Elektrobit

# EB tresos AutoCore

## Requirements for MCAL vendors


**Version:** 1.1.32

**Status:** proposed

**Release date:** 2019/07/26

Elektrobit Automotive GmbH
Am Wolfsmantel 46
D-91058 Erlangen
GERMANY

Phone: +49 9131 7701-0
Fax: +49 9131 7701-6333
http://www.elektrobit.com

## Document history

| Version | Date | Description | Editor |
|---|---|---|---|
| 1.0.0 | 2011/04/12 | Initial version for AUTOSAR 4.0 rev 2 | thga2649 |
| 1.0.1 | 2011/05/03 | Added section regarding BSWMD | thga2649 |
| 1.0.2 | 2011/05/04 | Updated module specific requirements for CAN and LIN to AUTOSAR 4.0 rev 3. | thga2649 |
| 1.1.0 | 2011/05/04 | Changed status to proposed after review. | thga2649 |
| 1.1.1 | 2011/05/10 | Corrected typos and formulated remaining BSWMD requirements as requirements with IDs. | thga2649 |
| 1.1.2 | 2011/05/11 | Added missing link between `BswImplementation` and `BswInternalBehavior` in requirements, examples, and strict schema. | thga2649 |
| 1.1.3 | 2011/05/31 | Updated templates for Makefiles (EB_MCAL_0076, EB_MCAL_0077), rephrase requirements on memory footprint (EB_MCAL_0007), added dependency for Can Assistant (EB_MCAL_0142), changed logic in code snippets for declaring own interrupt vectors and SchM macros in `Mcal.h` (EB_MCAL_0090, EB_MCAL_0091), and added clarifying introduction to Chapter 7, “Module specific requirements“. | thga2649 |
| 1.1.4 | 2011/05/31 | Added requirements for the existence and the content of common published information in the module's configuration scheme (EB_MCAL_0143 - EB_MCAL_0147). | thga2649 |
| 1.1.5 | 2011/06/30 | Removed duplicate requirement ID EB_MCAL_0106 by relabeling the requirement stating that "the elements `BSW-MODULE-DESCRIPTION`, `BSW-IMPLEMENTATION`, and `BSW-MODULE-ENTRY` shall be placed in seperate packages" as EB_MCAL_0105 (which was previously unused). Added EB's C style guide [8] to the package (`relevant_docs/C_Style_Guide.pdf`). Added example `anchors.xml` to the package (`samples/anchors.xml`). | thga2649 |
| 1.1.6 | 2011/07/25 | Removed incomplete sections (marked with TODO) for module specific requirements imposed by EB's implementation of Fee, Ea, and WdgIf since there aren't any. | thga2649 |
| 1.1.7 | 2011/08/01 | Added information and requirements about pre- and recommended configurations. | dake2402 |
| 1.1.8 | 2011/08/02 | Added information and requirements about service needs. | dake2402 |

| Version | Date | Description | Editor |
|---|---|---|---|
| 1.1.9 | 2011/08/08 | Adapted EB_MCAL_0073 to match changes in BSW00318 of [14]. | thga2649 |
| 1.1.10 | 2011/09/01 | Adapted EB_MCAL_0002, EB_MCAL_0003, EB_MCAL_0009 according to feedback from freescale. Added EB_MCAL_0173 and EB_MCAL_0174. | thga2649 |
| 1.1.11 | 2011/11/09 | Adapted EB_MCAL_0139 to match most recent draft of [5]. Updated schema/AUTOSAR_4-0-2_EB_strict.xsd to be based on strict AUTOSAR XML schema (schema/AUTOSAR_4-0-2_STRICT.xsd)) and updated example XML files accordingly to properly validate against updated schema/AUTOSAR_4-0-2_EB_strict.xsd. | thga2649 |
| 1.1.12 | 2011/11/10 | Removed requirement to include CONSTANT-MEMORYS and STATIC-MEMORYS in a module's INTERNAL-BEHAVIORS description since there's currently no use-case for having these. | thga2649 |
| 1.1.13 | 2011/11/18 | Added requirements EB_MCAL_0175 and EB_MCAL_0176. | thga2649 |
| 1.1.14 | 2011/12/02 | Updated to version 3.2.0 of [12]. Adapted samples XML files in samples/BSW_MD/autosar. | gebl2400 |
| 1.1.15 | 2012/01/25 | Resolved duplicate use of IDs EB_MCAL_0120 and EB_MCAL_0121. Changed ID of requirement regarding CAN driver HOH numbering from EB_MCAL_0120 to EB_MCAL_0120 to EB_MCAL_0177. Changed ID of requirement regarding CanTrcv channel configuration from EB_MCAL_0121 to EB_MCAL_0177. | thga2649 |
| 1.1.16 | 2012/02/15 | Added relevant AUTOSAR bugzilla issue to EB_MCAL_0106. Added relevant bugzilla issue to EB_MCAL_0014 together with an illustrating example. | thga2649 |
| 1.1.17 | 2012/02/17 | Update name and location for BSWMD (EB_MCAL_0102). Provided schema/AUTOSAR_4-0-3_EB_strict.xsd based on strict AUTOSAR XML schema (schema/AUTOSAR_4-0-3_STRICT.xsd) and updated example XML files accordingly to properly validate against updated schema/AUTOSAR_4-0-3_EB_strict.xsd (EB_MCAL_0104). Changed file names of example files to match the names of the virtual modules defined by AUTOSAR (e.g., renamed Can_DataTypes.xml to Can_GeneralTypes.arxml). | thga2649 |

| Version | Date | Description | Editor |
|---|---|---|---|
| 1.1.18 | 2012/05/04 | Added requirements EB_MCAL_0179 to EB_MCAL_0184 for proper handling of supported config variants and configuration classes. | thga2649 |
| 1.1.19 | 2012/06/05 | Revised Section 6.7, “Schedule manager usage” (EB_MCAL_0098 to EB_MCAL_0100) to match the AUTOSAR 4.0 SchM/Rte specification [10]. | thga2649 |
| 1.1.20 | 2012/07/27 | Updated ARXML examples in `samples/BSW_MD` and in `samples/BSW_MD/autosar` to adhere to the final decision of AUTOSAR in AUTOSAR Bugzilla issue #52928 regarding "Identifying M1 elements in packages" (i.e., `/AUTOSAR_<module>` instead of `/AUTOSAR/<module>`). Bumped to revision AUTOSAR 4.0 rev 3 and adapted relevant referenced documents accordingly. | thga2649 |
| 1.1.21 | 2012/08/22 | Corrected example for wrong AUTOSAR 4.0 SchM usage in Section 6.7, “Schedule manager usage”. Changed EB_MCAL_141 to EB_MCAL_0141. Changed EB_MCAL_01001 to EB_MCAL_0162. Changed duplicated EB_MCAL_0084 (registring a module's user documentation via the module's `anchor.xml` file) to EB_MCAL_0038. | thga2649 |
| 1.1.22 | 2012/10/09 | Updated EB_MCAL_0122 to explicitly exclude the scenario mandated by CAN409 to prevent that EB specific requirements are in contradiction to AUTOSAR. | thga2649 |
| 1.1.23 | 2012/11/29 | Removed requirements on service needs for SchM exclusive areas, since this is now handled via each module's BSWMD (i.e., removed section 6.8.3 "SchM exclusive areas"). Removed EB_MCAL_0160. | thga2649 |
| 1.1.24 | 2013/06/28 | Reworked BSWMD samples w.r.t. `MEMORY-SECTION` and `SW-ADDR-METHOD`: Now using `ADDR-METHOD-SHORT-NAME-AND-ALIGNMENT` as `MEMORY-ALLOCATION-KEY-WORD-POLICY` when the respective `MEMORY-SECTION` has an alignment suffix according to [3]. Futhermore using `SYMBOL` (with the same content as `SHORT-NAME`) since `SHORT-NAME` is only a default according to [5]. | thga2649 |
| 1.1.25 | 2014/04/17 | Added EB_MCAL_0185 for a CanTp specific requirement imposed in the CAN driver implementation. Added EB_MCAL_0186 to EB_MCAL_0190 as additional requirements imposed on the CAN driver for CAN-FD support (see Section 7.1.6, “CAN-FD support”). | thga2649 |

| Version | Date | Description | Editor |
|---|---|---|---|
| 1.1.26 | 2014/04/22 | Added EB_MCAL_0191 to EB_MCAL_0199 as additional requirements imposed on the GPT driver for global time synchronization support. Added EB_MCAL_0200 to EB_MCAL_0202 as additional requirement on the Ethernet driver (see Section 7.3, “EthIf”). | thga2649 |
| 1.1.27 | 2014/04/29 | Replaced EB_MCAL_0200 to EB_MCAL_0202 as additional requirement on the Ethernet driver with EB_MCAL_0203 to EB_MCAL_0212 (see Section 7.3, “EthIf”). | thga2649 |
| 1.1.28 | 2015/04/13 | Removed obsolete CanTp specific requirement EB_MCAL_0185 imposed on the CAN driver implementation. Removed obsolete Tm specific requirements EB_MCAL_0191 to EB_MCAL_0199 imposed on the Gpt implementation. | thga2649 |
| 1.1.29 | 2016/04/27 | Added EthIf specific requirement EB_MCAL_0213. | thga2649 |
| 1.1.30 | 2016/09/02 | Added EthIf specific requirement EB_MCAL_0214. | thga2649 |
| 1.1.31 | 2017/02/01 | Added `Lin_WakeupInternal()` to EB_MCAL_0128 and removed `Lin_GetStatus()`. | thga2649 |
| 1.1.32 | 2019/07/26 | Added qualification requirement EB_MCAL_0215 for compiler toolchain. | flro2674 |

# Table of Contents

# 1. Introduction

This document summarizes EB's requirements on AUTOSAR modules provided by MCAL vendors in order to facilitate the integration of the MCAL vendor's modules into the EB tresos AutoCore product.

The requirements formulated in this document make use of the requirement wording defined in [6], which has become the international standard for requirement wording.

In short: 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 described in [6]:

"MUST", "REQUIRED", "SHALL":
: These keywords are used for mandatory requirements or prohibitions.

"SHOULD", "RECOMMENDED":
: There may exist circumstances under which the requirement can be ignored but the reason must always be documented.

"MAY", "OPTIONAL":
: These keywords are used for nice-to-have requirements.

# 2. References

[1] *AUTOSAR Specification of ECU Configuration (Document Version 3.2.0)*, https://svn.autosar.org/repos/work/24_Sources/tags/R4.0.003/TPS_ECUConfiguration_087/AUTOSAR_TPS_ECUConfiguration.pdf, Release: 4.0 Revision: 3

[2] *AUTOSAR ECU Configuration Parameter Definition (Document Version 4.2.0)*, https://svn.autosar.org/repos/work/24_Sources/tags/R4.0.003/MOD_ECUConfigurationParameters_289/AUTOSAR_MOD_ECUConfigurationParameters.arxml, Release: 4.0 Revision: 3

[3] *AUTOSAR Specification of Memory Mapping (Document Version 1.4.0)*, https://svn.autosar.org/repos/work/24_Sources/tags/R4.0.003/SWS_MemoryMapping_128/AUTOSAR_SWS_MemoryMapping.pdf, Release: 4.0 Revision: 3

[4] *AUTOSAR Specification of Compiler Abstraction (Document Version 3.2.0)*, https://svn.autosar.org/repos/work/24_Sources/tags/R4.0.003/SWS_CompilerAbstraction_051/AUTOSAR_SWS_CompilerAbstraction.pdf, Release: 4.0 Revision: 3

[5] *AUTOSAR Specification of BSW Module Description Template (Document Version 2.2.0)*, https://svn.autosar.org/repos/work/24_Sources/tags/R4.0.003/TPS_BSWModuleDescriptionTemplate_089/AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf, Release: 4.0 Revision: 3

[6] *RFC 2119 - Key words for use in RFCs to Indicate Requirement Levels*, http://www.faqs.org/rfcs/rfc2119.html

[7] *MISRA-C:2004*, Guidelines for the use of the C language in critical systems

[8] *C Style Guide (Document Version 1.01)*, http://qms.ebgroup.elektrobit.com/qms/instructions/operative/software-entwicklung/progr-richtl/C_Style_Guide.pdf

[9] *EB tresos Studio developer's guide (product release 14.1)*, <EB tresos Studio install dir>/doc/2.0_EB_tresos_Studio/2.4_Studio_documentation_developers_guide.pdf

[10] *AUTOSAR Specification of RTE (Document Version 3.2.0)*, https://svn.autosar.org/repos/work/24_-Sources/tags/R4.0.003/SWS_RTE_084/AUTOSAR_SWS_RTE.pdf, Release: 4.0 Revision: 3

[11] *XML Path Language (XPath) - Version 1.0*, http://www.w3.org/TR/xpath/, W3C Recommendation 16 November 1999

[12] *AUTOSAR Generic Structure Template (Document Version 3.2.0)*, https://svn.autosar.org/repos/work/24_-Sources/tags/R4.0.003/TPS_GenericStructureTemplate_202/AUTOSAR_TPS_-GenericStructureTemplate.pdf, Release: 4.0 Revision: 3

[13] *AUTOSAR List of Basic Software Modules (Document Version 1.6.0)*, https://svn.autosar.org/repos/work/24_Sources/tags/R4.0.003/TR_BSWModuleList_150/AUTOSAR_TR_BSWModuleList.pdf, Release: 4.0 Revision: 3

[14] *AUTOSAR General Requirements on Basic Software Modules (Document Version 3.2.0 )*, https://svn.autosar.org/repos/work/24_Sources/tags/R4.0.003/SRS_BSWGeneral_043/AUTOSAR_SRS_BSWGeneral.pdf, Release: 4.0 Revision: 3

[15] *EB tresos Studio user's guide (product release 14.1)*, <EB tresos Studio install dir>/doc/2.0_EB_tresos_Studio/2.1_Studio_documentation_users_guide.pdf

[16] *AUTOSAR Specification of CAN Driver (Document Version 4.3.0)*, https://svn.autosar.org/repos/work/24_Sources/tags/R4.1.003/SWS_CANDriver_011/AUTOSAR_SWS_CANDriver.pdf, Release: 4.1 Revision: 3

[17] *AUTOSAR Specification of Ethernet Driver (Document Version 1.5.0)*, https://svn.autosar.org/repos/work/24_Sources/tags/R4.1.003/SWS_EthernetDriver_430/AUTOSAR_SWS_EthernetDriver.pdf, Release: 4.1 Revision: 3

[18] *Specification of GPT Driver (Document Version 3.3.1)*, https://svn.autosar.org/repos/work/24_Sources/tags/R4.1.003/SWS_GPTDriver_030/AUTOSAR_SWS_GPTDriver.pdf, Release: 4.1 Revision: 3

# 3. Glossary

| Definition/Acronyms | Description |
|---|---|
| API | Application Programming Interface |
| BSW | Basic Software |
| BSWBD | BSW Module Description |
| Dem | Diagnostic Event Manager |
| Det | Development Error Tracer |
| ECU | Electronic Control Unit |
| ICU | Interrupt Control Unit |
| ISR | Interrupt Service Routine |
| MCAL | Micro Controller Abstraction Layer |
| MCU | Micro Controller Unit |
| mangled Module Short Name | The module short name suffixed with the vendor API infix and the vendor ID as described in BSW00347 of [14]. |
| Module Short Name | The short name (a.k.a. module abbreviation) of a module according to [13] (e.g., `Det`). |
| SchM | BSW Schedule Manager |
| Rte | Run-time Environment |
| StMD | Standardized Module Definition |
| SRS | Software Requirements Specification |
| SWS | Software Specification |
| VSMD | Vendor Specific Module Definition |

Table 3.1. Glossary

# 4. Documentation base

## 4.1. General

**Requirement (EB_MCAL_0001)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
All AUTOSAR requirements shall be valid. This document defines additional requirements to the AUTOSAR requirements that shall be valid as well.

Several documents describe configuration parameters: the module's SWS documents, the document "AUTOSAR Specification of ECU Configuration" [1], and finally the XML file containing the "AUTOSAR ECU Configuration Parameter Definition" [2].

**Requirement (EB_MCAL_0002)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
If there are inconsistencies between the the module's SWS documents, the document "AUTOSAR Specification of ECU Configuration", and finally the XML file containing the "AUTOSAR ECU Configuration Parameter Definition", w.r.t. the modules' configuration parameters, the XML file shall be the source.

Rationale:
The XML file is automatically generated from the AUTOSAR UML meta model and thus not prone to copy/paste errors or missing manual updates. Furthermore the XML file allows for automated conformance checks.

# 4.2. AUTOSAR revisions

**Requirement (EB_MCAL_0003)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The modules' implementations shall adhere to the AUTOSAR 4.0 specification in revision 3.

Rationale:
Base for the EB tresos AutoCore 6/7/8 development is the AUTOSAR 4.0 specification in revision 3.

# 5. Timing and footprint

## 5.1. Interrupt handling

### 5.1.1. Interrupt lock time

**Requirement (EB_MCAL_0004)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The maximum interrupt lock time equals the maximum execution time of a critical section and shall be less 100 us.

### 5.1.2. Interrupt latency

The maximum interrupt latency is the maximum time an interrupt needs for execution. It is the worst case execution time but with a configuration which is optimised for latency. E.g. the worst case execution time for a CAN Rx ISR, with a CanIf configured for polling mode.

**Requirement (EB_MCAL_0005)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The maximum interrupt latency shall be less than 500 us.

## 5.2. Startup

The maximum start-up time is the maximum time from power on until network communication commences (e.-g., transmission of CAN messages take place)

**Requirement (EB_MCAL_0006)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The maximum startup time shall be less than 20 ms.

## 5.3. Memory footprint

The minimum MCAL memory footprint is the RAM/ROM consumption of the of the complete EB tresos AutoCore basic software MCAL modules using a minimum configuration.

**Requirement (EB_MCAL_0007)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The minimum MCAL memory footprint shall not exceed 30 kByte of ROM and 2 kByte of RAM in total.

# 6. Software construction

## 6.1. Coding guidelines

The coding guidelines, including C, XDM, generator code, are part of the software construction process.

### 6.1.1. C Coding for AUTOSAR BSW modules

#### 6.1.1.1. General rules

All AUTOSAR BSW modules shall comply with the guidelines given in the MISRA-C:2004 guidelines [7]. Any violations to the guidelines shall be documented in the source code according to EB's C style guide [8], which is provided in `relevant_docs/C_Style_Guide.pdf`.

**Requirement (EB_MCAL_0008)**

Coverage:
NONE

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
All AUTOSAR BSW modules shall comply with the guidelines given in the MISRA-C:2004 guidelines.

**Requirement (EB_MCAL_0009)**

Coverage:
NONE

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Any violations to the MISRA-C:2004 guidelines shall be documented in the source code according to EB's C style guide.

Compiler and/or platform specific keywords or types shall not be used in the BSW module's implementation. - The definitions provided by the "AUTOSAR Specification of Compiler Abstraction" document [4] shall be used instead for the sake of portability (e.g., `STATIC` or `NULL_PTR`).

If the use of inline assembler or `#pragma` is required, both shall be encapsulated using the compiler specific symbols from the "AUTOSAR Specification of Compiler Abstraction" document [4].

**Requirement (EB_MCAL_0010)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The BSW modules shall not use compiler specific keywords/types.

Rationale:
Compiler specific keywords/types are abstracted in Compiler.h. Use the keywords/types defined there.

**Requirement (EB_MCAL_0011)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The BSW modules shall not use platform specific keywords/types.

Rationale:
Platform specific keywords/types are abstracted in Platform_Types.h. Use the keywords/types defined there.

**Requirement (EB_MCAL_0012)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The BSW modules shall use the definitions provided by the "AUTOSAR Specification of Compiler Abstraction" document.

**Requirement (EB_MCAL_0013)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The BSW modules shall encapsulation any use of inline assembler or statements using the compiler specific symbols from the "AUTOSAR Specification of Compiler Abstraction" document.

For AUTOSAR BSW modules with an SUPPORTED-CONFIG-VARIANTS of VARIANT-POST-BUILD the name of the C-init-data struct containing the BSW modules configuration shall be the SHORT-NAME of the multiple configuration container of the module's XML configuration. The default name (NAME_PATTERN) of the multiple configuration container shall be <Module Short Name>_Config_<number>.

This is required since the EcuM module calls the module's init function with the address (i.e., the address operator (&) is applied) of this C-init-data struct as parameter (see AUTOSAR Bugzilla issue #53368) if the module's SUPPORTED-CONFIG-VARIANTS is VARIANT-POST-BUILD. Therefore the EcuM needs to know the name of the object. To use a common approach, this rule applies for all modules - even if the init function is not called from EcuM.

Thus if the SHORT-NAME of the multiple configuration container of the module's XML configuration is for example <Module Short Name>_Config_0, the module's code and generated configuration has to be structured in a way that the following code snippet in the C programming language compiles correctly:

```
<Module Short Name>_Init(&<Module Short Name>_Config_0);
```

AUTOSAR modules with a SUPPORTED-CONFIG-VARIANTS of VARIANT-PRE-COMPILE or VARIANT-LINK-TIME should chose the name of their C-init-data struct containing the BSW modules configuration to be the SHORT-NAME of the multiple configuration container of the module's XML configuration as well for the sake of consistency.

AUTOSAR modules with a SUPPORTED-CONFIG-VARIANTS of VARIANT-PRE-COMPILE or VARIANT-LINK-TIME shall be prepared that their module's init function is called with NULL_PTR as a parameter. This shall not cause a development error detection check to fail. The module shall silently ignore the fact that the module's init function is called with NULL_PTR (see AUTOSAR Bugzilla issue #45154) as a parameter since in case of SUPPORTED-CONFIG-VARIANTSs of VARIANT-PRE-COMPILE or VARIANT-LINK-TIME, the module shall not use the parameter anyway.

**Requirement (EB_MCAL_0014)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:

For AUTOSAR modules with a SUPPORTED-CONFIG-VARIANTS of VARIANT-POST-BUILD the name of the C-init-data struct containing the BSW modules configuration shall be the SHORT-NAME of the multiple configuration container of the module's XML configuration.

Rationale:

The EcuM module calls the module's init function with the address (i.e., the address operator (&) is applied) of this C-init-data struct as parameter (see AUTOSAR Bugzilla issue #53368) for modules with an SUPPORTED-CONFIG-VARIANTS of VARIANT-POST-BUILD.

**Requirement (EB_MCAL_0015)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

The default name (NAME_PATTERN) of the multiple configuration container shall be <Module Short Name>_Config_<number>.

Rationale:

Common naming scheme that ensures unique names for all C-init-data structs across all BSW modules.

**Requirement (EB_MCAL_0016)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

AUTOSAR modules with a SUPPORTED-CONFIG-VARIANTS of VARIANT-PRE-COMPILE or VARIANT-LINK-TIME should chose the name of their C-init-data struct containing the BSW modules configuration to be the SHORT-NAME of the multiple configuration container.

