Document Title	Specification of Function Inhibition Manager
Document Category	Software specification (SWS)
Document UID	82
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• New mechanism in FIM added to track and store the specific diagnostic events that cause the inhibition of a functionality (FID). Events are stored in the order of detection. • Data Type for CS Interface FunctionInhibitionCallback corrected to FiM_FidStatusChangeType • Functionality FiM_GetVersionInfo completely removed • Editorial changes: Added missing links, Restructure of Chapter 8 regarding Function Definitions and Callback notifications, Removed Chapter 7.3.4 Extende Production Errors
2025-11-27	R25-11	AUTOSAR Release Management	• No content changes • Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• [ECUC_FiM_00613] added • [ECUC_FiM_00614] added • [SWS_Fim_00109] added • [SWS_Fim_00110] added • [SWS_Fim_91001] added • [SWS_Fim_91002] added • [SWS_Fim_91003] added • [SWS_Fim_91004] added • [SWS_Fim_00101] changed • [ECUC_FiM_00096] description of enumeration changed

△

			△ • [ECUC_FiM_00039] changed/extended • Editorial changes
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes • Editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	• No content changes • Editorial changes
2021-11-25	R21-11	AUTOSAR Release Management	• SWS_Fim_CONSTR_0001 changed to SWS_Fim_CONSTR_00001
2020-11-30	R20-11	AUTOSAR Release Management	• requirement SWS_Fim_00010 degraded to explanatory description • requirement SWS_Fim_00062 removed
2019-11-28	R19-11	AUTOSAR Release Management	• No content changes • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Editorial changes • corrections regarding Dem and Fim interaction during start-up
2017-12-08	4.3.1	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	• Renaming of Event Status to Monitor Status following redesign of Dem/ DCM interface • Changed usage of Dem_GetEventStatus to Dem_GetMonitorStatus and renamed FiM_DemTriggerOnEventStatus to FiM_DemTriggerOnMonitorStatus Interfaces following redesign of Dem/ DCM interface • Removed requirement SWS_Fim_00073 • minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation

▽

△

2015-07-31	4.2.2	AUTOSAR Release Management	• Fim considers EventAvailbilty/ EventSuppression • Modified Initialization Sequence • minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2014-10-31	4.2.1	AUTOSAR Release Management	• Simplification of FiM configuration • Support of "Monitored Components" • Postbuild configuration clean up • Editorial changes
2014-03-31	4.1.3	AUTOSAR Release Management	• Revised development error codes • Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Change containers FiMFID and FiMInhibitationConfiguration • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Apply new requirement format and requirement IDs (leading zeros to reach 5 digits) • Move general requirements to AUTOSAR_SWS_BSWGeneral [1] • 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.
2011-12-22	4.0.3	AUTOSAR Administration	• Renaming of FiMCyclicEventEvaluation configuration parameter into FiMEventUpdateTriggeredByDem • Reformulation of [SWS_Fim_00070], SWS_Fim_00073 • Inhibition masks use TestFailed bit instead of TestFailedThisOperationCycle • File structure schema changed • Initialization sequence diagram added • Remove development error FIM_E_EVENTID_OUT_OF_RANGE

▽

△

2010-09-30	3.1.5	AUTOSAR Administration	• Intra module checks updated • Corrected multiplicity of configuration parameters FiMInhChoicedemRef and FiMInhChoiceSumRef • Introduction of ImplementationDataType replacing IntegerType and Boolean • Clarification of chapter describing interaction between Dem and FiM 7.2.2.3 • Relocation of [SWS_Fim_00067] explaining evaluation by the FiM of Dem events • Addition of a new requirement describing the standardized AUTOSAR interface [SWS_Fim_00090]
2010-02-02	3.1.4	AUTOSAR Administration	• Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• OBD related chapter added 7.2.3 • Corrected error description • Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• Error classification extended to report invocation with NULL pointer • Corrected InternalBehavior of FiM to fit to API's reentrant behavior • Minimum value of parameter FimMaxSummaryLinks fixed • Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• "Advice for users" revised • "Revision Information" added

▽

△

	2.1.14	AUTOSAR Administration	• Modification of the FiM data structure: Several summarized events can be assigned to the FimInhibition-Configuration • Inserted corrected sequence charts for FiM initialization phase and FiM_DemTriggerOnEventStatus • Added file MemMap.h to header file structure • Added requirement for extended header file structure (Schedule Manager) • Added SchM_FiM.h to header file structure • Legal disclaimer revised
2006-05-16	2.0	AUTOSAR Administration	• Initial Release

Disclaimer

This work (specification and/or software implementation) 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 work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work 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 work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 10
2 Acronyms and abbreviations 11
3 Related documentation 13
3.1 Input documents . . . 13
3.2 Related standards and norms . . . 14
3.3 Related specification . . . 14
4 Constraints and assumptions 15
4.1 Limitations . . . 15
4.2 Applicability to car domains . . . 15
5 Dependencies on other modules 17
5.1 Requirements . . . 17
5.1.1 Use Cases . . . 18
6 Requirements traceability 19
7 Functional specification 21
7.1 Background & Rationale . . . 21
7.2 Requirements . . . 21
7.2.1 FiM core variables . . . 21
7.2.1.1 Definition of 'Diagnostic Event' . . . 21
7.2.1.2 Definition of 'Monitor Status' . . . 21
7.2.1.3 Definition of 'Monitored Component' . . . 21
7.2.1.4 Definition of 'Summarized Event' . . . 22
7.2.1.5 Definition of 'Function Identifier' . . . 22
7.2.1.6 Definition of 'Function Identifier permission state' . . . 24
7.2.2 FiM core functionalities . . . 25
7.2.2.1 Initialization . . . 25
7.2.2.2 FiM Data Structure . . . 26
7.2.2.3 Interaction between Dem and Function Inhibition Manager (FiM) 28
7.2.2.4 Interaction between SW-Components and Function Inhibition Manager (FiM) . . . 31
7.2.2.5 Application example for FiM usage . . . 33
7.2.3 OBD-Functionality . . . 33
7.2.3.1 In-Use-Monitor Performance Ratio (IUMPR) Support . . . 33
7.3 Error classification . . . 34
7.3.1 Development Errors . . . 34
7.3.2 Runtime Errors . . . 34
7.3.3 Production Errors . . . 34
7.4 Configuration Constraints . . . 34

