

<b>Document Title</b>	Specification of Function Inhibition Manager
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	082
<b>Document Classification</b>	Standard
<b>Document Version</b>	2.5.0
<b>Document Status</b>	Final
<b>Part of Release</b>	4.1
<b>Revision</b>	3

<b>Document Change History</b>			
<b>Date</b>	<b>Version</b>	<b>Changed by</b>	<b>Change Description</b>
31.03.2014	2.5.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Revised development error codes</li> <li>Editorial changes</li> </ul>
31.10.2013	2.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Change containers FiMFID and FiMinhibitionConfiguration</li> <li>Editorial changes</li> <li>Removed chapter(s) on change documentation</li> </ul>
26.02.2013	2.3.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Apply new requirement format and requirement IDs (leading zeros to reach 5 digits)</li> <li>Move general requirements to AUTOSAR_SWS_BSWGeneral</li> <li>Add formal description of the Standardized AUTOSAR Interface for the Fim service. Types are formalized so that the types generated by the RTE can be used for the Fim APIs.</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Version</b>	<b>Changed by</b>	<b>Change Description</b>
02.11.2011	2.2.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Renaming of FiMCyclicEventEvaluation configuration parameter into FiMEventUpdateTriggeredByDem</li> <li>• Reformulation of SWS_Fim_00070, SWS_Fim_00073</li> <li>• Inhibition masks use TestFailed bit instead of TestFailedThisOperationCycle</li> <li>• File structure schema changed</li> <li>• Initialization sequence diagram added</li> <li>• Remove development error FIM_E_EVENTID_OUT_OF_RANGE</li> </ul>
25.11.2010	2.1.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Intra module checks updated</li> <li>• Corrected multiplicity of configuration parameters FiMInhChoicedemRef and FiMInhChoiceSumRef</li> <li>• Introduction of ImplementationDataType replacing IntegerType and Boolean</li> <li>• Clarification of chapter describing interaction between Dem and FiM (7.2.2.2)</li> <li>• Relocation of SWS_Fim_00067 explaining evaluation by the FiM of Dem events</li> <li>• Addition of a new requirement describing the standardized AUTOSAR interface (SWS_Fim_00090)</li> </ul>
03.12.2009	2.0.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Legal disclaimer revised</li> </ul>
02.07.2008	1.3.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• OBD related chapter added (7.2.3)</li> <li>• Corrected error description</li> <li>• Legal disclaimer revised</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Version</b>	<b>Changed by</b>	<b>Change Description</b>
18.12.2007	1.2.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Error classification extended to report invocation with NULL pointer</li> <li>• Corrected InternalBehavior of FiM to fit to API's reentrant behavior</li> <li>• Minimum value of parameter FimMaxSummaryLinks fixed</li> <li>• Document meta information extended</li> <li>• Small layout adaptations made</li> </ul>
24.01.2007	1.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• "Advice for users" revised</li> <li>• "Revision Information" added</li> </ul>
05.12.2006	1.1.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Modification of the FiM data structure: Several summarized events can be assigned to the FimInhibition-Configuration</li> <li>• Inserted corrected sequence charts for FiM initialization phase and <code>Fim_DemTriggerOnEventStatus</code></li> <li>• Added file <code>MemMap.h</code> to header file structure</li> <li>• Added requirement for extended header file structure (Schedule Manager)</li> <li>• Added <code>SchM_FiM.h</code> to header file structure</li> <li>• Legal disclaimer revised</li> </ul>
24.04.2006	1.0.0	AUTOSAR Administration	Initial Release

## **Disclaimer**

This specification and the material contained in it, as released by AUTOSAR is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## **Advice for users**

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

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

## Table of Contents

1	Introduction and functional overview .....	7
2	Acronyms and abbreviations .....	8
3	Related documentation.....	9
3.1	Input documents.....	10
3.2	Related standards and norms .....	11
3.3	Related specification .....	12
4	Constraints and assumptions .....	13
4.1	Limitations .....	14
4.2	Applicability to car domains.....	15
5	Dependencies on other modules.....	16
5.1	Requirements.....	17
5.1.1	Use Cases .....	17
5.2	File structure .....	18
6	Requirements traceability .....	20
6.1	Document: General Requirements on Basic Software Modules.....	26
6.2	Document: Requirements on FiM.....	31
7	Functional specification .....	32
7.1	Background & Rationale.....	33
7.2	Requirements.....	34
7.2.1	FiM core variables.....	34
7.2.2	FiM core functionalities .....	37
7.2.3	OBDFunctionality.....	42
7.2.4	Auxiliary explanations and definitions .....	43
7.2.5	Version check .....	43
7.3	Error classification .....	44
7.4	Error detection.....	45
7.5	Error notification .....	46
7.6	Debugging.....	47
8	API specification.....	48
8.1	API .....	49
8.1.1	Imported types .....	49
8.1.2	Type definitions.....	49
8.1.3	Function definitions.....	50
8.1.4	Expected Interfaces .....	56
8.2	Service interfaces.....	57
8.2.1	Client-Server-Interfaces .....	57
8.2.2	Implementation Data Types .....	58
8.2.3	Ports .....	58
8.2.4	Internal Behavior.....	58
9	Sequence diagrams .....	59
9.1	Initialization sequence of FiM .....	60

9.2	FiM_DemTriggerOnEventStatus .....	61
10	Configuration specification .....	62
10.1	How to read this chapter .....	63
10.2	Containers and configuration parameters .....	64
10.2.1	Variants .....	64
10.2.2	FiM.....	67
10.2.3	FiMGeneral.....	67
10.2.4	.....	70
10.2.5	FiMConfigSet.....	70
10.2.6	FiMFID.....	70
10.2.7	FiMEventSummary .....	71
10.2.8	FiMInhibitionConfiguration .....	72
10.2.9	FiMInhEventId .....	73
10.2.10	FiMInhChoiceDemRef .....	73
10.2.11	FiMInhChoiceSumRef.....	73
10.2.12	FiMInhRefChoice.....	74
10.2.13	FiMSummaryEventId .....	74
10.3	Published Information.....	75
11	Not applicable requirements .....	76

# 1 Introduction and functional overview

The Function Inhibition Manager is responsible for providing a control mechanism for software components and the functionality therein. In this context, a functionality can be built up of the contents of one, several or parts of runnable entities with the same set of permission / inhibit conditions. By means of the FiM, inhibiting (→ deactivation of application function) these functionalities can be configured and even modified during runtime (post-built configuration).

Functionality and runnable entity are different and independent types of classifications. Runnable entities are mainly characterized by their scheduling requirements. In contrast to that, functionalities are classified by their inhibit conditions. The services of the FiM focus on functionalities in SW-Cs, however, they are not limited to them. Functionalities of the BSW can also use the FiM services.

The functionalities are assigned to an identifier (FID – function identifier) along with the inhibit conditions for that particular identifier. The functionalities poll for the permission state of their respective FIDs before execution. If an inhibit condition comes true for a particular identifier, the corresponding functionality shall not be executed anymore.

The FiM is closely related to the Dem since diagnostic events and their status information are supported as inhibit conditions. Hence, functionality which needs to be stopped in case of a failure, e.g. of a certain sensor, can be represented by a particular identifier. If the failure is detected and the event is reported to the Dem, the FiM then inhibits the FID and therefore the corresponding functionality.

In order to handle the relation of functionality and linked events, the identifier and inhibit conditions of the functionality have been introduced into the SW-C template (equivalence for BSW) and during configuration, data structures are built up to deal with the sensitiveness of the identifiers against certain events

Software components can be integrated into a new environment as a collection of events which can be configured without big effort. Furthermore, system analysis is supported when questions as, for example, “Which functionality is inhibited if a particular event is detected?” arise. The data basis of the FiM serves as documentation of the configured relations between events and the SW-C to be inhibited.

In AUTOSAR, the RTE deals with SW-C in terms of their interfaces and scheduling requirements. In contrast to that, the FiM deals with inhibit conditions and provides supporting mechanisms for controlling functionalities via respective identifiers (FID). Therefore, the FiM concept and RTE concept do not interfere with each other.

The basic targets of the FiM specification document are:

- Standardization of APIs
- Introduction of possible implementation approaches
- Provide the ability for a common approach of OEM and supplier

## 2 Acronyms and abbreviations

<b>Abbreviation / Acronym:</b>	<b>Description:</b>
Activity state	The activity state is the status of a software component being executed. The activity state results from the permission state as a precondition and physical enable condition, too. It is not calculated by the FiM and not available as a status variable. It can only be derived from local information within a software component. For further details, see chapter 7.2.1.4.
API	Application Programming Interface
BSW	Basic Software
Dem	Diagnostic Event Manager
ECU	Electronic Control Unit
FID	Function Identifier
FiM	Function Inhibition Manager
Functionality	<p>Functionality comprises User-visible and User-non-visible functional aspects of a system (AUTOSAR_Glossary.pdf).</p> <p>In addition to that - in the FiM context - a functionality can be built up of the contents of one, several or parts of runnable entities with the same set of permission / inhibit conditions. By means of the FiM, the inhibition of these functionalities can be configured and even modified by calibration. Each functionality is represented by a unique FunctionId. A functionality is characterized by a specific set of inhibit condition in contrast to runnable entities having specific scheduling conditions.</p>
HW	Hardware
ID	Identification/Identifier
ISO	International Standardization Organization
MIL	Malfunction Indication Light
Monitoring function	<ul style="list-style-type: none"> <li>• Part of the Software Component.</li> <li>• Mechanism to monitor and finally to detect a fault of a certain sensor, actuator or could be a plausibility check</li> <li>• Reports states about events from internal processing of a SW-C or from further processing of return values of other basic software modules.</li> <li>• See also AUTOSAR_SWS_DiagnosticEventManager [10]</li> </ul>
NVRAM	Non volatile Memory
OBD	On-board Diagnostics
OBDII	Emission-related On-board Diagnostics
OEM	Original Equipment Manufacturer
OS	Operating System
Permission state	The permission state contains the information whether a functionality, represented by its FID, can be executed or whether it shall not run. The state is controlled by the FiM based on reported events. For further details, see chapter 7.2.1.4.
RAM	Random Access Memory
ROM	Read-only Memory
RTE	Runtime Environment
Runnable entity	A Runnable Entity is a part of an Atomic Software-Component, which can be executed and scheduled independently from the other Runnable Entities of this Atomic Software-Component. It is described by a sequence of instructions that can be started by the RTE. Each runnable entity is associated with exactly one EntryPoint.
SW-C	Software Component
UDS	Unified Diagnostic Services
WP	Autosar Work Package
Xxx_	Placeholder for an API provider



### 3 Related documentation

### 3.1 Input documents

- [1] List of Basic Software Modules  
AUTOSAR\_TR\_BSWModuleList.pdf
- [2] Layered Software Architecture  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf
- [3] General Requirements on Basic Software Modules  
AUTOSAR\_SRS\_BSWGeneral.pdf
- [4] Requirements on Function Inhibition Manager  
AUTOSAR\_SRS\_FunctionInhibitionManager.pdf
- [5] Specification of ECU Configuration  
AUTOSAR\_TPS\_ECUConfiguration.pdf
- [6] Software Component Template  
AUTOSAR\_TPS\_SoftwareComponentTemplate.pdf
- [7] Specification of RTE Software  
AUTOSAR\_SWS\_RTE.pdf
- [8] Specification of the Virtual Functional Bus  
AUTOSAR\_EXP\_VFB.pdf
- [9] Specification of Diagnostic Communication Manager  
AUTOSAR\_SWS\_DiagnosticCommunicationManager.pdf
- [10] Specification of Diagnostic Event Manager  
AUTOSAR\_SWS\_DiagnosticEventManager.pdf
- [11] Software Component Template  
AUTOSAR\_TPS\_SoftwareComponentTemplate.pdf
- [12] General Specification of Basic Software Modules  
AUTOSAR\_SWS\_BSWGeneral.pdf

### 3.2 Related standards and norms

- [13] IEC 7498-1 The Basic Model, IEC Norm, 1994
- [14] D1.5-General Architecture; ITEA/EAST-EEA, Version 1.0; chapter 3, page 72 et seq.
- [15] D2.1-Embedded Basic Software Structure Requirements; ITEA/EAST-EEA, Version 1.0 or higher
- [16] D2.2-Description of existing solutions; ITEA/EAST-EEA, Version 1.0 or higher.

### 3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [12] (SWS BSW General), which is also valid for Function Inhibition Manager.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Function Inhibition Manager.

## 4 Constraints and assumptions

**[SWS\_Fim\_00007]** FID numbers shall be unique per FiM. (SRS\_Fim\_04701)

Since communication between software components and basic software is limited to one ECU, the FiM can only control FIDs being located on the same ECU. Note that the RTE does currently not support communication between basic software and software components located on different ECUs.

## 4.1 Limitations

Timing constraints have to be considered for the whole system. Note that the process and response times strongly depend on the implementation of the FiM module. Hence, if there are explicit needs for faster responses of the FiM than the cycle (time slice of the task) these needs have to be considered by the FiM implementation specifically by the affected application. Special measures have to be implemented by the FiM which are not explicitly specified in this AUTOSAR document, since here, the implementation is – on purpose – not prescribed.

**[SWS\_Fim\_00043]** [The FiM shall compute the permission of a FID independently of the state of other FIDs.] (SRS\_Fim\_04706)

Interdependencies between FIDs are not supported by the FiM. That means an FID does not influence another FID.

## 4.2 Applicability to car domains

The FiM is designed to fulfill the design demands for ECUs with respect to a central handling of reactions of the system upon detected malfunctions, e.g. open circuit or shortcut. Therefore, the immediate domain of applicability of the FiM is currently body, chassis and powertrain ECUs. However, there is no reason that the FiM cannot be used in implementations of ECUs for other car domains as, for example, infotainment.

One major constraint is that the FiM alone will NOT be able to handle SW-Components that are:

1. time critical – They might be too slow for local reconfigurations (fast backup reaction in case of e.g. invalid signals).
2. physically interactive – They might not be sufficiently flexible.
3. safety critical – They might not have sufficient software integrity.