Rationale:

Consistency with the naming of scheme used for C-init-data structs of modules with SUPPORTED-CONFIG-VARIANTS of VARIANT-POST-BUILD.

**Requirement (EB_MCAL_0017)**

Coverage:

DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Calling the init function of an AUTOSAR module with a SUPPORTED-CONFIG-VARIANTS of VARIANT-PRE-COMPILE or VARIANT-LINK-TIME with a NULL_PTR as a parameter shall not cause a development error detection check to fail.

Rationale:
The EcuM module will call the init functions of such modules with a NULL_PTR as a parameter (see AUTOSAR Bugzilla issue #45154).

**Requirement (EB_MCAL_0018)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The module init function of an AUTOSAR module with a SUPPORTED-CONFIG-VARIANTS of VARIANT-PRE-COMPILE or VARIANT-LINK-TIME shall shall not use the parameter passed to the init function in any way.

Rationale:
Modules with a SUPPORTED-CONFIG-VARIANTS of VARIANT-LINK-TIME shall use the linker to resolve the references of the module's static code into the module's configuration.

**Requirement (EB_MCAL_0019)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
An AUTOSAR module with a SUPPORTED-CONFIG-VARIANTS of VARIANT-PRE-COMPILE or VARIANT-LINK-TIME shall silently ignore the fact that the module's init function is called with NULL_PTR as a parameter.

Rationale:
The EcuM module will call the init functions of such modules with a NULL_PTR as a parameter (see AUTOSAR Bugzilla issue #45154).

## 6.1.2. Configuration scheme for AUTOSAR BSW modules

### 6.1.2.1. General rules

The parameter definition for a BSW module shall be in the EB tresos XDM file format which is described in chapter 6.1. "The XDM format" of [9].

AUTOSAR defines rules for deriving a vendor specific module definitions (VSMD) from a standardized module definition (StMD) (see chapter 5.1 "Deriving vendor specific module definitions from standardized module definitions" of [1]).

The derived VSMD shall pass the VSMD check error free. Details about the VSMD check can be found in chapter 7.11.4.4. "Checking vendor-specific against standard module definitions" of [15].

Renaming enumeration literals (without changing the meaning) is not allowed (this includes that changing upper/lower case is also not allowed). Although [1] allows to add and remove enumeration literals (ecuc_sws_2084, ecuc_sws_6002), changing enums without changing the meaning results in incompabilities without any advantage. - Thus renaming enumeration literal from `RECEIVE` to `Receive` is not allowed for example!

According to ecuc_sws_5001 in [1] redundant entries in a module definition shall be avoided. If parameters in VSMD can be derived from the StMD directly or by computation, these parameters need not be defined in the VSMD.

No additional parameters should be added to a VSMD. Especially modules above the MCAL shall think twice if an extra parameter is really necessary. - In case a vendor specific parameter is required, the parameter's `ORIGIN` attribute must be set to the MCAL vendor's company name.

**Requirement (EB_MCAL_0140)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The parameter definition for a BSW module shall be in the EB tresos XDM file format.

**Requirement (EB_MCAL_0020)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The rules defined by AUTOSAR for deriving a vendor specific module definitions (VSMD) from a standardized module definition (StMD) shall not be broken.

**Requirement (EB_MCAL_0021)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The derived vendor specific module definitions (VSMD) shall pass the VSMD check error free.

**Requirement (EB_MCAL_0022)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Enumeration literals shall not be renamed (without changing the meaning). - This includes that changing upper/lower case is not allowed.

**Requirement (EB_MCAL_0023)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Redundant entries in the VSDM (i.e., entries that can be derived from existing entries in the StMD) shall be avoided. - If parameters in VSMD can be derived from the StMD directly or by computation, these parameters shall not be defined in the VSMD.

**Requirement (EB_MCAL_0024)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Additional (vendor-specific) parameters should not be added in the VSMD.

**Requirement (EB_MCAL_0025)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The attribute ORIGIN of all vendor specific parameters must be set to the vendor's company name

To increase the usablility of the MCAL module, every parameter should specify a reasonable default value (specified via the `DEFAULT` attribute). - This reduces the configuration effort of the user. The following XDM snippet contains an example taken from the EcuM module, where the default value of the `EcuMSleepModeId` is automatically calculated:

```
<v:var name="EcuMSleepModeId" type="INTEGER">
<a:a name="DEFAULT" type="XPath">
<a:tst expr="num:integer(../@index)"/>
</a:a>
... other attributes ...
</v:var>
```

For the configuration parameter `<Module Short Name>DevErrorDetect` a default value of `true` shall be used, whereas the default value of the parameter `<Module Short Name>VersionInfoApi` shall be set to `false`.

In order to prevent that constant configuration parameters can be changed/overwritten by a user's configuration, every definition of configuration parameter holding a constant value shall have type `INTEGER_LABEL`, `STRING_LABEL` or `FLOAT_LABEL`.

Optional parameters shall set the attribute `OPTIONAL` to `true`. An optional parameter is a parameter with lower multiplicity 0 and upper multiplicity 1. When transforming the StMD to the VSMD with EB tresos Studio this is achieved by using the command line argument `-DMapOptionalAsList=false`.

**Requirement (EB_MCAL_0026)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Every parameter should specify a reasonable default value.

Rationale:
When adding a new module the configuration project or adding new containers all empty parameters will throw errors.

**Requirement (EB_MCAL_0175)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
For the configuration parameter <Module Short Name>DevErrorDetect a default value of true shall be used.

Rationale:
Enable development error detection by default in order to facilitate detection of configuration errors during integration.

**Requirement (EB_MCAL_0176)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
For the configuration parameter <Module Short Name>VersionInfoApi a default value of false shall be used.

Rationale:
Reduce code size by omitting superfluous API function.

**Requirement (EB_MCAL_0027)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Every definition of configuration parameter holding a constant value shall have type INTEGER_LABEL, STRING_LABEL or FLOAT_LABEL.

Rationale:
Values of constant configuration parameters must not be overwritten by a configuration.

**Requirement (EB_MCAL_0141)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Optional parameters shall set the attribute OPTIONAL to true. An optional parameter is a parameter with lower multiplicity 0 and upper multiplicity 1.

Rationale:
The default representation of optional parameters in the EB tresos XDM file format is that of a list containing at most one element. This, however, has shown to be very unintuitive and confusing during configuration.

In order to facilitate the retrival of the the common published information without needing to read the module's basic software module description, the module's configuration scheme shall provide a container named `CommonPublishedInformation` holding the common published information of the module. The container `CommonPublishedInformation` shall be located below the container holding the module definition. The container `CommonPublishedInformation` contain the following parameters:

- `ModuleId` [`integer`]
- `VendorId` [`integer`]
- `ArMajorVersion` [`integer`]
- `ArMinorVersion` [`integer`]
- `ArPatchVersion` [`integer`]
- `SwMajorVersion` [`integer`]
- `SwMinorVersion` [`integer`]
- `SwPatchVersion` [`integer`]
- `VendorApiInfix` [`integer`]: optional, only if module has an upper multiplicity greater than 1 (i.e., can be instantiated multiple times on a single ECU)

Modules with an upper multiplicity greater than 1 shall specify the `VendorApiInfix` parameter within the `CommonPublishedInformation` container. - The `VendorApiInfix` parameter must not be left empty (i.-

e., must not be an empty string) in that case. Modules with an upper multiplicity equal to 1 shall not specify the `VendorApiInfix` parameter within the `CommonPublishedInformation` container.

**Requirement (EB_MCAL_0143)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The module's configuration scheme shall provide a container named CommonPublishedInformation holding the common published information of the module.

Rationale:
Facilitate the retrival of the the common published information without needing to read the module's basic software module description.

**Requirement (EB_MCAL_0144)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The container CommonPublishedInformation shall be located below the container holding the module definition.

Rationale:
Facilitate the retrival of the the common published information without needing to read the module's basic software module description.

**Requirement (EB_MCAL_0145)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The container CommonPublishedInformation contain the following parameters: ModuleId [integer], VendorId [integer] ArMajorVersion [integer], ArMinorVersion [integer], ArPatchVersion [integer], SwMajorVer-

sion [integer], SwMinorVersion [integer], SwPatchVersion [integer], VendorApiInfix [string] (optional, only present if module has an upper multiplicity greater than 1).

Rationale:

Facilitate the retrival of the the common published information without needing to read the module's basic software module description.

**Requirement (EB_MCAL_0146)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

Modules with an upper multiplicity greater than 1 shall specify the VendorApiInfix parameter within the CommonPublishedInformation container. - The VendorApiInfix parameter must not be left empty (i.e., must not be an empty string) in that case.

Rationale:

The VendorApiInfix is required to do proper name mangling in cases where the module is instantiated multiple times on the same ECU.

**Requirement (EB_MCAL_0147)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

Modules with an upper multiplicity equal to 1 shall not specify the VendorApiInfix parameter within the CommonPublishedInformation container.

Rationale:

The VendorApiInfix is superfluous information for modules with an upper multiplicity equal to 1. In those cases a missing VendorApiInfix signals that no name mangling needs to be performed.

The VSMD shall provide information on the supported configuration variants of a specific module using the `IMPLEMENTATION_CONFIG_VARIANT` element. For each supported configuration variant the `IMPLEMENTATION_CONFIG_VARIANT` element shall contain one literal out of `VariantPreCompile`, `VariantLinkTime`, `VariantPostBuildLoadable`, and `VariantPostBuildSelectable`. The default value for the `IMPLEMENTATION_CONFIG_VARIANT` shall be the highest of these supported configuration variants (where pre-compile time < link time < post-build loadable/selectable; post-build loadable == post-build selectable). E.g.,

for a module supporting a pre-compile time variant and a post-build loadable variant, the XDM snippet of the IMPLEMENTATION_CONFIG_VARIANT element will look as follows:

```
<v:var name="IMPLEMENTATION_CONFIG_VARIANT" type="ENUMERATION">
 [..]
 <a:da name="DEFAULT" value="VariantPostBuildLoadable"/>
 <a:da name="RANGE">
  <a:v>VariantPreCompile</a:v>
  <a:v>VariantPostBuildLoadable</a:v>
 </a:da>
</v:var>
```

For each configuration parameter the IMPLEMENTATIONCONFIGCLASS of the parameter shall be defined for each of the supported configuration variants. E.g., a the XDM snippet defining a parameter CanIfCtrlDrvTx-Cancellation for a module supporting a pre-compile time variant (VariantPreCompile) and a post-build loadable variant (VariantPostBuildLoadable), where the IMPLEMENTATIONCONFIGCLASS of the parameter is pre-compile (PreCompile) in both supported configuration variants will look as follows:

```
<v:var name="CanIfCtrlDrvTxCancellation" type="BOOLEAN">
 [..]
 <a:a name="IMPLEMENTATIONCONFIGCLASS"
  type="IMPLEMENTATIONCONFIGCLASS">
  <icc:v class="PreCompile">VariantPreCompile</icc:v>
  <icc:v class="PreCompile">VariantPostBuildLoadable</icc:v>
 </a:a>
 [..]
</v:var>
```

For each supported supported configuration variant of the module the IMPLEMENTATIONCONFIGCLASS of each config parameter (for the respective supported configuration variant) shall be smaller than or equal to the supported configuration variant. The following XDM snippet for example violates this requirement, since the IMPLEMENTATIONCONFIGCLASS of the parameter CanIfCtrlDrvTxCancellation is Link for the pre-compile time variant (VariantPreCompile) of the module:

```
<v:var name="CanIfCtrlDrvTxCancellation" type="BOOLEAN">
 [..]
 <a:a name="IMPLEMENTATIONCONFIGCLASS"
  type="IMPLEMENTATIONCONFIGCLASS">
```

```
  <icc:v class="Link">VariantPreCompile</icc:v>
 </a:a>
 [..]
</v:var>
```

For each supported supported configuration variant of the module any `INVALID` and/or `WARNING` checks of any given configuration parameter must not refer to configuration parameters of a larger configuration class for the respective supported configuration variant (e.g., `INVALID` checks of a configuration parameter with pre-compile time configuration class must not access configuration parameters with link time or post-build loadable/selectable configuration class). The following XDM snippet for example violates this requirement, since the `IMPLEMENTATIONCONFIGCLASS` of the parameter `Destination` is `Link` while the `IMPLEMENTATIONCONFIGCLASS` of the parameter `Destination` is `PreCompile` for the supported configuration variant `VariantPreCompile`:

```
<v:var name="Source" type="BOOLEAN">
 [..]
 <a:a name="IMPLEMENTATIONCONFIGCLASS"
  type="IMPLEMENTATIONCONFIGCLASS">
  <icc:v class="PreCompile">VariantPreCompile</icc:v>
  <icc:v class="Link">VariantPostBuildLoadable</icc:v>
 </a:a>
 [..]
 <a:da name="WARNING" type="XPath">
  <a:tst
   expr="((../Destination = &apos;DET&apos;)"
   true="Some error message"/>
 </a:da>
</v:var>
<v:var name="Destination" type="STRING">
 [..]
 <a:a name="IMPLEMENTATIONCONFIGCLASS"
  type="IMPLEMENTATIONCONFIGCLASS">
  <icc:v class="Link">VariantPreCompile</icc:v>
  <icc:v class="Link">VariantPostBuildLoadable</icc:v>
 </a:a>
 [..]
</v:var>
```

**Requirement (EB_MCAL_0179)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The VSMD shall provide information on the supported configuration variants of a specific module using the IMPLEMENTATION_CONFIG_VARIANT element.

Rationale:
AUTOSAR mandates the specification of the supported configuration variants.

**Requirement (EB_MCAL_0180)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
For each supported configuration variant the IMPLEMENTATION_CONFIG_VARIANT element shall contain one literal out of VariantPreCompile, VariantLinkTime, VariantPostBuildLoadable, and VariantPostBuildSelectable.

Rationale:
AUTOSAR mandates the specification of the supported configuration variants.

**Requirement (EB_MCAL_0181)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The default value for the IMPLEMENTATION_CONFIG_VARIANT shall be the highest of the supported configuration variants (where pre-compile time < link time < post-build loadable/selectable; post-build loadable == post-build selectable).

Rationale:
Provide a defined sane default for this parameter to ease configuration.

**Requirement (EB_MCAL_0182)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
For each configuration parameter the IMPLEMENTATIONCONFIGCLASS of the parameter shall be defined for each of the supported configuration variants.

Rationale:
Enforce consistency between the supported configuration variants of the module and the configuration classes of the configuration parameters.

**Requirement (EB_MCAL_0183)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
For each supported supported configuration variant of the module the IMPLEMENTATIONCONFIGCLASS of each config parameter (for the respective supported configuration variant) shall be smaller than or equal to the supported configuration variant.

Rationale:
Enforce consistency between the supported configuration variants of the module and the configuration classes of the configuration parameters.

**Requirement (EB_MCAL_0184)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
For each supported supported configuration variant of the module any INVALID and/or WARNING checks of any given configuration parameter must not refer to configuration parameters of a larger configuration class for the respective supported configuration variant (e.g., INVALID checks of a configuration parameter with pre-compile time configuration class must not access configuration parameters with link time or post-build loadable/selectable configuration class).

Rationale:
Prevent that checks of a given configuration parameter depend of the values of configuration parameters that will not be available at the time the respective parameter is configured.

### 6.1.2.2. Layout rules

Every container and parameter shall contain a detailed description (the XML attribute called `DESC`). - This `DESC` field is shown in the EB tresos Studio GUI. Therefore the user shall get a useful description of this parameter. Descriptions used from the AUTOSAR parameter definition shall be checked for correctness, consistency, and completeness. They shall be corrected and extended if necessary.

The description shall be formatted using XHTML tags. The complete description shall be enclosed by a `<html>` tag. - Only the following tags shall be used:

Tags for tables:

`<table>` `<tr>` `<th>` `<td>`

Tags for lists:

`<ul>` `<ol>` `<li>`

Preformated text:

`<pre>` (used for code examples spanning multiple lines)

Logical tags:

`<code>` `<strong>` `<em>` (used for (labels of) configuration parameters, variables, functions, and literals)

Text structure:

`<p>` `<br/>` (the `<p>` tag must not be nested)

Font tags:

`<b>` `<tt>`

Some of these tags are deprecated and thus should not be used:

Tags for structure:

`<br/>`

Font tags:

`<b>` `<tt>`

Consider the following description as an example:

```
<a:a name="DESC">
  <a:v>
    <![CDATA[
      <html>
        Numeric representation of the default shutdown target.
        <ul>
          <li><code>0x50 (dec 80)</code> : Sleep</li>
          <li><code>0x90 (dec 144)</code> : Reset</li>
          <li><code>0x80 (dec 128)</code> : Off</li>
        </ul>
```

```
        If default shutdown target is <strong>Sleep</strong> the default
        Sleep Mode has to be configured.
      </html>
    ]]>
  </a:v>
</a:a>
```

**Requirement (EB_MCAL_0028)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Every container and parameter shall contain a detailed description (using the XML attribute called DESC).

Rationale:
This DESC field is shown in the EB tresos Studio GUI. Therefore the user shall get a useful description of this parameter.

**Requirement (EB_MCAL_0029)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Descriptions used from the AUTOSAR parameter definition shall be checked for correctness, consistency, and completeness. They shall be corrected and extended if necessary.

Rationale:
This DESC field is shown in the EB tresos Studio GUI. Therefore the user shall get a useful description of this parameter.

**Requirement (EB_MCAL_0030)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The description shall be formatted using XHTML tags.

Rationale:
Provide proper markup of the description in the EB tresos Studio GUI.

**Requirement (EB_MCAL_0031)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The complete description shall be enclosed by a <html> tag.

Rationale:
Required to ensure XHTML validity.

**Requirement (EB_MCAL_0032)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Only the following tags shall be used: <table> <tr> <th> <td> <ul> <ol> <li> <pre> <code>   <p> <br/> <b> <tt>

Rationale:
Limitations of the EB tresos Studio GUI renderer and of the XSLT stylesheet for automatic conversion into DocBook.

Container and parameter shall not contain a tooltip (attribute `TOOLTIP`).

Every container and parameter should contain a label (attribute `LABEL`). If labels are used, they shall:

- contain the unit of the parameter in question (if applicable)
- use capitalisation like in headings (e.g. *D*evelopment *E*rror *D*etection)
- use following labels for common configuration parameters:
  - `<Module Short Name>DevErrorDetect`: 'Development Error Detection'
  - `<Module Short Name>VersionInfoApi`: 'Provide Version Info API'

- SUPPORTED-CONFIG-VARIANTS: 'Configuration Variant'

Consider the following label as an example:

```
<v:var name="EcuMCycleTime" type="INTEGER">
  <a:a name="LABEL" value="Cycle Time (ms)"/>
  ... other attributes ...
</v:var>
```

**Requirement (EB_MCAL_0033)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Container and parameter shall not contain a tooltip.

Rationale:
Tooltips are show during a short period of time only.

**Requirement (EB_MCAL_0034)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Every container and parameter may contain a label (attribute LABEL).

Rationale:
The default parameter name defined by AUTOSAR may not be easy to interpret.

**Requirement (EB_MCAL_0035)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
If applicable labels shall contain the unit of the parameter in question.

Rationale:
Probvide information about the unit of the parameter (in many cases this is not obvious).

**Requirement (EB_MCAL_0036)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Labels shall use capitalisation like in headings (e.g. Development Error Detection).

Rationale:
Have a consistent capitalization across all modules.

**Requirement (EB_MCAL_0037)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
For the parameter <Module Short Name>DevErrorDetect the label 'Development Error Detection' shall be used.

Rationale:
Have consistent naming for similar parameters across all modules.

**Requirement (EB_MCAL_0039)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
For the parameter <Module Short Name>VersionInfoApi the label 'Provide Version Info' shall be used.

Rationale:

Have consistent naming for similar parameters across all modules.

**Requirement (EB_MCAL_0040)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

For the parameter IMPLEMENTATION_CONFIG_VARIANT the label 'Configuration Variant' shall be used.

Rationale:

Have consistent naming for similar parameters across all modules.

Parameters of a container with multiplicity greater than 1 shall be displayed in the list view. - In case there are too much parameters in a container only the most important ones shall be shown. Consider the following container as an example:

```
<v:lst name="EcuMWakeupSourceConfig" type="MAP">
  <a:a name="COLUMNS">
    <a:v>EcuMWakeupSourceName</a:v>
    <a:v>EcuMWakeupSource</a:v>
    <a:v>EcuMValidationTimeout</a:v>
    <a:v>EcuMComChannel</a:v>
  </a:a>
  ... other attributes ...
</v:lst>
```

**Requirement (EB_MCAL_0041)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

Parameters of a container with multiplicity greater than 1 shall be displayed in the list view. - In case there are too much parameters in a container only the most important ones shall be shown.

Rationale:

Fast access to the most frequently used parameters.

### 6.1.3. Basic software module description (BSWMD)

According to AUTOSAR the interfaces, the implementation details, the requirements on other modules, and the definition of the dependencies and provided and required artifacts of an AUTOSAR BSW module are described by means of a *basic software module description (BSWMD)*.

For each AUTOSAR module exactly one BSWMD file shall be provided. The file containing the BSWMD shall be named `<Module Short Name>_Bswmd.arxml` and shall be placed in the directory `generate_swcd/swcd`.

This BSWMD shall adhere to the basic software module description template defined by AUTOSAR [5] and shall contain at least the following entities:

- `BSW-MODULE-DESCRIPTION`
  - `MODULE-ID`
  - `PROVIDED-ENTRYS`
  - `OUTGOING-CALLBACKS`
  - `BSW-MODULE-DEPENDENCYS`
    - `TARGET-MODULE-ID`
    - `REQUIRED-ENTRYS`
    - `EXPECTED-CALLBACKS`
  - `REQUIRED-MODE-GROUPS` (in case the triggering of the module's `MainFunction`s depends on the EcuM's current mode)
  - `BSW-INTERNAL-BEHAVIOR`
    - `EXCLUSIVE-AREAS`
    - `BSW-CALLED-ENTITY`
      - `IMPLEMENTED-ENTRY-REF`
      - `RUNS-INSIDE-EXCLUSIVE-AREA-REFS`
      - `CAN-ENTER-EXCLUSIVE-AREA-REF`
    - `BSW-SCHEDULABLE-ENTITY`
      - `IMPLEMENTED-ENTRY-REF`
      - `RUNS-INSIDE-EXCLUSIVE-AREA-REFS`
      - `CAN-ENTER-EXCLUSIVE-AREA-REF`
    - `BSW-INTERRUPT-ENTITY`

- IMPLEMENTED-ENTRY-REF
- INTERRUPT-CATEGORY
- RUNS-INSIDE-EXCLUSIVE-AREA-REFS
- CAN-ENTER-EXCLUSIVE-AREA-REF

- BSW-TIMING-EVENT (for the triggering of a module's MainFunctions)
  - DISABLED-IN-MODE-IREFS
    (in case the triggering of the MainFunction is disabled in some EcuM modes)
  - STARTS-ON-EVENT-REF
  - PERIOD

- BSW-IMPLEMENTATION
  - CODE-DESCRIPTORS
  - COMPILERS
  - LINKERS
  - GENERATED-ARTIFACTS
    - DEPENDENCY-ON-ARTIFACT
      - ARTIFACT-DESCRIPTOR
      - USAGES
  - PROGRAMMING-LANGUAGE
  - RESOURCE-CONSUMPTION
    - MEMORY-SECTIONS
  - SW-VERSION
  - VENDOR-ID
  - AR-RELEASE-VERSION
  - BEHAVIOR-REF
  - PRECONFIGURED-CONFIGURATION-REFS
  - RECOMMENDED-CONFIGURATION-REFS
  - VENDOR-API-INFIX
  - VENDOR-SPECIFIC-MODULE-DEF-REFS
- BSW-MODULE-ENTRY (for all public API functions provided by the module)
  - SERVICE-ID
  - IS-REENTRANT
  - IS-SYNCHRONOUS

- CALL-TYPE
- EXECUTION-CONTEXT
- SW-SERVICE-IMPL-POLICY

A modified XML schema (schema/AUTOSAR_4-0-3_EB_strict.xsd) is provided where the above elements are mandatory elements. This schema shall be used for validation of the provided BSWMD. - Any BSWMD that validates correctly against the modified XML schema (schema/AUTOSAR_4-0-3_EB_strict.xsd) validates correctly against the vanilla AUTOSAR schema as well.