8 API specification 36
8.1 Imported types . . . 36
8.2 Type definitions . . . 36
8.2.1 FiM_ConfigType . . . 36
8.3 Function definitions . . . 36
8.3.1 Interface ECUState Manager <-> FiM . . . 37
8.3.1.1 FiM_Init . . . 37
8.3.2 Interface SW-Components <-> FiM . . . 38
8.3.2.1 FiM_GetFunctionPermission . . . 38
8.3.2.2 FiM_ SetFunctionAvailable . . . 39
8.3.2.3 FiM_GetInhibitingEvents . . . 39
8.3.3 Interface Dem <-> FiM . . . 40
8.3.3.1 FiM_DemTriggerOnMonitorStatus . . . 40
8.3.3.2 FiM_ DemTriggerOnComponentStatus . . . 41
8.3.3.3 FiM_DemInit . . . 42
8.3.4 Callback notifications . . . 42
8.3.4.1 CBFidStatusChanged . . . 43
8.3.5 Scheduled functions . . . 43
8.3.5.1 FiM_MainFunction . . . 43
8.3.6 Expected Interfaces . . . 44
8.3.6.1 Mandatory Interfaces . . . 44
8.3.6.2 Optional Interfaces . . . 45
8.4 Service interfaces . . . 45
8.4.1 Client-Server-Interfaces . . . 45
8.4.1.1 FunctionInhibition . . . 45
8.4.1.2 ControlFunctionAvailable . . . 46
8.4.1.3 FunctionInhibitionCallback . . . 47
8.4.2 Implementation Data Types . . . 48
8.4.2.1 FiM_FunctionIdType . . . 48
8.4.2.2 FiM_EventIdArrayType . . . 48
8.4.2.3 FiM_FidStatusChangeType . . . 49
8.4.3 Ports . . . 49
8.4.4 Internal Behavior . . . 50
9 Sequence diagrams 51
9.1 Initialization sequence of FiM . . . 51
9.2 FiM_DemTriggerOnMonitorStatus . . . 51
10 Configuration specification 53
10.1 How to read this chapter . . . 53
10.2 Containers and configuration parameters . . . 53
10.2.1 FiM . . . 53
10.2.2 FiMGeneral . . . 54
10.2.3 FiMConfigSet . . . 59

10.2.4 FiMFID 60
10.2.5 FiMInhibitionConfiguration 61
10.2.6 FiMSummaryEvent 64
10.2.7 FiMCallbackFIDStatusChanged 66
10.3 Published Information 67
A Not applicable requirements 68
B Change history of AUTOSAR traceable items 69
B.1 Traceable item history of this document according to AUTOSAR Release R26-11 69
B.1.1 Added Specification Items in R26-11 69
B.1.2 Changed Specification Items in R26-11 69
B.1.3 Deleted Specification Items in R26-11 70
B.1.4 Added Constraints in R26-11 70
B.1.5 Changed Constraints in R26-11 70
B.1.6 Deleted Constraints in R26-11 70
B.2 Traceable item history of this document according to AUTOSAR Release R25-11 70
B.2.1 Added Specification Items in R25-11 70
B.2.2 Changed Specification Items in R25-11 70
B.2.3 Deleted Specification Items in R25-11 71
B.2.4 Added Constraints in R25-11 71
B.2.5 Changed Constraints in R25-11 71
B.2.6 Deleted Constraints in R25-11 71
B.3 Traceable item history of this document according to AUTOSAR Release R24-11 71
B.3.1 Added Specification Items in R24-11 71
B.3.2 Changed Specification Items in R24-11 72
B.3.3 Deleted Specification Items in R24-11 72
B.3.4 Added Constraints in R24-11 72
B.3.5 Changed Constraints in R24-11 72
B.3.6 Deleted Constraints in R24-11 72
B.4 Traceable item history of this document according to AUTOSAR Release R23-11 72
B.4.1 Added Specification Items in R23-11 72
B.4.2 Changed Specification Items in R23-11 72
B.4.3 Deleted Specification Items in R23-11 73
B.5 Traceable item history of this document according to AUTOSAR Release R22-11 73
B.5.1 Added Specification Items in R22-11 73
B.5.2 Changed Specification Items in R22-11 73
B.5.3 Deleted Specification Items in R22-11 73

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

Abbreviation / Acronym:	**Description:**
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 7.2.1.6.
API	Application Programming Interface
BSW	Basic Software
Dem	Diagnostic Event Manager
ECU	Electronic Control Unit
FID	Function Identifier
FiM	Function Inhibition Manager
Functionality	Functionality comprises User-visible and User-non-visible functional aspects of a system (see [2]). 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.
HW	Hardware
ID	Identification/Identifier
Inhibition Condition	The relation between one FID, an inhibition mask and the status of a Dem event/component. (see FiMInhibitionConfiguration)
ISO	International Standardization Organization
MIL	Malfunction Indication Light
Monitoring function	• Part of the Software Component. • Mechanism to monitor and finally to detect a fault of a certain sensor, actuator or could be a plausibility check • Reports states about events from internal processing of a SW-C or from further processing of return values of other basic software modules. • See also AUTOSAR_SWS_DiagnosticEventManager [3]
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 7.2.1.6.
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



△

Abbreviation / Acronym:	Description:
WP	Autosar Work Package
Xxx_	Placeholder for an API provider

Table 2.1: Abbreviations and Acronyms

3 Related documentation

3.1 Input documents

[1] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] Specification of Diagnostic Event Manager
AUTOSAR_CP_SWS_DiagnosticEventManager

[4] Requirements on Function Inhibition Manager
AUTOSAR_CP_RS_FunctionInhibitionManager

[5] Virtual Functional Bus
AUTOSAR_CP_TR_VFB

[6] Software Component Template
AUTOSAR_CP_TPS_SoftwareComponentTemplate

[7] Basic Software Module Description Template
AUTOSAR_CP_TPS_BSWModuleDescriptionTemplate

[8] Diagnostic Extract Template
AUTOSAR_CP_TPS_DiagnosticExtractTemplate

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 [1, 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]

Upstream requirements: SRS_Fim_04701

⌈FID numbers shall be unique per FiM.⌋

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 constrains 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]

Upstream requirements: SRS_Fim_04706

⌈The FiM shall compute the permission of a FID independently of the state of other FIDs.⌋

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]

Upstream requirements: SRS_BSW_00383

⌈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.⌋

- 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 monitor 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 [5] has to be considered as an additional requirement.
- For the formal description of the SW-C attributes [6] 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 ECUState Manager for initialization and shutdown) or need to query the FiM for execution permission themselves.

6 Requirements traceability

Requirement	Description	Satisfied by
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_Fim_00077]
[SRS_BSW_00172]	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	[SWS_Fim_00070] [SWS_Fim_00099]
[SRS_BSW_00304]	All AUTOSAR Basic Software Modules shall use only AUTOSAR data types instead of native C data types	[SWS_Fim_00027]
[SRS_BSW_00305]	Data types naming convention	[SWS_Fim_00027]
[SRS_BSW_00310]	API naming convention	[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_00331]	All Basic Software Modules shall strictly separate error and status information	[SWS_Fim_00015]
[SRS_BSW_00337]	Classification of development errors	[SWS_Fim_00076]
[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_00357]	For success/failure of an API call a standard return type shall be defined	[SWS_Fim_00097]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_Fim_00006] [SWS_Fim_00045] [SWS_Fim_00059] [SWS_Fim_00077]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[SWS_Fim_00060]
[SRS_BSW_00377]	A Basic Software Module can return a module specific types	[SWS_Fim_00027]
[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_00044] [SWS_Fim_00080] [SWS_Fim_00081]
[SRS_BSW_00384]	The Basic Software Module specifications shall specify at least in the description which other modules they require	[SWS_Fim_00080] [SWS_Fim_00081] [SWS_Fim_00099]
[SRS_BSW_00385]	List possible error notifications	[SWS_Fim_00076]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_Fim_00076]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_Fim_00092]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[SWS_Fim_00092]



