, Fee, and Ea components if the interface for those components is configured.

8.5 Dependencies to hardware

This component does not use a direct hardware interface.

9 Safety Manual NvM

9.1 Safety features

SMI-21
This component provides the following safety features:

ID	Safety feature
CREQ-724	NvM shall provide a service to read a single NvM block from NVRAM.
CREQ-725	NvM shall provide a service to write a single NvM block to NVRAM.
CREQ-730	NvM shall provide a service to read all possbile NvM blocks from NVRAM.
CREQ-731	NvM shall provide a service to write all possible NvM blocks to NVRAM.
CREQ-746	NvM shall provide configurable callbacks to synchronize block data.

NvM can detect missing, corruption and masquerading (lower layers provide the wrong block) of NvM blocks.

SMI-29
The user of MICROSAR Safe must design the system in a way that in case of the absence of nonâ€‘volatile data it is still safe (e.g. safe state or degradation). It cannot be assured by the memory stack that data is saved completely or at all because a reset or loss of energy might happen at any time, e.g. brown-out, black-out.
This also implies that it is in general impossible to guarantee that the latest information is available in the non-volatile memory, e.g. the system is reset before memory stack is even notified to write data to non-volatile memory.
Thus, safety-related functionality may not rely on the availability of data in non-volatile memory.
Since the availability of data in non-volatile memory cannot be guaranteed in any case, the only sensible use-case is reading safety-related calibration data.
Writing of data into non-volatile memory must be verified to assure that the information is available in non-volatile memory. Verification can only be done manually in a protected environment, e.g. at end of line, in a workshop, etc.
ECU software cannot verify if data was written, since at any time a reset could occur and the information that had to be written is lost immediately.
Reading of data does not modify data stored in non-volatile memory. Thus, reading can be used by safety-related functionality. The memory stack assures that the read data is identical to the data stored in non-volatile memory.
The availability may be increased by e.g. redundant storage.

9.2 Configuration constraints

SMI-25
The user of MICROSAR Safe shall configure and verify the following attributes for **each NvM block that contains safety-related data**:

- Set /MICROSAR/NvM/NvMBlockDescriptor/NvMBlockUseCrc to TRUE.
- Set /MICROSAR/NvM/NvMBlockDescriptor/NvMBlockCrcType to NVM_CRC32.

SMI-26
The user of MICROSAR Safe shall configure and verify the following attribute:

- Set /MICROSAR/NvM/NvMCommon/NvMUseBlockIdCheck to TRUE.

9.3 Additional verification measures

SMI-22
The user of MICROSAR Safe shall pass the intended block ID for reading and writing of a single NvM block. NvM cannot detect if an unintended block that is configured is provided by the user.
Verification can be performed during integration testing.

SMI-23
The user of MICROSAR Safe shall verify that the buffer passed for reading and writing of a single NvM block is valid and sufficiently large for the passed block ID.
Verification can be performed by a review of the generated configuration and the code passing the pointer and block ID to the NvM.

SMI-48
The user of MICROSAR Safe shall verify the size of the internal NvM buffer.

The buffer has the symbol name NvM_InternalBuffer_au8.
The buffer is generated when at least one of the following features is used:

- at least one block with explicit synchronization is configured
- repair of redundant blocks is enabled
- NvM internal CRC buffer is enabled

The buffer size shall be at least the size of the largest NVM block plus the size of the configured CRC value.

Verification can be performed e.g. by review.

9.4 Dependencies to other components

9.4.1 Safety features required from other components

SMI-28
The used Crc library shall provide the CRC calculation routines as safety feature.
If the Crc library from MICROSAR Safe is used, this dependency is fulfilled.

9.4.2 Coexistence with other components

SMI-27
This component requires coexistence with Dem, MemIf, RTE, BswM, and Det components if the interface for those components is configured.

9.5 Dependencies to hardware

This component does not use a direct hardware interface.