The elements BSW-MODULE-DESCRIPTION, BSW-IMPLEMENTATION, and BSW-MODULE-ENTRY shall be placed in seperate packages as specified in section 3.1 "Identifying M1 elements in packages" of [12] and further clarified by AUTOSAR Bugzilla issue #52928. These packages shall be named BswModuleDescriptions, Implementations, and BswModuleEntrys respectively.

The SHORT-NAME of the BSW-MODULE-DESCRIPTION shall contain the *unmangled* short name (i.e., the short name without vendor API infix (VENDOR-API-INFIX) and vendor ID (VENDOR-ID)) of the module (see rte_sws_7295 of [10]).

All modules with an upper multiplicity greater than 1 shall specify the vendor API infix in their BSWMD. - The vendor API infix must not be left empty in that case. All modules with an upper multiplicity equal to 1 shall not specify the vendor API infix in their BSWMD.

The SHORT-NAME of the DEPENDENCY-ON-ARTIFACT element shall be the AUTOSAR SWS compliant *unmangled* file name of the generated file (e.g., Can_PBcfg.c in case of the post-build configuration C-file (even if the implementation actually provides Can_1_EB_PBcfg.c)).

The SHORT-LABEL of the ARTIFACT-DESCRIPTOR and the AUTOSAR-ENGINEERING-OBJECT element shall be the *mangled* file name (i.e., the short name of the modules suffied with the vendor API infix and the vendor ID) of the the generated file including the relative path from the module's directory using :: as a path seperator (e.g., src::Can_1_EB_PBcfg.c, include::Can_1_EB_PBcfg.h).

The following values for the USAGE element of a DEPENDENCY-ON-ARTIFACT shall be used for the following purposes:

- COMPILE shall be used for pre-compile time configuration files (i.e., <Module Short Name>_Cfg.h).
- LINK shall be used for link time configuration files (i.e., <Module Short Name>_Lcfg.[ch]).
- EXECUTE shall be used for post-build time configuration files (i.e., <Module Short Name>_PBcfg.[ch]).

In addition to the CATEGORYs of AUTOSAR-ENGINEERING-OBJECTs that are already defined by AUTOSAR, the following additional CATEGORYs shall be used for the following purposes:

- SWMAKE shall be used for make file fragments (e.g., <Module Short Name>_rules.mak and <Module Short Name>defs.mak).
- SWSWCD shall be used for software component descriptions.

- SWTOOL shall be used for executable tools (e.g., an external configuration generator, some pre-processor).
- SWTEMPLATE shall be used for template files that can be used as blueprint for files created by the user.

BSW-MODULE-ENTRYs shall only be defined for API functions that are *not* defined by a module's AUTOSAR SWS. - BSW-MODULE-ENTRYs for API functions already defined by AUTOSAR SWS documents shall only be referenced by BSW-MODULE-ENTRY-REFs. The path for the reference shall adhere to the definitions listed in section 3.1 "Identifying M1 elements in packages" of [12] and further clarified by AUTOSAR Bugzilla issue #52928. As an example some BSW-MODULE-ENTRYs for a CAN driver are provided in samples/BSW_MD/autosar/Can_ModuleEntries.arxml.

IMPLEMENTATION-DATA-TYPEs and SW-BASE-TYPEs shall only be defined for data types that are *not* defined by a module's AUTOSAR SWS. - IMPLEMENTATION-DATA-TYPEs and SW-BASE-TYPEs for data types already defined by AUTOSAR SWS documents shall only be referenced by IMPLEMENTATION-DATA-TYPE-REFs and BASE-TYPE-REFs. The path for the reference shall adhere to the definitions listed in section 3.1 "Identifying M1 elements in packages" of [12] and further clarified by AUTOSAR Bugzilla issue #52928. As an example some IMPLEMENTATION-DATA-TYPEs and SW-BASE-TYPEs for a CAN driver are provided in samples/BSW_MD/autosar/Can_GeneralTypes.arxml.

COMPU-METHODs shall only be defined for enumeration literals or constants that are *not* defined by a module's AUTOSAR SWS. - COMPU-METHODs for enumeration literals or constants already defined by AUTOSAR SWS documents shall only be referenced by COMPU-METHOD-REFs. The path for the reference shall adhere to the definitions listed in section 3.1 "Identifying M1 elements in packages" of [12] and further clarified by AUTOSAR Bugzilla issue #52928. As an example some COMPU-METHODs for a CAN driver are provided in samples/BSW_MD/autosar/Can_GeneralTypes.arxml.

SW-ADDR-METHODs shall only be defined for memory sections that are *not* defined by a module's AUTOSAR SWS. - SW-ADDR-METHODs for memory sections already defined by AUTOSAR SWS documents shall only be referenced by SW-ADDRMETHOD-REFs. The path for the reference shall adhere to the definitions listed in section 3.1 "Identifying M1 elements in packages" of [12] and further clarified by AUTOSAR Bugzilla issue #52928. For illustration purposes the SW-ADDR-METHODs for some of the memory sections defined in [3] are provided in samples/BSW_MD/autosar/MemMap.xml.

The RETURN-TYPE element of BSW-MODULE-ENTRYs must not be used for API functions with return type void, since this is explicitly mandated by constr_4056 of the "AUTOSAR Specification of BSW Module Description Template" [5].

For illustration purposes the BSWMD for an AUTOSAR Can driver (samples/BSW_MD/Can_Bswmd.arxml) is provided. This BSWMD illustrates the description of a Can driver provided by the vendor EB (with vendor ID 1) in the package EB_Can_T2D11M1I0R0.

The mapping of AUTOSAR-ENGINEERING-OBJECTs to the respective FILEs (including path names relative to a module's directory) takes place in a seperate catalog file according to section 4.9 "EngineeringObject" of [12].

This catalog file shall be named <Module Short Name>_Catalog.xml and shall be placed in the directory autosar. For each AUTOSAR module exactly one catalog file shall be provided. This catalog file shall be validated against the schema schema/catalog_V3_0_0.ml.xsd.

For illustration purposes a catalog file for an AUTOSAR Can driver (samples/BSW_MD/Can_Catalog.xml) is provided.

**Requirement (EB_MCAL_0101)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Each AUTOSAR BSW module shall provide a basic software module description as defined by AUTOSAR.

Rationale:
Required for integration in the EB build environment and for the integration of the SchM functionality into the Rte.

**Requirement (EB_MCAL_0102)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The file containing the basic software module description shall be named <Module Short Name>_Bswmd.arxml and shall be placed in the module's generate_swcd/swcd directory.

Rationale:
Have a standardized name and location for the file.

**Requirement (EB_MCAL_0103)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The basic software module description file shall adhere to the basic software module description template defined by AUTOSAR and shall contain at least the elements (and their mandatory sub-elements) listed above.

Rationale:

Required for integration in the EB build environment and for the integration of the SchM functionality into the Rte.

**Requirement (EB_MCAL_0104)**

Coverage:

TEST

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

The basic software module description file shall validate correctly against the strict schema schema/AUTOSAR_4-0-3_EB_strict.xsd.

Rationale:

Enforce AUTOSAR compliance as well as the existence of mandatory parameters by means of the schema.

**Requirement (EB_MCAL_0105)**

Coverage:

TEST

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

The elements BSW-MODULE-DESCRIPTION, BSW-IMPLEMENTATION, and BSW-MODULE-ENTRY shall be placed in seperate packages. These packages shall be named BswModuleDescriptions, Implementations, and BswModuleEntrys respectively.

Rationale:

AUTOSAR specifies this that way.

**Requirement (EB_MCAL_0107)**

Coverage:

TEST

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

The SHORT-NAME of the BSW-MODULE-DESCRIPTION shall contain the unmangled short name (i.e., the short name without vendor API infix (VENDOR-API-INFIX) and vendor ID (VENDOR-ID)) of the module.

Rationale:

The SHORT-NAME is used by the Rte generator to generate SchM API functions (that must not contain vendor API infix and vendor ID).

**Requirement (EB_MCAL_0108)**

Coverage:

TEST

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

Modules with an upper multiplicity greater than 1 shall specify the vendor API infix in their BSWMD. - The vendor API infix must not be left empty in that case.

Rationale:

The vendor API infix is required to create mangled names in to prevent name clashes among different instances of the module.

**Requirement (EB_MCAL_0109)**

Coverage:

TEST

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

Modules with an upper multiplicity equal to 1 shall not specify the vendor API infix in their BSWMD.

Rationale:

No name mangling is required for those modules.

**Requirement (EB_MCAL_0110)**

Coverage:

TEST

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

The SHORT-NAME of the DEPENDENCY-ON-ARTIFACT element shall be the AUTOSAR SWS compliant unmangled file name of the generated file (e.g., Can_PBcfg.c in case of the post-build configuration C-file (even if the implementation actually provides Can_1_EB_PBcfg.c)).

**Requirement (EB_MCAL_0111)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The SHORT-LABEL of the ARTIFACT-DESCRIPTOR and the AUTOSAR-ENGINEERING-OBJECT element shall be the mangled file name (i.e., the short name of the modules suffied with the vendor API infix and the vendor ID) of the the generated file including the relative path from the module's directory using :: as a path seperator (e.g., src::Can_1_EB_PBcfg.c, include::Can_1_EB_PBcfg.h).

Rationale:
Using the mangled file name here (instead of implicitly mangling the name in some tooling) provides more flexibility. Giving the relative path facilitates placing the files in dedicated directories.

**Requirement (EB_MCAL_0112)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The value COMPILE for the USAGE element of a DEPENDENCY-ON-ARTIFACT shall be used for pre-compile time configuration files (i.e., <Module Short Name>_Cfg.h).

Rationale:
Explicitly mark that this is a pre-compile time configuration file.

**Requirement (EB_MCAL_0113)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The value LINK for the USAGE element of a DEPENDENCY-ON-ARTIFACT shall be used for link time configuration files (i.e., <Module Short Name>_Lcfg.[ch]).

Rationale:
Explicitly mark that this is a link time configuration file.

**Requirement (EB_MCAL_0114)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The value EXECUTE for the USAGE element of a DEPENDENCY-ON-ARTIFACT shall be used for post-build time configuration files (i.e., <Module Short Name>_PBcfg.[ch]).

Rationale:
Explicitly mark that this is a post-build time configuration file.

**Requirement (EB_MCAL_0115)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The value SWMAKE for the CATEGORYs of AUTOSAR-ENGINEERING-OBJECTs shall be used for make file fragments (e.g., <Module Short Name>_rules.mak and <Module Short Name>defs.mak).

Rationale:
Explicitly mark that this is a make file fragment.

**Requirement (EB_MCAL_0116)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The value SWSWCD for the CATEGORYs of AUTOSAR-ENGINEERING-OBJECTs shall be used for software component descriptions.

Rationale:
Explicitly mark that this is a software component description.

**Requirement (EB_MCAL_0117)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The value SWTOOL for the CATEGORYs of AUTOSAR-ENGINEERING-OBJECTs shall be used for executable tools (e.g., an external configuration generator, some pre-processor).

Rationale:
Explicitly mark that this is an executable tool.

**Requirement (EB_MCAL_0118)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The value SWTEMPLATE for the CATEGORYs of AUTOSAR-ENGINEERING-OBJECTs shall be used for template files that can be used as blueprint for files created by the user.

Rationale:
Explicitly mark that this is an template file.

**Requirement (EB_MCAL_0129)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
BSW-MODULE-ENTRYs shall only be defined for API functions that are not defined by a module's AUTOSAR SWS.

Rationale:
In the long run, AUTOSAR will provide the BSW-MODULE-ENTRYs for the API functions standardized by the AUTOSAR SWS documents.

**Requirement (EB_MCAL_0130)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
BSW-MODULE-ENTRYs for API functions already defined by AUTOSAR SWS documents shall only be referenced by BSW-MODULE-ENTRY-REFs. The path for the reference shall adhere to the definitions listed in section 3.1 "Identifying M1 elements in packages" of the "AUTOSAR Generic Structure Template" and further clarified by AUTOSAR Bugzilla issue #52928.

Rationale:
Prevent duplication of information and have a standardized path in the reference.

**Requirement (EB_MCAL_0131)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
IMPLEMENTATION-DATA-TYPEs shall only be defined for data types that are not defined by a module's AUTOSAR SWS.

Rationale:
In the long run, AUTOSAR will provide the IMPLEMENTATION-DATA-TYPEs for the data types standardized by the AUTOSAR SWS documents.

**Requirement (EB_MCAL_0132)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
IMPLEMENTATION-DATA-TYPEs for data types already defined by AUTOSAR SWS documents shall only be referenced by IMPLEMENTATION-DATA-TYPE-REFs. The path for the reference shall adhere to the definitions listed in section 3.1 "Identifying M1 elements in packages" of the "AUTOSAR Generic Structure Template" and further clarified by AUTOSAR Bugzilla issue #52928.

Rationale:
Prevent duplication of information and have a standardized path in the reference.

**Requirement (EB_MCAL_0133)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
SW-BASE-TYPEs shall only be defined for data types that are not defined by a module's AUTOSAR SWS.

Rationale:
In the long run, AUTOSAR will provide the SW-BASE-TYPEs for the data types standardized by the AUTOSAR SWS documents.

**Requirement (EB_MCAL_0134)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
SW-BASE-TYPEs for data types already defined by AUTOSAR SWS documents shall only be referenced by SW-BASE-TYPE-REFs. The path for the reference shall adhere to the definitions listed in section 3.1 "Identifying M1 elements in packages" of the "AUTOSAR Generic Structure Template" and further clarified by AUTOSAR Bugzilla issue #52928..

Rationale:
Prevent duplication of information and have a standardized path in the reference.

**Requirement (EB_MCAL_0135)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
COMPU-METHODs shall only be defined for enumeration literals or constants that are not defined by a module's AUTOSAR SWS.

Rationale:
In the long run, AUTOSAR will provide the COMPU-METHODs for the enumeration literals standardized by the AUTOSAR SWS documents.

**Requirement (EB_MCAL_0136)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
COMPU-METHODs for enumeration literals or constants already defined by AUTOSAR SWS documents shall only be referenced by COMPU-METHOD-REFs. The path for the reference shall adhere to the definitions listed in section 3.1 "Identifying M1 elements in packages" of the "AUTOSAR Generic Structure Template" and further clarified by AUTOSAR Bugzilla issue #52928.

Rationale:
Prevent duplication of information and have a standardized path in the reference.

**Requirement (EB_MCAL_0137)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
SW-ADDR-METHODs shall only be defined for memory sections that are not defined by a module's AUTOSAR SWS.

Rationale:
In the long run, AUTOSAR will provide the SW-ADDR-METHODs for the memory sections standardized by the AUTOSAR SWS documents.

**Requirement (EB_MCAL_0138)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
SW-ADDR-METHODs for memory sections already defined by AUTOSAR SWS documents shall only be referenced by SW-ADDRMETHOD-REFs. The path for the reference shall adhere to the definitions listed in section 3.1 "Identifying M1 elements in packages" of the "AUTOSAR Generic Structure Template" and further clarified by AUTOSAR Bugzilla issue #52928.

Rationale:
Prevent duplication of information and have a standardized path in the reference.

**Requirement (EB_MCAL_0139)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The RETURN-TYPE element of BSW-MODULE-ENTRYs must not be used for API functions with return type void.

Rationale:
"AUTOSAR Specification of BSW Module Description Template" explitly mandates this in constr_4056.

**Requirement (EB_MCAL_0119)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Each AUTOSAR BSW module shall provide exactly one catalog file containing a mapping from AUTOSAR-ENGINEERING-OBJECTs and ARTIFACT to FILEs as defined by AUTOSAR.

Rationale:
Required for integration in the EB build environment.

**Requirement (EB_MCAL_0120)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The catalog file shall be named <Module Short Name>_Catalog.xml and shall be placed in the modules autosar directory.

Rationale:
Have a standardized name and location for the file.

**Requirement (EB_MCAL_0121)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The catalog file shall validate correctly against the schema schema/catalog_V3_0_0.ml.xsd.

Rationale:
Enforce the compliance to the ASAM standard by means of the schema.

## 6.1.4. Template based code generator for AUTOSAR BSW modules

### 6.1.4.1. General rules

Files included in code templates shall be protected from multiple inclusion by an include guard mechanism to prevent unintentional multiple inclusion of the respective file. Consider the following code snippet as an example:

```
!/* *** multiple inclusion protection *** */!]
[!IF "not(var:defined('LIN_CHECKS_M'))"!]
[!VAR "LIN_CHECKS_M"="'true'"!]

... rest of file ...

[!ENDIF!]
```

EB tresos Studio uses a mixed interpretation scheme to evaluate conditional expressions, e.g. in `IF` statements. If the conditional consists of one node path to a boolean parameter only (e.g. `[!IF "SomeParameter"!]`), then the parameter value is implicitly compared to `'true'`. If the conditional expression is not primary, then it is interpreted according to the XPath specification [11]. This means that in `[!IF "Par1 and Par2"!]` the subexpressions `Par1` and `Par2` evaluate to `true` if and only if `Par1` and `Par2` are a non empty node-sets. - For clarity and simplicity these detailed knowledge about the evaluation logic shall not be used.

Use `[!IF "SomeParameter = 'true'"!]` or `[!IF "SomeParameter > 0"!]` or `[!IF "SomeParameter != ''"!]` or `[!IF "node:exists(SomeParameter)"!]` and *do not* use `[!IF "SomeParameter"!].` - The same recommendation applies to conditional expressions in XDM files.

**Requirement (EB_MCAL_0042)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Files included in code templates shall be protected from multiple inclusion by an include guard mechanism.

Rationale:
Multiple inclusion often happens by accident and is not intentional. - Having a multiple inclusion guard in place prevents damage in case a file in (unintentionally) included multiple times.

**Requirement (EB_MCAL_0043)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
CamelCase should be used for the naming of all identifiers (macros, variables). - E.g.: FrIfPduReference-Count

Rationale:
Have a common capitalization across all module's template files.

**Requirement (EB_MCAL_0044)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
In expressions which evaluate to a boolean result, comparison operators or boolean evaluation functions shall be used explicitly. The implicit conversion of node-sets to boolean values shall not be used.

Rationale:
EB tresos Studio uses a mixed interpretation scheme to evaluate conditional expressions. - For clarity and simplicity these detailed knowledge about the evaluation logic shall not be used.

### 6.1.4.2. Layout

The section inside opening ([!XXX!]) and closing ([!ENDXXX!]) template code generator constructs shall be indented (e.g. for XXX=IF, LOOP, FOR). The indentation depth shall be the same as for C code which is defined in EB's C style guide [8] - The indentation rule applies to all lines inside these constructs, independent of whether they contain C source code or template code.

The section between lines which contain opening ({) and closing ({) braces on the same column shall be indented just like C source code. - The indentation rule applies to all lines inside this section, independent of whether they contain C source code or template code.

Consider the following code snippet as an example:

```
[!IF "<expression>"]
  [!VAR "SomeCount" = "0"!]
  /** \brief This array holds the configuration parameters listed in ModContainer */
  STATIC CONST(Mod_Type, MODULE_CONST) Mod_ConstArray[[!"num:i(count(ModContainer/*))"!]] =
  {
    [!LOOP "node:order(ModContainer/*, 'ModParameter1')"!]
      {
        [!"num:hex(ModParameter1)"!],    /* [!"name(ModParameter)"!] */
        [!"ModParameter2"!],    /* [!"name(ModParameter2)"!] */
        [!"ModParameter3"!],    /* [!"name(ModParameter3)"!] */
      },
      [!VAR "SomeCount" = "$SomeCount + ModParameter3"!]
    [!ENDLOOP!]
  }
[!ENDIF!]
```

To increase readability and maintainability of the template code, the keyword [!AUTOSPACING!] shall be used whenever possible. When using this keyword Additional 'line-end markers' like '[!//' to suppress new lines can be avoided.

Consider the following code snippet as an example:

```
/* --------{ EB Automotive C Source File }-------- */

/* This file contains [...] */

[!AUTOSPACING!]
< some more template variable initializations, macros, etc. >
[!//
```

```
/*=================[inclusions]=============================================*/
```

If switched on `[!AUTOSPACING!]` has the following effect for commands which do not produce output:

- if there are only spaces in front of the command, they will be removed
- if there are only spaces and a newline after the command, the spaces and the newline will be removed

Currently there are some known issues in the template-based generator itself, which prevent the `[!AUTOSPACING!]` from working as expected (see ASCCCB-1330). Until these issues have been fixed the following things have to be kept in mind:

- Spaces are being removed before first generator directive.

  Example:

  ```
  #define MOD_VALUE [!IF "<condition>"!]1U[!ELSE!]2U[!ENDIF!]
  ```

  leads to

  ```
  #define MOD_VALUE1U or #define MOD_VALUE2U
  ```

  Workaround:

  ```
  #define MOD_VALUE [!IF "<condition>"!] 1U[!ELSE!] 2U[!ENDIF!]
  ```

- `[!/*comment*/!]` is generating a new line. - This applies if the line contains the code generator comment only.

  Workaround:

  ```
  [!/*comment*/!][!//
  ```

- When entering `[!ELSE!]` or `[!ELSEIF "<expression>"!]` a new line is being generated.

Workaround:

```
[!ELSE!][!//
[!ELSEIF "<expression>"!][!//
```

**Requirement (EB_MCAL_0045)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The section inside opening ([!"XXX"!]) and closing ([!END"XXX"!]) template code generator constructs shall be indented (e.g. for "XXX"=IF, LOOP, FOR). The indentation depth shall be the same as for C code which is defined in EB's C style guide .

Rationale:
Readability shall be like C code.