## 5 Dependencies on other modules

[SWS\_Fim\_00044] 「The AUTOSAR **Function Inhibition Manager (FiM)** has interfaces and dependencies on the Diagnostic Event Manager (Dem), the Software Components (SW-C) with FID interface, the ECU State Manager, the RTE and the BSW modules supposed to be inhibited by the FiM.」(SRS\_BSW\_00384)

- The **Diagnostic Event Manager (Dem)** is in charge of handling detected malfunctions denoted as events and reported by monitoring functions. The Dem informs and updates the Function Inhibition Manager (FiM) upon changes of the event status in order to stop or release functionalities according to assigned dependencies.
- **SW-Components (SW-C) with FID interface** query for permission to execute functionality identified by an FID at the FiM. The FIDs have to be provided by the SW components.
- **ECU State manager** is responsible for the basic initialization and de-initialization of BSW-components.
- **BSW module(s)** that are supposed to be inhibited by the FiM shall use the FiM interface to ask for permission. Therefore, the affected BSW modules have to provide the corresponding configuration data (EventID – FID – Inhibition mask relation) at configuration time realized by using a template similar to the SW-component template. The interface handling for BSW modules corresponds to the interface handling for SW-components.
- **The RTE** implements scheduling mechanisms for BSW, e.g. assigns priority and memory protection to each BSW module used in an ECU.



## 5.1 Requirements

There are three sources of requirements for this specification:

- The requirements for the functionality of the FiM service are specified in [4]. In order to model the VFB view of the Service, the chapter on AUTOSAR Services of the VFB specification [8] has to be considered as an additional requirement.
- For the formal description of the SW-C attributes [11] gives the requirements.

### 5.1.1 Use Cases

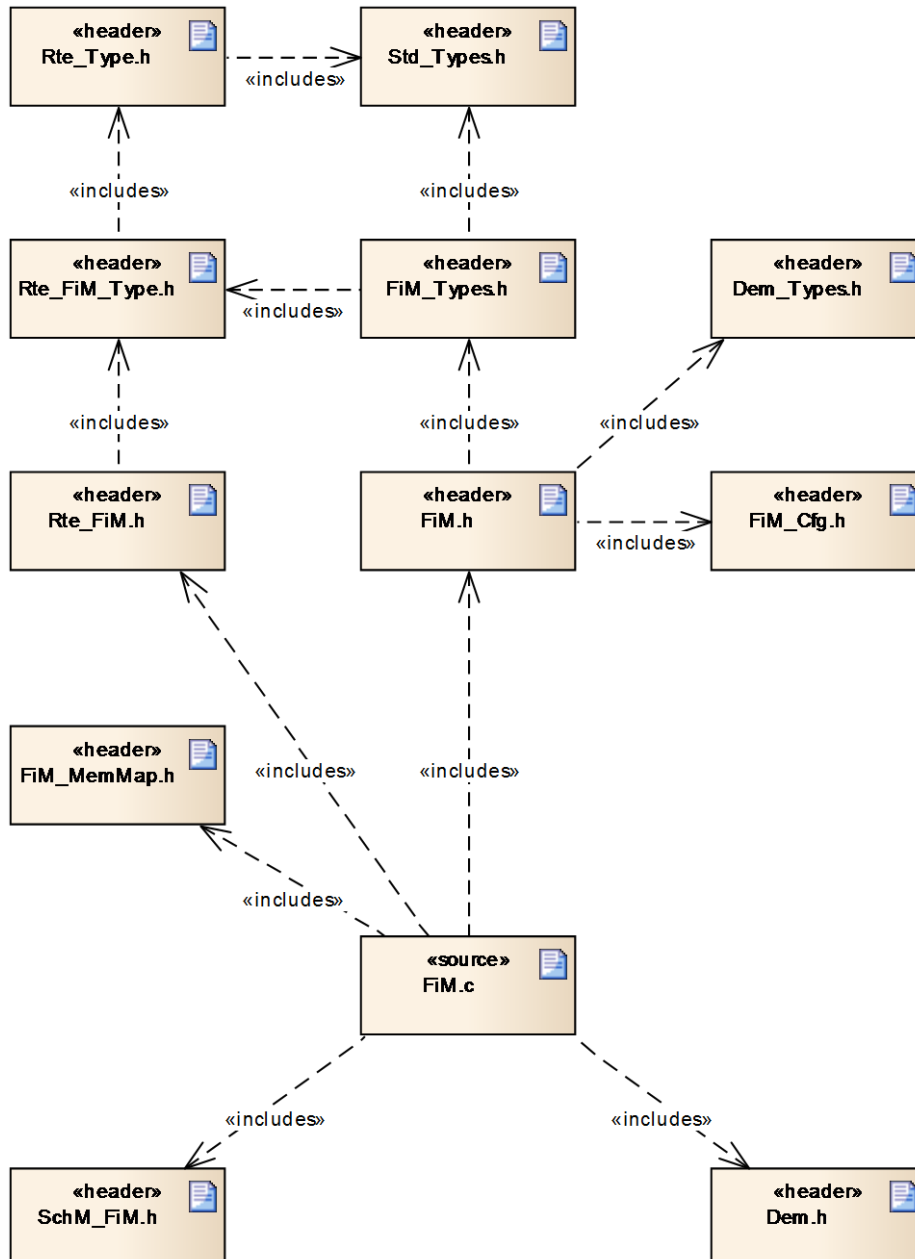
On each ECU, typically one instance of the FiM Service and several Atomic Software Component instances using this Service are employed. The Atomic Software Components are named “clients” further on in this document.

Additionally, there are parts of the basic software, which either control the FiM Manager (e.g. the ECU State Manager for initialization and shutdown) or need to query the FiM for execution permission themselves.

## 5.2 File structure

**[SWS\_Fim\_00029]** The FiM module shall adhere to the following include file structure:

(SRS\_BSW\_00346, SRS\_BSW\_00348, SRS\_BSW\_00381, SRS\_BSW\_00383, SRS\_BSW\_00412, SRS\_BSW\_00415, SRS\_BSW\_00435, SRS\_BSW\_00436, SRS\_BSW\_00447)



**[SWS\_Fim\_00031]** The FiM module shall include the `Dem.h` file.

(SRS\_BSW\_00383)

By this inclusion, EventId Symbols and the API to read the event status are included.

**[SWS\_Fim\_00096]** 「The file `Fim_Types.h` shall include `Rte_FiM_Type.h` to include the types which are common used by BSW Modules and Software Components. `Fim_Types.h` and `Fim.h` shall only contain types that are not already defined in `Rte_FiM_Type.h`.」()

## 6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_Fim_00025
-	-	SWS_Fim_00064
-	-	SWS_Fim_00065
-	-	SWS_Fim_00066
-	-	SWS_Fim_00067
-	-	SWS_Fim_00069
-	-	SWS_Fim_00070
-	-	SWS_Fim_00071
-	-	SWS_Fim_00072
-	-	SWS_Fim_00073
-	-	SWS_Fim_00076
-	-	SWS_Fim_00077
-	-	SWS_Fim_00078
-	-	SWS_Fim_00079
-	-	SWS_Fim_00080
-	-	SWS_Fim_00081
-	-	SWS_Fim_00082
-	-	SWS_Fim_00089
-	-	SWS_Fim_00091
-	-	SWS_Fim_00092
-	-	SWS_Fim_00096
BSW00005	-	SWS_Fim_00999
BSW00006	-	SWS_Fim_00999
BSW00007	-	SWS_Fim_00999
BSW00009	-	SWS_Fim_00999
BSW00010	-	SWS_Fim_00999
BSW00101	-	SWS_Fim_00004, SWS_Fim_00006
BSW00158	-	SWS_Fim_00013
BSW00159	-	SWS_Fim_00999
BSW00160	-	SWS_Fim_00999
BSW00161	-	SWS_Fim_00999
BSW00162	-	SWS_Fim_00999
BSW00164	-	SWS_Fim_00999
BSW00166	-	SWS_Fim_00004, SWS_Fim_00006, SWS_Fim_00011, SWS_Fim_00021
BSW00167	-	SWS_Fim_00999
BSW00168	-	SWS_Fim_00999

BSW00170	-	SWS_Fim_00999
BSW00172	-	SWS_Fim_00999
BSW00324	-	SWS_Fim_00999
BSW00382	-	SWS_Fim_00999
BSW00420	-	SWS_Fim_00999
BSW00421	-	SWS_Fim_00999
BSW00431	-	SWS_Fim_00999
BSW00434	-	SWS_Fim_00999
BSW0339	-	SWS_Fim_00999
BSW0341	-	SWS_Fim_00999
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_Fim_00999
SRS_BSW_00302	All AUTOSAR Basic Software Modules shall only export information needed by other modules	SWS_Fim_00999
SRS_BSW_00304	-	SWS_Fim_00027
SRS_BSW_00305	Data types naming convention	SWS_Fim_00027
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_Fim_00999
SRS_BSW_00307	Global variables naming convention	SWS_Fim_00999
SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_Fim_00999
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_Fim_00999
SRS_BSW_00310	API naming convention	SWS_Fim_00004, SWS_Fim_00006, SWS_Fim_00011, SWS_Fim_00021
SRS_BSW_00312	Shared code shall be reentrant	SWS_Fim_00011, SWS_Fim_00021
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_Fim_00999
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_Fim_00999
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_Fim_00999
SRS_BSW_00326	-	SWS_Fim_00999
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_Fim_00999
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_Fim_00999
SRS_BSW_00331	All Basic Software Modules shall strictly	SWS_Fim_00015

	separate error and status information	
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_Fim_00999
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_Fim_00999
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_Fim_00999
SRS_BSW_00342	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	SWS_Fim_00999
SRS_BSW_00343	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	SWS_Fim_00999
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_Fim_00013
SRS_BSW_00345	BSW Modules shall support pre-compile configuration	SWS_Fim_00013
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_Fim_00029
SRS_BSW_00347	A Naming separation of different instances of BSW drivers shall be in place	SWS_Fim_00999
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_Fim_00029
SRS_BSW_00353	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	SWS_Fim_00999
SRS_BSW_00355	-	SWS_Fim_00999
SRS_BSW_00357	For success/failure of an API call a standard return type shall be defined	SWS_Fim_00999
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_Fim_00004, SWS_Fim_00006, SWS_Fim_00045, SWS_Fim_00059
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_Fim_00999
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_Fim_00999
SRS_BSW_00361	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header	SWS_Fim_00999
SRS_BSW_00370	-	SWS_Fim_00999
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined	SWS_Fim_00060

	convention	
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_Fim_00999
SRS_BSW_00376	-	SWS_Fim_00060
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_Fim_00027
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_Fim_00999
SRS_BSW_00381	The pre-compile time parameters shall be placed into a separate configuration header file	SWS_Fim_00029
SRS_BSW_00383	The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description	SWS_Fim_00029, SWS_Fim_00031
SRS_BSW_00384	The Basic Software Module specifications shall specify at least in the description which other modules they require	SWS_Fim_00004, SWS_Fim_00044
SRS_BSW_00386	The BSW shall specify the configuration for detecting an error	SWS_Fim_00999
SRS_BSW_00387	The Basic Software Module specifications shall specify how the callback function is to be implemented	SWS_Fim_00999
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_Fim_00062
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_Fim_00062
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_Fim_00045, SWS_Fim_00055, SWS_Fim_00056, SWS_Fim_00057, SWS_Fim_00058, SWS_Fim_00059
SRS_BSW_00409	All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from Dem configuration	SWS_Fim_00999
SRS_BSW_00412	References to c-configuration parameters shall be placed into a separate h-file	SWS_Fim_00029
SRS_BSW_00414	The init function may have parameters	SWS_Fim_00004
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_Fim_00029
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_Fim_00004, SWS_Fim_00018
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_Fim_00999
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_Fim_00999
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the	SWS_Fim_00999

	SW-C Template	
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_Fim_00999
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_Fim_00999
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_Fim_00999
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_Fim_00999
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_Fim_00999
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_Fim_00999
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_Fim_00999
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_Fim_00999
SRS_BSW_00435	-	SWS_Fim_00029
SRS_BSW_00436	-	SWS_Fim_00029
SRS_BSW_00447	Standardizing Include file structure of BSW Modules Implementing Autosar Service	SWS_Fim_00029
SRS_Fim_04700	An Interface for querying the FID permission status shall be provided	SWS_Fim_00010, SWS_Fim_00011, SWS_Fim_00090, SWS_Fim_00094
SRS_Fim_04701	The Functionalities supervised by the FIM shall be defined by static configuration	SWS_Fim_00002, SWS_Fim_00003, SWS_Fim_00007
SRS_Fim_04702	The FIM shall support different inhibit options	SWS_Fim_00012
SRS_Fim_04706	Individual configuration of inhibit conditions of functionalities shall be available	SWS_Fim_00008, SWS_Fim_00013, SWS_Fim_00016, SWS_Fim_00043
SRS_Fim_04709	The permission state shall be evaluated before executing functionalities	SWS_Fim_00011
SRS_Fim_04712	The permission states at start up shall be initialized	SWS_Fim_00004, SWS_Fim_00018
SRS_Fim_04713	Methods for the computation of permission states shall be provided	SWS_Fim_00009, SWS_Fim_00015, SWS_Fim_00020
SRS_Fim_04717	The permission states shall be updated	SWS_Fim_00021, SWS_Fim_00022
SRS_Fim_04719	Mechanism for summarized diagnostic event states shall be provided	SWS_Fim_00061
SRS_Fim_04721	OBD Functionalities shall be supported	SWS_Fim_00999
SRS_Fim_04722	-	SWS_Fim_00011