10 Safety Manual Rte

10.1 Safety features

SMI-323
This component provides the following safety features:

ID	Safety feature
CREQ-1024	Rte shall provide a service to initiate the transmission of data elements with last-is-best semantic for explicit S/R communication mode.
CREQ-1021	Rte shall provide a service to copy the received data element to a buffer with last-is-best semantic for explicit S/R communication mode.
CREQ-1022	Rte shall provide a service to get the value of the received data element with last-is-best semantic for explicit S/R communication mode.
CREQ-1023	Rte shall provide a service to initiate the transmission of data element with queued semantic for explicit S/R communication mode.
CREQ-1025	Rte shall provide a service to initiate the reception of data element queued semantic for explicit S/R communication mode.
CREQ-1031	Rte shall provide a service to read a data element for implicit S/R communication mode.
CREQ-1029	Rte shall provide a service to write a data element for implicit S/R communication mode.
CREQ-1041	Rte shall provide a service to get the reference of a data element to be written for implicit S/R communication mode.
CREQ-1037	Rte shall provide a service to get the status of a data element for implicit S/R communication mode.
CREQ-1042	Rte shall provide a service to initiate a client-server communication.
CREQ-1043	Rte shall provide a service to get the result of an asynchronous client-server call.
CREQ-1046	Rte shall provide a service to read Inter-Runnable Variables with explicit behavior.
CREQ-1048	Rte shall provide a service to write Inter-Runnable Variables with explicit behavior.
CREQ-1047	Rte shall provide a service to read Inter-Runnable Variables with implicit behavior.
CREQ-1044	Rte shall provide a service to write Inter-Runnable Variables with implicit behavior.
CREQ-1045	Rte shall provide a service to access per-instance memory.
CREQ-1146	Rte shall provide a service to access a single ports.
CREQ-1143	Rte shall provide a service to access an array of ports.
CREQ-	Rte shall provide a service to get the number of ports.

1145	
CREQ-1150	Rte shall provide a callback to copy data from a NVM buffer to RTE.
CREQ-1148	Rte shall provide a callback to copy data from RTE to a NVM buffer.
CREQ-1144	Rte shall provide a callback to get notified about a finished NVM job.
CREQ-1147	Rte shall provide a callback to get notified about a requested mirror initialization.
CREQ-1299	Rte shall provide Nv data communication.
CREQ-1038	Rte shall provide N:1 ("Fan-In") S/R communication.
CREQ-1102	Rte shall provide 1:M ("Fan-Out") S/R communication.
CREQ-1104	Rte shall provide N:1 C/S communication.
CREQ-1156	Rte shall support data exchange between different OS applications.
CREQ-1151	Rte shall provide the optional usage of one or several data transformers.

10.2 Configuration constraints

SMI-2066
The user of MICROSAR Safe shall disable online calibration and measurement during series production.

The RTE can be generated with online calibration and measurement enabled for series production, but they shall be made inoperable for normal operation. Vector's XCP can e.g. be disabled safely during runtime by ASIL software.

10.3 Additional verification measures

SMI-322
The user of MICROSAR Safe shall execute the RTE Analyzer.
The RTE Analyzer performs checks to identify faults in the generated RTE. Especially out-of-bounds accesses within the RTE are detected. Moreover it provides the user of MICROSAR Safe with feedback what was generated. This feedback can be used during integration testing.
Please see the Technical Reference of the RTE Analyzer how to execute it.

The RTE Analyzer lists RTE APIs that read or write end-to-end protected data (see SMI-98).

10.3.1 Guided integration testing

Some faults in the RTE Generator can only be found during integration testing. Vector assumes that the user of MICROSAR Safe performs an integration testing and verification of software safety requirements according to ISO 26262 Part 6 Clauses 10 and 11 (see also SMI-4). To support this integration testing the RTE Analyzer produces a configuration feedback report.
Please refer to the Technical Reference of the RTE Analyzer for a description of the configuration feedback report.
This section describes the requirements that must be fulfilled during integration testing and verification of software safety requirements.