**Requirement (EB_MCAL_0046)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The generated C-code shall meet all the requirements defined in EB's C style guide .

Rationale:
The generated C-code shall not differ in style from non-generated C-code.

**Requirement (EB_MCAL_0047)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The section between lines which contain opening ({) and closing (}) braces on the same column shall be indented just like C source code. - The indentation rule applies to all lines inside this section, independent of whether they contain C source code or template code.

Rationale:
The generated C-code shall not differ in style from non-generated C-code.

**Requirement (EB_MCAL_0048)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The keyword [!AUTOSPACING!] shall be used for all template code.

Rationale:
Additional line-end markers like '[!//' to suppress new lines can be avoided. Template code indentation can easily be used, as leading spaces are not generated. This increases readability and maintainability.

### 6.1.4.3. Statements

In order to facilitate overage analyzes of the code templates, the two forms of the `[!ASSERT!]` statement of the EB tresos Studio template based generator language shall not be used. A combination of `[!IF!]` and `[!ERROR!]` statement shall be used instead.

Furthermore the `[!LOOP!]` statement shall be used only to loop over node-sets, i.e. where 0, 1 or more nodes can be present. The `[!SELECT!]` statement on the other hand shall only be used to select single nodes.

**Requirement (EB_MCAL_0049)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The two forms of the [!ASSERT!] statement of the EB tresos Studio template based generator language shall not be used. A combination of [!IF!] and [!ERROR!] statement shall be used instead.

Rationale:
Coverage analyzes of template code is simplified.

**Requirement (EB_MCAL_0050)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The [!LOOP!] statement shall only be used to loop over node-sets, i.e. where 0, 1 or more nodes can be present. The [!SELECT!] statement shall only be used to select single nodes.

Rationale:
Clearly indicate when something is a loop.

### 6.1.4.4. Variables

The variables predefined by the code generator (such as `$date`, `$target`, etc.) shall not be overwritten. - The developer documentation of EB tresos Studio [9] contains a list of all pre-defined variables.

**Requirement (EB_MCAL_0051)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The variables predefined by the code generator (such as $date, $target, etc.) shall not be overwritten.

Rationale:
Overwriting these variables would lead to unexpected behavior.

### 6.1.4.5. Macros

Macros do not introduce a new namespace. - Therefore in order to prevent name clashes the following rules shall be obeyed:

- Names of macro parameters shall not have the same name as a variable found somewhere else in the code template.
- If variables are defined within a macro and shall only be used by this macro, their name shall be prefixed (for example with `Mac`) and they shall be defined before their first use.
- Variables used as *output* parameters shall be prefixed with `Out`.

**Requirement (EB_MCAL_0052)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Names of macro parameters shall not have the same name as a variable found somewhere else in the code template.

Rationale:
Macros do not have an own namespace.

**Requirement (EB_MCAL_0053)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
If variables are defined within a macro and shall only be used by this macro, their name shall be prefixed (for example with Mac) and they shall be defined before their first use.

Rationale:
Macros do not have an own namespace.

**Requirement (EB_MCAL_0054)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Variables used as parameters shall be prefixed with Out.

Rationale:

Macros do not have an own namespace.

## 6.1.5. Consistency checks of AUTOSAR BSW modules

This section defines rules for checking constraints among the parameters of a single AUTOSAR BSW module (intra module checks) as well as among different AUTOSAR BSW modules (inter module checks).

### 6.1.5.1. General rules

**Requirement (EB_MCAL_0055)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

The error message shall be concise and short to show the user what is wrong. Code generator template checks especially need to show the user which parameter is wrong, for example by providing the path to the offending parameter(s) starting from module-level.

Consider the following two error messages as an example:

- CAN controller `[!"$CanControllerName"!]` needs to be activated for use with the CAN Interface.
- If 'Zero Cost Operation' is enabled the parameters 'PduRSingleIf' and 'PduRSingleTp' have to be set.

#### 6.1.5.1.1. Intra-module checks

Intra-module checks take care that configuration parameter within a module are consistent.

**Requirement (EB_MCAL_0067)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:
For every integer parameter the range shall be checked if this is not done in the AUTOSAR parameter definition.

**Requirement (EB_MCAL_0068)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
All parameter constraints specified in the SWS shall be checked.

#### 6.1.5.1.2. Inter-module checks

Intra-module checks take care that configuration parameter of a module is consistent w.r.t. configuration parameters of another module.

**Requirement (EB_MCAL_0069)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Inter-module-checks shall only access those parameters from other modules that are defined in the StMD provided by AUTOSAR.

Rationale:
The interchangeability is reduced by accessing non standard parameters by a inter module check.

**Requirement (EB_MCAL_0070)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Inter-module-checks should only check those parameters that are absolutely necessary for code generation.

Rationale:
The interchangeability is reduced by accessing non standard parameters by a inter module check.

**Requirement (EB_MCAL_0071)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
If constraints checked are necessary for a successful code generation, a failed check shall raise an error. All other tests shall raise a warning.

Rationale:
Modules may be stubbed or exist outside of the tool chain, so only produce an error, if the configuration is absolutely necessary.

**Requirement (EB_MCAL_0072)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
In rare cases one parameter (semantically) is located in two module configurations. If this happens it shall be checked that both parameters contain the same value. Both modules shall to perform this check.

Rationale:
Ensure consistency of redundant parameters.

For example the PDU Id of a single CAN Interface Rx-PDU is defined both in the CAN Interface and PDU Router configurations. In that case a consistency check implemented in the CAN Interface module and in the PDU Router module shall enforce that the PDU Id is identically configured in both modules.

### 6.1.5.2. Checks performed in the schema (`<Module Short Name>.xdm` file)

**Requirement (EB_MCAL_0056)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Any constraint should preferably be checked in the schema if possible (and not in the code generator template)

Rationale:
Give the user an interactive feedback if the EB tresos Studio GUI is used for configuration.

**Requirement (EB_MCAL_0057)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Checks performed in the schema shall be so fast that they are not notable by the user. The user interface shall show visual feedback within 1 second for realistic configurations. In case a check requires too much runtime, it shall be performed in the code generator template.

Rationale:
The checks shall not hinder or slow down the user in her configuration work.

A *realistic configuration* consists of about:

- 100 PDUs
- 1000 signals
- 200 memory blocks
- 500 diagnostic events

Consider the following hints to create high-performant checks in the schema:

- Exploit lazy evaluation: If many expressions are combined with `and` or `or`, place those expressions at the beginning that are easy to evaluate and most likely cause the evaluation of the whole expression to be finished.
- Use the `/*[...]` construct with care, because it can easily operate on huge nodesets. Try to reduce the nodeset as early as possible.

For further developer hints regarding the performance of X-Path expressions, take a look in the BSW Developer Hints Wiki page

A *warning* shall be thrown if

- an inconsistency has been detected with an external module which results in a compile or link error. E.-g. check for required Dem symbols.

An *error* shall be thrown if

- an inconsistency has been detected with an external module which results in wrong runtime behavior.
- a module internal inconsistency has been detected.
- the generation of the configuration code is not possible due to missing information.

**Requirement (EB_MCAL_0058)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
A warning shall be thrown if an inconsistency has been detected with an external module which results in a compile or link error. E.g. check for required Dem symbols.

Rationale:
Modules may be stubbed or exist outside of the tool chain. The symbols may then be present at compile time without being part of the ECU configuration.

**Requirement (EB_MCAL_0059)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
An error shall be thrown if an inconsistency has been detected with an external module which results in wrong runtime behavior.

Rationale:
Incorrect run-time behavior due to a configuration inconsistency shall be prevented.

**Requirement (EB_MCAL_0060)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
An error shall be thrown if a module internal inconsistency has been detected.

Rationale:
Module internal inconsistencies always lead to a malfunction of the module upon run-time.

**Requirement (EB_MCAL_0061)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
An error shall be thrown if the generation of the configuration code is not possible due to missing information.

Rationale:
The user shall be informed about this problems as early as possible instead of producing an incomplete configuration code which yields compile-time errors.

Consider the following code snippet as an example warning for an inconsistency with an external module:

```
<a:da name="WARNING" type="XPath">
  <a:tst
    expr="count(as:modconf('Dem')[1]/DemConfigSet/*[1]/DemEventParameter/
      *[@name = 'LINIF_E_RESPONSE']) = 1"
    false="LIN Interface needs the DEM event /Dem/DemConfigSet/
      DemEventParameter/LINIF_E_RESPONSE for a successful build"/>
</a:da>
```

**Requirement (EB_MCAL_0062)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:

Any CALCULATION-FORMULA of a configuration parameter listed in the StMD shall be implemented using the functionality of the XDM format.

Implement the `CALCULATION-FORMULA`s using `INVALID`, `DEFAULT` and similar parameter attributes. - When using parameters of other modules also check for consistency (name, semantic, ...).

**Requirement (EB_MCAL_0063)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

A module shall check the presence of configuration of the modules it requires for a successful generation of its own configuration code. If a required module is missing, an error shall be issued. The error message format is: The <my module's long name> module needs the <required module's long name> module.

Consider the following code snippet as an example warning in case of a missing module configuration:

```
<a:tst expr="count(as:modconf('PduR')) = 1"
  false="The LIN Interface module needs the PDU Router module."/>
```

### 6.1.5.3. Code generator template checks

**Requirement (EB_MCAL_0064)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

The checks should be located in the file generate/include/<Module Short Name>_checks.m if they are architecture independent. Architecture and/or derivative specific checks should be located in an equally

named file in the architecture/derivative specific subdirectory. Each check file should be included only once by a single template file.

**Requirement (EB_MCAL_0065)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The code generator template shall not access possibly non-existent parameters without checking their existence in advance. Possibly non-existent parameters are those that have zero as lower multiplicity or are located within a container (below module level) that has zero as lower multiplicity. This also applies to optional parameters.

Rationale:
Accessing a non-existent parameter leads to an XPath-error which is unlikely to help the user to find the source of the problem.

Hint: Use the X-Path functions `node:exists(<node>)`, `node:refexists(<reference>)`, and `node:refvalid(<reference>)` to check whether a specific node or a reference exists or references a valid node.

Consider the following code snippet as an example for using `node:refvalid()` to check whether a reference (`EcuMMcuInitConfig`) exists and points to a valid node prior to accessing this node:

```
[!IF "node:refvalid(./EcuMMcuInitConfig)"!]
[!VAR "EcuMMcuConfigPtr"="name(as:ref(./EcuMMcuInitConfig))"!]
```

**Requirement (EB_MCAL_0066)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Checks performed in the code generator template shall not run for longer than 10 seconds (for a huge but realistic configuration - see EB_MCAL_0057). If there are checks that run longer escalate to EB Project

Software Architect (PSA) for a decision. Possible decisions: The check is removed, other checks are removed, or the increased runtime is accepted. In either case the decision needs to be documented in the module's User's Guide, chapter "Known Problems".

#### 6.1.5.4. BSW compile time and runtime SW consistency checks

**Requirement (EB_MCAL_0073)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
A BSW module's main implementation file (i.e., <Module Short Name>.c) shall check consistency of the versions defined in the modules's version information file (i.e., <Module Short Name>_Version.h). The version-checks shall only be performed in one *.c file. The version-checks shall not be performed in any *.h file. This check applies for the AUTOSAR release version macros <MODULE SHORT NAME>_AR_RELEASE_MAJOR_VERSION, <MODULE SHORT NAME>_AR_RELEASE_MINOR_VERSION, <MODULE SHORT NAME>_AR_RELEASE_REVISION_VERSION, the AUTOSAR SWS version macros <MODULE SHORT NAME>_AR_MAJOR_VERSION, <MODULE SHORT NAME>_AR_MINOR_VERSION, <MODULE SHORT NAME>_AR_PATCH_VERSION, for the module version macros <MODULE SHORT NAME>_SW_MAJOR_VERSION, <MODULE SHORT NAME>_SW_MINOR_VERSION, <MODULE SHORT NAME>_SW_PATCH_VERSION, and for the vendor ID <MODULE SHORT NAME>_VENDOR_ID.

Rationale:
Meet AUTOSAR's requirements and keep maintainance effort low in case of updates.

Consider the following code snippet as an example for checking the `AR_MAJOR_VERSION` in the `WdgM.c` file:

```
#if (!defined WDGM_AR_MAJOR_VERSION) /* configuration check */
#error WDGM_AR_MAJOR_VERSION must be defined
#endif /* if (!defined WDGM_AR_MAJOR_VERSION) */

/* major version check */
#if (WDGM_AR_MAJOR_VERSION != 4U)
#error WDGM_AR_MAJOR_VERSION wrong (!= 4U)
#endif /* if WDGM_AR_MAJOR_VERSION */
```

**Requirement (EB_MCAL_0074)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Although required by the SRS General, BSW004, inter-module version-checks shall not be done. A deviation is documented globally for all BSW modules within the release notes and shall not be documented for each module.

Rationale:
Inter-module checks are very error prone and could not be handled between different suppliers. Facilitate the use of AUTOSAR 4.0 MCAL modules in an AUTOSAR 4.1 stack.

# 6.2. Naming conventions

## 6.2.1. BSW module naming convention

To be able to coexist with other modules in the same directory, a module's root directory must follow a specific name scheme: `<Module Short Name>_TS_<Version String>`· Hereby `Version String` is a short string containing numbers for architecture, derivative and version numbers, e.g. `T21D1M2I2R0`.

- `T`: target ID
- `D`: derivative ID
- `M`: module's major version number
- `I`: module's minor version number
- `R`: reserved (must be `0`)

**Requirement (EB_MCAL_0078)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The name of the module's root directory shall comply to the naming scheme outlined above.

Rationale:
Coexistence with other BSW modules in the same directory.

#### 6.2.1.1. Target and derivative naming

The support of multiple sub derivatives can be handled with the ECU resource properties file concept (see Section 6.5, “ECU resource properties file concept”).

The target derivative mapping is defined between EB and the chip manufacturer in the statement of work (SOW) document. The administration and the assignment of IDs is in the responsibility of EB.

A complete list of the target derivative numbers could be found at Encoding Architecture and Derivative Wiki Page.

**Requirement (EB_MCAL_0079)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The target ID and the derivative ID of the module shall comply to the IDs defined in the SOW document.

Rationale:
Coexistence with BSW modules for other architectures/derivatives in the same directory.

### 6.2.2. API naming convention

The names of APIs of modules with an upper multiplicity greater than 1 (e.g., Can, Fr) shall be constructed according to the following pattern: `<Module Short Name>_<Vendor Id>_<Vendor Api Infix>_<Service Name>()` (see AUTOSAR Bugzilla issue #53325). Thus for example the service `AckIRQ()` of the module `Fr` of a vendor with vendor ID `1` using a vendor API infix of `MFR4300` shall look the following way: `Fr_1_MFR4300_AckIRQ()`. The same pattern shall be applied for published parameters (e.g., `FR_1_MFR4300_SHADOW_BUFFER_NUM`), for memory mapping and compiler abstraction `#defines` (e.g., `FR_1_MFR4300_`-

STOP_SEC_VAR_FAST_UNSPECIFIED and FR_1_MFR4300_CODE), and for public typedefs that are specific to the respective module (e.g., Fr_1_MFR4300_ConfigOffsetType).

**Requirement (EB_MCAL_0106)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The names of APIs of modules with an upper multiplicity greater than 1 shall be constructed according to the following pattern: <Module Short Name>_<VendorId>_<VendorApiInfix>_<ServiceName>() (see AUTOSAR Bugzilla issue #53325). The same pattern shall be applied for published parameters, for memory mapping and compiler abstraction #defines and for public typedefs that are specific to the respective module.

Rationale:
Prevent name clashes upon deployment of multiple different instances of these modules within a single ECU.

# 6.3. Structure of an AUTOSAR BSW module

## 6.3.1. Overview

Each AUTOSAR module contains three kinds of files:

Generated files:
Files that are directly changed or generated by the configuration tool (i.e, EB tresos Studio including its Java and template-based generator) or some external generator. - The files <Module Short Name>_Cfg.h, <Module Short Name>_Lcfg.[ch], and <Module Short Name>_PBcfg.[ch] fall into that category.

Dependent files:
Files that are dependent on input from generated files. Dependent files are usually files that need pre-compile time configuration from generated files, e.g. preprocessor macros.

Static files:
Files that are neither generated by the configuration tool nor depend on information from generated files. Static files are thus files that do not need any pre-compile time configuration or any other information from generated files.

The complete build process of an BSW module can be divided into the following three different steps:

1. The generation of the configuration `*.c` and `*.h` files from the `*.epc` or `*.xdm` files by configuration tool (or some external generator) is called the *configuration generation* of the module.
2. The compilation of all static, dependent, and generated files is called the *module compilation process*.
3. The linking of all object files resulting from the previous step is called *module linking process*.

The general file/directory structure of an AUTOSAR BSW module shall be as follows:

`autosar`:
directory for configuration scheme file(s) for this AUTOSAR BSW module in ARXML format (e.g., mainly the `<Module Short Name>.bmd` file) and for the corresponding catalog files (`<Module Short Name>_Catalog.xml`).

`config`:
directory for configuration scheme file(s) for this AUTOSAR BSW module in XDM format (e.g., mainly the `<Module Short Name>.xdm` file). - Requirements for the configuration scheme are listed in Section 6.1.2, “Configuration scheme for AUTOSAR BSW modules”.

`config_ext`:
directory for pre- and recommended configuration file(s) for this AUTOSAR BSW module in XDM format (optional). Requirements for the pre- and recommended configurations are listed in Section 6.3.2.5, “Pre- and recommended configuration”.

`generate`:
directory for the generated module configuration files. - This directory shall contain the following sub-directories:

`include`:
generated module configuration include files (e.g., `<Module Short Name>_Cfg.h`, `<Module Short Name>_Lcfg.h`, and `<Module Short Name>_PBcfg.h`).

`src`:
generated module configuration source files (e.g., `<Module Short Name>_Lcfg.c`, and `<Module Short Name>_PBcfg.c`). - Note that there is no such thing as a `<Module Short Name>_Cfg.c` file, since any `*.c` file is per definition link time or post-build time configuration.

`generate_swcd`:
directory for the (generated) basic software module desription files. - This directory shall contain the following sub-directories:

`swcd`:
directory for the basic software modules description files (`<Module Short Name>_Bswmd.arxml`).

`include`:
directory for the header files of this AUTOSAR BSW module.

`resources`:
directory for service needs requests (optional). For details, refer to Section 6.8, “Service needs”.

`src`:
directory for the source (a.k.a. implementation) files of this AUTOSAR BSW module.

`make`:
directory the make files of this AUTOSAR BSW module (i.e., `<Module Short Name>_defs.mak` and `<Module Short Name>_rules.mak`). - Examples for these files will be given later on.

`META-INF`:
directory containing parts of the plugin information used by EB tresos Studio. - A file named `MANIFEST.MF` must be contained in that directory.

`plugin.xml`:
file containing further parts of the plugin information used by EB tresos Studio.

`anchors.xml`:
file containing plugin information about the location of documentation shown in EB tresos Studio. - This file shall be used to refer to the documentation (e.g., PDF document) of the MCAL module.

**Requirement (EB_MCAL_0075)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The general file/directory structure of an AUTOSAR BSW module shall match the directory structure outlined above.

Rationale:
Interoperability with EB tresos Studio and the EB tresos AutoCore (user) build environment.

Templates for the make files (`module_defs.mak` and `module_rules.mak`) for an AUTOSAR BSW module are provided in the directory `samples/Makefiles`. - The string `<Module Short Name>` shall be replaced by your module's short name.

**Requirement (EB_MCAL_0076)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The <Module Short Name>_rules.mak file shall have the layout illustrated by the template samples/Makefiles/module_rules.mak.

Rationale:
Interoperability with EB tresos Studio and the EB tresos AutoCore (user) build environment.

**Requirement (EB_MCAL_0077)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The <Module Short Name>_defs.mak file shall have the layout illustrated by the template samples/Make-files/module_defs.mak.

Rationale:
Interoperability with EB tresos Studio and the EB tresos AutoCore (user) build environment.

## 6.3.2. `plugin.xml`

Details about the content of the `plugin.xml` file can be found in the developer documentation of EB tresos Studio [9].

### 6.3.2.1. Com Importer registration

#### 6.3.2.1.1. Overview

In order to provide support for importing *CAN configuration information* from a DBC file or from an AUTOSAR system description, *LIN configuration information* from an LDF file or from an AUTOSAR system description, and *FlexRay configuration* from a FIBEX file or from an AUTOSAR system description, the so-called Com Importer needs to be registered in the CAN/LIN/FlexRay driver module's `plugin.xml` file.

In the following a full-fledged example based on the Com importer registration for the AUTOSAR 3.x CAN driver for the NEC V850ECAG4M is provided:

```
<extension point="dreisoft.tresos.comimporter.api.plugin.comtransformer">
  <comtransformer
```

```
    id="ComTransformer_<Module ID>"
    moduleId="<Module ID>">
    <transformer
      class="dreisoft.tresos.comimporter.api.transformer.asr40.
        <Module Short Name>Transformer">
      <parameter name="canfiltermask.dontcarebit" value="1"/>
      <parameter name="canfiltermask.std.shiftleft" value="18"/>
      <parameter name="canfiltermask.ext.shiftleft" value="0"/>
    </transformer>
    <variant value="CanConfigSet.ImportAsSingleContainer"/>
  </comtransformer>
</extension>
```

In the following subsections the different important elements in the registration are explained in detail.

#### 6.3.2.1.2. Choosing the proper transformer class based on the AUTOSAR version

The first important thing regarding the registration in `plugin.xml` is the fact that depending on whether the module is an AUTOSAR 2.1, an AUTOSAR 3.0/3.1, or an AUTOSAR 4.0 module, different transformer classes have to be registered. - For AUTOSAR 2.1 the following class shall be used: `dreisoft.tresos.comimporter.api.transformer.<Module Short Name>Transformer`

For AUTOSAR 3.0/3.1 the following class is required: `dreisoft.tresos.comimporter.api.transformer.asr30.<Module Short Name>Transformer`

For AUTOSAR 4.0 the following class is required: `dreisoft.tresos.comimporter.api.transformer.asr40.<Module Short Name>Transformer`

#### 6.3.2.1.3. Special issue - Com Importer for CAN drivers

##### 6.3.2.1.3.1. Defining the behavior when computing filter masks

Upon import the Com Importer calculates filter masks for the CAN hardware objects. The behavior when calculating filter masks upon import can be modified via the following parameters:

`canfiltermask.dontcarebit`:
: This parameter is used to define whether a `0` or a `1` is used as don't care bit in the CAN filter masks. In case this parameter is not present, as default value of `0` is used.

`canfiltermask.std.shiftleft`:
This parameter is used to define how many bits the filter masks for standard CAN IDs have to be shifted to the left. In case this parameter is not present, as default value of `0` (i.e., no shifting) is used.

`canfiltermask.ext.shiftleft`:
This parameter is used to define how many bits the filter masks for extended CAN IDs have to be shifted to the left. In case this parameter is not present, as default value of `0` (i.e., no shifting) is used.