## 6.1 Document: General Requirements on Basic Software Modules

<b>Requirement</b>	<b>Satisfied by</b>
[BSW00003] Version identification	<a href="#">SWS Fim_00023</a>
[BSW00004] Version check	<a href="#">SWS Fim_00023</a>
[BSW00006] Platform independency	Not applicable (Implementation requirement)
[BSW00007] HIS MISRA C	Not applicable (Implementation requirement)
[BSW00005] No hard coded horizontal interfaces within MCAL	Not applicable (requirement for $\mu$ C abstraction layer)
[BSW00009] Module User Documentation	Not applicable (Documentation requirement)
[BSW00010] Memory resource documentation	Not applicable (Documentation requirement)
[BSW00101] Initialization interface	<a href="#">SWS Fim_00004</a> , <a href="#">SWS Fim_00006</a>
[BSW00158] Separation of configuration from implementation	<a href="#">SWS Fim_00013</a>
[BSW00159] Tool-based configuration	Not applicable (Requirement for tool WP)
[BSW00160] Human-readable configuration data	Not applicable (Requirement for tool WP)
[BSW00161] Microcontroller abstraction	Not applicable (requirement for $\mu$ C abstraction layer)
[BSW00162] ECU layout abstraction	Not applicable (requirement for $\mu$ C abstraction layer)
[BSW00164] Implementation of interrupt service routines	Not applicable (requirement for OS, complex drivers and $\mu$ C abstraction)
[BSW00166] BSW Module interfaces	<a href="#">SWS Fim_00004</a> , <a href="#">SWS Fim_00006</a> , <a href="#">SWS Fim_00011</a> , <a href="#">SWS Fim_00021</a>
[BSW00167] Static configuration checking	Not applicable (Requirement for tool WP)
[BSW00168] Diagnostic Interface of SW components	Not applicable (requirement for DCM)
[BSW00170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable (requirement for application layer)
[BSW00171] Configurability of optional functionality	<a href="#">SWS Fim_00032</a> , <a href="#">SWS Fim_00033</a> , <a href="#">FIM040</a>
[BSW00172] Compatibility and documentation of scheduling strategy	Not applicable (Documentation requirement)
[SRS_BSW_00300] Module naming convention	Implemented
[SRS_BSW_00301] Limit imported information	Not applicable (Implementation requirement)
[SRS_BSW_00302] Limit exported information	Not applicable (Implementation requirement)
[SRS_BSW_00304] AUTOSAR integer data types	<a href="#">SWS Fim_00027</a> , <a href="#">FIM041</a>
[SRS_BSW_00305] Self-defined data types naming convention	<a href="#">SWS Fim_00027</a>
[SRS_BSW_00306] Avoid direct use of compiler and platform specific keywords	Not applicable (Implementation requirement)
[SRS_BSW_00307] Global variables naming convention	Not applicable (Implementation requirement)
[SRS_BSW_00308] Definition of global data	Not applicable

<b>Requirement</b>	<b>Satisfied by</b>
	(Implementation requirement)
[SRS_BSW_00309] Global data with read-only constraint	Not applicable (Implementation requirement)
[SRS_BSW_00310] API naming convention	<a href="#">SWS_Fim_00004</a> , <a href="#">SWS_Fim_00006</a> , <a href="#">SWS_Fim_00011</a> , <a href="#">SWS_Fim_00021</a>
[SRS_BSW_00312] Shared code shall be reentrant	<a href="#">SWS_Fim_00011</a> , <a href="#">SWS_Fim_00021</a>
[SRS_BSW_00314] Separation of interrupt frames and service routines	Not applicable (Implementation requirement)
[SRS_BSW_00318] Format of module version numbers	<a href="#">FIM041</a>
[SRS_BSW_00321] Enumeration of module version numbers	<a href="#">FIM041</a>
[SRS_BSW_00323] API parameter checking	Not applicable (Implementation requirement)
[BSW00324] Do not use HIS I/O Library	Not applicable (requirement for $\mu$ C abstraction layer)
[SRS_BSW_00325] Runtime of interrupt service routines	Not applicable (Implementation requirement)
[SRS_BSW_00326] Transition from ISRs to OS tasks	Not applicable (requirement for OS and RTE)
[SRS_BSW_00327] Error values naming convention	<a href="#">SWS_Fim_00047</a>
[SRS_BSW_00328] Avoid duplication of code	Not applicable (Implementation requirement)
[SRS_BSW_00329] Avoidance of generic interfaces	Implemented
[SRS_BSW_00330] Usage of macros / inline functions instead of functions	Not applicable (Implementation requirement)
[SRS_BSW_00331] Separation of error and status values	<a href="#">SWS_Fim_00015</a> , <a href="#">SWS_Fim_00047</a>
[SRS_BSW_00333] Documentation of callback function context	Not applicable (Documentation requirement)
[SRS_BSW_00334] Provision of XML file	Not applicable (Implementation requirement)
[SRS_BSW_00335] Status values naming convention	Implemented
[SRS_BSW_00336] Shutdown interface	Not applicable (No shutdown interface required for FiM)
[SRS_BSW_00337] Classification of errors	<a href="#">SWS_Fim_00047</a> , <a href="#">SWS_Fim_00048</a> , <a href="#">SWS_Fim_00049</a>
[SRS_BSW_00338] Detection and Reporting of development errors	<a href="#">SWS_Fim_00051</a>
[SRS_BSW_00339] Reporting of production relevant errors and exceptions	Not applicable (No production relevant errors available)
[SRS_BSW_00341] Microcontroller compatibility documentation	Not applicable (no $\mu$ C incompatibility)
[SRS_BSW_00342] Usage of source code and object code	Not applicable (Implementation requirement)
[SRS_BSW_00343] Specification and configuration of time	Not applicable (no timing dependency specified)
[SRS_BSW_00344] Post-Build configuration	<a href="#">SWS_Fim_00013</a>
[SRS_BSW_00345] Pre-Build configuration	<a href="#">SWS_Fim_00013</a>
[SRS_BSW_00346] Basic set of module files	<a href="#">SWS_Fim_00029</a>
[SRS_BSW_00347] Naming separation of drivers	Not applicable (requirement for $\mu$ C abstraction layer)

<b>Requirement</b>	<b>Satisfied by</b>
[SRS_BSW_00348] Standard type header	<a href="#">SWS_Fim_00029</a>
[SRS_BSW_00350] Development error detection keyword	<a href="#">FIM040</a>
[SRS_BSW_00353] Platform specific type header	Not applicable (hardware dependency)
[SRS_BSW_00355] Do not redefine AUTOSAR integer data types	Not applicable (Implementation requirement)
[SRS_BSW_00357] Standard API return type	Not applicable (Only void functions used)
[SRS_BSW_00358] Return type of init() functions	<a href="#">SWS_Fim_00004</a> , <a href="#">SWS_Fim_00006</a> , <a href="#">SWS_Fim_00045</a> , <a href="#">SWS_Fim_00059</a>
[SRS_BSW_00359] Return type of callback functions	Not applicable (No callback function used)
[SRS_BSW_00360] Parameters of callback functions	Not applicable (No callback function used)
[SRS_BSW_00361] Compiler specific language extension header	Not applicable (hardware dependency)
[SRS_BSW_00369] Do not return development error codes via API	<a href="#">SWS_Fim_00051</a>
[SRS_BSW_00370] Separation of callback interface from API	Not applicable (Implementation requirement)
[SRS_BSW_00371] Do not pass function pointers via API	Implemented
[SRS_BSW_00373] Main processing function naming convention	<a href="#">SWS_Fim_00060</a>
[SRS_BSW_00374] Module vendor identification	<a href="#">FIM041</a>
[SRS_BSW_00375] Notification of wake-up reason	Not applicable (no wake-up interrupt specified)
[SRS_BSW_00376] Return type and parameters of main processing functions	<a href="#">SWS_Fim_00060</a>
[SRS_BSW_00377] Module specific API return types	<a href="#">SWS_Fim_00027</a>
[SRS_BSW_00378] AUTOSAR boolean type	Not applicable (Implementation requirement)
[SRS_BSW_00379] Module identification	<a href="#">SWS_Fim_00032</a> , <a href="#">SWS_Fim_00033</a> , <a href="#">FIM041</a>
[SRS_BSW_00380] Separate C-Files for configuration parameters	<a href="#">SWS_Fim_00030</a>
[SRS_BSW_00381] Separate configuration header file for pre-compile time parameters	<a href="#">SWS_Fim_00029</a>
[BSW00382] Not-used configuration elements need to be listed	Not applicable (no configuration element available that is not used)
[SRS_BSW_00383] List dependencies of configuration files	<a href="#">SWS_Fim_00029</a> , <a href="#">SWS_Fim_00031</a>
[SRS_BSW_00384] List dependencies to other modules	<a href="#">SWS_Fim_00004</a> , <a href="#">SWS_Fim_00044</a>
[SRS_BSW_00385] List possible error notifications	<a href="#">SWS_Fim_00051</a>
[SRS_BSW_00386] Configuration for detecting an error	Not applicable (FiM cannot detect errors)
[SRS_BSW_00387] Specify the configuration class of callback function	Not applicable (No callback function used)
[SRS_BSW_00388] Introduce containers	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00389] Containers shall have names	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00390] Parameter content shall be unique within the module	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00391] Parameter shall have unique names	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00392] Parameters shall have a type	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00393] Parameters shall have a range	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>

<b>Requirement</b>	<b>Satisfied by</b>
[SRS_BSW_00394] Specify the scope of the parameters	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00395] List the required parameters (per parameter)	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00396] Configuration classes	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00397] Pre-compile-time parameters	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00398] Link-time parameters	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00399] Loadable Post-build time parameters	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00400] Selectable Post-build time parameters	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00401] Documentation of multiple instances of configuration parameters	<a href="#">FIM037</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00402] Published information	<a href="#">FIM041</a>
[SRS_BSW_00404] Reference to post build time configuration	<a href="#">SWS_Fim_00062</a>
[SRS_BSW_00405] Reference to multiple configuration sets	<a href="#">SWS_Fim_00062</a>
[SRS_BSW_00406] Check module initialization	<a href="#">SWS_Fim_00045</a> , <a href="#">SWS_Fim_00055</a> , <a href="#">SWS_Fim_00056</a> , <a href="#">SWS_Fim_00057</a> , <a href="#">SWS_Fim_00058</a> , <a href="#">SWS_Fim_00059</a>
[SRS_BSW_00407] Function to read out published parameters	<a href="#">SWS_Fim_00032</a> , <a href="#">SWS_Fim_00033</a>
[SRS_BSW_00408] Configuration parameter naming convention	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00409] Header files for production code error IDs	Not applicable (No production code errors)
[SRS_BSW_00410] Compiler switches shall have defined values	<a href="#">FIM037</a> , <a href="#">FIM038</a> , <a href="#">FIM039</a> , <a href="#">FIM040</a>
[SRS_BSW_00411] Get version info keyword	<a href="#">SWS_Fim_00032</a> , <a href="#">SWS_Fim_00033</a>
[SRS_BSW_00412] Separate H-File for configuration parameters	<a href="#">SWS_Fim_00029</a>
[SRS_BSW_00413] Accessing instances of BSW modules	<a href="#">FIM041</a>
[SRS_BSW_00414] Parameter of init function	<a href="#">SWS_Fim_00004</a>
[SRS_BSW_00415] User dependent include files	<a href="#">SWS_Fim_00029</a>
[SRS_BSW_00416] Sequence of Initialization	<a href="#">SWS_Fim_00004</a> , <a href="#">SWS_Fim_00018</a>
[SRS_BSW_00417] Reporting of Error Events by Non-Basic Software	Not applicable (FiM is BSW module)
[SRS_BSW_00419] Separate C-Files for pre-compile time configuration parameters	<a href="#">SWS_Fim_00030</a>
[BSW00420] Production relevant error event rate detection	Not applicable (requirement for Dem)
[BSW00421] Reporting of production relevant error events	Not applicable (No production code errors)
[SRS_BSW_00422] Debouncing of production relevant error status	Not applicable (requirement for Dem)
[SRS_BSW_00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (Implementation requirement)
[SRS_BSW_00424] BSW main processing function task allocation	Not applicable (Implementation requirement)
[SRS_BSW_00425] Trigger conditions for schedulable objects	Not applicable (Implementation requirement)
[SRS_BSW_00426] Exclusive areas in BSW modules	Not applicable (Implementation requirement)
[SRS_BSW_00427] ISR description for BSW modules	Not applicable (Implementation requirement)
[SRS_BSW_00428] Execution order dependencies of main processing functions	Not applicable (Implementation requirement)

<b>Requirement</b>	<b>Satisfied by</b>
[SRS_BSW_00429] Restricted BSW OS functionality access	Not applicable (Implementation requirement)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (Implementation requirement)
[SRS_BSW_00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (no requirement for FiM)
[SRS_BSW_00433] Calling of main processing functions	Not applicable (requirement for scheduler)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (requirement for scheduler)
[SRS_BSW_00435] Header File Structure for the Basic Software Scheduler	<a href="#">SWS_Fim_00029</a>
[SRS_BSW_00436] Module Header File Structure for the Basic Software	<a href="#">SWS_Fim_00029</a>
Memory Mapping	Not applicable (Implementation requirement)
[SRS_BSW_00447] Standardizing Include file structure of BSW Modules Implementing Autosar Service	<a href="#">SWS_Fim_00029</a>

## 6.2 Document: Requirements on FiM

<b>Requirement</b>	<b>Satisfied by</b>
[SRS_Fim_04700] Interface for querying the FID permission status	<a href="#">SWS_Fim_00010</a> , <a href="#">SWS_Fim_00011</a>
[SRS_Fim_04701] Functionality supervised by the FiM	<a href="#">SWS_Fim_00002</a> , <a href="#">SWS_Fim_00003</a> , <a href="#">SWS_Fim_00007</a>
[SRS_Fim_04702] Support of inhibit options	<a href="#">SWS_Fim_00012</a>
[SRS_Fim_04706] Individual configuration of inhibit conditions of functionalities	<a href="#">SWS_Fim_00008</a> , <a href="#">SWS_Fim_00013</a> , <a href="#">SWS_Fim_00016</a> , <a href="#">SWS_Fim_00043</a>
[SRS_Fim_04709] Evaluation of permission state before executing functionalities	<a href="#">SWS_Fim_00011</a>
[SRS_Fim_04712] Initialization of the permission states at start up	<a href="#">SWS_Fim_00004</a> , <a href="#">SWS_Fim_00018</a>
[SRS_Fim_04713] Methods for the computation of permission states	<a href="#">SWS_Fim_00009</a> , <a href="#">SWS_Fim_00015</a> , <a href="#">SWS_Fim_00020</a>
[SRS_Fim_04717] Updating the permission states	<a href="#">SWS_Fim_00021</a> , <a href="#">SWS_Fim_00022</a>
[SRS_Fim_04719] Mechanism for summarized diagnostic event states	<a href="#">FIM037</a> , <a href="#">SWS_Fim_00061</a>
[SRS_Fim_04721] Support OBD Functionality	Not applicable (requirement for Dem)
[SRS_Fim_04722] Return type for API accessed through RTE	<a href="#">SWS_Fim_00011</a>

## 7 Functional specification



## 7.1 Background & Rationale

The Function Inhibition Manager allows querying the permission / inhibition status of software components and the functionality therein. In the FiM context an FID (FID – function identifier) identifies an application functionality along with the inhibit conditions for that particular identifier. The functionalities poll for the permission state of their FID before execution. If an inhibit condition applies for a particular identifier, the corresponding functionality is not allowed to be executed anymore. By means of the FiM, the inhibition of these functionalities can be configured and even modified by calibration. Dem events and their status information are supported as inhibit conditions.