10.3.1.1 BSW configuration

SMI-2124
The user of MICROSAR Safe shall ensure that the RTE and the operating system assume the same scheduling properties.

The scheduling properties of the RTE tasks depend on the configuration of the operating system. The scheduling properties are e.g. preemptability, core assignment or task priority.

The RTE Analyzer lists the scheduling properties in the configuration feedback report to assist in this integration step.

SMI-2129
The user of MICROSAR Safe shall ensure that the assumptions of the operating system and the RTE are the same with regards to the locking behavior of the spinlocks.

The RTE generator uses spinlocks from the operating system to protect inter-core communication. Spinlocks must not be called concurrently on the same core. The operating system optionally provides spinlocks that can prevent these concurrent accesses on the same core. If this protection by the operating system is not used, the RTE generator has to prevent concurrent calls to the spinlock APIs on the same core.

Verification can e.g. be performed by review of the configuration feedback report.

The RTE Analyzer lists spinlocks that are not protected by the RTE to assist in this integration step.

SMI-684
The user of MICROSAR Safe shall ensure that the configuration of COM and RTE are consistent.

The interfaces to the COM that are used for signal reception use *void* pointers as parameter. Inconsistencies between the configuration of the COM and the RTE might lead to memory corruption by the COM. During integration the size assumptions between the COM and the RTE shall be compared.

Verification can be performed by review of the generated configuration and/or static code analysis.

The RTE Analyzer lists relevant calls to assist in this integration step.

SMI-685
The user of MICROSAR Safe shall ensure that the configuration of NVM and RTE are consistent.

The interfaces to the NVM that are used to handle NV Block SWCs use *void* pointers as parameters. Inconsistencies between the configuration of the NVM and the RTE might lead to memory corruption by the RTE. During integration the size assumptions between the NVM and the RTE shall be compared.

Verification can be performed by review of the generated configuration and/or static code analysis.

The RTE Analyzer lists relevant calls to assist in this integration step.

10.3.1.2 Executable Entity Scheduling

SMI-2063
The user of MICROSAR Safe shall ensure that all safety-related executable entities are triggered with their correct conditions.

These conditions are:

- cylic triggers with cycle time and offset
- init triggers
- background triggers
- triggers fired by RTE APIs

If triggers have dependency on modes this has to be done for all modes as *Rte_Switch* APIs modify the scheduling.
Triggers can be decoupled by the minimum start interval functionality and the data reception filter functionality.
The scheduling of the executable entities also depends on the configuration of the operating system, the used controller, other running tasks and interrupt service routines and the resource usage of the entities that are implemented by the user.

Vector recommends not using the minimum start interval and the data reception filter functionality for safety-related runnables.
Vector recommends not using background triggers for safety-related functionality.
Vector recommends using cyclic scheduling without mode dependencies and using of a watchdog as safety mechanism for safety-related entities where possible.

Cyclic triggers are e.g. scheduled deterministically. Thus, an integration test verifying that safety-related functionality is scheduled at the expected times may be sufficient.

The RTE Analyzer lists the executable entities of SWCs and the tasks in which they are executed to assist in this integration step.

SMI-2128
The user of MICROSAR Safe shall ensure that reentrant runnables are reentrant.

Runnables can be called reentrantly from multiple tasks. Their implementation needs to support this use case.

Verification can e.g. be performed by review, static code analysis and/or integration testing.

The RTE Analyzer lists all runnables of SWCs that are called from concurrent tasks to assist in this integration step.

SMI-2064
The user of MICROSAR Safe shall ensure that the timeouts configured for blocking APIs that are used in safety-related executable entities are adequately addressed. If a timeout for a blocking API is used as a safety mechanism (i.e. e.g. no checkpoint with deadline monitoring in the task), the user of MICROSAR Safe shall also ensure that the timeout value is adequate.

Relevant timeouts are:

- timeouts of *Rte_Call* APIs
- timeouts of *Rte_Result* APIs
- timeouts of *Rte_Receive* APIs
- timeouts of *Rte_Feedback* APIs
- timeouts of *Rte_SwitchAck* APIs

The timeouts also depend on the configuration of the operating system, the used controller, other running tasks and interrupt service routines and the resource usage of entities that are implemented by the user.