Note that this behavior solely depends on how the CAN drivers configuration generator wants filter masks to be defined. - This might be different from the handling of the filter masks on the actual hardware.

#### 6.3.2.1.3.2. Defining the behavior regarding the multiplicity of `CanConfigSet`

Based on the existence (or non existence) of the variant `CanConfigSet.ImportAsSingleContainer`, the Com Importer creates `CanConfigSet` as a single container (which is not correct according to AUTOSAR) or as a list of containers (which is correct according to AUTOSAR).

Based on the structure of the `Can.xdm` schema file provided by the MCAL vendor, the variant thus has to be included/excluded in the registration of the Com Importer in order to ensure a correct import here.

**NOTE**

In the long run, the MCAL vendors must use a list of containers (i.e., an upper multiplicity > 1) here since using a single container only is a deviation from AUTOSAR.

In case the `Can.xdm` looks like the following:

```
<v:lst name="CanConfigSet" type="MULTIPLE-CONFIGURATION-CONTAINER">
  <a:da name="MIN" value="1"/>
   <v:ctr name="CanConfigSet" type="MULTIPLE-CONFIGURATION-CONTAINER"´>
     <a:a name="DESC" value="EN: This is the multiple configuration [..] "/>
```

(note the presence of the `<v:lst>` tag here), the variant *must not* be used in the registration.

In case the `Can.xdm` looks like the following:

```
<v:ctr name="CanConfigSet" type="IDENTIFYABLE"´>
  <a:a name="DESC" value="EN: This is the multiple configuration [..] "/>
```

(note the absence of the <v:lst> tag here), the variant *must* be used in the registration.

#### 6.3.2.1.4. Special issue - Com Importer for LIN drivers

##### 6.3.2.1.4.1. Defining the behavior regarding the multiplicity of `LinGlobalConfig`

Based on the existence (or non existence) of the variant `LinGlobalConfig.ImportAsSingleContainer`, the Com Importer creates `LinGlobalConfig` as a single container (which is not correct according to AUTOSAR) or as a list of containers (which is correct according to AUTOSAR).

Based on the structure of the `Lin.xdm` schema file provided by the MCAL vendor, the variant thus has to be included/excluded in the registration of the Com Importer in order to ensure a correct import here.

**NOTE**

In the long run, the MCAL vendors must use a list of containers (i.e., an upper multiplicity > 1) here since using a single container only is a deviation from AUTOSAR.

In case the `Lin.xdm` looks like the following:

```
<v:lst name="LinGlobalConfig" type="MULTIPLE-CONFIGURATION-CONTAINER">
  <a:da name="MIN" value="1"/>
  <v:ctr name="LinGlobalConfig" type="MULTIPLE-CONFIGURATION-CONTAINER">
    <a:a name="DESC" value="EN: This container contains the global
      configuration parameter of the Lin driver. This container is
      a MultipleConfigurationContainer, i.e. this container and its
      sub-containers exist once per configuration set."/>
```

(note the presence of the <v:lst> tag here), the variant *must not* be used in the registration.

In case the `Lin.xdm` looks like the following:

```
<v:ctr name="LinGlobalConfig" type="IDENTIFIABLE">
  <a:a name="DESC" value="EN: This container contains the global
    configuration parameter of the Lin driver. This container is
    a MultipleConfigurationContainer, i.e. this container and its
    sub-containers exit once per configuration set."/>
```

(note the absence of the <v:lst> tag here), the variant *must* be used in the registration.

**Requirement (EB_MCAL_0080)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
For the modules Lin, Can, and Fr the module's plugin.xml file shall contain a proper Com Importer registration as described above.

Rationale:
Support the import of module configuration parameters from AUTOSAR system description, FIBEX files, LDF files, and DBC files.

### 6.3.2.2. ECU resource properties file registration

The ECU resource properties file(s) (see (see Section 6.5, “ECU resource properties file concept”)) must be registered in the `plugin.xml` file according to the developer documentation of EB tresos Studio [9].

**Requirement (EB_MCAL_0081)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The module should make use of ECU resource properties file by registering them in the plugin.xml file.

Rationale:
Support for sub derivative handling.

### 6.3.2.3. Handle ID calculator registration

#### 6.3.2.3.1. Overview

In order to automatically calculate the handle ID parameters (e.g., PDU IDs, CAN hardware object handle IDs etc.) EB tresos Studio provides a generic handle ID calculator. MCAL modules can use this generic handle

ID calculator to facilitate the calculation of their handle IDs by proper registration of the handle ID calculator via their `plugin.xml` files.

The registration is done via the extension point `dreisoft.tresos.guidedconfig.api.plugin.pushservice`. At this extension point a pushoperation with a unique ID and a name that is displayed in the EB tresos Studio GUI can be registered. In order to ensure a unique ID we recommend to encode the handle ID parameter name, the module name and the module variant string into the ID string (e.g., `ComIPduRxHandleId_Com_TS_T19D1M4I0R0`). When chosing the name for the pushoperation, the `/` character can be used to introduce a grouping of the different pushoperations. Here the pushoperations shall be grouped by the module name, yielding the following naming scheme: `<Module Short Name>/<Parameter Name>` (e.g., `Com/ComIPduRxHandleId`).

The generic handle ID calculator implementation provides serveral policies (called *operation classes* (`OpClass`) for the calculation of the handle ID values. Each of these operation classes is described in the following sections together with its parameters and a real-life example for the proper use of the policy.

#### 6.3.2.3.2. `SimpleHandleIdPushOperation`

`OpClass=dreisoft.tresos.guidedconfig.api.pushservice.operations.handleid.SimpleHandleIdPushOperation`

This policy can be applied if all the information for handle ID calculation is available in the module configuration in which the handle ID parameters are residing. The module ID for which the policy is applied is provided in the parameter `moduleId`. Element lists of containers containing the handle ID (*handle ID elements*) are referenced as SchemaPath via parameter `listPath`. `idPath` is the path seen relatively from one handle ID element. The policy supports the following features:

##### 6.3.2.3.2.1. Explicit grouping/sorting via `groupBy`, `sortOrder`, `ignore` parameters

`groupBy` specifies the relative path from the handle ID element to the parameter/container by the value of which the handle ID shall be grouped. Allowed are references to (choice) containers, reference and value parameters.

`sortOrder` denotes a comma-separated list by which the handle ID groups can be sorted, i.e. elements having `sortOrders` leftmost value will get the handle IDs first.

`ignore` is a comma-separated list indicating the handle ID groups that shall be ignored, i.e. elements having a value contained in the `ignore` list won't get a handle ID.

If `zbcGroups` (`zbc` stands for "zero-based and consecutive") is set to `true`, all groups will start their handle IDs with the initial value `startingId`, if `zbcGroups` is set to `false`, the first element of the second group will get the handle ID following the handle ID of the first groups last element.

#### 6.3.2.3.2.2. Implicit grouping via multiple handle ID elements in one configuration

If the handle ID elements are already grouped implicitly, i.e. there can be more than one handle ID element list, the parameter `zbcGroups` indicates whether every list shall have its own number range (`zbcGroups = true`), or there is still one global number range (`zbcGroups = false`).

#### 6.3.2.3.2.3. Definition of the first ID value

`startingId` denotes the first handle ID value of a group or, if no grouping is selected, the global starting ID (since there is only one global group then).

#### 6.3.2.3.2.4. Definition of an additional (positive) filter

`filterPath` specifies the relative path from the handle ID element to the parameter the value of which shall be used as a filter.

`filterValue` is a comma separated list specifying the values which shall "pass through" the filter, i.e. for which handle IDs shall be generated.

**NOTE**

Please note that if you must refer to a handle ID within a choice container, you also have to specify the choice container value as filter in order to avoid error messages occurring if the choice container has a value which does not contain the handle ID parameter.

#### 6.3.2.3.2.5. Complete parameter list

`moduleId`:
: The ID of the module for which to generated handle IDs.

`listPath`:
: Schema path to the list with the handle IDs.

`idPath`:
: Path relative to each list entry where the handle ID variable can be found.

`groupBy`:
: Relative path from the list entry to the parameter that is used as grouping key.

`sortOrder`:
: Parameter providing a comma separated list of values that are used as keys in the `groupBy` parameter. If a list entry starts with ~ this entry is interpreted as a regular expression.

`ignore`:
: Parameter providing a comma separated list of values that are used as group keys that should be ignored. If a list entry starts with ~ this entry is interpreted as a regular expression.

`zbcGroups`:
If not set to `false` each group starts its IDs by zero.

`startingId`:
If not set to each count starts with zero.

`filterPath`:
Schema path to an element parameter on which a filter function can be applied.

`filterValues`:
Parameter providing a comma separated list of values which are passing the filter function. If a list entry starts with ~ this entry is interpreted as a regular expression.

#### 6.3.2.3.2.6. Example: Com 3.1 Rx PDU handle IDs

```
<extension point="dreisoft.tresos.guidedconfig.api.plugin.pushservice">
  <pushoperation id="ComIPduRxHandleId_Com_TS_T19D1M4I0R0" <!-- unique ID -->
    name="Com/ComIPduRxHandleId" <!-- description seen in gui ('/' is used -->
                                 <!-- for grouping - only one level of     -->
                                 <!-- grouping supported                   -->
    desc="">
    <operationclass
      class="dreisoft.tresos.guidedconfig.api.pushservice.operations.handleid.
        SimpleHandleIdPushOperation">
      <parameter name="moduleId" value="Com_TS_T19D1M4I0R0"/>
      <parameter
        name="listPath"
        value="/AUTOSAR/TOP-LEVEL-PACKAGES/TS_T19D1M4I0R0/ELEMENTS/Com/ComConfig/
          ComConfig/ComIPdu"/>
      <parameter name="idPath" value="ComIPduRxHandleId"/>
      <parameter name="groupBy" value="ComIpduDirection"/>
      <parameter name="ignore" value="~^$,SEND"/> <!-- "~^$" means: empty value -->
      <parameter name="zbcGroups" value="true"/>
    </operationclass>
    <event>
      <and>
        <with variable="class">
          <equals value="dreisoft.tresos.guidedconfig.api.pushservice.operations.
            handleid.HandleIdEvent"/>
        </with>
        <with variable="moduleId.Com_TS_T19D1M4I0R0">
          <equals value="true"/>
        </with>
        <with variable="relVersion">
          <equals value="3.1"/> <!-- please specify ASR Release here -->
```

```
        </with>
      </and>
    </event>
  </pushoperation>
</extension>
```

#### 6.3.2.3.3. `PduLookupHandleIdPushOperation`

`OpClass=dreisoft.tresos.guidedconfig.api.-pushservice.operations.handleid.PduLookupHandleIdPushOperation`

This policy can be applied if the handle IDs shall be grouped according to a *foreign* module, which is associated by referencing common PDU elements in the global EcuC PDU container. The policy is currently only used by the PduR. It supports the same features as `SimpleHandleIdPushOperation`, but provides additional parameters:

`pduRefPath`:
Relative path from the list entry to its reference to the EcuC PDUs.

`foreignModuleIdType`:
Type if the foreign module, without target extensions, e.g. `CanIf`, `FrTp`, `Com`, ...

`foreignListPath`:
Same as `listPath` of the `SimpleHandleIdPushOperation`, only referencing the elements in the foreign module. The schema path must start with the first path element after `ELEMENTS` (i.e. no foreign `TS_` extension must be provided).

`foreignPduRefPath`:
Same as `pduRefPath` of the `SimpleHandleIdPushOperation`, only for the foreign moodule.

`foreignFilterPath`:
See `filterPath` of the `SimpleHandleIdPushOperation`, only for the foreign module.

`foreignFilterValues`:
See `filterValues` of the `SimpleHandleIdPushOperation`, only for the foreign module.

##### 6.3.2.3.3.1. Example: PduR 3.1 Handle IDs for FrIf

```
<extension point="dreisoft.tresos.guidedconfig.api.plugin.pushservice">
  <pushoperation id="FrIfRxPduId_PduR_TS_T19D1M3I0R0"
    name="PduR/SrcPdu HandleId (FrIf)"
```

```
    desc="">
    <operationclass class="dreisoft.tresos.guidedconfig.api.pushservice.operations.
      handleid.PduLookupHandleIdPushOperation">
      <parameter name="moduleId" value="PduR_TS_T19D1M3I0R0"/>
      <parameter
        name="listPath"
        value="/AUTOSAR/TOP-LEVEL-PACKAGES/TS_T19D1M3I0R0/ELEMENTS/PduR/
          PduRGlobalConfig/PduRGlobalConfig/PduRRoutingTable/PduRRoutingPath"/>
      <parameter name="idPath" value="PduRSrcPdu/HandleId"/>
      <parameter name="pduRefPath" value="PduRSrcPdu/SrcPduRef"/>
      <parameter name="foreignModuleIdType" value="FrIf"/>
      <parameter name="foreignListPath" value="/FrIf/FrIfConfig/FrIfConfig/FrIfPdu"/>
      <parameter name="foreignPduRefPath" value="FrIfPduDirection/FrIfPduRef"/>
      <parameter name="foreignFilterPath" value="FrIfPduDirection"/>
      <parameter name="foreignFilterValues" value="FrIfRxPdu"/>
      <parameter name="zbcGroups" value="false"/>
    </operationclass>
    <event>
      <and>
        <with variable="class">
          <equals value="dreisoft.tresos.guidedconfig.api.pushservice.operations.
            handleid.HandleIdEvent"/>
        </with>
        <with variable="moduleId.PduR_TS_T19D1M3I0R0">
          <equals value="true"/>
        </with>
        <with variable="relVersion">
          <equals value="3.0"/>
        </with>
      </and>
    </event>
  </pushoperation>
</extension>
```

#### 6.3.2.3.4. PduLookupHandleIdCopyPushOperation

`OpClass=dreisoft.tresos.guidedconfig.api.-pushservice.operations.handleid.PduLookupHandleIdCopyPushOperation`

This policy is used if the own module needs to copy its handle IDs from a related module (relationship is established via a common link to EcuC's PDU collection). This policy supports the following parameters:

`moduleId`, `listPath`, `idPath`, `pduRefPath`:
See `PduLookupHandleIdPushOperation` and `SimpleHandleIdPushOperation`

Furthermore, the policy provides for <n> foreign modules where handle IDs can be copied from the following parameter set:

foreignModuleIdType_<n>:
The type of the <n>-th foreign module from which the handle IDs are copied (see foreignModuleIdType parameter).

foreignListPath_<n>:
Schema path to the corresponding list entries of the <n>-th foreign module (see foreignListPath parameter).

foreignPduRefPath_<n>:
Path relative to each foreign list entry where the reference to the global PDU can be found in the <n>-th foreign module (see foreignPduRefPath parameter).

foreignHandleIdPath_<n>:
Path relative to each foreign list entry where the handle ID to copy can be found in the <n>-th foreign module (see foreignHandleIdPath parameter).

This policy does not support the feature of grouping and filtering; if there is more than one schema path in a foreign module via which elements containing handle IDs can be retrieved, a new parameter set can be introduced for each of those schema paths.

Please note that the copy policy needs handle ids from foreign modules, which themselves are calculated. In order to ensure that the source handle ID has already been calculated, the PduLookupHandleIdCopyPush-Operation policy must be called with an order parameter which is lower than 500 (default order parameter of the other handle ID calculation policies). See the following example.

#### 6.3.2.3.4.1. Example: CanIf 2.1 CanIfRxTargetPduID

```
<extension point="dreisoft.tresos.guidedconfig.api.plugin.pushservice">
  <pushoperation id="CanIfRxTargetPduID_CanIf_TS_T19D1M3I0R0"
    name="CanIf/CanIfRxTargetPduID"
    desc=""
    order="250">
    <!--
      default order value is 500, order=250 means that this op
      is executed _after_ the other ops to ensure that the source
      handle ids have already been set before they are copied here
    -->

    <operationclass class="dreisoft.tresos.guidedconfig.api.pushservice.operations.
      handleid.PduLookupHandleIdCopyPushOperation">
      <parameter name="moduleId" value="CanIf_TS_T19D1M3I0R0"/>
      <parameter
```

```
        name="listPath"
        value="/AUTOSAR/TOP-LEVEL-PACKAGES/TS_T19D1M3I0R0/ELEMENTS/CanIf/
          CanIfInitConfigSet/CanIfInitConfig/CanIfInitConfig/CanIfRxPduConfig"/>
      <parameter name="idPath" value="CanIfRxTargetPduID"/>
      <parameter name="pduRefPath" value="PduIdRef"/>

      <parameter name="foreignModuleIdType_0" value="PduR"/>
      <parameter
        name="foreignListPath_0"
        value="/PduR/PduRGlobalConfig/PduRRoutingTable/PduRRoutingPath"/>
      <parameter name="foreignPduRefPath_0" value="PduRSrcPdu/SrcPduRef"/>
      <parameter name="foreignHandleIdPath_0" value="PduRSrcPdu/HandleId"/>

      <parameter name="foreignModuleIdType_1" value="CanTp"/>
      <parameter name="foreignListPath_1" value="/CanTp/CanTpRxNSdu"/>
      <parameter name="foreignPduRefPath_1" value="CanTpRxNPdu/CanTpRxNPduRef"/>
      <parameter name="foreignHandleIdPath_1" value="CanTpRxNPdu/CanTpRxNPduId"/>

      <parameter name="foreignModuleIdType_2" value="CanTp"/>
      <parameter name="foreignListPath_2" value="/CanTp/CanTpTxNSdu"/>
      <parameter name="foreignPduRefPath_2"
        value="CanTpRxFcNPdu/CanTpRxFcNPduRef"/>
      <parameter name="foreignHandleIdPath_2"
        value="CanTpRxFcNPdu/CanTpRxFcNPduId"/>
    </operationclass>
    <event>
      <and>
        <with variable="class">
          <equals value="dreisoft.tresos.guidedconfig.api.pushservice.operations.
            handleid.HandleIdEvent"/>
        </with>
        <with variable="moduleId.CanIf_TS_T19D1M3I0R0">
          <equals value="true"/>
        </with>
        <with variable="relVersion">
   <equals value="2.1"/>
        </with>
      </and>
    </event>
  </pushoperation>
</extension>
```

**Requirement (EB_MCAL_0162)**

Coverage:
DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

The automatic handle ID calculator shall be registered in the module's plugin.xml file according to the previous description for all modules containing handle ID parameters in their configuration.

Rationale:

Reduce the user's manual configuration work.

### 6.3.2.4. Cluster definition

EB tresos Studio version 10.x and above provides a new "Module Configurations" dialog, which allows to add whole clusters of modules.

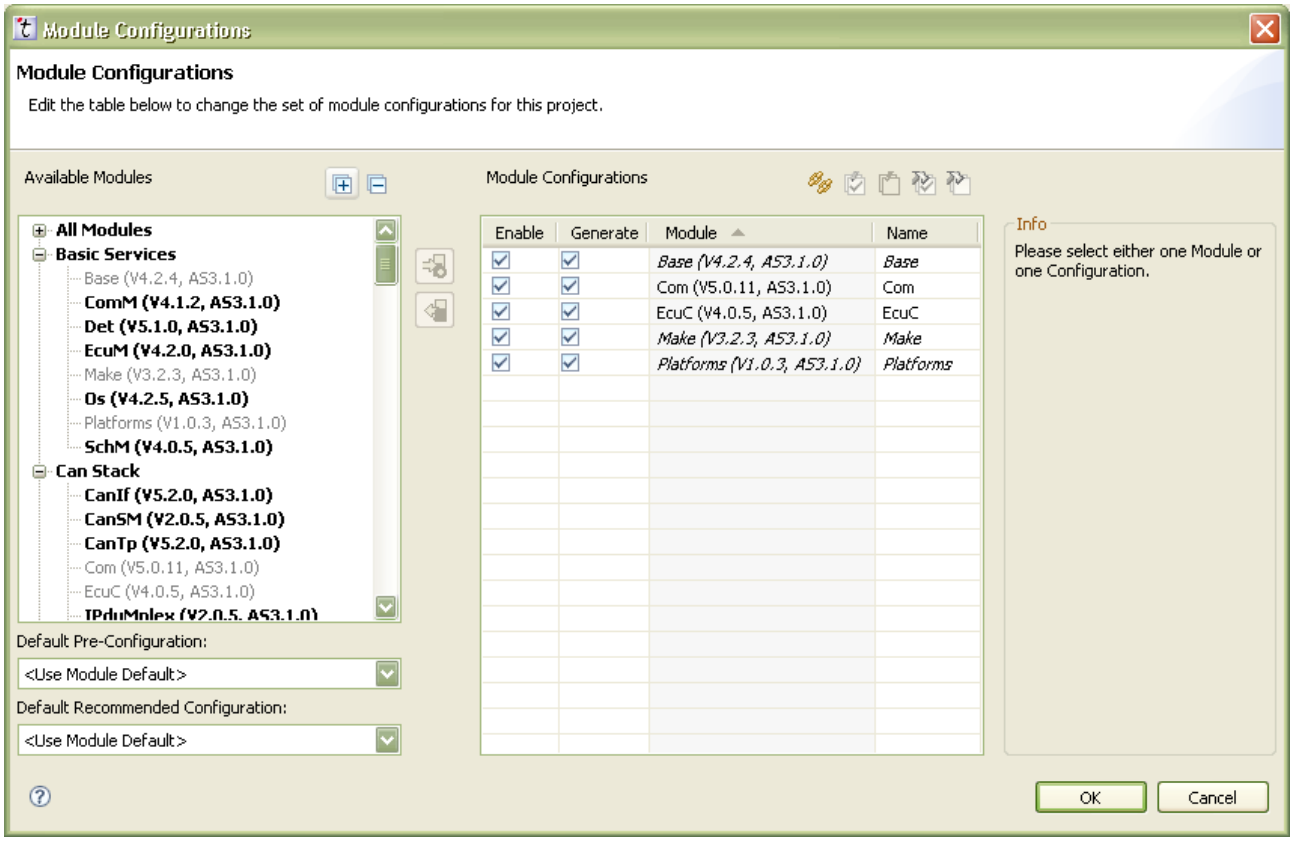


Figure 6.1. Screenshot EB tresos Studio "Module Configurations" dialog

Therefore each module plugin has to define the cluster it belongs to. The example below shows the setting for a CAN driver. A list (containing only the subcontractor modules) for the cluster assignments can be found after the example.

```
<extension point="dreisoft.tresos.launcher2.plugin.module"
  id="Can_TS_T0D1M2I0R0_ModuleId"
  name="Can_TS_T0D1M2I0R0 Module">

  <module id="Can_TS_T0D1M2I0R0"
    label="Can"
    mandatory="false"
    .....>
    <cluster name="Can Stack"/>
  </module>
</extension>
```

The cluster `Can Stack` includes the following modules:

- Can
- CanTrcv

The cluster `FlexRay Stack` includes the following modules:

- Fr
- FrTrcv

The cluster `Lin Stack` includes the following modules:

- Lin
- LinTrcv

The cluster `Ethernet Stack` includes the following modules:

- Eth
- EthTrcv

The cluster `Memory Stack` includes the following modules:

- Ea
- Fls
- Fee
- Eep

The cluster `MCAL` includes the following modules:

- Ads
- Dio

- Gpt
- Icu
- Mcu
- Port
- Pwm
- Spi

The cluster `Ttcan Stack` includes the following modules:

- Ttcan

The cluster `Watchdog Stack` includes the following modules:

- Wdg
- WdgIf

The cluster `Tests` includes the following modules:

- CoreTst
- FlsTst
- RamTst

**Requirement (EB_MCAL_0082)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The module shall contain a proper cluster definition according to the above information in its plugin.xml file.