In order to handle the relation of functionality and associated affecting events, the identifier (FID) and inhibit conditions (events) of the functionality are included in the SW component template (equivalence for BSW). During configuration of the FiM, data structures (i.e. an inhibit matrix) are built up to deal with the sensitiveness of the identifiers against certain events.

## 7.2 Requirements

### 7.2.1 FiM core variables

#### 7.2.1.1 Definition of 'Diagnostic Event'

A 'Diagnostic Event' is an identifier provided by the Dem to a specific diagnostic monitor function to report an error. The status of a 'Diagnostic Event' represents the result of a monitoring function or the report of a Basic Software Module. See AUTOSAR\_SWS\_DiagnosticEventManager document for further details [10].

#### 7.2.1.2 Definition of 'Summarized Event'

**[SWS\_Fim\_00061]** [The FiM configuration shall support summarizing events. A summarized event consists of multiple single diagnostic events.](SRS\_Fim\_04719)

During the configuration process, these single events can be combined to a summarized event ([ECUC FIM 00037](#)). A summarized event simplifies dealing with the multiple events that are associated with or represented by the particular summarized event. For simplicity, this particular summarized event can be used as an inhibit condition in the SW-C templates.

**[SWS\_Fim\_00064]** [The FiM shall also be able to process the inhibit conditions of all FIDs associated to one summarized event if one of the Dem Events associated to this summarized event is reported to the FiM.]( )

Hence, the particular summarized event is just a representative of multiple diagnostic events (ref.10.2.4). A use case for summarized events is for example the combination of all error conditions that indicate a failed sensor:

A sensor X has multiple diagnostics, e.g. short cut ground, battery and open circuit: X\_SCG, X\_SCB and X\_OC. The functions FID\_0, FID\_1, ..., FID\_N are to be inhibited in case of this fault.

A direct configuration requires  $3 * N$  containers `FiMinhibitionConfiguration` with `FIM_INH_EVENT_ID = X_SCG/SCB/OC` and `FIM_INH_FUNCTION_ID = FID_0/.../N`.

With summarized events, a configuration can be created with less effort: A single summarized event is created by a container `FiMSummary`, `FIM_SUMMARIZED_EVENT = X_SUMMARY`. Then three `FiMEventSummary` containers are configured: `FIM_INPUT_SUMMARIZED_EVENT = X_SCG/SCB/OC` and `FIM_OUTPUT_SUMMARIZED_EVENT = X_SUMMARY`. As inhibition only  $N$  `FiMinhibitionConfiguration` with `FIM_INH_EVENT_ID = X_SUMMARY` and `FIM_INH_FUNCTION_ID = FID_0/.../N` are needed.

### 7.2.1.3 Definition of 'Function Identifier'

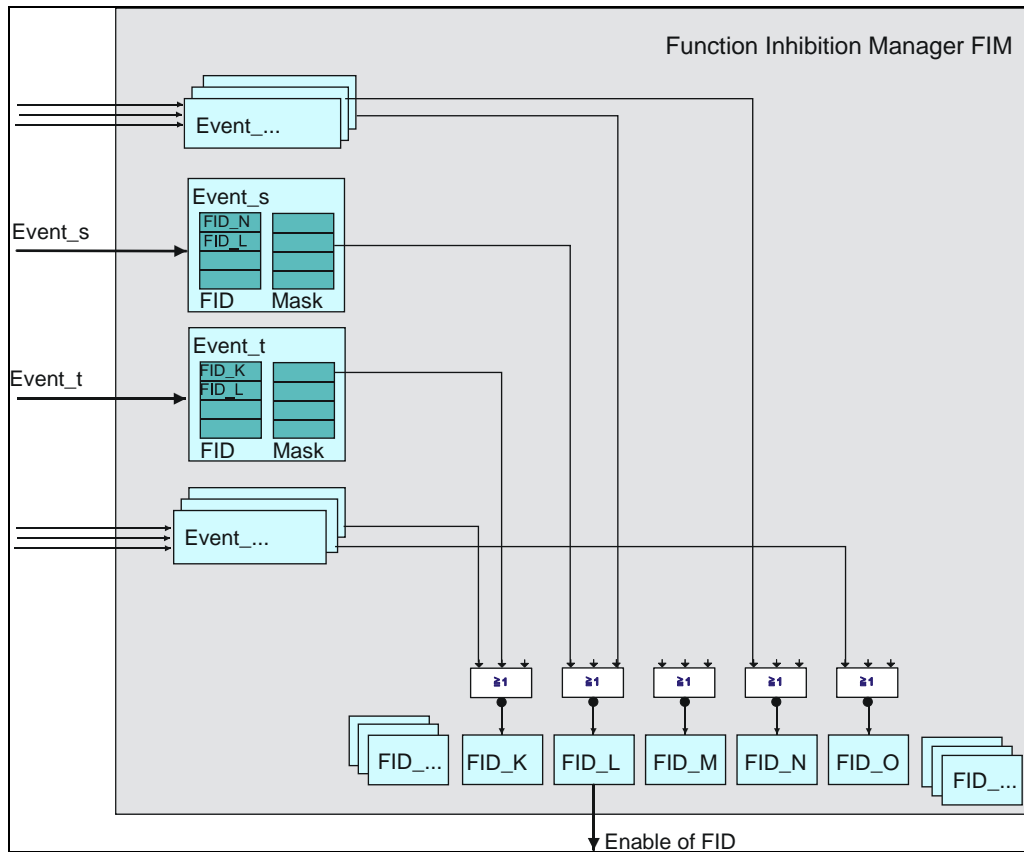
The functionalities are addressed by numbers called function identifiers (FID). A FID represents the contents of one or more parts of runnable entities or BSW functionality with the same set of permission / inhibition conditions.

**[SWS\_Fim\_00002]** 「The configuration process shall guarantee that FunctionIds are unique per FiM. Two distinct functionalities with different dependencies on events shall never have the same FunctionId (see also [SWS\\_Fim\\_00007](#)). 」  
(SRS\_Fim\_04701)

**[SWS\_Fim\_00003]** 「The FiM module's environment shall use the FunctionId to directly point to the associated functionality information (permission status etc.)」  
(SRS\_Fim\_04701)

Note: The SW-C template contains the symbolic names of all FIDs ("FID\_XXX") relevant for the respective SW-C. The subsequent numbering of all FIDs within a node is accomplished by the configuration process.

**[SWS\_Fim\_00010]** 「The flow of information starts with the API call of the Dem providing changes of the event information. This information is processed and dependencies to FIDs are evaluated. Finally, the permission state of the FIDs is accessed via API through the RTE (Figure 1).」(SRS\_Fim\_04700)



**Figure 1: Logical information flow to determine FID permission states for an implementation with permission state stored in RAM**

The permission state of each FID is calculated based on the EventIds assigned to a specific FID. Afterwards, the calculated permission states of each FID (e.g. FID\_K) are “and-ed” to determine the resulting permission state. This implies an implementation where the FiM stores the permission state of the FIDs in RAM.

Alternatively, the FiM can poll the event status to re-calculate the permission state. The polling is triggered either by a functionality requesting its permission state (SW-C or BSW) or in a cyclic task. In this case, there is no increased process effort within the FiM at changes of any event.

**7.2.1.4 Definition of ‘Function Identifier permission state’**

**[SWS\_Fim\_00015]** [The FID permission state contains the information whether a functionality represented by its FID can be executed. If the permission state == TRUE, the functionality associated with the FID is permitted to be executed. If the permission state == FALSE, the functionality associated with the FID is not allowed to be executed.](SRS\_BSW\_00331, SRS\_Fim\_04713)

The permission state is based on events reported by the Dem. Therefore, the permission state does not directly consider physical conditions (e.g. temperature, engine speed...) but those conditions reported to the Dem (e.g. sensor defect).

Additionally to the permission state as prerequisite, the activity state (is the function active or not) includes physical enable conditions representing whether the functionality is indeed executed or not, i.e. is active or not.

As stated above, one possible implementation is to provide the permission state in status variables. An alternative is to compute the permission on the query based on the underlying dependencies.

Hint: If the permission states are stored in status variables, they are unique values per FID. SW-components access the status via `FiM_GetFunctionPermission`.

**[SWS\_Fim\_00009]** ¶ If the implementation uses status variables for the permission of the FIDs, the status variables shall be readable for tracking purposes by the calibration system (to be defined by AUTOSAR) during the development phase of the ECU. ¶ (SRS\_Fim\_04713)

Note that reliable permission states can only be expected after complete FiM initialization, which follows the Dem initialization. Therefore, reactions on BSW malfunctions via FiM might be delayed.

## 7.2.2 FiM core functionalities

### 7.2.2.1 FiM Data Structure

**[SWS\_Fim\_00013]** ¶ The configuration process of the FiM shall create data structures within the FiM module to store the inhibit relations (EventID – FID – applicable mask). ¶ (BSW00158, SRS\_BSW\_00344, SRS\_BSW\_00345, SRS\_Fim\_04706)

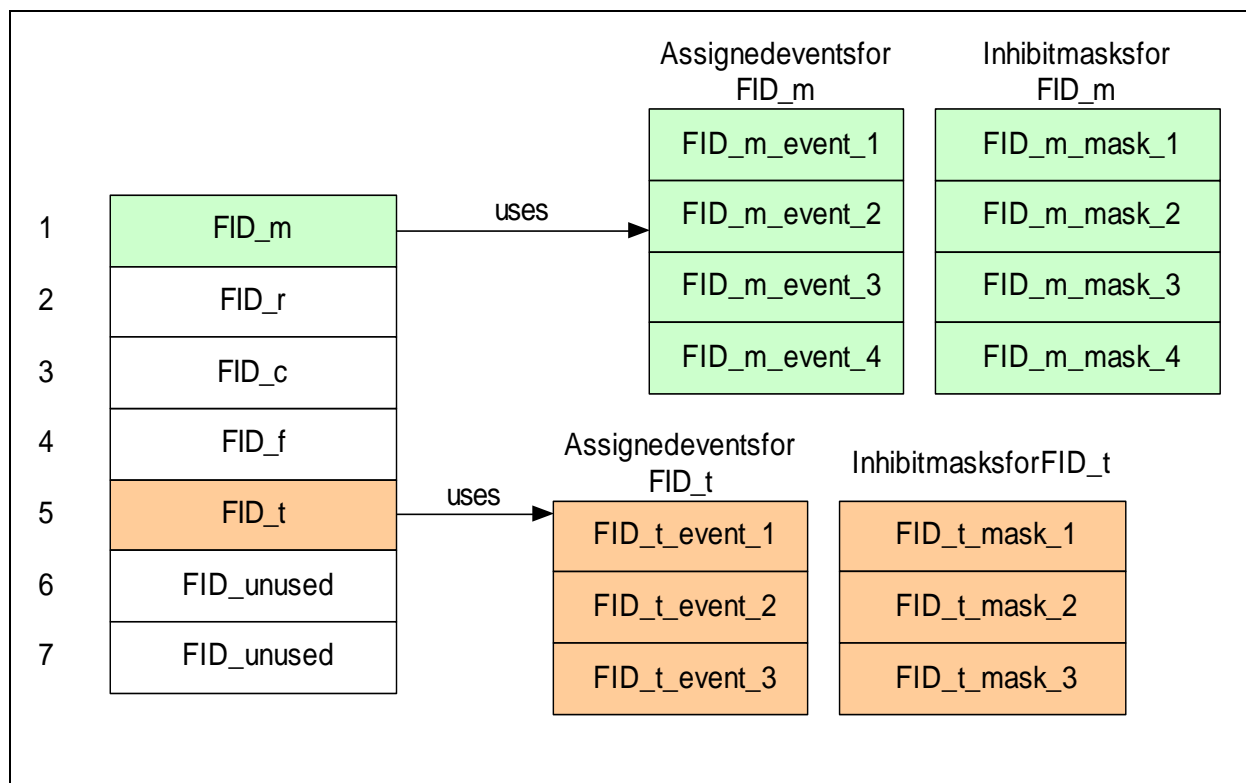
A configurable number of EventIds and inhibition masks are assigned to one FID. The number of EventIds and inhibit masks per FID have to match so that for each configured event, a corresponding inhibit mask exists.

The inhibition mask contains the inhibition conditions for a FID provided that the associated EventIds have a certain status (`Dem_EventStatusExtendedType`). These masks define which states of an event the FID is sensitive to. However, the mask does not only address certain bits according to the `Dem_EventStatusExtendedType`, it rather selects an algorithm to calculate the boolean inhibition condition from the `Dem_EventStatusExtendedType`.

The implementation of the FiM data structure cannot be prescribed. A possible implementation of the inhibit matrix could be a block of calibration values for each

inhibit source (=EventId). That means for each EventId a list of FIDs and masks is available that shall be inhibited by this EventId. A possible FiM structure consisting of such a configuration and a FID status array is exemplarily shown in Figure 2.

There is an inhibition mask assigned to every FID and both are assigned to a particular EventId. If this event has a certain state, the inhibition of the FID becomes active if the event state matches the configured mask.



**Figure 2: Inhibit mask**

**[SWS\_Fim\_00008]** [The FiM module shall provide the possibility to modify the inhibit conditions by post-built configuration.](SRS\_Fim\_04706)

Depending on the implementation, it might not be possible to:

- Add new events.
- Extend the number of inhibited FID's per event.
- Extend the specified configuration parameters concerning number of events, number of FIDs and number of links.