△

Requirement	Description	Satisfied by
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_Fim_00045] [SWS_Fim_00055] [SWS_Fim_00056] [SWS_Fim_00057] [SWS_Fim_00058] [SWS_Fim_00059] [SWS_Fim_00104] [SWS_Fim_00124] [SWS_Fim_00126]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_Fim_00077]
[SRS_BSW_00416]	The sequence of modules to be initialized shall be configurable	[SWS_Fim_00018]
[SRS_BSW_00461]	Modules called by generic modules shall satisfy all interfaces requested by the generic module	[SWS_Fim_00079]
[SRS_BSW_00480]	Null pointer errors shall follow a naming rule	[SWS_Fim_00076]
[SRS_Fim_04700]	Information about the FID state	[SWS_Fim_00011] [SWS_Fim_00025] [SWS_Fim_00066] [SWS_Fim_00090] [SWS_Fim_00094] [SWS_Fim_00109] [SWS_Fim_00110] [SWS_Fim_91001] [SWS_Fim_91002] [SWS_Fim_91003] [SWS_Fim_91004]
[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] [SWS_Fim_00098] [SWS_Fim_00101]
[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] [SWS_Fim_00065]
[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_00018] [SWS_Fim_00069] [SWS_Fim_00082] [SWS_Fim_00102]
[SRS_Fim_04713]	Methods for the computation of permission states shall be provided	[SWS_Fim_00009] [SWS_Fim_00015] [SWS_Fim_00020] [SWS_Fim_00098]
[SRS_Fim_04717]	The permission states shall be updated	[SWS_Fim_00021] [SWS_Fim_00022] [SWS_Fim_00067] [SWS_Fim_00100]
[SRS_Fim_04719]	Mechanism for summarized diagnostic event states shall be provided	[SWS_Fim_00061] [SWS_Fim_00064] [SWS_Fim_00099]
[SRS_Fim_04723]	The FIM shall provide a boolean configuration option per FID.	[SWS_Fim_00105] [SWS_Fim_00106] [SWS_Fim_00107] [SWS_Fim_00108]
[SRS_Fim_04724]	The FIM shall provide a mechanism to track the events that cause function inhibitions.	[SWS_FiM_91005] [SWS_FiM_91006] [SWS_Fim_00120] [SWS_Fim_00121] [SWS_Fim_00122] [SWS_Fim_00123] [SWS_Fim_00125]

Table 6.1: Requirements Tracing

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.

See AUTOSAR_SWS_DiagnosticEventManager document for further details [3].

7.2.1.2 Definition of 'Monitor Status'

A 'monitor status' is the status calculated by the Dem according to the reported values of monitor functions. Possible values are defined by Dem_MonitorStatusType.

See AUTOSAR_SWS_DiagnosticEventManager document for further details [3].

7.2.1.3 Definition of 'Monitored Component'

A 'Monitored Component' is an identifier provided by the Dem to a specific monitored component (hardware component or signal). The FAILED status of a 'monitored component' represents the result of all assigned monitoring functions and inherited failure information from other DemComponents.

See AUTOSAR_SWS_DiagnosticEventManager document for further details [3].

7.2.1.4 Definition of 'Summarized Event'

[SWS_Fim_00061]

Upstream requirements: SRS_Fim_04719

⌈The FiM configuration shall support summarizing events. A summarized event consists of multiple single diagnostic events.⌋

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]

Upstream requirements: SRS_Fim_04719

⌈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.10.2.3). 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 (`FiMSummaryEvent`), a group of events can be reused for several inhibition configurations, by selecting it as `FiMInhSumRef`. This may simplify configuration.

7.2.1.5 Definition of 'Function Identifier'

The Fim implements the calculation of function permissions. Object to those calculations are SW-Components or logical units, which receive the information "Permission granted" / "permission denied".

To address those components, these have to be configured in FIM and a Function Identifier is assigned to address them via interfaces.

[SWS_Fim_00002]

Upstream requirements: SRS_Fim_04701

⌈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]).⌋

[SWS_Fim_00003]

Upstream requirements: SRS_Fim_04701

⌈The FiM module's environment shall use the FunctionId to directly point to the associated functionality information (permission status etc.)⌋

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 7.1).

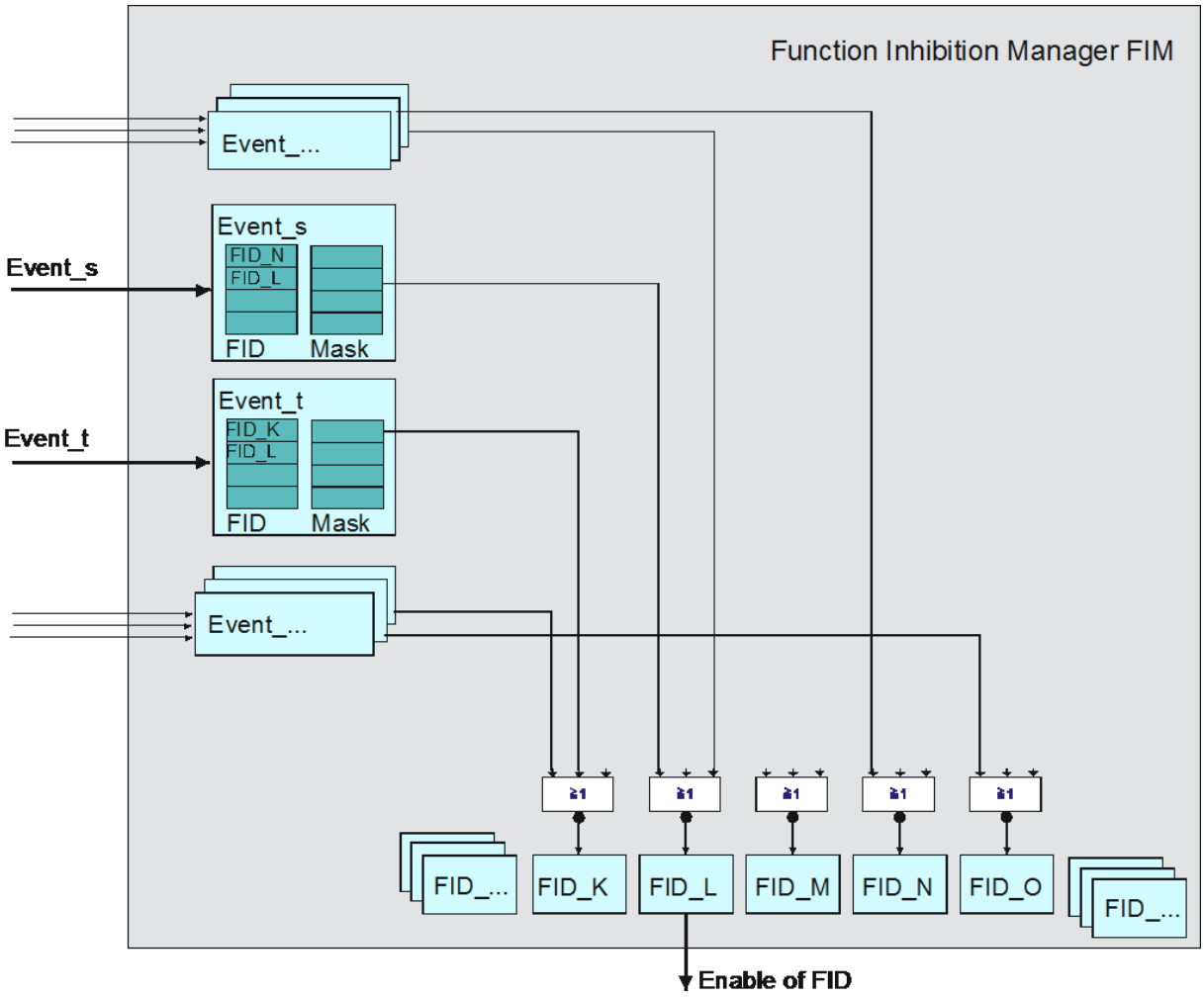

Figure 7.1: Logical information flow to determine FID permission states for an implementation with permission state stored in RAM

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 monitor 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.