Vector recommends not using blocking APIs in safety-related entities except for cross partition client-/server communication.
Vector strongly recommends not using blocking APIs without timeout.

A review may be sufficient to verify that timeout handling is implemented properly by the SWC.
If no other safety mechanism is in place, a test that the timeout is notified at the expected time by the RTE can be used as means of verification.

The RTE Analyzer lists the blocking APIs of SWCs to assist in this integration step.

SMI-2122
The user of MICROSAR Safe shall ensure that the correct implementation method has been chosen for every exclusive area.

Exclusive areas can be used to ensure data consistency (see SMI-11).
The implementation depends on the requirements of the application and on other factors like the expected duration of the exclusive area. Interrupt locks are typically faster than resources but can only be used for short sequences due to the blocking of interrupts. Operating system interrupts only block the interrupts of the operating system whereas **all** interrupts blocks all interrupts.

Verification can e.g. be performed by review, static code analysis and/or integration testing.

The RTE Analyzer lists the exclusive areas and their implementation method to assist in this integration step.

10.3.1.3 SWC Communication

SMI-324
The user of MICROSAR Safe shall ensure that the connections between SWCs are as intended.

Many types of faults can lead to a mix of connections between SWCs. These are unlikely and usually already addressed by straight forward integration testing.
The list of senders needs to be correct for every receiver and the subset of the received data needs to be correct.

Vector recommends the following RTE subset for safety-related SWCs:

- use only 1:1 connections.
- use the same datatype on both sides of a connection
- avoid data conversion

Information that is used from non-safety-related SWCs has to be checked for plausability. If such a data path is found during integration this is an indicator that your safety analysis has to be reconsidered.

Verification can be performed by review and/or an integration test testing the normal operation.

The RTE Analyzer list the connections between SWCs to assist in this integration step.

SMI-2065
The user of MICROSAR Safe shall ensure that inter-ECU sender-/receiver and inter-ECU client-/server communication work as expected.

This requires verification of:

- data needs to be routed to the correct ECU by the underlying communication stack. This includes 1:1, 1:N, N:1 and reception of partial signal data.

- both ECUs need to use the same data representation (datatypes, endianess, serialization)

Vector requires using E2E protection for safety-related signals.
Vector recommends using 1:1 connections.
Vector recommends always sending and receiving complete data elements.

Integration testing on vehicle network level using fault-injection can be used. Vector assumes that this is normally done to verify the effectiveness of the E2E protection.

The RTE Analyzer lists the APIs of SWCs with inter-ECU communication to assist in this integration step.

SMI-2123
The user of MICROSAR Safe shall ensure that all connected SWCs expect the same converted data.

The RTE offers conversions that can be applied to specific connections.

Vector recommends not using data conversion for safety-related connections.

Verification can e.g. be performed by review or integration testing of all data conversions.

The RTE Analyzer lists all APIs of SWCs that perform data conversion to assist in this integration step.

10.3.1.4 Usage of RTE Headers

SMI-2067
The user of MICROSAR Safe shall ensure that the *defines* and *typedefs* that are generated by the RTE match the expectations of the SWCs that use them.

Inconsistencies may lead to e.g, memory corruption when a runnable uses an RTE array datatype within its implementation and writes beyond the bounds of this array.
Moreover, different SWCs may have different assumptions with regards to the meaning of communicated values, e.g. if one SWC uses the symbolic name, another SWC the integral value of an enumerated type.

The following code is provided:

- Configured Application Error Defines for Client-/Server Communication
- Configured AUTOSAR Datatypes
- Configured Upper and Lower Limit Defines for Primitive Application Data Types
- Configured Init Value Defines for Sender-/Receiver Communication
- Configured InvalidValue Defines for Sender-/Receiver Communication
- Configured Enumeration Defines for CompuMethods

- Configured Mask Defines for CompuMethods
- Configured Mode Defines for Mode Communication
- Configured ActivationReasons

The defines and typedefs are part of the *Rte_Type.h*, *Rte_*<SWC>.*h* and *Rte_*<SWC>_Type.*h* headers.