**7.2.2.2 Interaction between Dem and Function Inhibition Manager (FiM)**

**[SWS\_Fim\_00022]** [The purpose of the FiM module is to provide services to control (permit / inhibit) functionality within SW-Cs based on Dem events being supported as inhibit conditions.](SRS\_Fim\_04717)

**[SWS\_Fim\_00065]** [The Function Inhibition Manager shall use the FID – EventIDs – inhibition masks relations provided by the software components to determine the permission state for all configured FIDs.]()

Upon changes of a reported event status, the Dem shall inform the FiM (or other SW-C) about the new status if the FiM is not configured to use polling mode. For this purpose, it shall use the API function `FiM_DemTriggerOnEventStatus`.

Using this API call, the inhibit links assigned to this source can be updated immediately every time an inhibit source changes its status.

1. Note: From the function point of view, synchronous update of inhibit / release conditions can be made either within or outside of `FiM_MainFunction` API.

As mentioned in chapter (4.1), the implementation of the FiM highly depends on requirements (e.g. timing requirements) derived from applications. If an application requires fast reaction times the FiM has to provide FID information sufficiently fast to allow triggering limp-home functionality.

The API `FiM_DemTriggerOnEventStatus` is only relevant if a status variable per FID is stored. In an alternative implementation when no status is stored and the permission status is calculated every time when queried, the API `FiM_DemTriggerOnEventStatus` is without effect.

As an example of implementation, Figure 3 shows the calculation of a single EventId-FID link. On the left hand side, the event status is reported by the Dem as `Dem_EventStatusExtendedType`. This status is compared to the mask configured for the EventId associated with the FID.

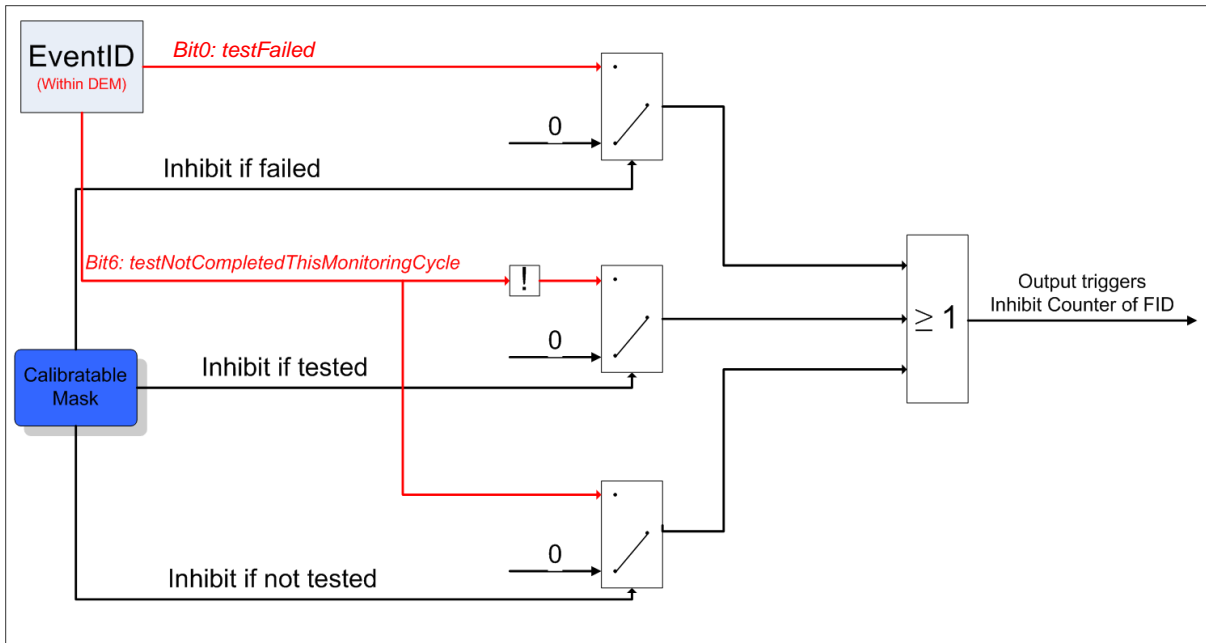
An inhibition counter is assigned to each FID. The inhibition counter contains the number of currently inhibiting EventIds.

If the calculation is performed cyclically (event status is read through `Dem_GetEventStatus`), the inhibition counter shall be incremented if the status and the mask match; otherwise, the inhibition counter is not updated. This is applicable for `FiM_GetFunctionPermission` (if the permission state has to be computed upon the query) and `FiM_MainFunction` APIs.

In the trigger on event status change, the stored currently inhibiting EventIds (inhibition counter) shall be used for the computation for the permission state. If there is an event status change reported by `FiM_DemTriggerOnEventStatus`, then the following shall be performed:

- a. If the change in status for the EventId results in a released state (mask does not match with the event status), then the inhibition counter has to be decremented.
- b. If the change in status for the EventId results in an inhibited state (mask matches with the event status), then the inhibition counter has to be incremented.

If the inhibition counter is > 0, then the FID permission state shall be set to FALSE, otherwise the FID permission state shall be set to TRUE.



**Figure 3: Calculation of permission state based on event status information**

**[SWS\_Fim\_00012]** [The FiM module shall calculate the inhibit status based on the actual status of the inhibit source and the calibrated mask which exists for each inhibit source (ref. 10.2.7). The FiM module shall inhibit the FID if the Event status is equal to the calibrated mask (=Defect, Tested, NotTested). The inhibition is deactivated if the mask of the event does not match anymore the calibrated value.] (SRS\_Fim\_04702)

Optionally, the tested status can be used for inhibiting. Depending on the inhibition condition, the inhibition can be active if the event has status "Tested" or "NotTested". If no tested value is selected, the tested status is not relevant.

The available combinations of status flags are assigned to a predefined value which has verbal representation like "Tested", "Not\_Tested" or Last\_Failed".



**7.2.2.3 Interaction between SW-Components and Function Inhibition Manager (FiM)**

**[SWS\_Fim\_00016]** 「The configuration engineer shall provide at compile time the inhibit conditions for each FID required for handling the dependencies of functionalities and events in the FiM module.」(SRS\_Fim\_04706)

Note, that modifications by calibration shall be possible. The configuration mechanism of the FiM using SW-component template contents shall consider these requirements.

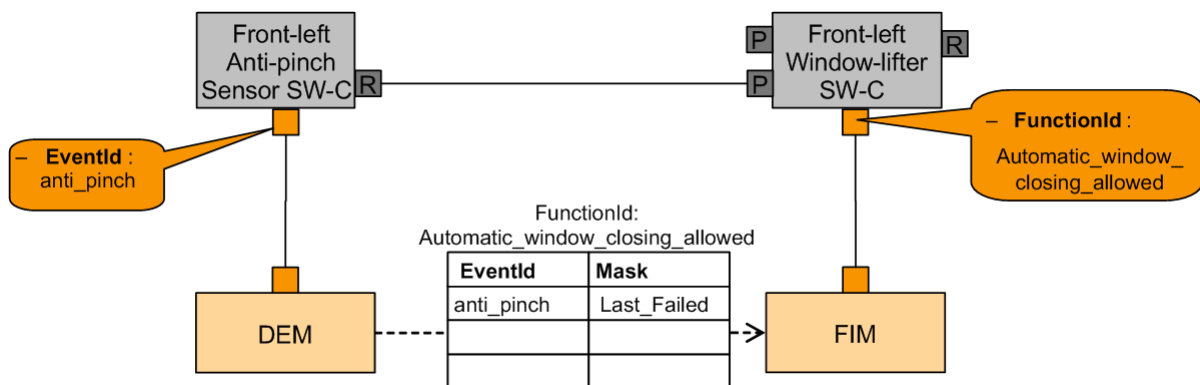
First, the FID needs to be introduced and allocated. Furthermore, for each FID a list of events plus associated mask causing the inhibition of the FID shall be provided by the SW-component. Chapter 10 introduces how the SW-component template considers these configuration requirements.

During the configuration process, the data structures are built up. Depending on the implementation this could, e.g. be a mapping of an event onto all affected FIDs or alternatively vice versa, a mapping of a FID onto all events affecting it.

Controlling implies that within the implemented functionality, the permission of a FID is queried via AUTOSAR service.

**[SWS\_Fim\_00020]** 「 The FiM module shall ensure an immediate control of functionality by synchronously responding to an incoming permission query. The FiM module shall realize this behavior either by storing the permission state as a status variable or by evaluation of the event states upon permission query. 」 (SRS\_Fim\_04713)

**7.2.2.4 Application example for FiM usage**



**Figure 4: FiM usage**

- The configuration of the FiM actually establishes the relationship between the EventId and the assigned FunctionId(s)
- The required information is:
  - For each FunctionId: How does the status of the FunctionId depend on the status of one/several EventIds?
    - The mask determines the relationship between the EventId status and the inhibit status of the FunctionId.
    - The row result is 'OR'ed to come up with the overall result for one FunctionId if it depends on several EventIds.

### 7.2.2.5 Initialization

**[SWS\_Fim\_00018]** «If Dem events status information is used, the FiM module shall compute the permission states for all FIDs at its initialization based on all restored event status information (not only events stored in the fault memory) of the Dem.»  
(SRS\_BSW\_00416, SRS\_Fim\_04712)

Hence, the FiM needs to be initialized after the Dem.

If FiM and Dem are delivered as one bundle, the FiM may have direct access to event information (structure) in the Dem rather than using the API `Dem_GetEventStatus`. In this way, a time-efficient initialization of the FiM can be achieved.

## 7.2.3 OBD-Functionality

### 7.2.3.1 In-Use-Monitor Performance Ratio (IUMPR) Support

In order to track the behavior of diagnostic functions in every day usage, in particular the capability to find malfunctions, the regulations require the tracking of this performance in relation to a standardized driving profile. This is called "In-Use Monitor Performance Ratio" (IUMPR) defined as the number of times a fault could have been found (=numerator) divided by the number of times the standardized driving profile has been fulfilled (=denominator). The relevant data recording is allocated in the Dem based on FIDs and EventIDs.

Thus, based on the FiM configuration of the referenced FIDs it can be evaluated whether a Ratio Id specific data record needs to be stopped. In particular, IUMPR tracking shall be stopped as long as the entry remains visible in service \$07.

The Dem may use the FiM configuration for its IUMPR calculation or by call of `FiM_GetFunctionPermission` of a dedicated FID.

Note: The FiM does not provide special OBDII functionality but uses already existing mechanisms for OBDII.

## 7.2.4 Auxiliary explanations and definitions

### 7.2.4.1 Output for other WPs

In order to be runtime-efficient, the event status information needs to be evaluated quickly, e.g. in the `FiM_Init` function. If Dem and FiM are implemented as one package, the Dem-APIs with access to event status information are not necessarily used and so direct access to event status information is allowed (see AUTOSAR conformance classes).

## 7.2.5 Version check

For details refer to the chapter 5.1.8 “Version Check” in *SWS\_BSWGeneral*.

### 7.3 Error classification

The FiM checks for certain faults during development and integration phase.

**[SWS\_Fim\_00076]** [The FiM module shall detect the following errors and exceptions depending on its configuration (development/production): ]()

<i>Type or error</i>	<i>Relevance</i>	<i>Related error code</i>	<i>Value [hex]</i>
API function called before the FiM module has been full initialized or after the FiM module has been shut down	Development	FIM_E_UNINIT	0x01
FiM_GetFunctionPermission called with wrong FID	Development	FIM_E_FID_OUT_OF_RANGE	0x02
Dem calls FiM with invalid EventId	Development	FIM_E_EVENTID_OUT_OF_RANGE	0x03
API is invoked with NULL Pointer	Development	FIM_E_INVALID_POINTER	0x04

## 7.4 Error detection

For details refer to the chapter 7.3 “Error Detection” in *SWS\_BSWGeneral*.

## 7.5 Error notification

For details refer to the chapter 7.4 “Error notification” in *SWS\_BSWGeneral*.

## 7.6 Debugging

For details refer to the chapter 7.1.17 “Debugging support” in *SWS\_BSWGeneral*.

## 8 API specification



## 8.1 API

### 8.1.1 Imported types

In this chapter, all types included from the following files are listed:

[SWS\_Fim\_00081] ⌈

<i>Module</i>	<i>Imported Type</i>
Dem	Dem_EventIdType
	Dem_UdsStatusByteType
SchM	SchM_ReturnType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋()

### 8.1.2 Type definitions

#### 8.1.2.1 FiM\_FunctionIdType

[SWS\_Fim\_00027] ⌈

<b>Name:</b>	FiM FunctionIdType	
<b>Type:</b>	uint8, uint16	
<b>Range:</b>	0..255, 0..65535	-- Identifier of functionality Configurable, size depends on System complexity. Remark: Not all numbers are valid. The FIM data generation tool shall only assign valid values.
<b>Description:</b>	Type for the FunctionID	

⌋(SRS\_BSW\_00304, SRS\_BSW\_00305, SRS\_BSW\_00377)

#### 8.1.2.2 FiM\_ConfigType

[SWS\_Fim\_00092] ⌈

<b>Name:</b>	FiM ConfigType	
<b>Type:</b>	Structure	
<b>Range:</b>	--	implementation specific
<b>Description:</b>	This type defines a data structure for the post build parameters of the FIM. At initialization the FIM gets a pointer to a structure of this type to get access to its configuration data, which is necessary for initialization.	

⌋()

### 8.1.3 Function definitions

This is a list of functions provided for upper layer modules.

#### 8.1.3.1 Interface ECU State Manager ↔ FiM

##### 8.1.3.1.1 FiM\_Init

[SWS\_Fim\_00077] ↑

<b>Service name:</b>	FiM_Init
<b>Syntax:</b>	void FiM_Init( const FiM_ConfigType* FiMConfigPtr )
<b>Service ID[hex]:</b>	0x00
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	FiMConfigPtr --
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This service initializes the FiM.

↓()

The ECU State Manager shall call the function `FiM_Init` during the startup phase of the ECU in order to initialize the permission states of the FIDs based on the event data of the Dem.

The FiM is not functional until this function has been called.