7.2.1.6 Definition of 'Function Identifier permission state'

[SWS_Fim_00015]

Upstream requirements: SRS_BSW_00331, SRS_Fim_04713

⌈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.⌋

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]

Upstream requirements: SRS_Fim_04713

⌈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.⌋

7.2.2 FiM core functionalities

7.2.2.1 Initialization

[SWS_Fim_00069]

Upstream requirements: SRS_Fim_04712

⌈The function `FiM_DemInit` shall compute the permission state for all FIDs.⌋

[SWS_Fim_00082]

Upstream requirements: SRS_Fim_04712

⌈The function `FiM_DemInit` shall access the EventId states via the function `Dem_GetMonitorStatus` and the component information via `Dem_GetComponentFailed`.⌋

[SWS_Fim_00109] Initial FID state information callback

Upstream requirements: SRS_Fim_04700

⌈If the FiM initializes an initial FID state as defined in [SWS_Fim_00069] and the container `FiMCallbackFIDStatusChanged` is configured for this `FiMFID`, the FiM shall call the `CBFidStatusChanged` callback with `FiM_FidStatusChangeType` set to INIT and Permission set to the new initialized FID state. [SRS_Fim_04700]

The FiM will trigger the `CBFidStatusChanged` when the FID is initialized or the FiD states. The callback may be called at a time the destination component is yet not finalized by it's init function. It is up to the component to handle calls of `CBFidStatusChanged` at any time.⌋

[SWS_Fim_00018]

Upstream requirements: SRS_BSW_00416, SRS_Fim_04712

⌈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 monitor status information (not only events stored in the fault memory) of the Dem.⌋

The FiM is designed that it requires Dem monitor states during initialization. Therefore the Dem needs to ensure that at the point in time the FiM is initialized, the Dem is ready to provide monitor states via `Dem_GetMonitorStatus`. The Fim is not able to detect a not initialized Dem due to possible disabled events and will always behave as described in [SWS_Fim_00097].

[SWS_Fim_00102]

Upstream requirements: SRS_Fim_04712

⌈The initialization of Dem and Fim shall always follow the below order :

step 0) `Dem_PreInit`

step 1) Non-volatile memory data has to be available.

step 2) FiM_Init (setting up internal variables); after FiM_Init, the Fim is not yet ready to be used.

step 3) Dem_Init: do the internal DEM initialization and use FiM_DemInit to finally initialize the FIM⌋

Note: From step 3 onwards, the Dem and Fim are finally initialized and ready to be used.

[SWS_Fim_00104]

Upstream requirements: SRS_BSW_00406

⌈If FiM_GetFunctionPermission is called before the FiM is initialized, the FiM shall return E_NOT_OK.(FiM_DemInit).⌋

[SWS_Fim_00124] Initial state of FiM_GetInhibitingEvents

Upstream requirements: SRS_BSW_00406

⌈If FiM_GetInhibitingEvents is called before the FiM is initialized, the FiM shall return E_NOT_OK.(FiM_DemInit).⌋

7.2.2.2 FiM Data Structure

[SWS_Fim_00013]

Upstream requirements: SRS_BSW_00344, SRS_BSW_00345, SRS_Fim_04706

⌈The configuration process of the FiM shall create data structures within the FiM module to store the inhibit relations (EventID - FID - applicable mask).⌋

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_UdsStatusByteType). 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_UdsStatusByteType, it rather selects an algorithm to calculate the boolean inhibition condition from the Dem_UdsStatusByteType.

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.

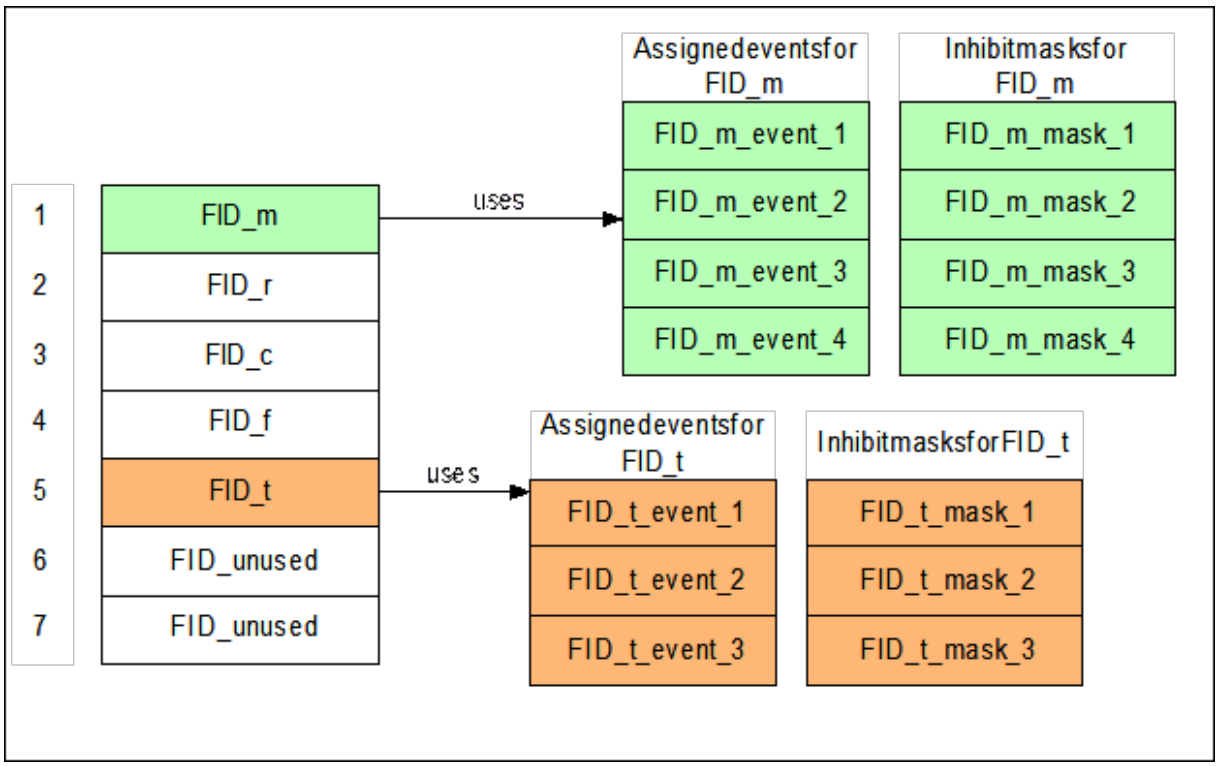

Figure 7.2: Inhibit Mask

[SWS_Fim_00008]

Upstream requirements: SRS_Fim_04706

⌈The FiM module shall provide the possibility to modify the inhibit conditions by post-built configuration.⌋

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.