Rationale:
Proper cluster grouping in the EB tresos Studio "Module Configurations" dialog.

### 6.3.2.5. Pre- and recommended configuration

EB tresos Studio supports that modules come along with pre- and recommended configurations. Pre- and recommended configurations are (data) XDM files or fragments in which certain parameters are already set to specific values, certain lists already contain some containers etc.

The difference between a pre- and recommended configuration is that pre-configuration parameters cannot be changed by the user anymore whereas recommended configuration parameters can be changed.

A module may come along with a single pre-configuration and multiple recommended configurations. When the user adds the module to the project, the user can select the pre/recommended configuration which shall be applied.

The usage of recommended configurations can significantly reduce the time for configuring the module since in many cases, it is sufficient to first start with reasonable default values. Later, specific optimizations might be applied.

Use cases for recommended configurations in the MCAL are:

- The Mcu may have different recommended configurations for different PLL settings. The Mcu recommended configuration may be ready to use for the evaluation board, so the user does not have to modify the Mcu configuration for an initial project setup.
- The Can module may have different recommended configurations for different baud rates/osciallator frequencies.
- The Port module may have a recommended configuration where all ports which are exisiting on a specific derivative are already configured.
- the Fls module may have a recommended configuration where all flash sectors are already configured.

The following example shows how a pre- and a recommended configuration is registered in the `plugin.xml` file.

```
<extension point="dreisoft.tresos.launcher2.plugin.configuration"
    id="Mcu_TS_T2D17M0I9R0_ConfigId"
    name="Mcu_TS_T2D17M0I9R0 Configuration">
 <configuration moduleId="Mcu_TS_T2D17M0I9R0">
  <schema>
   <manager class="dreisoft.tresos.autosar2.resourcehandling.AutosarSchemaManager"/>
   <resource value="config/Mcu.xdm" type="xdm"/>

   <!-- register the pre-configuration configuration Mcu -->
   <resource value="config_ext/McuPreConfiguration.xdm" type="xdm"
     id="res_McuPreConfiguration"/>
   <preconfig
    name="McuPreConfiguration"
    default="true"
    description="Mcu Pre-Configuration"
    resourceId="res_McuPreConfiguration"
    path="ASPath:/TS_T2D17M0I9R0/McuPreConfiguration"/>

   <!-- register the recommended configuration Mcu -->
   <resource value="config_ext/McuRecConfiguration160MHz.xdm" type="xdm"
     id="res_McuRecConfiguration160Mhz"/>
```

```
   <recconfig
    name="McuRecConfiguration160MHz"
    default="false"
    description="Recommended configuration for 160 MHz system frequency"
    resourceId="res_McuRecConfiguration160Mhz"
    path="ASPath:/TS_T2D17M0I9R0/McuRecConfiguration160MHz"/>

   <resource value="config_ext/McuRecConfiguration200MHz.xdm" type="xdm"
    id="res_McuRecConfiguration200Mhz"/>
   <recconfig
    name="McuRecConfiguration200MHz"
    default="false"
    description="Recommended configuration for 200 MHz system frequency"
    resourceId="res_McuRecConfiguration200Mhz"
    path="ASPath:/TS_T2D17M0I9R0/McuRecConfiguration200MHz"/>
  </schema>
 </configuration>
</extension>
```

The parameters of `preconfig` and `recconfig` are as follows:

`name`:
The `name` attribute provides the name of the pre-/recommended configuration that is shown to the user.

`default`:
The `default` attributes selects the pre-/recommended configuration that is presented to the user a initially selected preconfiguration when adding a new module configuration.

`description`:
The `description` attributes provides a supplementary text that is shown to the user when selecting the pre-/recommended configuration in the GUI. The description shall be limited to a single line in order to fit into the GUI.

`resourceId`:
The `resourceId` attribute selects the resource file from which to load the pre-/recommended configuration. The supplied id must match an id of a resource tag supplied for this module.

`path`:
The `path` attributes provides an XPath or ASPath to the module configuration in the referenced XDM that forms the pre-/recommended configuration.

**Requirement (EB_MCAL_0148)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
At least the modules Mcu, Can, Port, Fls, Eep, Lin and Fr shall contain at least one proper recommended configuration definition according to the above information in its plugin.xml file. Other modules may have a recommended configuration.

Rationale:
A recommended configuration eases the initial setup of the module.

**Requirement (EB_MCAL_0149)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
A module may contain a pre-configuration definition according to the above information in its plugin.xml file.

Rationale:
A pre-configuration configuration eases the initial setup of the module.

**Requirement (EB_MCAL_0150)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Pre- and recommended configuration xdm files shall be located in a sub-directory called config_ext.

Rationale:
All pre- and recommended configuration files shall be stored in a standardized directory.

**Requirement (EB_MCAL_0151)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
For each pre-configuration a .xdm file named <Module Short Name>PreConfiguration<Identifier of pre-configuration>.xdm shall exist in the config_ext directory of the module.

Rationale:
All configuration files shall be named according to a standardized scheme.

**Requirement (EB_MCAL_0152)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The name of a pre-configuration shall be <ModuleShortName>PreConfiguration<Identifier of pre-configuration>.

Rationale:
All pre-configuration names shall follow a standardized scheme.

**Requirement (EB_MCAL_0153)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
For each recommended configuration a .xdm file named <Module Short Name>RecConfiguration<Identifier of recommended configuration>.xdm shall exist in the config_ext directory of the module.

Rationale:
All configuration files shall be named according to a standardized scheme.

**Requirement (EB_MCAL_0154)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The name of a recommended configuration shall be <ModuleShortName>RecConfiguration<Identifier of recommended configuration>.

Rationale:

All recommended configuration names shall follow a standardized scheme.

### 6.3.2.6. Service needs requests

For proper operation of the Service Needs Calculator (see Section 6.8, “Service needs”), the service needs requests must be registered in the `plugin.xml` file.

The following example shows how service needs requests are registered in the `plugin.xml` file.

```
<extension point="dreisoft.tresos.guidedconfig.api.plugin.pushservice">
 <pushoperation
   id="SvcAsReq_<Module ID>"
   name="<Module Short Name> service needs requests">
   <operationclass class="eb.tresos.svclib.api.pushservice.SvcRequestOperation">
     <parameter name="module" value="<Module Short Name>" />
     <parameter name="req://schm.mainFunctions:1"
       value="resources/<Module Short Name>_SchM_MainFunctions.xml" />
     <parameter name="req://dem.events:1"
       value="resources/<Module Short Name>_Dem_Events.xml" />
     <parameter name="req://ecum.initFunctions:1"
       value="resources/<Module Short Name>_EcuM_InitFunctions.xml" />
     <parameter name="req://os.isrs:1"
       value="resources/<Module Short Name>_Os_Isrs.xml" />
   </operationclass>
   <event>
      <and>
        <with variable="class">
          <equals value="eb.tresos.svclib.api.event.SvcRequestsEvent" />
        </with>
        <with variable="svc.type">
          <equals value="request" />
        </with>
        <with variable="ecuConfig.moduleId.<Module ID>">
          <equals value="true" />
        </with>
        <or>
          <with variable="ecuConfig.relVersion">
            <equals value="4.0" />
          </with>
          <with variable="ecuConfig.relVersion">
            <equals value="0.0" />
          </with>
```

```
        </or>
      </and>
   </event>
 </pushoperation>
</extension>
```

**Requirement (EB_MCAL_0156)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Each MCAL module shall specify its service needs requests. Available service providers are Dem events, EcuM init functions, Os ISRs, and SchM main functions.

Rationale:
Service needs requests enable the usage of the Service Needs Calculator and ease the configuration of the dependencies of the module.

**Requirement (EB_MCAL_0157)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The service needs requests shall fit to the module's implementation.

Rationale:
The request description must be consistent with the module's implementation, otherwise the Service Needs Calculator cannot configure the dependencies correctly.

**Requirement (EB_MCAL_0158)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:

When the module has service needs requests defined, the module's plugin.xml file shall register the push operation eb.tresos.svclib.api.pushservice.SvcRequestOperation with the variables svc.type=request and ecuConfig.moduleType.<Module Short Name>=true.

Rationale:

The push operation is required to assure that the Service Needs Calculator can evaluate the service needs requests.

**Requirement (EB_MCAL_0159)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

When the module uses SchM main functions, the parameter req://schm.mainFunctions:1 shall be added to the SvcRequestOperation. The value of the parameter shall be the relative path and name of the XML file in which the SchM main function service needs requests are described.

Rationale:

Each service needs request type has to be registered in the SvcRequestOperation, otherwise the Service Needs Calculator cannot process the requests.

**Requirement (EB_MCAL_0161)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

When the module uses Dem events, the parameter req://dem.events:1 shall be added to the SvcRequest-Operation. The value of the parameter shall be the relative path and name of the XML file in which the Dem event service needs requests are described.

Rationale:

Each service needs request type has to be registered in the SvcRequestOperation, otherwise the Service Needs Calculator cannot process the requests.

**Requirement (EB_MCAL_0163)**

Coverage:

DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
When the module uses EcuM init functions, the parameter req://ecum.initFunctions:1 shall be added to the SvcRequestOperation. The value of the parameter shall be the relative path and name of the XML file in which the EcuM init function service needs requests are described.

Rationale:
Each service needs request type has to be registered in the SvcRequestOperation, otherwise the Service Needs Calculator cannot process the requests.

**Requirement (EB_MCAL_0164)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
When the module uses Os ISRs, the parameter req://os.isrs:1 shall be added to the SvcRequestOperation. The value of the parameter shall be the relative path and name of the XML file in which the OS ISR service needs requests are described.

Rationale:
Each service needs request type has to be registered in the SvcRequestOperation, otherwise the Service Needs Calculator cannot process the requests.

**Requirement (EB_MCAL_0165)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Any service needs request XML file shall be located in a sub-directory resources. The file name consists of <Module Short Name>_<Service Provider Module Short Name>_<Service Name>.xml.

Rationale:
The file names shall be named according to a standardized scheme in all modules.

**Requirement (EB_MCAL_0166)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Each service needs request XML file shall be valid according to the XSD schema file provided by the service provider.

Rationale:
The service needs request files must have a valid format.

## 6.3.3. MANIFEST.MF

Details about the content of the MANIFEST.MF file can be found in the developer documentation of EB tresos Studio [9].

### 6.3.3.1. Com Importer bundle dependency

For proper operation the Com importer plugin must be listed in the Require-Bundle section of the MANIFEST.MF file for Can/Lin/FlexRay driver modules:

```
Require-Bundle:
[..]
dreisoft.tresos.comimporter.api.plugin,
[..]
```

**Requirement (EB_MCAL_0083)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Can/Lin/FlexRay driver modules shall contain a dependency to the dreisoft.tresos.comimporter.api.plugin in the Require-Bundle section of their MANIFEST.MF file.

Rationale:
Ensure correct loading of the Com Importer plugin.

#### 6.3.3.2. Can Assistant bundle dependency

For proper operation of the Can Assistant, the Guide Configuration plugin must be listed in the `Require-Bundle` section of the `MANIFEST.MF` file for Can driver modules:

```
Require-Bundle:
[..]
dreisoft.tresos.guidedconfig.api.plugin,
[..]
```

**Requirement (EB_MCAL_0142)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Can driver modules shall contain a dependency to the dreisoft.tresos.guidedconfig.api.plugin in the Require-Bundle section of their MANIFEST.MF file.

Rationale:
Ensure correct loading of the Can Assistant plugin.

#### 6.3.3.3. ECU Resource Manger bundle dependency

For the proper handling of ECU resource property files (see Section 6.5, “ECU resource properties file concept”), the ECU Resource Manager must be enabled in the module's `MANIFEST.MF` file according to the developer documentation of EB tresos Studio [9].

**Requirement (EB_MCAL_0084)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Modules that make use of ECU resource properties file shall enable the ECU Resource Manager in the module's MANIFEST.MF file.

Rationale:
Support for sub derivative handling.

#### 6.3.3.4. Service needs requests bundle dependency

For proper operation of the Service Needs Calculator (see Section 6.8, “Service needs”), the SvcAs plugin must be listed in the `Require-Bundle` section of the `MANIFEST.MF` file for all modules which have service needs requests:

```
Require-Bundle:
[..]
SvcAsLib;resolution:=optional,
[..]
```

**Requirement (EB_MCAL_0155)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Modules which have service needs requests shall contain an optional dependency to the SvcAsLib in the Require-Bundle section of their MANIFEST.MF file.

Rationale:
Ensure correct working of the Service Needs Calculator plugin.

### 6.3.4. `anchors.xml`

The `anchors.xml` file is used to register user documentation. The file is located directly in the top level directory of the plugin, e.g., `<path of your EB tresos Studio installation>/plugins/Can_TS_T2D11M0I9R0/anchors.xml`.

The content of the `anchor.xml` file is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<?NLS TYPE="org.eclipse.help.toc"?>
```

```
<toc
  label="<Module Short Name> <Architecture>/<Derivative>"
  link_to="../eb.tresos.doc.autocore.mcal.releasenotes/
    anchors.xml#EB_tresos_AutoCore_MCAL_releasenotes"
  topic="$nl$/index.html">
  <topic label="<Module Short Name>" href="">
    <topic label="<document-1-name>" href="<document-1-path-filename>"/>
    <topic label="<document-N-name>" href="<document-N-path-filename>"/>
  </topic>
</toc>
```

`<Module Short Name>`:
has to be replaced by your module's short name, e.g., `Can`.

`<Architecture>`:
has to be replaced by the architecture which is supported by your module, e.g., `PA`.

`<Derivative>`:
has to be replaced by the derivative which is supported by your module, e.g., `XPC560XP`.

`<document-1-name>`:
has to be replaced by the name of your document. This text is shown in the EB tresos Studio help window. E.g., `Users Guide &quot; XPC560XP_MCAL_Integration_Manual_CAN.pdf &quot; (.doc)`

`<document-1-path-filename>`:
has to be replaced by the path/filename, where your documentation is installed, relative to the EB tresos Studio installation folder, e.g., `doc/XPC560XP_MCAL_Integration_Manual_CAN.pdf`

`<document-N-name>`:
optional - only needed if you have more than one document

`<document-N-path-filename>`:
optional - only needed if you have more than one document

Below you find an example of an `anchor.xml` together with the corresponding screenshot of the EB tresos Studio help window for:

- `<Module Short Name>`: `Can`
- `<Architecture>`: `PA`
- `<Derivative>`: `XPC560XP`
- `<document-1-name>`: `Users Guide &quot; XPC560XP_MCAL_Integration_Manual_CAN.pdf &quot; (.doc)`
- `<document-1-path-filename>`: `doc/XPC560XP_MCAL_Integration_Manual_CAN.pdf`

```
<?xml version="1.0" encoding="UTF-8"?>
<?NLS TYPE="org.eclipse.help.toc"?>

<toc
  label="Can PA/XPC560XP"
  link_to="../eb.tresos.doc.autocore.mcal.releasenotes/
    anchors.xml#EB_tresos_AutoCore_MCAL_releasenotes"
  topic="$nl$/index.html">
  <topic label="Can" href="">
    <topic
      label="Users Guide &quot; XPC560XP_MCAL_Integration_Manual_CAN.pdf &quot; (.doc)"
      href="doc/XPC560XP_MCAL_Integration_Manual_CAN.pdf"/>
    <topic
      label="Users Guide &quot; XPC560XP_MCAL_User_Manual_CAN.pdf &quot; (.doc)"
      href="doc/XPC560XP_MCAL_User_Manual_CAN.pdf"/>
  </topic>
</toc>
```

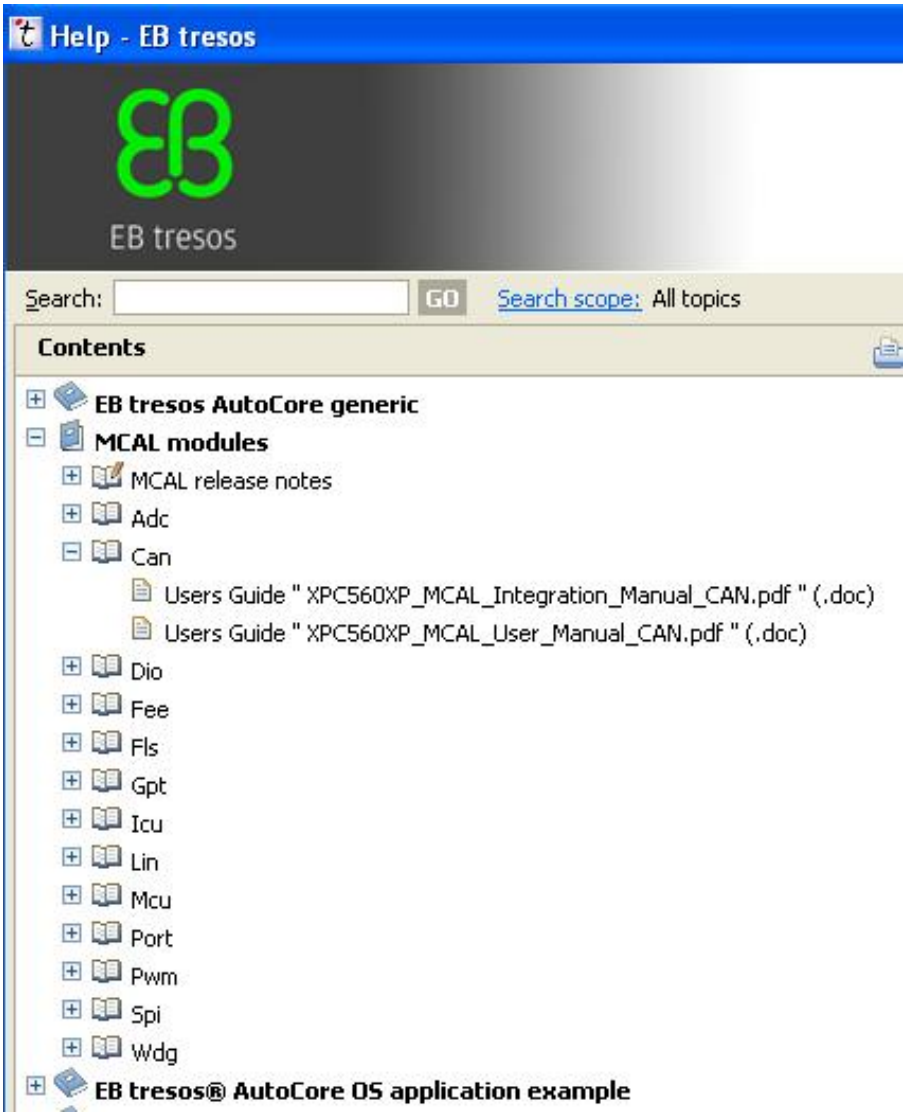


Figure 6.2. Screenshot EB tresos Studio help window

This example `anchor.xml` is also provided in `samples/anchors.xml`.

**Requirement (EB_MCAL_0038)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The module's user documentation shall be properly registered via the module's anchor.xml file.

Rationale:

Provide linkage to the module's user documentation in EB tresos Studio's help window.

## 6.3.5. MCAL include structure

An overview of the include structure of MCAL modules is depicted in Figure 6.3, “MCAL include structure”.

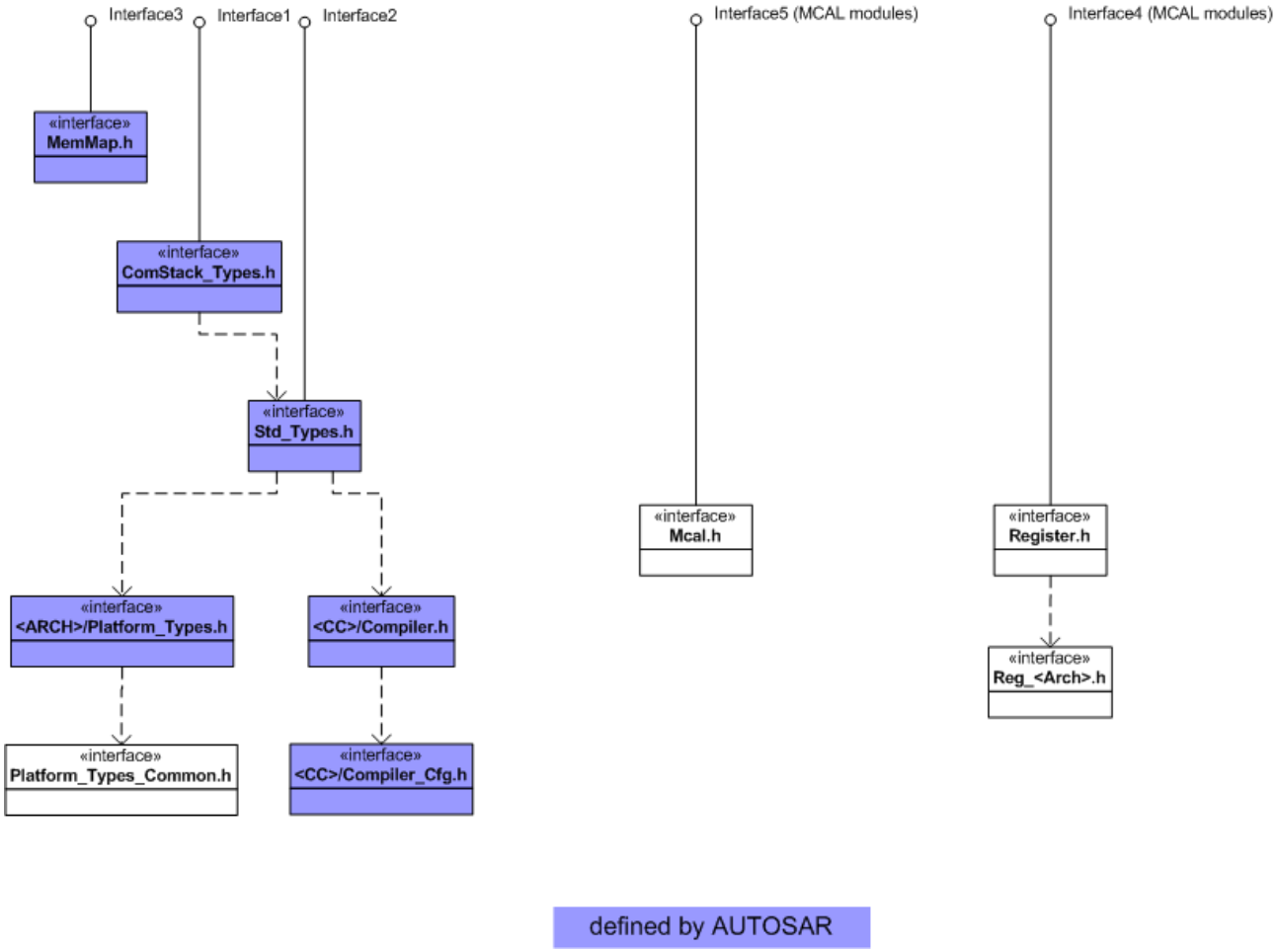


Figure 6.3. MCAL include structure

### 6.3.5.1. Module header file

**Requirement (EB_MCAL_0087)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The module's header file <Module Short Name>.h shall only export those interfaces which are absolutely required by upper layers.

Rationale:
Information hiding (MCAL internal interfaces shall not be visible at upper layers), prevention of namespace pollution, reduction of possible name clashes.

#### 6.3.5.2. `Register.h`

All register definitions of the MCAL shall be places in the file `Register.h`. - Since these definitions are MCAL internal information, they shall not be exported (directly or indirectly) via the module's external interface (i.e., `<Module Short Name>.h`.

**Requirement (EB_MCAL_0088)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The file Register.h shall contain all register definitions of the MCAL module.

**Requirement (EB_MCAL_0089)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The content of the Register.h file shall not be visible at the external interface (<Module Short Name>.h)

Rationale:
Information hiding (MCAL internal interfaces shall not be visible at upper layers), prevention of namespace pollution, reduction of possible name clashes.

#### 6.3.5.3. `Mcal.h`

The file `Mcal.h` may be used to include `os.h` or alternatively *declare* interrupt vectors. - Consider the following example code snippet:

```
#ifdef NO_AUTOSAR_OS
/* my own interrupt vector declaration */
#elif
#include os.h
#endif
```

The *definition* of own interrupt vectors shall not be done in the module's c file (e.g., `<Module Short Name>.c`). Instead the `Mcal.c` file should be used.

The file `Mcal.h` may be used to define the SchM macros for integration purposes. - Consider the following example code snippet:

```
#ifdef NO_AUTOSAR_SCHM
/* my own SchM mapping */
#elif
#include SchM_<Module Short Name>.h
#endif
```

The content of the `Mcal.h` file shall not be visible at the external interface (`<Module Short Name>.h`).

**Requirement (EB_MCAL_0090)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The file Mcal.h may be used to include os.h or alternatively declare interrupt vectors. The definition of own interrupt vectors shall not be done in the module's c file (e.g., <Module Short Name>.c). Instead the Mcal.c file should be used.

Rationale:
It is not allowed that the module's C files include os.h directly or define the interrupt vectors directly.

**Requirement (EB_MCAL_0091)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The file Mcal.h may be used to define the SchM macros for integration purposes.

**Requirement (EB_MCAL_0092)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The content of the Mcal.h file shall not be visible at the external interface (<Module Short Name>.h)

Rationale:
Information hiding (MCAL internal interfaces shall not be visible at upper layers), prevention of namespace pollution, reduction of possible name clashes.

## 6.4. Module configuration transformer

A module configuration transformer is used to convert an existing configuration based on AUTOSAR release X to a configuration based on AUTOSAR release Y.

**Requirement (EB_MCAL_0085)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
A module configuration transformer is mandatory for all modules and shall be developed in cooperation with EB.

Rationale:
Give the user the possibility to convert an existing configuration based on a previous AUTOSAR release into a configuration based on the current AUTOSAR release.

## 6.5. ECU resource properties file concept

The ECU resource properties file concept allows to handle multiple sub derivatives within the same plugins and is strongly recommended to simplify the variant handling and the maintenance of plugins. Details can be found in the developer documentation of EB tresos Studio [9].

**Requirement (EB_MCAL_0086)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The ECU resource properties file concept should be used for the handling of sub derivatives.

Rationale:
Simplification of variant handling.

## 6.6. Compiler and memory abstraction usage

The compiler and memory abstraction shall be used as defined by AUTOSAR.

The files `MemMap.h`, `Compiler.h`, and `Compiler_Cfg.h` shall not be used for other purposes as defined by AUTOSAR. - Especially the definition of intrinsics or the inclusion of other files shall not be done in those files.

It is allowed to extend the files

- `MemMap.h` by additional memory sections.
- `Compiler.h` by additional compiler key words.
- `Compiler_Cfg.h` by additional memory or pointer classes.

EB adds additional definitions to the files which are provided by EB's base module for the complete EB tresos AutoCore.

**Requirement (EB_MCAL_0093)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The compiler and memory abstraction shall be used as defined by AUTOSAR.

**Requirement (EB_MCAL_0094)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The files MemMap.h, Compiler.h and Compiler_Cfg.h shall not be used for other purposes as defined by AUTOSAR (e.g., to define intrinsics, to include files).

**Requirement (EB_MCAL_0095)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The file MemMap.h may be extended by additional memory sections.

Rationale:
Certain on-board peripherals might required special memory mapping.

**Requirement (EB_MCAL_0096)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The file Compiler.h may be extended by additional compiler keywords.

Rationale:
Certain MCAL modules might require additional compiler keywords.

**Requirement (EB_MCAL_0097)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:

The file Compiler_Cfg.h may be extended by additional memory or pointer classes.

Rationale:

Access to certain on-board peripherals might required special memory or pointer classes.

# 6.7. Schedule manager usage

As defined by AUTOSAR, all locks (critical sections) shall be realised via SchM/Rte calls. - Any direct call to interrupt disabling APIs or similar things are strictly prohibited.

Each module shall use its own critical section using its own module specific SchM/Rte macros (see [10] for details (e.g., rte_sws_7250, rte_sws_7253):

- `SchM_Enter_<mp>[_<vendorId>_<vendorApiInfix>]_<eaid>()`
- `SchM_Exit_<mp>[_<vendorId>_<vendorApiInfix>]_<eaid>()`

The actual values for `mp` (module prefix), `vendorId`, `vendorApiInfix`, and `eaid` (exclusive area id) are obtained from the module's BSWMD. Hereby `mp` is the `SHORT-NAME` of the calling module's `BSW-MODULE-DESCRIPTION` (see rte_sws_7250 of [10]). The infix `_<vendorId>_<vendorApiInfix>` is optional and shall only be used if the `VENDOR-API-INFIX` element is present in the module's `BSW-IMPLEMENTATION`. In that case `vendorId` shall be the value of the `VENDOR-ID` element and `vendorApiInfix` shall be the value of the `VENDOR-API-INFIX` and shall be the value of the (see rte_sws_7528 of [10]). The `eaid` is the `SHORT-NAME` of the respective `EXCLUSIVE-AREA` element.

The following example of critical section usage is *correct*:

Can:

```
SchM_Enter_Can_SCHM_CAN_EXCLUSIVE_AREA_0();
SchM_Exit_Can_SCHM_CAN_EXCLUSIVE_AREA_0();
```

Lin:

```
SchM_Enter_Lin_SCHM_LIN_EXCLUSIVE_AREA_0();
SchM_Exit_Lin_SCHM_LIN_EXCLUSIVE_AREA_0();
```

The following example of critical section usage is *wrong*:

Can:

```
SchM_Enter_MCAL_SCHM_MCAL_EXCLUSIVE_AREA_0();
SchM_Exit_MCAL_SCHM_MCAL_EXCLUSIVE_AREA_0();
```

Lin:

```
SchM_Enter_MCAL_SCHM_MCAL_EXCLUSIVE_AREA_0();
SchM_Exit_MCAL_SCHM_MCAL_EXCLUSIVE_AREA_0();
```

These module specific SchM/Rte macros must be accompanied by a matching SchM/Rte configuration (integration work).

If critical sections within one module could be called from different contexts, also different sections for the SchM/Rte shall be defined, since this allows for optimization during system integration.

E.g., a driver implements functions which are called from an interrupt. On system A, this interrupt is not preemtible. On this system the system integrator could map this section to 'no lock necessary'. On system B, this interrupt is preemptible (nested interrupts possible). On this system the system integrator must map this section to a lock mechanism. Another section within the module is always called from task context. This section must be mapped always to an appropriate lock mechanism by the system integrator. The described scenario could only work if different critical sections are used.

**Requirement (EB_MCAL_0098)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
All locks (i.e., critical sections) shall be realised via SchM/Rte calls. - Any direct call to interrupt disabling APIs or similar things are strictly prohibited.

Rationale:
Allow for optimizations during system integration.

**Requirement (EB_MCAL_0099)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Each module shall use its own critical section using its own module specific SchM/Rte macros.

Rationale:
Allow for optimizations during system integration on a per-module basis.

**Requirement (EB_MCAL_0100)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
If critical sections within one module could be called from different contexts, also different sections for the SchM/Rte shall be defined.

Rationale:
Allow for fine-grained optimizations during system integration.

## 6.8. Service needs

AUTOSAR modules have several dependencies among each other e.g. some modules require that specific interrupt service routines are configured in the Os or that specific Dem events are configured in the Dem module. Sometimes these dependencies are even configuration-dependent e.g. a Dem event exists only if the Dem reporting is enabled.

Configuring all these dependencies manually is complex and time-consuming. That's why the EB tresos AutoCore provides a *Service Needs Calculator*. The Service Needs Calculator collects the requests from the requester modules and performs the requested configuration changes in the provider modules automatically.

The following services are available for MCAL modules:

- SchM main functions
- EcuM init functions
- Dem events
- Os ISRs

The service requester modules have to specify their *service needs* in specific XML files which have to be registered in the `plugin.xml` file of the module (see Section 6.3.2.6, “Service needs requests”). Each type of service requests have to be specified in a separate XML file.

Each service provides a XSD schema file with a separate namespace which shall be used to edit and validate the XML request files. Each XML request file consists of a various number of elements representing one requested service entry each. There are some basic attributes resp. elements that apply to all request entries.

It is recommended that you use an XML editor which can validate an XML file against the corresponding XSD schema file. The XSD schema files are located in the `schema` directory which is delivered together with this document.

It's mandatory that the XML file validates against the respective schema.

The following subsections give an overview of the service needs request XML files.

The XSD schema also contains a more detailed description about each XML element/attribute. Please refer to the XSD schema files for a complete documentation.

## 6.8.1. General

This section describes XML attributes and elements which shall/may be used in all service needs requests.

| Attribute | Optional | Description |
|---|---|---|
| `name` | no | The mandatory attribute representing the name of the requested entry in plain text (no XPath expressions available). It must be unique across the entries of the file and comply to the AUTOSAR format for short names since it specifies the name of the configuration container of the requested service entry. It is recommended to use the module short name as prefix for the name. |
| `base` | yes | Optional attribute defining a configuration path that is used as base for evaluating XPath expressions within the current requested element. This means that XPath parameters of the request can be specified relative to this base path. The `base` attribute is always treated as XPath expression. If the `base` attribute doesn't evaluate to a single configuration parameter but to a list of configuration nodes (e.g. when pointing to all children of a list node) then the requested entry is duplicated for each node. The `base` attribute can be used if the number of service needs requests (e.g. number of main functions) depends on the number of configuration containers (e.g. ComM channels) in the ECU configuration. |
| `suffix` | yes | Optional attribute that is only evaluated if the `base` attribute is specified and eval- |

| Attribute | Optional | Description |
|---|---|---|
| | | uates to multiple nodes. The `suffix` attribute is always treated as XPath expression. The value of the `suffix` attribute is appended to the value of the `name` attribute to assure the uniqueness of a request entry if the `base` attribute evaluates to multiple nodes and the whole request entry is duplicated for each node. |
| `enable` | yes | Optional boolean parameter that specifies whether the request entry has to be handled by the service needs calculator or shall be ignored. Since the `enable` parameter can be either specified as plain text (`true` or `false`) or as XPath expression it is an XML element with the optional attribute `type` (`PLAIN`, which is the default value or `XPATH`) and the attribute `value`. The `enable` parameter can be used if the existence of the service needs request depends on settings in the ECU configuration. |
| updateReference | yes | Optional parameter that specifies a path of a reference node (normally within the configuration of the requesting module) whose value will be updated to the requested entry by the service needs provider. The value of this parameter is always treated as path to a configuration node. |

The following examples shows how the `base` and `suffix` attributes can be used to specify a configuration-dependent number of main functions. The `base` attribute contains an XPath expression. The XPath expression evaluates to multiple nodes. For each node, a separate main function is allocated. The name of the main function is consists of the name of the request specified in the `name` attribute and is concatenated with the string to which the XPath expression in the `base` attribute evaluates.

Assuming there are two entries in the `CanMainFunctionReadPeriod` list, then the Service Needs Calculator will add two main functions to the SchM configuration, `Can_MainFunction_Read_1` and `Can_MainFunction_Read_2`.

```
<?xml version="1.0" encoding="UTF-8"?>
<schm:mainFunctions
 xmlns:schm="http://www.tresos.de/_project/AutoCore/schm.mainFunctions_1.xsd"
 xmlns:svc="http://www.tresos.de/_project/AutoCore/svc.base_1.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.tresos.de/_project/AutoCore/schm.mainFunctions_1.xsd">
 <mainFunction name="Can_MainFunction_Read"
  base="as:modconf( &apos;Can&apos; )[1]/CanGeneral/CanMainFunctionReadPeriod/*"
  suffix="concat( &apos;_&apos;, num:i( ./@index + 1 ) )">
  <header value="Can.h" />
  <cycleTime type="XPATH" value="." />
  <taskPrefix value="SchMComTask" />
    <taskPriority value="150" />
 </mainFunction>
</schm:mainFunctions>
```

The next example shows a request for one Dem event with the name `CANTP_E_OPER_NOT_SUPPORTED`. The requester module has several references to this Dem event which have to be set by the Service Needs Calculator. The `base` attribute evaluates to multiple container nodes. The Dem event is requested for each container node. The `suffix` is not used i.e. the Dem event has the same name for each container node. The `updateReference` specifies one reference node within the container node which shall refer to the requested Dem event. Since the `enable` element is used, the request is only processed when the `value` attribute of the `enable` element evaluates to true i.e. in this case only when Dem event reporting is enabled for this event.

```
<?xml version="1.0" encoding="UTF-8"?>
<dem:events xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.tresos.de/_project/AutoCore/dem.events_1.xsd"
  xmlns:dem="http://www.tresos.de/_project/AutoCore/dem.events_1.xsd">
  <event name="CANTP_E_OPER_NOT_SUPPORTED"
   base="as:modconf('CanTp')[1]/CanTpConfig/*[1]/CanTpChannel/*">
   <enable type="XPATH"
       value="as:modconf('CanTp')[1]/ReportToDem/CanTpOperNotSupportedReportToDem='DEM'" />
   <updateReference
       value="./CanTpChannelDemEventParameterRefs/CANTP_E_OPER_NOT_SUPPORTED" />
   <diagEventDebounceAlgorithm value="DiagEventDebounceCounterBased" />
   <counterDecrementStepSize value="1" />
   <counterIncrementStepSize value="1" />
   <counterJumpDown value="false" />
   <counterJumpUp value="false" />
   <counterPassedThreshold value="1" />
  </event>
 </dem:events>
```

## 6.8.2. SchM main functions

The schema for SchM main function requests is specified in the file `schm.mainFunctions_1.xsd`.

The following XML code describes a request for a SchM main function of the IpduM.

```
<?xml version="1.0" encoding="UTF-8"?>
<schm:mainFunctions
 xmlns:schm="http://www.tresos.de/_project/AutoCore/schm.mainFunctions_1.xsd"
 xmlns:svc="http://www.tresos.de/_project/AutoCore/svc.base_1.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation=
    "http://www.tresos.de/_project/AutoCore/schm.mainFunctions_1.xsd">

 <mainFunction name="IpduM_MainFunction">
  <header value="IpduM.h" />
  <cycleTime type="XPATH"
   value="as:modconf( &apos;IpduM&apos; )[1]/IpduMGeneral/IpduMConfigurationTimeBase" />
  <taskPrefix value="SchMComTask" />
  <taskPriority value="150" />
  <callBefore>
      <mainFunction>Com_MainFunctionTx</mainFunction>
      <mainFunction>Can_MainFunction_Write</mainFunction>
      <mainFunction>FrIf_MainFunction</mainFunction>
      <mainFunction>LinIf_MainFunction</mainFunction>
      <mainFunction>Xcp_MainFunction</mainFunction>
      <mainFunction>Spi_MainFunction_Handling</mainFunction>
  </callBefore>
  <callAfter>
      <mainFunction>Can_MainFunction_BusOff</mainFunction>
      <mainFunction>Can_MainFunction_Wakeup</mainFunction>
      <mainFunction>Can_MainFunction_Mode</mainFunction>
      <mainFunction>CanTrcv_MainFunction</mainFunction>
      <mainFunction>Can_MainFunction_Read</mainFunction>
      <mainFunction>Com_MainFunctionRx</mainFunction>
      <mainFunction>Com_MainFunctionRouteSignals</mainFunction>
      <mainFunction>Dcm_MainFunction</mainFunction>
      <mainFunction>CanTp_MainFunction</mainFunction>
      <mainFunction>FrTp_MainFunction</mainFunction>
  </callAfter>
 </mainFunction>
</schm:mainFunctions>
```

The `header` element specifies the name of the header file which contains the function prototype of the main function.

The SchM main function service automatically requests an Os task entry per requested main function and maps the main function to this task. The cycle time for the main function is given by the `cycleTime` element. When the module has a configuration parameter for the cycle time, the cycle time shall be read from this configuration parameter so that it is under user's control.

The value of the `taskPrefix` element specifies the prefix for the task name on which the main function is mapped. The Service Needs Calculator automatically adds the task period as suffix to the task name e.g. `SchMComTask_5ms`. The task priority is given by the element `taskPriority`.

With the optional `callBefore` and `callAfter` elements it is possible to describe the order in which the main functions are called within the task. In this example, the requested main function shall be called before the functions listed in the `callBefore` list and shall be called afer the functions listed in the `callAfter` list. Each value is treated as prefix; any main function whose name starts with a value defined at the `callBefore` or `callAfter` list is taken into account for ordering.

## 6.8.3. EcuM init functions

The schema for EcuM init function requests is specified in the file `schm.initFunctions_1.xsd`.

The following XML code describes a request for an EcuM init function of the Adc module.

```
<?xml version="1.0" encoding="UTF-8"?>
<ecum:initFunctions
 xmlns:ecum="http://www.tresos.de/_project/AutoCore/ecum.initFunctions_1.xsd"
 xmlns:svc="http://www.tresos.de/_project/AutoCore/svc.base_1.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.tresos.de/_project/AutoCore/ecum.initFunctions_1.xsd">
 <initFunction name="Adc_Init">
  <moduleId value="Adc" />
  <service value="Init" />
  <initList value="1" />
  <moduleConfigurationRef type="XPATH"
   value="as:modconf( &apos;Adc&apos; )[1]/AdcConfigSet/*[1]" />
  <initializeAfter>
   <moduleId>Mcu</moduleId>
   <moduleId>Port</moduleId>
  </initializeAfter>
```

```
 </initFunction>
</ecum:initFunctions>
```

The `moduleId` element specifies the Id of the module, the `service` element specifies the suffix for the name of the init function. The `initList` element specifies to which init list the function is added.

When the init function has a parameter, the value of the `moduleConfigurationRef` element contains an XPath expression which evaluates to one multiple configuration container. The EcuM uses the name of of this multiple configuration container as name for the init config structure which is passed by reference to the init function.

With the `initializeAfter` element, it is possible to specify modules which must be initialized before this module.

## 6.8.4. Dem events

The schema for Dem event requests is specified in the file `dem.events_1.xsd`.

The following XML code describes a request for a Dem event `CAN_E_TIMEOUT`.

```
<?xml version="1.0" encoding="UTF-8"?>
<dem:events
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.tresos.de/_project/AutoCore/dem.events_1.xsd"
 xmlns:dem="http://www.tresos.de/_project/AutoCore/dem.events_1.xsd">
 <event name="CAN_E_TIMEOUT">
  <diagEventDebounceAlgorithm value="DiagEventDebounceMonitorInternal" />
 </event>
</dem:events>
```

The value of the `diagEventDebounceAlgorithm` element is set to `DiagEventDebounceMonitorInternal` i.e. the module does not use debouncing to report the event.

## 6.8.5. Os ISRs

The schema for Os ISR requests is specified in the file `os.isrs_1.xsd`.

The following XML code describes a request for the Os ISR `Can_Rx_Interrupt` of the Can module.