**[SWS\_Fim\_00004]** 「The FiM calculates the permission states based on event status information stored in the Dem and inhibition mask configuration.」(BSW00101, BSW00166, SRS\_BSW\_00310, SRS\_BSW\_00358, SRS\_BSW\_00384, SRS\_BSW\_00414, SRS\_BSW\_00416, SRS\_Fim\_04712)

Therefore, the Dem must be initialized before the FiM. SW-Components controlled by FID interface are initialized afterwards.

**[SWS\_Fim\_00071]** 「The FiM module's environment shall make sure that the Dem has been initialized before the FiM is initialized.」()

**[SWS\_Fim\_00045]** 「If development error detection is turned on the FiM module shall report an error to the DET if it has not successfully completed the initialization and has detected not permitted access.」(SRS\_BSW\_00358, SRS\_BSW\_00406)

**[SWS\_Fim\_00059]** 「A static status variable denoting if the FiM is initialized shall be initialized with value 0 before any APIs of the FiM is called.

`FiM_Init` shall set the static status variable to a value not equal to 0.」  
(SRS\_BSW\_00358, SRS\_BSW\_00406)

In order to restore the permission states quickly, it is recommended that the Dem provides direct access to event status information if Dem and FiM are implemented as a cluster. In this case, the FiM needs to have knowledge about the data structure of the Dem so that it can directly access EventId states.

**[SWS\_Fim\_00072]** 「If Dem and FiM are implemented as two separate modules, the FiM module shall access the EventId states through the API call

`Dem_GetEventStatus.」()`

**Note:** There is no explicit action during shutdown. The permission states remain valid until the ECU is shut down since they directly depend on the event status information.

### 8.1.3.2 Interface SW-Components ↔ FiM

#### 8.1.3.2.1 FiM\_GetFunctionPermission

[SWS\_Fim\_00011] ⌈

<b>Service name:</b>	FiM_GetFunctionPermission	
<b>Syntax:</b>	Std_ReturnType FiM_GetFunctionPermission( FiM_FunctionIdType FID, boolean* Permission )	
<b>Service ID[hex]:</b>	0x01	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	FID	Identification of a functionality by assigned FID. The FunctionId is configured in the FiM.  Min.: 1 (0: Indication of no functionality) Max.: Result of configuration of FIDs in FiM (Max is either 255 or 65535)
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	Permission	TRUE: FID has permission to run FALSE: FID has no permission to run, i.e. shall not be executed
<b>Return value:</b>	Std_ReturnType	E_OK: The request is accepted E_NOT_OK: The request is not accepted, ie. initialization of FiM not completed
<b>Description:</b>	This service reports the permission state to the functionality.	

⌋(BSW00166, SRS\_BSW\_00310, SRS\_BSW\_00312, SRS\_Fim\_04700, SRS\_Fim\_04709, SRS\_Fim\_04722)

[SWS\_Fim\_00066] ⌈The SW Components and the BSW shall use the function `FiM_GetFunctionPermission` to query for the permission to execute a certain functionality represented by the respective FID.⌋()

[SWS\_Fim\_00025] ⌈The function `FiM_GetFunctionPermission` shall deliver the return value synchronously to enable direct use of this information for controlling and executing the underlying code in the software component.⌋()

[SWS\_Fim\_00055] ⌈If development error detection for the module FiM is enabled: the function `FiM_GetFunctionPermission` shall perform a plausibility check on the FID range. If a FID is out of range, the function shall raise a development error and return no permission (FALSE).⌋(SRS\_BSW\_00406)

[SWS\_Fim\_00056] ⌈If development error detection for the module FiM is enabled: the function `FiM_GetFunctionPermission` shall check that the initialization of the module FiM has been completed. If the function detects that the initialization is not

complete, it shall raise a development error and return no permission (FALSE).」  
(SRS\_BSW\_00406)

### 8.1.3.3 Interface Dem ↔ FiM

#### 8.1.3.3.1 FiM\_DemTriggerOnEventStatus

[SWS\_Fim\_00021]「

<b>Service name:</b>	FiM_DemTriggerOnEventStatus	
<b>Syntax:</b>	<pre>void FiM_DemTriggerOnEventStatus (     Dem_EventIdType EventId,     Dem_UdsStatusByteType EventStatusByteOld,     Dem_UdsStatusByteType EventStatusByteNew )</pre>	
<b>Service ID[hex]:</b>	0x02	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	EventId	Identification of an Event by assigned event number. The Event Number is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.: Result of configuration of Event Numbers in DEM (Max is either 255 or 65535)
	EventStatusByteOld	Extended event status before change
	EventStatusByteNew	Detected / reported of event status
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	This service is provided by the Dem in order to call FiM upon status changes.	

」(BSW00166, SRS\_BSW\_00310, SRS\_BSW\_00312, SRS\_Fim\_04717)

[SWS\_Fim\_00073]「If the FiM module is notified by Dem about event status changes (as defined in configuration parameter FIM\_EVENT\_UPDATE\_TRIGGERED\_BY\_DEM = TRUE), the Dem module shall call the function FiM\_DemTriggerOnEventStatus whenever the status of an event changes.」()

In case, the FiM module polls event status (as defined in configuration parameter FIM\_EVENT\_UPDATE\_TRIGGERED\_BY\_DEM = FALSE), the FiM has to query event status information from the Dem. In that case, the Dem does not have to call the function FiM\_DemTriggerOnEventStatus. FiM module can decide whether to poll all event status in a cyclic manner, or one by one upon a call to FiM\_GetFunctionPermission.

[SWS\_Fim\_00057]「If development error detection for the module FiM is enabled: the function FiM\_DemTriggerOnEventStatus shall perform a plausibility check on

the EventId. If the requested EventId is not existing in the Dem configuration, the function shall raise the development error FIM\_E\_EVENTID\_OUT\_OF\_RANGE.」(SRS\_BSW\_00406)

**[SWS\_Fim\_00058]** 「If development error detection for the module FiM is enabled: The function `FiM_DemTriggerOnEventStatus` shall check for complete initialization of the FiM. If the function detects that the initialization is not complete, it shall raise a development error.」(SRS\_BSW\_00406)

### 8.1.3.3.2 FiM\_DemInit

**[SWS\_Fim\_00006]** 「

<b>Service name:</b>	FiM_DemInit
<b>Syntax:</b>	<pre>void FiM_DemInit(     void )</pre>
<b>Service ID[hex]:</b>	0x03
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This service re-initializes the FiM.

」(BSW00101, BSW00166, SRS\_BSW\_00310, SRS\_BSW\_00358)

**[SWS\_Fim\_00069]** 「The function `FiM_DemInit` shall re-compute the permission state for all FIDs.」()

**[SWS\_Fim\_00082]** 「If Dem and FiM are implemented as two separate modules, the function `FiM_DemInit` shall synchronously access the EventId states via the function `Dem_GetEventStatus`.」()

In case Dem and FiM are implemented as one bundle, the FiM module needs to have knowledge about the data structure of the Dem so that it can directly access the EventId states.

### 8.1.3.4 FiM\_GetVersionInfo

[SWS\_Fim\_00078] ⌈

<b>Service name:</b>	FiM_GetVersionInfo	
<b>Syntax:</b>	<pre>void FiM_GetVersionInfo(     Std_VersionInfoType* versioninfo )</pre>	
<b>Service ID[hex]:</b>	0x04	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	None	
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	versioninfo	Pointer to where to store the version information of this module.
<b>Return value:</b>	None	
<b>Description:</b>	This service returns the version information of this module.	

⌋()

### 8.1.3.5 Call-back notifications

This chapter lists all functions provided by the FiM module and used by lower layer modules.

No callback notification is specified.

### 8.1.3.6 Scheduled functions

This chapter lists all functions provided by the FiM module and called directly by the Basic Software Module Scheduler.

#### 8.1.3.6.1 FiM\_MainFunction

[SWS\_Fim\_00060] ⌈

<b>Service name:</b>	FiM_MainFunction	
<b>Syntax:</b>	<pre>void FiM_MainFunction(     void )</pre>	
<b>Service ID[hex]:</b>	0x05	
<b>Description:</b>	--	

⌋(SRS\_BSW\_00373, SRS\_BSW\_00376)

The evaluation of permission states can be performed either on event change or cyclically.

[SWS\_Fim\_00070] ⌈If FiM module polls event status (as defined in configuration parameter `FIM_EVENT_UPDATE_TRIGGERED_BY_DEM = FALSE`) and decides to

do it in a cyclic manner, `FiM_MainFunction` shall be used to calculate the permission states of all `EventIds` using their inhibition masks. The API `Dem_GetEventStatus` shall be used to get status information of `EventIds`. `⌋()`

**[SWS\_Fim\_00067]** `⌈`The FiM shall perform the evaluation of actual `EventIds` status information cyclically for all the `EventIds` using the inhibition mask and then calculate the corresponding FID permission states. FiM shall access the event status information using the API `Dem_GetEventStatus` if Dem and FiM are implemented as separate modules. FiM shall access the event status structure of Dem if Dem and FiM are implemented as a bundle. `⌋()`

### 8.1.4 Expected Interfaces

This chapter lists all functions the module FiM requires from other modules.

#### 8.1.4.1 Mandatory Interfaces

This chapter defines all interfaces, which are required to fulfill the core functionality of the module.

**[SWS\_Fim\_00079]** `⌈`

<i>API function</i>	<i>Description</i>
<code>Dem_GetEventStatus</code>	Gets the current extended event status of an event.
<code>SchM_ActMainFunction_FiM</code>	Invokes the <code>SchM_ActMainFunction</code> function to trigger the activation of a corresponding main processing function.
<code>SchM_CancelMainFunction_FiM</code>	Invokes the <code>SchM_CancelMainFunction</code> function to trigger the cancellation of the requested activation of a corresponding main processing function.

`⌋()`

#### 8.1.4.2 Optional Interfaces

This chapter defines all interfaces, which are required to fulfill an optional functionality of the module.

**[SWS\_Fim\_00080]** `⌈`

<i>API function</i>	<i>Description</i>
<code>Det_ReportError</code>	Service to report development errors.

`⌋()`



## 8.2 Service interfaces

This chapter specifies the ports and port interfaces to operate the FiM functionality over the VFB.

### 8.2.1 Client-Server-Interfaces

#### 8.2.1.1 FiM\_FunctionInhibition

Using the concepts of the SW-C template, the interface is defined as follows:

##### [SWS\_Fim\_00090] ▮

Name	FunctionInhibition	
Comment	The SW Components can use this service to query for the permission to execute a certain functionality represented by a FID.	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK

##### Operations

GetFunctionPermission		
Comments	Get the permission state of the respective FID.	
Variation	--	
Parameters	Permission	
	Comment	The permission of the requested FID. TRUE: FID has permission to run FALSE: FID has no permission to run, i.e. shall not be executed
	Type	boolean
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	The request is not accepted, ie. initialization of FIM not completed

▮(SRS\_Fim\_04700)

## 8.2.2 Implementation Data Types

None.

## 8.2.3 Ports

[SWS\_Fim\_00094]⌈

Name	Func_{Name}		
Kind	ProvidedPort	Interface	FunctionInhibition
Description	A client can query the FiM for execution permission for a specific function. The FIDs which represent the functions are not directly used by the client SW-C. Instead, the mechanism of "port-defined argument values" is used and every FID is mapped to a separate port that is responsible for the data exchange via RTE.		
Variation	Name = {ecuc(Fim/FimConfigSet/FiMFID.SHORT-NAME)}		

⌋( SRS\_Fim\_04700)

## 8.2.4 Internal Behavior

The InternalBehavior of the FiM Service is only seen by the local RTE. Additionally to the definition of the function identifiers as port defined arguments, the InternalBehavior has to specify the operation invoked runnables:

```
InternalBehavior FiM {
    // definition of associated operation-invoked RTE-events not shown
    // (it is done in the same way as for any SWC type)

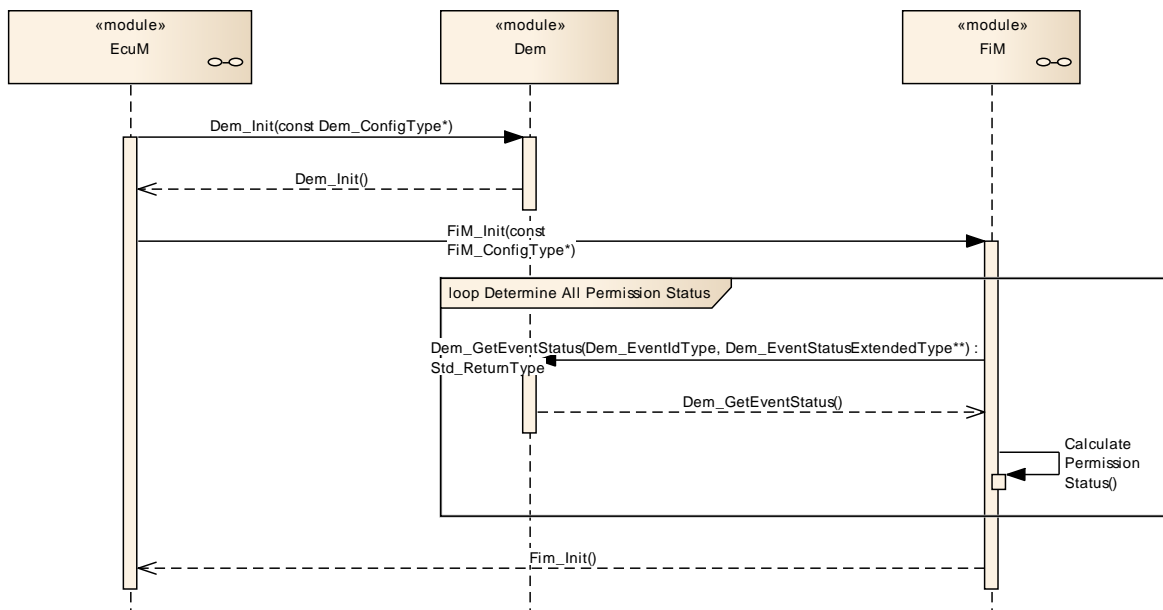
    // section "runnable entities":
    RunnableEntity GetFunctionPermission
        symbol "FiM_GetFunctionPermission"
        canbeInvokedConcurrently = TRUE
}
```