7.2.2.3 Interaction between Dem and Function Inhibition Manager (FiM)

[SWS_Fim_00022]

Upstream requirements: SRS_Fim_04717

⌈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.⌋

[SWS_Fim_00065]

Upstream requirements: SRS_Fim_04706

⌈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 in the monitor status of a reported event, the Dem informs the FiM about the monitor status change via the API function `FiM_DemTriggerOnMonitorStatus`, if DemTriggerFiMReports is enabled.

On being informed about a monitor status change, the Fim uses the Api `Dem_GetMonitorStatus` to recalculate the function inhibitions.

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_DemTriggerOnMonitorStatus` 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_DemTriggerOnMonitorStatus` is without effect.

As an example of implementation, Figure 3 shows the calculation of a single Event Id-FID link. On the left hand side, the monitor 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 (monitor status is read through `Dem_GetMonitorStatus`), 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 monitor status change, the stored currently inhibiting EventIds (inhibition counter) shall be used for the computation for the permission state. If there is an

monitor status change reported by `FiM_DemTriggerOnMonitorStatus`, 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 monitor 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 monitor 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.

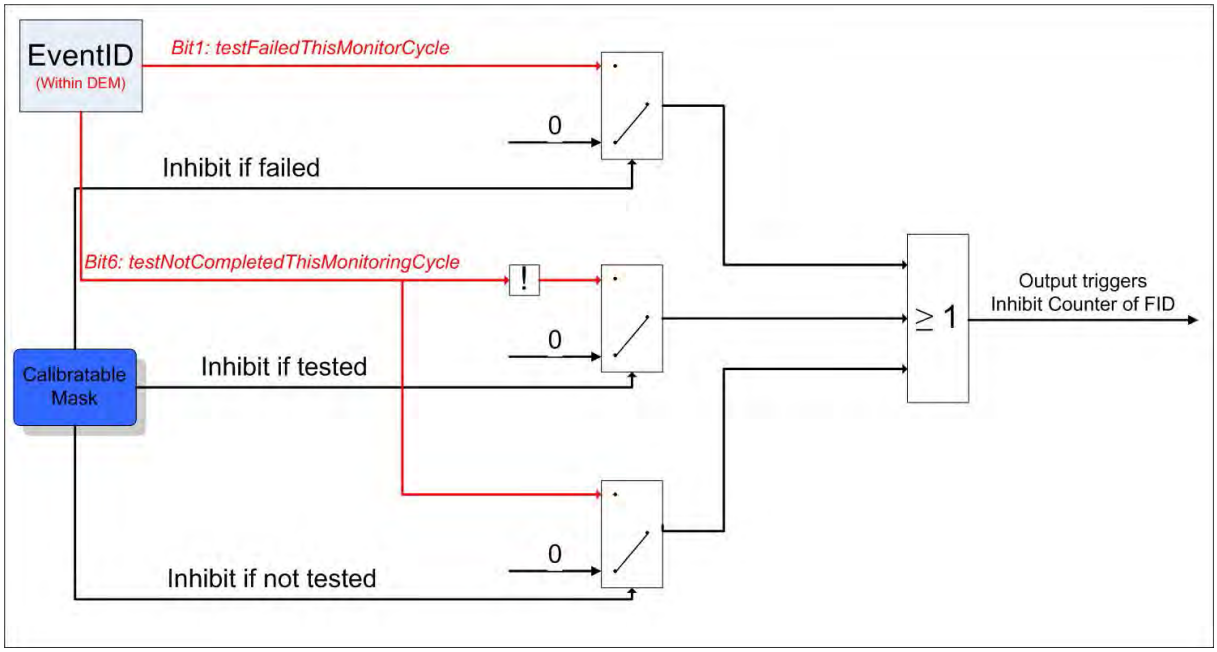

Figure 7.3: Calculation of permission state based on monitor status information

[SWS_Fim_00012]

Upstream requirements: SRS_Fim_04702

⌈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 Monitor 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.⌋

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".

[SWS_Fim_00098]

Upstream requirements: SRS_Fim_04713, SRS_Fim_04702

⌈The Function Inhibition Manager shall use the FID - DemComponentId - inhibition configuration to determine the permission state for the configured FID.

Upon changes of the FAILED status of a `DemComponent`, the function status shall be recalculated. Whenever the component status is FAILED (ComponentFailedStatus = TRUE), the FID is inhibited.⌋

[SWS_Fim_00099]

Upstream requirements: SRS_Fim_04719, SRS_BSW_00384, SRS_BSW_00172

⌈The FIM shall use the API `Dem_GetComponentFailed` to get the current FAILED status of a component.⌋

[SWS_Fim_00100]

Upstream requirements: SRS_Fim_04717

⌈If the FIM is configured for being triggered on eventStatus (FiMCyclicEventEvaluation), the FIM shall accept the status changed information of a `DemComponent` by providing the function `FiM_DemTriggerOnComponentStatus`.⌋

[SWS_Fim_00120] Storing the inhibiting EventId

Upstream requirements: SRS_Fim_04724

⌈If the status of a Dem_EventIdType changes such that it satisfies the configured inhibition mask for a specific FID (causing the FID to be inhibited), the FiM shall store the inhibiting events strictly in the order of their detection by FIM, placing each new EventId in the next available empty location in the array, starting from index 0 up to `FiMInhibitionTrackingDepth`-1. If `FiMInhibitionTracking` is enabled for this FID.⌋

[SWS_Fim_00121] Tracking Array Depth

Upstream requirements: SRS_Fim_04724

⌈The number of EventIds stored in the tracking array for a given FID is limited by the configuration parameter `FiMInhibitionTrackingDepth` [ECUC_FiM_00615]. This parameter shall be configured with care, as it may lead to significant CPU load and memory usage. If the array is full, any additional inhibiting events for that FID will be ignored and will not overwrite the existing entries.⌋

[SWS_Fim_00122] Displacement of the inhibiting EventId

Upstream requirements: SRS_Fim_04724

⌈If the status of a Dem_EventIdType changes so that it no longer fulfills the configured inhibition mask for a given FID, The FiM shall update the array for a given FID by removing the EventId, and the remaining elements shall be displaced accordingly if `FiMInhibitionTracking` is enabled for this FID.⌋

[SWS_Fim_00123] Initial values of the tracking array

Upstream requirements: SRS_Fim_04724

⌈When the FiM retrieves event states during `FiM_DemInit` and identifies events that already cause inhibition, these events shall be stored in the tracking array in the order of their detection by FIM if `FiMInhibitionTracking` is enabled for this FID.⌋

Note: In multi-partition or multi-core environments utilizing a Master-Satellite architecture, the exact mechanism for tracking inhibition events across different OS partitions is vendor-specific. Integrators should refer to the BSW vendor's technical documentation for implementation details and limitations.