Vector recommends using the headers generated by the RTE when a static code analysis is performed on the application code.
Vector recommends only using the *defines* instead of the defined values in the code.
Vector recommends using the *defines* only when needed (Mode, Application Error Defines).
Vector recommends reviewing the used *defines* for safety-related SWCs.

Verification for correct usage of datatypes may be performed by review and/or static code analysis. Consistent usage of *defines* can be verified by review and/or integration testing with all used values.

The RTE Analyzer verifies that all accesses within the RTE do not lead to memory corruption.

SMI-2071
The user of MICROSAR Safe shall ensure that the indirect API is used consistently.

Indirect API functionality consists of the APIs:

- *Rte_Port*
- *Rte_Ports*
- *Rte_NPorts*

The indirect API makes it possible to call different APIs through an array access. The indirect API functionality can be enabled individually per port.
A wrong configuration switch can easily lead to a call outside of the array returned by the Rte_Ports API.

Vector recommends not using the indirect API.

Verification can e.g. be performed by review that the intended APIs are returned.

The RTE Analyzer does not assist in this integration step.

10.3.1.5 Usage of RTE APIs

SMI-2072
The user of MICROSAR Safe shall ensure that the RTE and all of its users have the same assumptions with regards to the sizes of the datatypes.

The RTE supports the configuration of custom datatypes for its APIs. The RTE specification mandates that arrays are passed as pointer to the array base type.
The RTE does not enforce that both sides of a connection use arrays of the same size.

No NULL pointers or invalid pointers shall be passed to RTE APIs.
The object to which the pointer points needs to have at least the size of the pointer base type.

Vector recommends using the headers generated by the RTE when static code analysis is performed on the application code.
Vector recommends using the same datatypes on both sides of a connection.
Arrays and void pointers on interfaces where the called function writes to them are considered especially problematic.

Verification can e.g. be performed by review and/or static code analysis.

The RTE Analyzer lists APIs and runnables that use such parameters to assist in this integration step.

SMI-2073
The user of MICROSAR Safe shall ensure that RTE APIs are only called from their configured contexts.

Fast response times are crucial in embedded systems. Therefore, the RTE generator analyzes the call contexts of all APIs in order to optimize away unneeded interrupt locks. When the application calls the APIs from a different context than the RTE assumes, data consistency problems may arise.

In systems with ASIL partitions, the RTE generator uses a conservative locking strategy. Locks are only optimized away if all accesses are done within the same task.

Verification can e.g. be performed by review and/or static code analysis.

The RTE Analyzer lists the optimized APIs of SWCs to assist in this integration step.

10.3.1.6 Configuration of RTE APIs

SMI-2074
The user of MICROSAR Safe shall ensure that the receivers can handle the initial value provided by the RTE if no write or calibration occurred. For implicit accesses it has to be assured that the correct initial value is sent when *Rte_IWrite* or *Rte_IWriteRef* were not called in the runnable.

Initial values can be configured for:

- non-queued sender-/receiver communication
- inter-runnable variables
- mode ports

- calibration parameters

The initial value is returned when no sending API was called before the first read or no calibration was performed before the first read. The initial values depend on the connected components.

Vector recommends using the same initial value on all port data elements that are connected with each other.

The RTE Analyzer lists all APIs of SWCs that may provide an initial value to assist in this integration step. If possible, the RTE Analyzer reports the initial value generated by the RTE into the configuration feedback report.

SMI-2075
The user of MICROSAR Safe shall ensure that the alive timeout by the COM is not used for safety-related inter-ECU sender/receiver communication.

safety-related communication must be protected by E2E protection. The decision if new data is available (alive) can only be made by the E2E mechanism. Data is not interpreted by the COM. For example the sending ECU might repeat old data. This is only detected by the cycle counter that is part of the E2E protection.