```
 <?xml version="1.0" encoding="UTF-8"?>
<os:isrs xmlns:os="http://www.tresos.de/_project/AutoCore/os.isrs_1.xsd"
 xmlns:svc="http://www.tresos.de/_project/AutoCore/svc.base_1.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.tresos.de/_project/AutoCore/os.isrs_1.xsd">
 <isr name="Can_Rx_Interrupt">
  <category value="CATEGORY_2" />
  <stackSize value="256" />
  <irqLevelParameterName value="OsWindowsIrqLevel" />
  <irqLevel value="1" />
  <vectorParameterName value="OsWindowsVector" />
  <vector value="INTUSER01" />
 </isr>
</os:isrs>
```

The `category` element specifies the category of the ISR, the `stackSize` the default stack size for the ISR.

For ISRs, also the level/priority and the name of the interrupt vector are part of the request. Since these parameters of the Os configuration are not specified by AUTOSAR, the names of these parameters in the specific Os implementation have to be part of the request.

In this example, the level parameter name is `OsWindowsIrqLevel` as specified by the `irqLevelParameterName` element. The name of the vector parameter is `OsWindowsVector` as specified by the `vectorParameterName` element.

The actual values for the level is given by the `irqLevel` element, the value for the vector is given by the `vector` element.

## 6.8.6. Requirements

**Requirement (EB_MCAL_0167)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The main functions of the memory stack modules shall be mapped by default to a task with the prefix SchMMemTask.

Rationale:
All memory stack main functions shall be processed in the same task.

**Requirement (EB_MCAL_0168)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The main functions of the communication stack modules shall be mapped by default to a task with the prefix SchMComTask.

Rationale:
All communication stack main functions shall be processed in the same task.

**Requirement (EB_MCAL_0169)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The main function period in the service needs request shall be derived from a main function period parameter if such a parameter exists in the module's configuration.

Rationale:
The main function period shall be under user's control.

**Requirement (EB_MCAL_0170)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The default period for the main functions of the memory stack modules shall be 10 ms.

Rationale:
All memory stack main functions shall be processed in a task with the same period.

**Requirement (EB_MCAL_0171)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The default period for the main functions of the communication stack modules shall be 5 ms.

Rationale:
All communication stack main functions shall be processed in a task with the same period.

**Requirement (EB_MCAL_0172)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Only the following MCAL modules shall specify an EcuM init function in init list one: Mcu, Port, Dio, Gpt, Wdg, Adc, Icu, Pwm.

Rationale:
With the EcuMFlex, not all modules are initialized by the EcuM. The other MCAL modules are initialized by the BswM.

## 6.9. Qualification

The following section lists miscellaneous requirements on the build environment and toolchain that need to be considered during integration/qualification of the MCAL.

**Requirement (EB_MCAL_0215)**

Coverage:
TEST

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:

When integrating or qualifying the MCAL together with the EB Base module the compiler option that allows detection of " unsued local typedefs " is not supported. This compiler option is usually known as -Wunused-local-typedefs.

Rationale:

The EB Base module contains an implementation for static compile time assertions. This implementation makes use of unique typedefs that are defined but not used.

# 7. Module specific requirements

This section contains module specific requirements on the MCAL modules. - E.g., Section 7.1, “CanIf” contains requirements imposed by the CanIf and the CanTp module on the MCAL modules (especially on Can and on CanTrcv).

## 7.1. CanIf

### 7.1.1. Configuration

#### 7.1.1.1. CAN driver HOH numbering

The CAN driver configuration must support the HOH numbering as decided in AUTOSAR Bugzilla issue #11714:

The numbering for HTHs and HRHs together shall be unique and sequential within the driver and start from 0. Restriction: The HRHs and the HTHs shall be grouped, where HRHs come first.

Example: HRH0 - 0, HRH1 - 1, HTH0 - 2, HTH1 - 3

This must especially be supported for configurations containing several controllers what in this case means that the HRHs of all controllers are grouped before the HTHs of all controllers (there must not be any HTH of any controller with an object ID lower than that of any HRH configured for this driver).

**Requirement (EB_MCAL_0177)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The CAN driver configuration shall support the HOH numbering as decided in AUTOSAR Bugzilla issue #11714. - The numbering for HTHs and HRHs together shall be unique and sequential within the driver and start from 0. Restriction: The HRHs and the HTHs shall be grouped, where HRHs come first. This shall especially be supported for configurations containing several controllers what in this case means that the HRHs of all controllers shall grouped before the HTHs of all controllers (there shall not be any HTH of any controller with an object ID lower than that of any HRH configured for this driver).

Rationale:
Allow for an efficient implementation (i.e., without index translation table) in the Can interface module.

#### 7.1.1.2. CanTrcv channel configuration

The CAN interface assumes that the CAN transceiver driver provides a *list* of transceiver channel configurations (container `CanTrcvChannel`). Therefore it must be ensured, that the CanTrcv VSMD contains a list of `CanTrcvChannel` containers if it is delivered in EB tresos XDM file format (i.e., as `*.xdm` configuration description file), even if only a single `CanTrcvChannel` is supported.

**Requirement (EB_MCAL_0178)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The CAN transceiver driver VSMD shall contain a list of CanTrcvChannel containers if it is delivered in EB tresos XDM file format (i.e., as *.xdm configuration description file), even if only a single CanTrcvChannel is supported.

Rationale:
Replacing the list by a single element would lead to a change in the X-Path required to access the CanTrcvChannel container and thus leads to incompatibility.

### 7.1.2. Indications from Can and CanTrcv

In CANIF653 the CanIf SWS cleary distiguishes between `Controller`s of the Can module (identified via the `CanControllerId` configuration parameter (CAN316_Conf) of the Can module) and abstracted `ControllerId`s of the CanIf module (identified via the `CanIfCtrlId` configuration parameter (CANIF647_Conf) of the CanIf module).

Since the signatures of the `CanIf_ControllerBusOff()` and `CanIf_ControllerModeIndication()` explicitly name the relevant parameter `Controller` and not `ControllerId`, the Can module shall use its own `CanControllerId` configuration parameter as `Controller` parameter when calling these CanIf API functions.

**Requirement (EB_MCAL_0173)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The CAN driver shall use its own CanControllerId configuration parameter as Controller parameter when calling the CanIf_ControllerBusOff() and the CanIf_ControllerModeIndication() functions.

Rationale:
CANIF653 clearly distinguishes between Controllers of the Can modules (identified via the CanControllerId configuration parameter (CAN316_Conf) of the Can module) and abstracted ControllerIds of the CanIf module (identified via the CanIfCtrlId configuration parameter (CANIF647_Conf) of the CanIf module). - CANIF218 and CANIF699 explicitly use Controller als name for the relevant parameter in the synopsis of CanIf_ControllerBusOff() and CanIf_ControllerModeIndication() respectively.

In CANIF655 the CanIf SWS cleary distiguishes between `Transceiver`s (actually transceiver channels according to CanTrcvs terminology) of the CanTrcv module (identified via the `CanTrcvChannelId` configuration parameter (CanTrcv155_Conf) of the CanTrcv module) and abstracted `TransceiverId`s of the CanIf module (identified via the `CanIfTrcvId` configuration parameter (CANIF654_Conf) of the CanIf module).

Since the signature of the `CanIf_TrcvModeIndication()` explicitly names the relevant parameter `Transceiver` and not `TransceiverId`, the CanTrcv module shall use its own `CanTrcvChannelId` configuration parameter as `Transceiver` parameter when calling this CanIf API function.

**Requirement (EB_MCAL_0174)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The CAN transceiver driver shall use its own CanTrcvChannelId configuration parameter as Transceiver parameter when calling the CanIf_TrcvModeIndication() function.

Rationale:
CANIF655 clearly distinguishes between Transceivers of the CanTrcv modules (identified via the CanTrcvChannelId configuration parameter (CanTrcv155_Conf) of the CanTrcv module) and abstracted TransceiverIds of the CanIf module (identified via the CanIfTrcvId configuration parameter (CANIF654_Conf) of the CanIf module). - CANIF705 explicitly uses Transceiver as name for the relevant parameter in the synopsis of CanIf_TrcvModeIndication().

## 7.1.3. CAN driver mode handling

The CAN driver function `Can_SetControllerMode()` shall be able to handle mode requests to the current mode without errors meaning:

- no development error shall be triggered and
- the function returns with value `CAN_OK`.

This behaviour must be ensured with development error detection being enabled and with development error detection being disabled.

Examples:

- `Can_SetControllerMode(..., CAN_T_STOP)` in mode `STOPPED`
- `Can_SetControllerMode(..., CAN_T_WAKEUP)` in mode `STOPPED`

The only exception to this general rule is a mode request into the mode `STARTED` while residing in mode `STARTED` (Rationale: For this scenario, AUTOSAR explicitly mandates in CAN409 that this shall cause a development error to be raised and that the function returns with value `CAN_NOT_OK` (see also comment #7 of AUTOSAR Bugzilla issue #27245).).

**Requirement (EB_MCAL_0122)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The CAN driver function Can_SetControllerMode() shall be able to handle mode requests to the current mode without raising any errors. No development error shall be triggered and the function shall simply return CAN_OK in those cases. The only exception to this general rule is a mode request into the mode STARTED while residing in mode STARTED.

Rationale:
This behavior eases the state machine implementation in the CAN interface. For the scenario that while residing in mode STARTED a transition into mode STARTED is requested, AUTOSAR however explicitly mandates in CAN409 that this shall cause a development error to be raised and that the function returns with value CAN_NOT_OK.

## 7.1.4. Interrupt locks and CAN driver API calls

The following CAN driver API functions are called with interrupts disabled and must therefore be able to work in this context:

- `Can_Write()`
- `Can_CheckWakeup()`

**Requirement (EB_MCAL_0123)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The CAN driver API functions Can_Write() and Can_CheckWakeup() shall be able to operate correctly when called with interrupts disabled.

Rationale:
This behaviour is required to ensure data consistency at the interface between CAN driver and CAN interface.

## 7.1.5. Message reception

### 7.1.5.1. CAN driver data pointer

To improve performance during message reception, the CanIf receive indication function (`CanIf_RxIndication()`) propagates the data pointer provided by the CAN driver directly to the upper layer receive indication function.

Therefore the CAN driver must make sure that at least as much bytes can be accessed via the data pointer, as are named in the CAN frame's data length code (DLC) and that the data can be consistently read until the receive indication function returns to the CAN driver (please compare CAN299 and CAN300).

**Requirement (EB_MCAL_0124)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The CAN driver must ensure that at least the number of bytes indicated by the CAN frame's data length code (DLC) can be consistently accessed in the context of the CanIf_RxIndication() function.

Rationale:
For performance reasons the CAN interface module simply forwards the data pointer obtained from the CAN driver in the context of the CanIf_RxIndication() function to the CanIf's upper layer without copying the data addressed by the pointer.

### 7.1.5.2. Upper layer data pointer

The data pointer delivered to all upper layers is the same that was delivered by the CAN driver.

Therefore all upper layers may only read but not modify the content of the memory area, this data pointer is pointing to. If the data shall be modified by the module itself or by any other upper layers to which it is delivered, it must be copied to a RAM section first before modifying it.

**Requirement (EB_MCAL_0125)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
All upper layers of the CAN interface shall access the data pointer delivered by the CAN interface in a read-only fashion. Write accesses to the memory location referenced by the data pointer shall not be performed.

Rationale:
This way copying the data several times can be reduced to the point where this is really necessary to improve performance in the CAN stack.

## 7.1.6. CAN-FD support

The CAN driver shall support for CAN FD according to the AUTOSAR 4.1.3 CAN driver SWS [16] as a compatible extension. This applies to the provided API functions as well as to the configuration parameters.

The AUTOSAR 4.0.3 APIs shall be kept for the sake of backwards compatibility though.

Thus this basically means:

- The new API `Can_SetBaudrate()` shall be provided as specified in the AUTOSAR 4.1.3 CAN driver SWS [16] (see SWS_CAN_00491).
- The deprecated API `Can_ChangeBaudrate()` shall be kept for AUTOSAR 4.0.3 compatibility as specified in the AUTOSAR 4.0.3 CAN driver SWS (see CAN449).
- The deprecated API `Can_CheckBaudrate()` shall be kept for AUTOSAR 4.0.3 compatibility as specified in the AUTOSAR 4.0.3 CAN driver SWS (see CAN454).
- The API `Can_Write()` shall be extended as specified in the AUTOSAR 4.1.3 CAN driver SWS [16], i.e., this API shall detect if a CAN FD frame is to be sent by checking if the CAN FD bit is set in CAN ID. In that case a CAN FD frame shall be sent if the CAN FD baud rate is active (see SWS_CAN_00486).

- An additonal configuration container named `CanControllerFdBaudrateConfig` shall be provided as specified in the AUTOSAR 4.1.3 CAN driver SWS [16] (see ECUC_Can_00473).

**Requirement (EB_MCAL_0186)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The CAN driver shall provide the Can_SetBaudRate() API as specified in the AUTOSAR 4.1.3 CAN driver SWS (see SWS_CAN_00491).

Rationale:
Required for CAN-FD support.

**Requirement (EB_MCAL_0187)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The CAN driver shall provide an extension to the Can_Write() API as specified in the AUTOSAR 4.1.3 CAN driver SWS (see SWS_CAN_00486). - Thus API shall detect if a CAN FD frame is to be sent by checking if the CAN FD bit is set in CAN ID. If that is the case a CAN FD frame shall be sent if the CAN FD baud rate is active.

Rationale:
Required for CAN-FD support.

**Requirement (EB_MCAL_0188)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The CAN driver shall support an additional configuration container named as specified in the AUTOSAR 4.0.3 CAN driver SWS (see ECUC_Can_00473).

Rationale:
Required for the configuration of the second baud rate for CAN-FD.

**Requirement (EB_MCAL_0189)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The CAN driver shall provide the Can_ChangeBaudrate() API as specified in the AUTOSAR 4.0.3 CAN driver SWS (see CAN449).

Rationale:
Backwards compatibility with AUTOSAR 4.0.3.

**Requirement (EB_MCAL_0190)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The CAN driver shall provide the Can_CheckBaudrate() API as specified in the AUTOSAR 4.0.3 CAN driver SWS (see CAN454).

Rationale:
Backwards compatibility with AUTOSAR 4.0.3.

# 7.2. LinIf

## 7.2.1. Configuration

### 7.2.1.1. LinTrcv channel configuration

The LIN interface assumes that the LIN transceiver driver provides a *list* of transceiver channel configurations (container `LinTrcvChannel`). Therefore it must be ensured, that the LinTrcv VSMD contains a list of `LinTr-`

`cvChannel` containers if it is delivered in EB tresos XDM file format (i.e., as `*.xdm` configuration description file), even if only a single `LinTrcvChannel` is supported.

**Requirement (EB_MCAL_0127)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The LIN transceiver driver VSMD shall contain a list of LinTrcvChannel containers if it is delivered in EB tresos XDM file format (i.e., as *.xdm configuration description file), even if only a single LinTrcvChannel is supported.

Rationale:
Replacing the list by a single element would lead to a change in the X-Path required to access the LinTrcvChannel container and thus leads to incompatibility.

## 7.2.2. Interrupt locks and LIN driver API calls

The following LIN driver API functions are called with interrupts disabled and must therefore be able to work in this context:

- `Lin_Wakeup()`
- `Lin_WakeupInternal()` (in case of a LIN driver according to AUTOSAR 4.2 or 4.3)

**Requirement (EB_MCAL_0128)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The LIN driver API functions Lin_Wakeup() and Lin_WakeupInternal() (in case of a LIN driver according to AUTOSAR 4.2 or 4.3) shall be able to operate correctly when called with interrupts disabled.

Rationale:
This behaviour is required to ensure data consistency at the interface between LIN driver and LIN interface.

# 7.3. Ethlf

## 7.3.1. Eth controller and configuration

The Ethernet interface assumes that the Ethernet driver provides a *list* of controller configurations (container `EthCtrlConfig`). Therefore it must be ensured, that the Eth VSMD contains a list of `EthCtrlConfig` containers if it is delivered in EB tresos XDM file format (i.e., as `*.xdm` configuration description file), even if only a single `EthCtrlConfig` is supported.

**Requirement (EB_MCAL_0203)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The Ethernet driver VSMD shall contain a list of EthCtrlConfig containers if it is delivered in EB tresos XDM file format (i.e., as *.xdm configuration description file), even if only a single EthCtrlConfig is supported.

Rationale:
Replacing the list by a single element would lead to a change in the X-Path required to access the EthCtrlConfig container and thus leads to incompatibility.

## 7.3.2. Eth physical address configuration

For usability reasons the Ethernet driver shall use a type of `STRING` (i.e., a `ECUC-STRING-PARAM-DEF`) for its `EthCtrlPhyAddress` parameter. This parameter shall allow the specification of the unique 48-bit physical address (MAC address) of the controller in network byte order according to the following regular expression: `[0-9a-fA-F]{2}[[:-][0-9a-fA-F]{2}]{5}` (see also AUTOSAR Bugzilla issue #57307 and the AUTOSAR 4.1.3 Ethernet driver SWS [17]).

**Requirement (EB_MCAL_0204)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:

The Ethernet driver shall use a type of STRING (i.e., a ECUC-STRING-PARAM-DEF) for its EthCtrlPhyAddress parameter. This parameter shall allow the specification of the unique 48-bit physical address (MAC address) of the controller in network byte order according to the following regular expression: [0-9a-fA-F]{2}[[:-][0-9a-fA-F]{2}]{5}.

Rationale:

Usability - makes the configuration for the user easier.

## 7.3.3. Eth physical address and reception API

The Ethernet driver implementation shall provide the `Eth_UpdatePhysAddrFilter()`, the `Eth_SetPhysAddr()`, and the `Eth_Receive()` API according to the AUTOSAR 4.1.3 Ethernet driver SWS [17].

Thus this basically means:

- The new API `Eth_SetPhysAddr()` shall be provided as specified in the AUTOSAR 4.1.3 Ethernet driver SWS [17] (see SWS_Eth_00151 and SWS_Eth_00139 - SWS_Eth_00143).
- The new API `Eth_UpdatePhysAddrFilter()` shall be provided as specified in the AUTOSAR 4.1.3 Ethernet driver SWS [17] (see SWS_Eth_00152, SWS_Eth_00150, SWS_Eth_00164 - SWS_Eth_00167, SWS_Eth_00144, SWS_Eth_00146, SWS_Eth_00147).
- The configuration parameter `EthUpdatePhysAddrFilter` for the new API `Eth_UpdatePhysAddrFilter()` shall be provided as specified in the AUTOSAR 4.1.3 Ethernet driver SWS [17] (see ECUC_Eth_00019).
- The API `Eth_Receive()` shall be provided as specified in the AUTOSAR 4.1.3 Ethernet driver SWS [17] (see SWS_Eth_00095 - SWS_Eth_00098, SWS_Eth_00132, SWS_Eth_00155, SWS_Eth_00153, SWS_Eth_00099).
- `Eth_GeneralTypes.h` shall be included by `Eth.h` as specified in the AUTOSAR 4.1.3 Ethernet driver SWS [17] (see SWS_Eth_00148).
- The data types `Eth_DataType`, `Eth_FilterActionType`, `Eth_FrameType`, `Eth_ModeType`, `Eth_ReturnType`, and `Eth_RxStatusType` shall be provided by `Eth_GeneralTypes.h` as specified in the AUTOSAR 4.1.3 Ethernet driver SWS [17] (see SWS_Eth_00026).
- The data type `Eth_FilterActionType` shall provide the literals `ETH_ADD_TO_FILTER` and `ETH_REMOVE_FROM_FILTER` as specified in the AUTOSAR 4.1.3 Ethernet driver SWS [17] (see SWS_Eth_00163).
- The data type `Eth_RxStatusType` shall provide the literals `ETH_RECEIVED`, `ETH_NOT_RECEIVED`, `ETH_RECEIVED_MORE_DATA_AVAILABLE`, and `ETH_RECEIVED_FRAMES_LOST` as specified in the AUTOSAR 4.1.3 Ethernet driver SWS [17] (see SWS_Eth_00162).

**Requirement (EB_MCAL_0205)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The new API Eth_SetPhysAddr() shall be provided as specified in the AUTOSAR 4.1.3 Ethernet driver SWS (see SWS_Eth_00151 and SWS_Eth_00139 - SWS_Eth_00143).

Rationale:
Interoperability with AUTOSAR 4.1.x EthIf.

**Requirement (EB_MCAL_0206)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The new API Eth_UpdatePhysAddrFilter() shall be provided as specified in the AUTOSAR 4.1.3 Ethernet driver SWS (see SWS_Eth_00152, SWS_Eth_00150, SWS_Eth_00164 - SWS_Eth_00167, SWS_Eth_-00144, SWS_Eth_00146, SWS_Eth_00147).

Rationale:
Interoperability with AUTOSAR 4.1.x EthIf.

**Requirement (EB_MCAL_0207)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The configuration parameter EthUpdatePhysAddrFilter for the new API Eth_UpdatePhysAddrFilter() shall be provided as specified in the AUTOSAR 4.1.3 Ethernet driver SWS (see ECUC_Eth_00019).

Rationale:
Interoperability with AUTOSAR 4.1.x EthIf.

**Requirement (EB_MCAL_0208)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The API Eth_Receive() shall be provided as specified in the AUTOSAR 4.1.3 Ethernet driver SWS (see SWS_Eth_00095 - SWS_Eth_00098, SWS_Eth_00132, SWS_Eth_00155, SWS_Eth_00153, SWS_Eth_00099).

Rationale:
Interoperability with AUTOSAR 4.1.x EthIf.

**Requirement (EB_MCAL_0209)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
Eth_GeneralTypes.h shall be included by Eth.h as specified in the AUTOSAR 4.1.3 Ethernet driver SWS (see SWS_Eth_00148).

Rationale:
Interoperability with AUTOSAR 4.1.x EthIf.

**Requirement (EB_MCAL_0210)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The data types Eth_DataType, Eth_FilterActionType, Eth_FrameType, Eth_ModeType, Eth_ReturnType, and Eth_RxStatusType shall be provided by Eth_GeneralTypes.h as specified in the AUTOSAR 4.1.3 Ethernet driver SWS (see SWS_Eth_00026).

Rationale:
Interoperability with AUTOSAR 4.1.x EthIf.

**Requirement (EB_MCAL_0211)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The data type Eth_FilterActionType shall provide the literals ETH_ADD_TO_FILTER and ETH_REMOVE_FROM_FILTER as specified in the AUTOSAR 4.1.3 Ethernet driver SWS (see SWS_Eth_00163).

Rationale:
Interoperability with AUTOSAR 4.1.x EthIf.

**Requirement (EB_MCAL_0212)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:
The data type Eth_RxStatusType shall provide the literals ETH_RECEIVED, ETH_NOT_RECEIVED, ETH_RECEIVED_MORE_DATA_AVAILABLE, and ETH_RECEIVED_FRAMES_LOST as specified in the AUTOSAR 4.1.3 Ethernet driver SWS (see SWS_Eth_00162).

Rationale:
Interoperability with AUTOSAR 4.1.x EthIf.

## 7.3.4. Eth MII API

The Ethernet driver implementation shall not raise the development error `ETH_E_INV_MODE` in case `Eth_WriteMii()` is called while the controller mode is different from `ETH_MODE_ACTIVE`. This basically means that the AUTOSAR 4.0.3 Ethernet driver SWS requirement ETH128 shall not be implemented. - See also AUTOSAR Bugzilla issue #57299.

**Requirement (EB_MCAL_0213)**

Coverage:
DESIGN

Source:
Elektrobit Automotive GmbH

Release: 1.0.0

Description:

The Ethernet driver implementation shall not raise the development error ETH_E_INV_MODE in case Eth_WriteMii() is called while the controller mode is different from ETH_MODE_ACTIVE. This basically means that the AUTOSAR 4.0.3 Ethernet driver SWS requirement ETH128 shall not be implemented.

Rationale:

As stated in AUTOSAR Bugzilla issue #57299 ETH128 prohibits the configuration of the transceiver before the controller is fully active. Based on this AUTOSAR Bugzilla issue ETH128 has been removed with AUTOSAR 4.1.1 and is thus no longer present in the AUTOSAR 4.1.3 Ethernet driver SWS .

## 7.3.5. Eth include structure

The Ethernet driver implementation shall not assume that the file `Eth_GeneralTypes.h` includes the file `Eth_Cfg.h`. Instead the driver implementation shall include the file `Eth_Cfg.h` directly in its file `Eth_Types.h` if this is needed. - See also AUTOSAR Bugzilla issue #74677.

**Requirement (EB_MCAL_0214)**

Coverage:

DESIGN

Source:

Elektrobit Automotive GmbH

Release: 1.0.0

Description:

The Ethernet driver implementation shall not assume that the file Eth_GeneralTypes.h includes the file Eth_Cfg.h. Instead the driver implementation shall include the file Eth_Cfg.h directly in its file Eth_Types.h if this is needed.

Rationale:

As stated in AUTOSAR Bugzilla issue #74677 by including Eth_Cfg.h Eth_GeneralTypes.h will become dependent on the configuration of the Ethernet driver(s). - Since Eth_GeneralTypes.h is a general file (which is independent from any particular driver implementation) this must not be the case.