Note: During Dem_FimInit the inhibiting events are calculated based on the monitor state provided by the Dem module. The order in which the events are stored within the inhibiting events array is undefined. This means that after a reset the order of the events can be different. Furthermore, in case there are more active inhibiting events than the buffer allows, different events can be stored. The first inhibiting event found is stored in this array and this is not necessarily the event that was the first event that caused the inhibition of the FID. The Dem has no general timestamp of event occurrence and therefore it is not possible to restore the exact same inhibiting order.

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

[SWS_Fim_00016]

Upstream requirements: SRS_Fim_04706

⌈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.⌋

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]

Upstream requirements: SRS_Fim_04713

⌈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.⌋

[SWS_Fim_00105]

Upstream requirements: SRS_Fim_04723

⌈If a function (FID) is set to not available using the interface `FiM_SetFunctionAvailable`, its permission state `FiM_GetFunctionPermission` shall always return FALSE.⌋

Polling the FID state from an application or SW-Component via the `FiM_GetFunctionPermission` can result in considerable CPU load, especially if a very large number of FIDs (e.g. 2000 or more) are used in the system and the FIDs are cyclically polled within a short time. In a typical vehicle operation, errors occur seldomly and the FID state mostly indicates a permission to run. So instead of polling the inhibition state frequently with always the same result (permission to run) it can be more efficient for an application to get triggered by a function state change. Herefore the FiM provides an optional callback mechanism that is triggered each time a FID state changes its value.

[SWS_Fim_00110] FID state change callback

Upstream requirements: SRS_Fim_04700

⌈If the state of an FID changes and the container `FiMCallbackFIDStatusChanged` is configured for this `FiMFID`, the FiM shall call the `CBFidStatusChanged` callback with `FiM_FidStatusChangeType` set to CHANGED and Permission set to the new FID state.⌋

The FID state change callback is provided via both: via C-Function or RTE call on a C/S interface. If `FiMCallbackFIDStatusChangedFnc` is not empty, the FiM uses this function name as C function callback with signature based on `CBFidStatusChanged`. If `FiMCallbackFIDStatusChangedFnc` is empty, the FiM uses the operation `CBFidStatusChanged` on the C/S Interface `FunctionInhibitionCallback` for the callback.

7.2.2.5 Application example for FiM usage

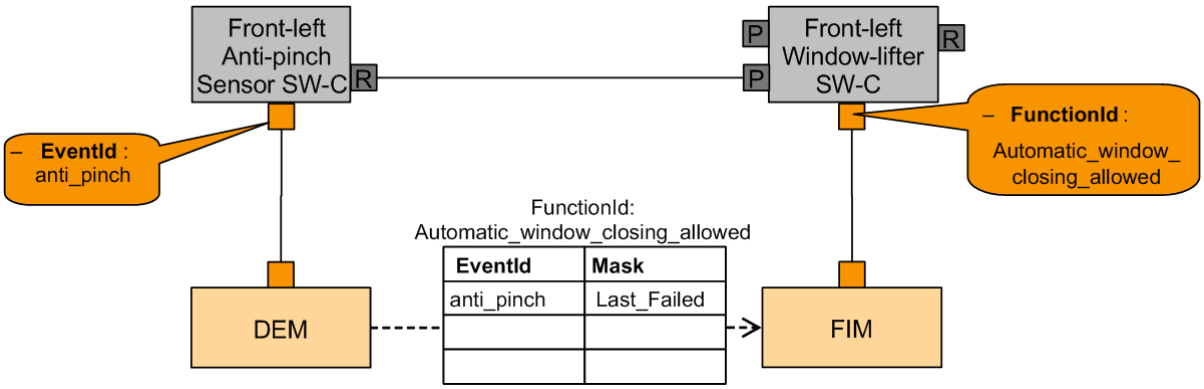

Figure 7.4: FiM usage

- 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.

7.2.3 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.

7.3 Error classification

Chapter [1, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.3.1 Development Errors

[SWS_Fim_00076] Definition of development errors in module FiM

Upstream requirements: SRS_BSW_00480, SRS_BSW_00386, SRS_BSW_00385, SRS_BSW_00337

⌈

Type of error	*Related error code*	*Error value*
API function called before the FiM module has been full initialized or after the FiM module has been shut down	FIM_E_UNINIT	0x01
FiM_GetFunctionPermission called with wrong FID	FIM_E_FID_OUT_OF_RANGE	0x02
Dem calls FiM with invalid EventId	FIM_E_EVENTID_OUT_OF_RANGE	0x03
API is invoked with NULL Pointer.	FIM_E_PARAM_POINTER	0x04
Invalid configuration set selection	FIM_E_INIT_FAILED	0x05

⌋

7.3.2 Runtime Errors

There are no runtime errors.

7.3.3 Production Errors

There are no productions errors.

7.4 Configuration Constraints

[SWS_Fim_CONSTR_00001] ⌈

For each configured FiMInhibitionConfiguration, at least one of FiMInhSumRef or FiMInhEventRef or FiMInhComponentRef shall be configured.⌋

[SWS_Fim_CONSTR_00002] Parameter FiMEventUpdateTriggeredByDem ⌈

For event inhibition tracking to be enabled (FiMInhibitionTrackingDepth > 0), the configuration parameter FiMEventUpdateTriggeredByDem shall be configured to TRUE.⌋

[SWS_Fim_CONSTR_00003] Multiple inhibition masks ⌈

For event inhibition tracking to be enabled (FiMInhibitionTrackingDepth > 0), configuring multiple inhibition masks for the same combination Dem_EventId and FiM_FunctionId is strictly prohibited.⌋

8 API specification

8.1 Imported types

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

[SWS_Fim_00081] Definition of imported datatypes of module FiM

Upstream requirements: SRS_BSW_00383, SRS_BSW_00384

⌈

Module	*Header File*	*Imported Type*
Dem	Dem.h	Dem_ComponentIdType
	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_MonitorStatusType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

8.2.1 FiM_ConfigType

[SWS_Fim_00092] Definition of datatype FiM_ConfigType

Upstream requirements: SRS_BSW_00404, SRS_BSW_00405

⌈

Name	FiM_ConfigType	
Kind	Structure	
Elements	--	
	Type	–
	Comment	implementation specific
Description	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 initializsation.	
Available via	FiM.h	

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 Interface ECUState Manager <-> FiM

8.3.1.1 FiM_Init

[SWS_Fim_00077] Definition of API function FiM_Init

Upstream requirements: SRS_BSW_00414, SRS_BSW_00358, SRS_BSW_00101

⌈

Service Name	FiM_Init	
Syntax	`void FiM_Init ( const FiM_ConfigType* FiMConfigPtr )`	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	FiMConfigPtr	–
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This service initializes the FIM.	
Available via	FiM.h	

⌋

Note: see Chapter 9.1

[SWS_Fim_00045]

Upstream requirements: SRS_BSW_00358, SRS_BSW_00406

⌈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.⌋

[SWS_Fim_00059]

Upstream requirements: SRS_BSW_00358, SRS_BSW_00406

⌈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.⌋

In order to restore the permission states quickly, it is recommended that the Dem provides direct access to monitor 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.

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 monitor status information.