The RTE Analyzer lists the reading APIs that provide the alive timeout status to assist in this integration step.

SMI-2126
The user of MICROSAR Safe shall ensure that SWCs handle the *RTE_E_INVALID* return code properly.

The RTE offers a functionality to invalidate signals.

Vector requires using end-to-end protection for safety-related signals. Relying on the invalidation mechanism for safety-related signals is not an option.

Vector recommends not using the invalidation for safety-related intra-ECU connections.

Verification can e.g. be performed by review and/or integration testing.

The RTE Analyzer lists RTE APIs that return the *RTE_E_INVALID* return code to assist in this integration step.

SMI-2127
The user of MICROSAR Safe shall ensure that SWCs handle the *RTE_E_NEVER_RECEIVED* return code properly.

The RTE offers a functionality to report if a signals was received after the ECU was started.

Vector requires using end-to-end protection for safety-related signals. Relying on the never received mechanism for safety-related signals is not an option. Especially, when using the E2E transformer, its return value shall be evaluated.

Verification can e.g. be performed by review or integration testing.

The RTE Analyzer lists RTE APIs that return the *RTE_E_NEVER_RECEIVED* return code to assist in this integration step.

SMI-2125
The user of MICROSAR Safe shall ensure that the queue sizes that were chosen during the configuration are sufficient for the integrated system.

The RTE uses queues for mode communication, sender-/receiver communication and mapped client-/server communication. The queue sizes depend on the scheduling of entities and the call sequences of the APIs.

Vector recommends not using APIs with queues for safety-related functionality.

Verification can e.g. be performed by stress testing.

The RTE Analyzer lists the queue sizes to assist in this integration step.

10.4 Safety features required from other components

SMI-2121
RTE requires the following functionality as safety feature from the operating system:

- Interrupt enabling/disabling
- Inter OS Application Communicator (IOC) sending and receiving functionality
- Spin-lock functionality

This requirement is fulfilled if an ASIL operating system by Vector is used.

10.5 Dependencies to hardware

This component does not use a direct hardware interface.

11 Glossary and Abbreviations

11.1 Glossary

Term	Definition
User of MICROSAR Safe	Integrator and user of components from MICROSAR Safe provided by Vector.
MICROSAR Safe	MICROSAR Safe comprises MICROSAR SafeBSW and MICROSAR SafeRTE as Safety Element out of Context. MICROSAR SafeBSW is a set of components, that are developed according to ISO 26262 [1], and are provided by Vector in the context of this delivery. The list of MICROSAR Safe components in this delivery can be taken from the documentation of the delivery.
Critical section	A section of code that needs to be protected from concurrent access. A critical section may be protected by using the AUTOSAR exclusive area concept.
Configuration data	Data that is used to adapt the MICROSAR Safe component to the specific use-case of the user of MICROSAR Safe. Configuration data typically comprises among others: feature selection, routing tables, channel tables, task priorities, memory block descriptions.
Generated code	Source code that is generated as a result of the configuration in DaVinci Configurator Pro
Partition	A set of memory regions that is accessible by tasks and ISRs. Synonym to OSApplication.

11.2 Abbreviations

Abbreviation	Description
ASIL	Automotive Safety Integrity Level
BSWMD	Basic Software Module Description
CPU	Central Processing Unit
CREQ	Component Requirement
EEPROM	Eletronically Ereasable and Programmable Read-only Memory
ECC	Error Correcting Code
ECU	Electronic Control Unit
EXT	Driver for an external hardware unit
ISO	International Standardization Organization
MCAL	Microcontroller Abstraction
MIP	Module Implementation Prefix
MSSV	MICROSAR Safe Silence Verifier
OS	Operating System
PDU	Protocol Data Unit
QM	Quality Management

RAM	Random Access Memory
SMI	Safety Manual Item
TCL	Tool Confidence Level

12 Contact

Visit our website for more information on

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

www.vector.com