## 9 Sequence diagrams

## 9.1 Initialization sequence of FiM

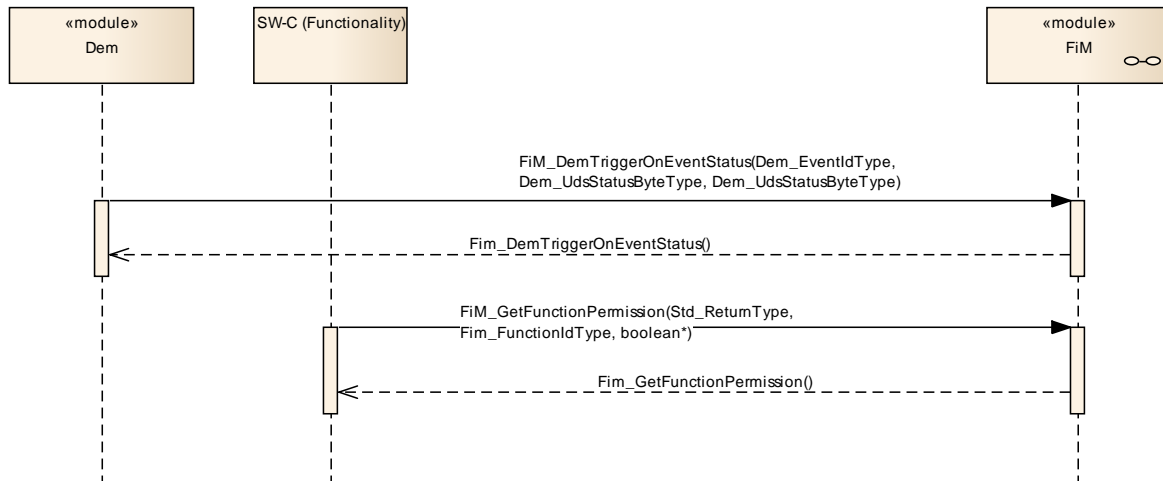
Dem\_Init is called by the ECU State Manager as soon as the NVRAM Manager has copied the error memory relevant data block(s) from NVRAM to RAM ([10], DEM065). After the initialization of the Dem, the ECU State Manager is able to call FiM\_Init and the FiM is able to calculate the permission states of all FIDs based on the event status information afterwards. The FiM loops over all configured FIDs and reads the event status information by calling Dem\_GetEventStatus.

The FiM assumes that the copying process of NVRAM data has been successful when FiM\_Init is called.



## 9.2 FiM\_DemTriggerOnEventStatus

The sequence diagram below illustrates how the Dem informs the FiM about the change of a certain event status by calling `FiM_DemTriggerOnEventStatus`. Furthermore, it indicates how the FID is affected by requesting permission status using `FiM_GetFunctionPermission`.



## 10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification, Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module FiM.

Chapter 10.3 specifies published information of the module FiM.

## 10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS\_BSWGeneral*.

## 10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described in Chapter 7 and Chapter 7.3.

[SWS\_Fim\_00062] ▮

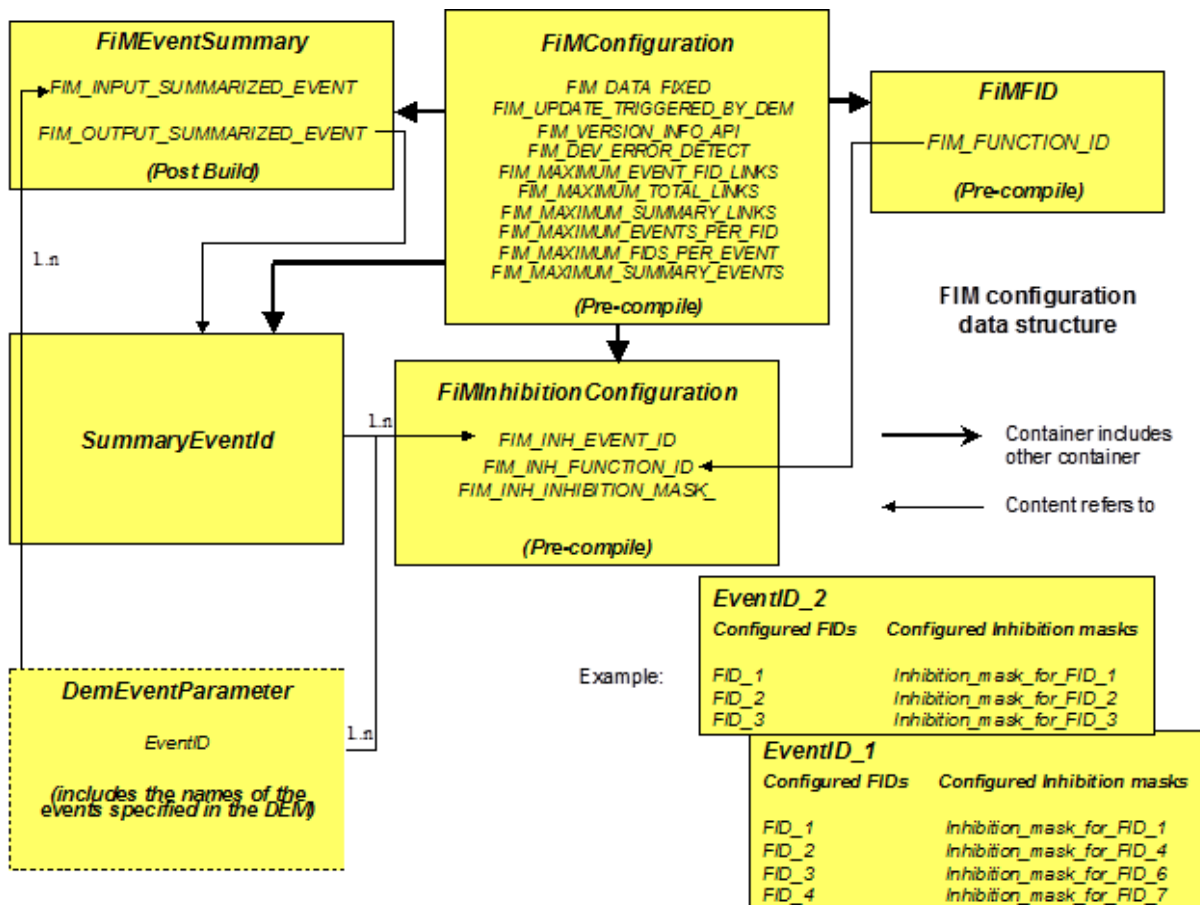


Figure 5: FiM configuration (SRS\_BSW\_00404, SRS\_BSW\_00405)

### 10.2.1 Variants

Variants describe sets of configuration parameters. Thus describe the possible configuration variants of this module

[SWS\_Fim\_00089] ▮ VARIANT-PRE-COMPILE: Only parameters with "Pre-compile time" configuration are allowed in this variant. ▮()



[SWS\_Fim\_00091] [VARIANT-POST-BUILD: Parameters with "Pre-compile time", "Link time" and "Post-build time" are allowed in this variant. ]()

The inhibit conditions provided by the software components can be considered as a superset of all inhibit conditions for all variants. Based on that, the inhibit configuration has to be derived for all events and FIDs in one project. If an EventId or FID is not supported in a certain variant, the link between them has no effect. The requirements SRS\_BSW\_00404 and SRS\_BSW\_00405 are therefore not applicable.

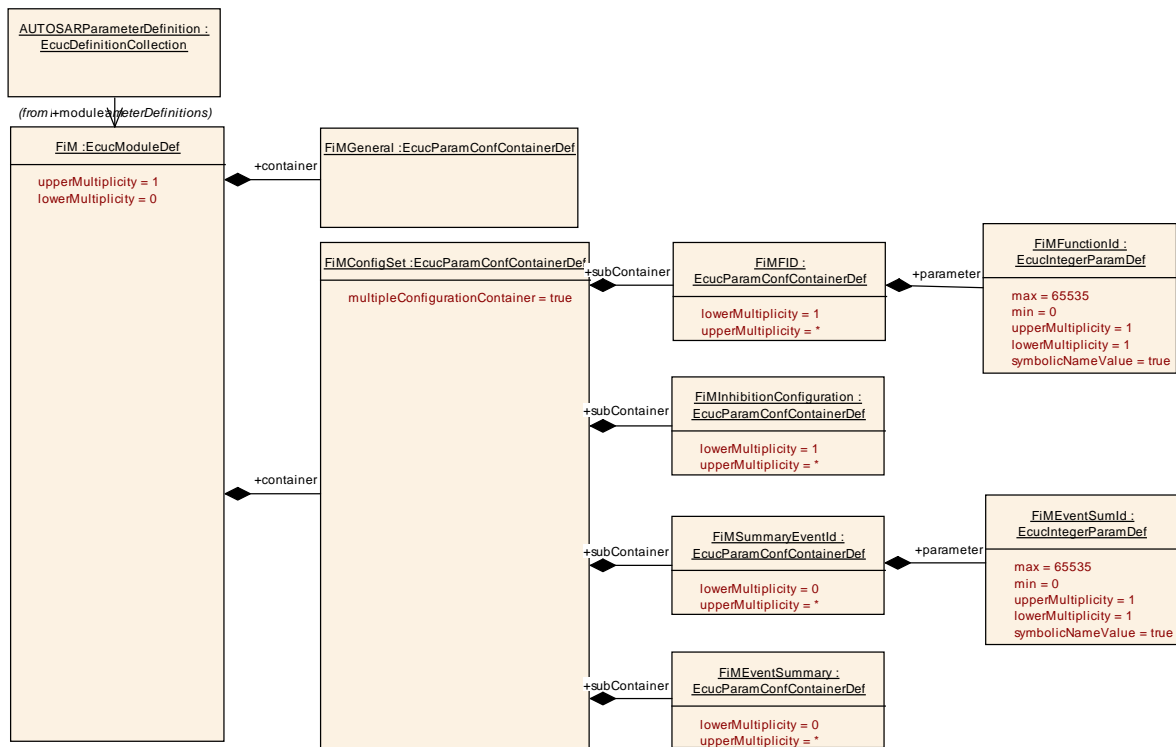


Figure 6: Configuration overview for FIM

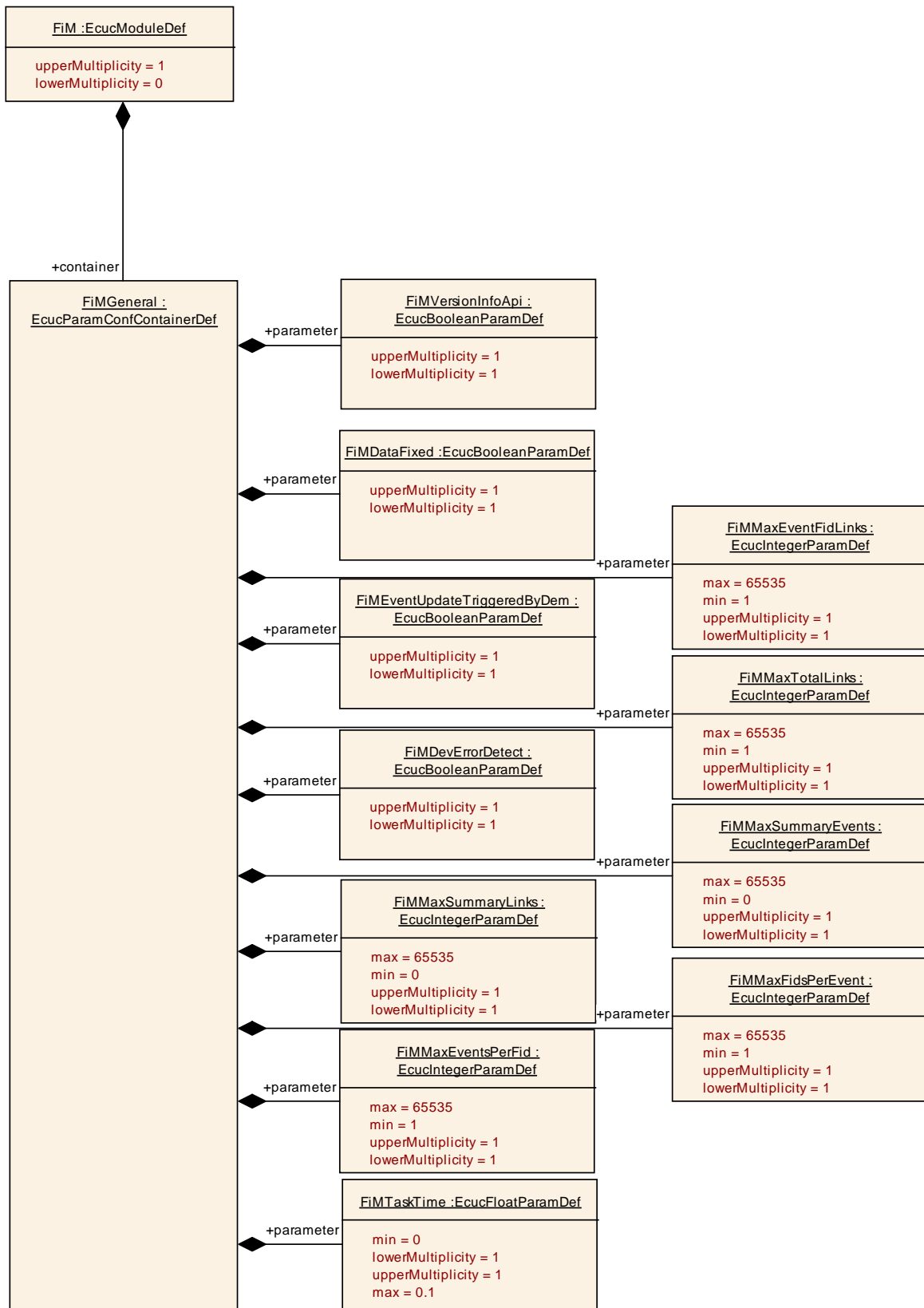


Figure 7: Configuration overview for FiMGeneral

## 10.2.2 FiM

<b>Module Name</b>	FiM
<b>Module Description</b>	Configuration of the FiM (Function Inhibition Manager) module.