8.3.2 Interface SW-Components <-> FiM

8.3.2.1 FiM_GetFunctionPermission

[SWS_Fim_00011] Definition of API function FiM_GetFunctionPermission

Upstream requirements: SRS_BSW_00310, SRS_BSW_00312, SRS_Fim_04700, SRS_Fim_04709

⌈

Service Name	FiM_GetFunctionPermission	
Syntax	`Std_ReturnType FiM_GetFunctionPermission ( FiM_FunctionIdType FID, boolean* Permission )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	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)
Parameters (inout)	None	
Parameters (out)	Permission	TRUE: FID has permission to run FALSE: FID has no permission to run, i.e. shall not be executed
Return value	Std_ReturnType	`E_OK`: The request is accepted `E_NOT_OK`: The request is not accepted, ie. initialization of FIM not completed
Description	This service reports the permission state to the functionality.	
Available via	FiM.h	

⌋

[SWS_Fim_00066]

Upstream requirements: SRS_Fim_04700

⌈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]

Upstream requirements: SRS_Fim_04700

⌈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]

Upstream requirements: SRS_BSW_00406

⌈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).⌋

[SWS_Fim_00056]

Upstream requirements: SRS_BSW_00406

⌈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).⌋

8.3.2.2 FiM_ SetFunctionAvailable

[SWS_Fim_00106] Definition of API function FiM_SetFunctionAvailable

Upstream requirements: SRS_Fim_04723

⌈

Service Name	FiM_SetFunctionAvailable	
Syntax	`Std_ReturnType FiM_SetFunctionAvailable ( FiM_FunctionIdType FID, boolean Availability )`	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	FID	Identification of a functionality by assigned FID.
	Availability	The permission of the requested FID: TRUE: Function is available. FALSE: Function is not available.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request is accepted `E_NOT_OK`: Request is not accepted (e.g. invalid FID is given)
Description	This service sets the availability of a function. The function is only available if FiMAvailability Support is configured as True.	
Available via	FiM.h	

⌋

8.3.2.3 FiM_GetInhibitingEvents

[SWS_FiM_91005] Definition of API function FiM_GetInhibitingEvents

Upstream requirements: SRS_Fim_04724

⌈

Service Name	FiM_GetInhibitingEvents
Syntax	`Std_ReturnType FiM_GetInhibitingEvents ( FiM_FunctionIdType FID, FiM_EventIdArrayType* EventIdBuffer )`

▽

△

Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different FID, non-reentrant for same FID	
Variation	({ecuc(FiM/FiMGeneral/FiMInhibitionTrackingDepth)} > 0)	
Parameters (in)	FID	Identification of the Functionality by assigned ID.
Parameters (inout)	None	
Parameters (out)	EventIdBuffer	Pointer to an array where the FiM will write the stored inhibiting Event IDs.
Return value	Std_ReturnType	`E_OK`: The request was successful. `E_NOT_OK`: The request failed (e.g., FIM not initalized, provided buffer not enough, invalid FID or null pointers).
Description	This service retrieves the list of Dem Event IDs that are currently causing the specified FID to be inhibited.	
Available via	FiM.h	

⌋

[SWS_Fim_00125] Usage of EventIdBuffer

Upstream requirements: SRS_Fim_04724

⌈When the API `FiM_GetInhibitingEvents` is called, the FiM shall copy the contents of the internal tracking array for the requested FID into the provided EventIdBuffer.⌋

[SWS_Fim_00126] Development error detection of FiM_GetInhibitingEvents

Upstream requirements: SRS_BSW_00406

⌈If development error detection for the module FiM is enabled: the function `FiM_GetInhibitingEvents` 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 list.⌋

8.3.3 Interface Dem <-> FiM

8.3.3.1 FiM_DemTriggerOnMonitorStatus

[SWS_Fim_00021] Definition of API function FiM_DemTriggerOnMonitorStatus

Upstream requirements: SRS_BSW_00310, SRS_BSW_00312, SRS_Fim_04717

⌈

Service Name	FiM_DemTriggerOnMonitorStatus

Syntax	`void FiM_DemTriggerOnMonitorStatus (`
	`  Dem_EventIdType EventId`
	`)`

Service ID [hex]	0x02
Sync/Async	Synchronous

▽

△

Reentrancy	Reentrant	
Parameters (in)	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)
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This service is provided to be called by the Dem in order to inform the Fim about monitor status changes.	
Available via	FiM_Dem.h	

⌋

[SWS_Fim_00057]

Upstream requirements: SRS_BSW_00406

⌈If development error detection for the module FiM is enabled: the function `FiM_DemTriggerOnMonitorStatus` 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`.⌋

[SWS_Fim_00058]

Upstream requirements: SRS_BSW_00406

⌈If development error detection for the module FiM is enabled: The function `FiM_DemTriggerOnMonitorStatus` shall check for complete initialization of the FiM. If the function detects that the initialization is not complete, it shall raise a development error.⌋

8.3.3.2 FiM_ DemTriggerOnComponentStatus

[SWS_Fim_00101] Definition of API function FiM_DemTriggerOnComponentStatus

Upstream requirements: SRS_Fim_04702

⌈

Service Name	FiM_DemTriggerOnComponentStatus	
Syntax	`void FiM_DemTriggerOnComponentStatus ( Dem_ComponentIdType ComponentId )`	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ComponentId	Identification of a DemComponent.
Parameters (inout)	None	

▽

△

Parameters (out)	None
Return value	None
Description	Triggers on changes of the component failed status.
Available via	FiM_Dem.h

⌋

8.3.3.3 FiM_DemInit

[SWS_Fim_00006] Definition of API function FiM_DemInit

Upstream requirements: SRS_BSW_00310, SRS_BSW_00358

⌈

Service Name	FiM_DemInit
Syntax	`void FiM_DemInit ( void )`
Service ID [hex]	0x03
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	This service re-initializes the FIM.
Available via	FiM_Dem.h

⌋

8.3.4 Callback notifications

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

8.3.4.1 CBFidStatusChanged

[SWS_Fim_91002] Definition of configurable interface CBFidStatusChanged

Upstream requirements: SRS_Fim_04700

⌈

Service Name	CBFidStatusChanged	
Syntax	`Std_ReturnType CBFidStatusChanged ( FiM_FidStatusChangeType Status, boolean Permission )`	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Status	Set to INIT if the FiM is calling this callback due to the initialisation of the module. If set to CHANGED, this callback indicates a FID state change
	Permission	Indicates the initial/new FID permission state. TRUE: FID has permission to run FALSE: FID has no permission to run.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: this function always returns `E_OK` for compatibility reasons with RTE
Description	–	
Available via	FiM.h	

⌋

8.3.5 Scheduled functions

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

8.3.5.1 FiM_MainFunction

[SWS_Fim_00060] Definition of scheduled function FiM_MainFunction

Upstream requirements: SRS_BSW_00373

⌈

Service Name	FiM_MainFunction
Syntax	`void FiM_MainFunction ( void )`
Service ID [hex]	0x05
Description	–
Available via	SchM_FiM.h

⌋

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

[SWS_Fim_00070]

Upstream requirements: SRS_BSW_00172

⌈If FiM module polls monitor status (as defined in configuration parameter `FiMEventUpdateTriggeredByDem` = 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_GetMonitorStatus` shall be used to get status information of EventIds.⌋