Included Containers		
Container Name	Multiplicity	Scope / Dependency
FiMConfigSet	1	This container contains the configuration parameters and sub containers of the FiM module supporting multiple configuration sets. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
FiMGeneral	1	--

## 10.2.3 FiMGeneral

<b>SWS Item</b>	ECUC_FiM_00040 :
<b>Container Name</b>	FiMGeneral
<b>Description</b>	--
<b>Configuration Parameters</b>	

<b>SWS Item</b>	ECUC_FiM_00008 :									
<b>Name</b>	FiMDataFixed {FIM_DATA_FIXED}									
<b>Description</b>	Enable or disable calibration of inhibit relations The scope of the parameter is to meet the requirement (FIM008) to have the option to calibrate inhibit data on the one hand side and also to provide the option to protect inhibit data for consistency reasons.									
<b>Multiplicity</b>	1									
<b>Type</b>	EcucBooleanParamDef									
<b>Default value</b>	--									
<b>ConfigurationClass</b>	<table border="1"> <tr> <td><b>Pre-compile time</b></td> <td>X</td> <td>All Variants</td> </tr> <tr> <td><b>Link time</b></td> <td>--</td> <td></td> </tr> <tr> <td><b>Post-build time</b></td> <td>--</td> <td></td> </tr> </table>	<b>Pre-compile time</b>	X	All Variants	<b>Link time</b>	--		<b>Post-build time</b>	--	
<b>Pre-compile time</b>	X	All Variants								
<b>Link time</b>	--									
<b>Post-build time</b>	--									
<b>Scope / Dependency</b>	scope: local									

<b>SWS Item</b>	ECUC_FiM_00087 :									
<b>Name</b>	FiMDevErrorDetect {FIM_DEV_ERROR_DETECT}									
<b>Description</b>	This configuration parameter is used to switch on or to switch off the detection of development errors during development.									
<b>Multiplicity</b>	1									
<b>Type</b>	EcucBooleanParamDef									
<b>Default value</b>	--									
<b>ConfigurationClass</b>	<table border="1"> <tr> <td><b>Pre-compile time</b></td> <td>X</td> <td>All Variants</td> </tr> <tr> <td><b>Link time</b></td> <td>--</td> <td></td> </tr> <tr> <td><b>Post-build time</b></td> <td>--</td> <td></td> </tr> </table>	<b>Pre-compile time</b>	X	All Variants	<b>Link time</b>	--		<b>Post-build time</b>	--	
<b>Pre-compile time</b>	X	All Variants								
<b>Link time</b>	--									
<b>Post-build time</b>	--									
<b>Scope / Dependency</b>	scope: local									

<b>SWS Item</b>	ECUC_FiM_00086 :
<b>Name</b>	FiMEventUpdateTriggeredByDem {FIM_EVENT_UPDATE_TRIGGERED_BY_DEM}
<b>Description</b>	This configuration parameter specifies the way FIM obtains status of EventIds. TRUE: the DEM informs FIM about changes of event status, FALSE: the

	FIM polls event status from the DEM module either cyclically or on demand.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FiM_00088 :</b>		
<b>Name</b>	FiMMaxEventFidLinks {FIM_MAXIMUM_EVENT_FID_LINKS}		
<b>Description</b>	This configuration parameter specifies the total maximum number of links between EventIds and FIDs.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FiM_00089 :</b>		
<b>Name</b>	FiMMaxEventsPerFid {FIM_MAXIMUM_EVENTS_PER_FID}		
<b>Description</b>	This configuration parameter specifies the maximum number of EventIds that can be linked to a single FID.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FiM_00090 :</b>		
<b>Name</b>	FiMMaxFidsPerEvent {FIM_MAXIMUM_FIDS_PER_EVENT}		
<b>Description</b>	This configuration parameter specifies the maximum number of FIDs that can be linked to a single event.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FiM_00091 :</b>		
<b>Name</b>	FiMMaxSummaryEvents {FIM_MAXIMUM_SUMMARY_EVENTS}		
<b>Description</b>	This configuration parameter specifies the maximum number of summarized events that can be configured.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		

<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FiM_00092 :</b>		
<b>Name</b>	FiMMaxSummaryLinks {FIM_MAXIMUM_SUMMARY_LINKS}		
<b>Description</b>	This configuration parameter specifies the total maximum number of links between EventIds and summarized events.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FiM_00093 :</b>		
<b>Name</b>	FiMMaxTotalLinks {FIM_MAXIMUM_TOTAL_LINKS}		
<b>Description</b>	This configuration parameter specifies the total maximum number of links between EventIds and FIDs plus the number of links between EventIds and summarized events.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FiM_00600 :</b>		
<b>Name</b>	FiMTaskTime {FIM_TASK_TIME}		
<b>Description</b>	<p>Allow to configure the time for the periodic cyclic task. Please note: This configuration value shall be equal to the value in the Basic Software Scheduler configuration of the RTE module.</p> <p>The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. DEM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DEM.</p> <p>min: A negative value is not allowed.</p> <p>max: FID must be set after a maximal time of 100ms after DEM status is set.</p> <p>upperMultiplicity: Exactly one TaskTime must be specified per configuration.</p> <p>lowerMultiplicity: Exactly one TaskTime must be specified per configuration.</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	0 .. 0.1		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>ECUC_FiM_00094 :</b>		
<b>Name</b>	FiMVersionInfoApi {FIM_VERSION_INFO_API}		
<b>Description</b>	This configuration parameter is used to switch on or to switch off the API to get the version information.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

## 10.2.4

### 10.2.5 FiMConfigSet

<b>SWS Item</b>	<b>ECUC_FiM_00601 :</b>		
<b>Container Name</b>	FiMConfigSet [Multi Config Container]		
<b>Description</b>	This container contains the configuration parameters and sub containers of the FiM module supporting multiple configuration sets. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.		
<b>Configuration Parameters</b>			

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
FiMEventSummary	0..*	The summarized EventId definition record consists of a summarized event ID and a specific EventId. This record means that a particular FID that has to be disabled in case of summarized event (defined above) is to be disabled in any of the specific events. A possible solution could be assigning events as summarized events along with a list of specific events. During the configuration process the summarized event substitutes the referenced single events. However, it is not outlined how this requirement is solved - whether by configuration process or by implementation within the FiM. The FiM configuration tool could also build up a suitable data structure for summarized events and deal with it in the FiM implementation.
FiMFID	1..*	This container includes symbolic names of all FIDs.
FiMinhibitionConfiguration	1..*	This container includes all configuration parameters concerning the relationship between event and FID.
FiMSummaryEventId	0..*	This container defines the name of a summarized event.

### 10.2.6 FiMFID

<b>SWS Item</b>	<b>ECUC_FiM_00039 :</b>		
<b>Container Name</b>	FiMFID{FiMFID}		

<b>Description</b>	This container includes symbolic names of all FIDs.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_FiM_00085 :</b>		
<b>Name</b>	FiMFunctionId {FIM_FUNCTION_ID}		
<b>Description</b>	The configuration parameter is used as an ID which represents a functionality. FiMFunctionId is the unique identifier assigned during FIM configuration. Implementation Type: FIM_FunctionIdType.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: ECU		

<b>No Included Containers</b>
-------------------------------

### 10.2.7 FiMEventSummary

<b>SWS Item</b>	<b>ECUC_FiM_00037 :</b>		
<b>Container Name</b>	FiMEventSummary		
<b>Description</b>	The summarized EventId definition record consists of a summarized event ID and a specific EventId. This record means that a particular FID that has to be disabled in case of summarized event (defined above) is to be disabled in any of the specific events. A possible solution could be assigning events as summarized events along with a list of specific events. During the configuration process the summarized event substitutes the referenced single events. However, it is not outlined how this requirement is solved - whether by configuration process or by implementation within the FiM. The FiM configuration tool could also build up a suitable data structure for summarized events and deal with it in the FiM implementation.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_FiM_00083 :</b>		
<b>Name</b>	FiMInputSumEventRef {FIM_INPUT_SUMMARIZED_EVENT}		
<b>Description</b>	--		
<b>Multiplicity</b>	1..*		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FiM_00084 :</b>		
<b>Name</b>	FiMOutputSumEventRef {FIM_OUTPUT_SUMMARIZED_EVENT}		
<b>Description</b>	--		
<b>Multiplicity</b>	1		
<b>Type</b>	Symbolic name reference to [ FiMSummaryEventId ]		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	

	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

### 10.2.8 FiMinhibitionConfiguration

<b>SWS Item</b>	<b>ECUC_FiM_00038 :</b>
<b>Container Name</b>	FiMinhibitionConfiguration
<b>Description</b>	This container includes all configuration parameters concerning the relationship between event and FID.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_FiM_00096 :</b>		
<b>Name</b>	FiMinhInhibitionMask {FIM_INH_INHIBITION_MASK}		
<b>Description</b>	The configuration parameter is used to specify the inhibition mask for an event - FID relation.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	FIM_LAST_FAILED	Last Failed - DEM_UDS_STATUS_TF flag of Dem Eventstatus is set Use case: Re-configuration, avoiding follow-up errors	
	FIM_NOT_TESTED	Not Tested this cycle - DEM_UDS_STATUS_TNCTOC flag of Dem Eventstatus is set. Use case: Scheduling of monitors.	
	FIM_TESTED	Tested - DEM_UDS_STATUS_TNCTOC flag of Dem Eventstatus is not set. Use case: Self deactivation, check during driving cycle.	
	FIM_TESTED_AND_FAILED	Tested and Failed - DEM_UDS_STATUS_TF flag of Dem Eventstatus is set and DEM_UDS_STATUS_TNCTOC flag is not set Use case: Avoiding deadlocks, repeated monitoring.	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FiM_00095 :</b>		
<b>Name</b>	FiMinhFunctionIdRef {FIM_INH_FUNCTION_ID}		
<b>Description</b>	--		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ FiMFID ]		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
FiMinhEventId	1	The configuration parameter is used for an existing DEM event



		and summarized events as well.
--	--	--------------------------------

### 10.2.9 FiMInhEventId

<b>SWS Item</b>	<b>ECUC_FiM_00097 :</b>	
<b>Container Name</b>	FiMInhEventId{FIM_INH_EVENT_ID}	
<b>Description</b>	The configuration parameter is used for an existing DEM event and summarized events as well.	
<b>Configuration Parameters</b>		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
FiMInhRefChoice	1	--

### 10.2.10 FiMInhChoiceDemRef

<b>SWS Item</b>	<b>ECUC_FiM_00099 :</b>	
<b>Container Name</b>	FiMInhChoiceDemRef	
<b>Description</b>	--	
<b>Configuration Parameters</b>		

<b>SWS Item</b>	<b>ECUC_FiM_00100 :</b>	
<b>Name</b>	FiMInhEventRef	
<b>Description</b>	--	
<b>Multiplicity</b>	1	
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--
	<b>Link time</b>	--
	<b>Post-build time</b>	--
<b>Scope / Dependency</b>	scope: local	

<b>No Included Containers</b>		
-------------------------------	--	--

### 10.2.11 FiMInhChoiceSumRef

<b>SWS Item</b>	<b>ECUC_FiM_00101 :</b>	
<b>Container Name</b>	FiMInhChoiceSumRef	
<b>Description</b>	--	
<b>Configuration Parameters</b>		

<b>SWS Item</b>	<b>ECUC_FiM_00102 :</b>	
<b>Name</b>	FiMInhSumRef	
<b>Description</b>	--	
<b>Multiplicity</b>	1..*	
<b>Type</b>	Symbolic name reference to [ FiMSummaryEventId ]	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--
	<b>Link time</b>	--
	<b>Post-build time</b>	--
<b>Scope / Dependency</b>	scope: local	

**No Included Containers**

### 10.2.12 FiMInhRefChoice

<b>SWS Item</b>	ECUC_FiM_00098 :
<b>Choice container Name</b>	FiMInhRefChoice
<b>Description</b>	--

<b>Container Choices</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
FiMInhChoiceDemRef	0..1	--
FiMInhChoiceSumRef	0..1	--

### 10.2.13 FiMSummaryEventId

<b>SWS Item</b>	ECUC_FiM_00103 :
<b>Container Name</b>	FiMSummaryEventId
<b>Description</b>	This container defines the name of a summarized event.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	ECUC_FiM_00104 :		
<b>Name</b>	FiMEventSumId {FIM_SUMMARIZED_EVENT}		
<b>Description</b>	The summarized EventId definition record defines the existence of a summarized event with a specific name. This summarized event can be referenced in the FiMEventSummary (as FiMSummaryEventId) and Inhibition configuration (as FiMInhEventId).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

## 10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in *SWS\_BSWGeneral*.

## 11 Not applicable requirements

**[SWS\_Fim\_00999]** 「 These requirements are not applicable to this specification. 」

(BSW00006, BSW00007, BSW00005, BSW00009, BSW00010, BSW00159, BSW00160, BSW00161, BSW00162, BSW00164, BSW00167, BSW00168, BSW00170, BSW00172, SRS\_BSW\_00301, SRS\_BSW\_00302, SRS\_BSW\_00306, SRS\_BSW\_00307, SRS\_BSW\_00308, SRS\_BSW\_00309, SRS\_BSW\_00314, SRS\_BSW\_00323, BSW00324, SRS\_BSW\_00325, SRS\_BSW\_00326, SRS\_BSW\_00328, SRS\_BSW\_00330, SRS\_BSW\_00333, SRS\_BSW\_00334, SRS\_BSW\_00336, BSW0339, BSW0341, SRS\_BSW\_00342, SRS\_BSW\_00343, SRS\_BSW\_00347, SRS\_BSW\_00353, SRS\_BSW\_00355, SRS\_BSW\_00357, SRS\_BSW\_00359, SRS\_BSW\_00360, SRS\_BSW\_00361, SRS\_BSW\_00370, SRS\_BSW\_00375, SRS\_BSW\_00378, BSW00382, SRS\_BSW\_00386, SRS\_BSW\_00387, SRS\_BSW\_00409, SRS\_BSW\_00417, BSW00420, BSW00421, SRS\_BSW\_00422, SRS\_BSW\_00423, SRS\_BSW\_00424, SRS\_BSW\_00425, SRS\_BSW\_00426, SRS\_BSW\_00427, SRS\_BSW\_00428, SRS\_BSW\_00429, BSW00431, SRS\_BSW\_00432, SRS\_BSW\_00433, BSW00434, SRS\_Fim\_04721)