[SWS_Fim_00097]

Upstream requirements: SRS_BSW_00357

⌈If `Dem_GetMonitorStatus` returns E_NOT_OK, the FIM shall not consider this event in its inhibition mask calculation⌋

[SWS_Fim_00067]

Upstream requirements: SRS_Fim_04717

⌈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 monitor status information using the API `Dem_GetMonitorStatus` if Dem and FiM are implemented as separate modules. FiM shall access the monitor status structure of Dem if Dem and FiM are implemented as a bundle.⌋

8.3.6 Expected Interfaces

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

8.3.6.1 Mandatory Interfaces

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

[SWS_Fim_00079] Definition of mandatory interfaces required by module FiM

Upstream requirements: SRS_BSW_00461

⌈

API Function	*Header File*	*Description*
Dem_GetMonitorStatus	Dem.h	Gets the current monitor status for an event.
SchM_ActMainFunction_FiM	<none>	Invokes the SchM_ActMainFunction function to trigger the activation of a corresponding main processing function.

⌋

8.3.6.2 Optional Interfaces

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

[SWS_Fim_00080] Definition of optional interfaces requested by module FiM

Upstream requirements: SRS_BSW_00383, SRS_BSW_00384

⌈

API Function	*Header File*	*Description*
Det_ReportError	Det.h	Service to report development errors.

⌋

8.4 Service interfaces

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

8.4.1 Client-Server-Interfaces

8.4.1.1 FunctionInhibition

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

[SWS_Fim_00090] Definition of ClientServerInterface FunctionInhibition

Upstream requirements: SRS_Fim_04700

⌈

Name	FunctionInhibition		
Comment	The service interface is used by the SW-C to query the permission state of a specific FID and its corresponding inhibiting events.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	GetFunctionPermission	
Comment	Get the permission state of the respective FID.	
Relates to	FiM_GetFunctionPermission	
Variation	–	
Parameters	Permission	
	Type	boolean
	Direction	OUT

▽

△

	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
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetInhibitingEvents	
Comment	Retrieves the list of Dem_EventIdTypes that are currently inhibiting the FID.	
Relates to	FiM_GetInhibitingEvents	
Variation	({ecuc(FiM/FiMGeneral/FiMInhibitionTrackingDepth)} > 0)	
Parameters	EventIdBuffer	
	Type	FiM_EventIdArrayType
	Direction	OUT
	Comment	Pointer to an array where the FiM will write the stored inhibiting Event IDs.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋

8.4.1.2 ControlFunctionAvailable

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

[SWS_Fim_00107] Definition of ClientServerInterface ControlFunctionAvailable

Upstream requirements: SRS_Fim_04723

⌈

Name	ControlFunctionAvailable		
Comment	SW Components can use this service to set the availability of a function.		
IsService	true		
Variation	({ecuc(FiM/FiMGeneral/FiMAvailabilitySupport)} == True)		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	SetFunctionAvailable	
Comment	Sets the availability of a function.	
Relates to	FiM_SetFunctionAvailable	
Variation	–	
Parameters	Availability	
	Type	boolean
	Direction	IN

▽

△

	Comment	The permission of the requested FID: TRUE: Function is available. FALSE: Function is not available.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋

8.4.1.3 FunctionInhibitionCallback

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

[SWS_Fim_91004] Definition of ClientServerInterface FunctionInhibitionCallback

Upstream requirements: SRS_Fim_04700

⌈

Name	FunctionInhibitionCallback		
Comment	The SW Components can use this service to get informed on permission state changes of a FID.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful

Operation	CBFidStatusChanged	
Comment	–	
Relates to	CBFidStatusChanged	
Variation	–	
Parameters	Status	
	Type	FiM_FidStatusChangeType
	Direction	IN
	Comment	Set to INIT if the FiM is calling this callback due to the initialisation of the module. If set to CHANGED, this callback indicates a FID state change
	Variation	–
	Permission	
	Type	boolean
	Direction	IN
	Comment	Indicates the initial/new FID permission state. TRUE: FID has permission to run FALSE: FID has no permission to run.
	Variation	–
Possible Errors	E_OK	

⌋

8.4.2 Implementation Data Types

8.4.2.1 FiM_FunctionIdType

[SWS_Fim_00027] Definition of ImplementationDataType FiM_FunctionIdType

Upstream requirements: SRS_BSW_00304, SRS_BSW_00305, SRS_BSW_00377

⌈

Name	FiM_FunctionIdType		
Kind	Type		
Derived from	***Basetype***	***Variation***	
	uint16	platform depended	
	uint8	platform depended	
Range	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.
Description	Type for the FunctionID		
Variation	–		
Available via	Rte_FiM_Type.h		

⌋

8.4.2.2 FiM_EventIdArrayType

[SWS_FiM_91006] Definition of ImplementationDataType FiM_EventIdArrayType

Upstream requirements: SRS_Fim_04724

⌈

Name	FiM_EventIdArrayType		
Kind	Array	***Element type***	Dem_EventIdType
Size	FiMInhibitionTrackingDepth Elements		
Description	An array type used by the RTE to transport the list of inhibiting Dem Event IDs from the FiM to the application SWC. The size of the array is statically configured by FiMInhibitionTrackingDepth.		
Variation	–		
Available via	Rte_FiM_Type.h		

⌋

8.4.2.3 FiM_FidStatusChangeType

[SWS_Fim_91001] Definition of ImplementationDataType FiM_FidStatusChangeType

Upstream requirements: SRS_Fim_04700

⌈

Name	FiM_FidStatusChangeType		
Kind	Type		
Derived from	uint8		
Range	INIT	0x00	The callback is called triggered by the Fim initialisation.
	CHANGED	0x01	The callback is called triggered by a FID state change.
Description	This type is used to distinguish an initial FID state from a FID state change callback call.		
Variation	–		
Available via	Rte_FiM_Type.h		

⌋

8.4.3 Ports

[SWS_Fim_00094] Definition of Port Func_{Name} provided by module FiM

Upstream requirements: SRS_Fim_04700

⌈

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.		
Port Defined Argument Value(s)	***Type***	FiM_FunctionIdType	
	Value	{ecuc(FiM/FiMConfigSet/FiMFID/FiMFunctionId.value)}	
Variation	Name = {ecuc(FiM/FiMConfigSet/FiMFID.SHORT-NAME)}		

⌋

[SWS_Fim_00108] Definition of Port Control_{Name} provided by module FiM

Upstream requirements: SRS_Fim_04723

⌈

Name	Control_{Name}		
Kind	ProvidedPort	***Interface***	ControlFunctionAvailable
Description	A client can set the availability for a specific function.		

▽

△

Port Defined Argument Value(s)	***Type***	FiM_FunctionIdType
	Value	{ecuc(FiM/FiMConfigSet/FiMFID/FiMFunctionId.value)}
Variation	({ecuc(FiM/FiMGeneral/FiMAvailabilitySupport)} == True) Name = {ecuc(FiM/FiMConfigSet/FiMFID.SHORT-NAME)}	

⌋

[SWS_Fim_91003] Definition of Port CBFIDStatusChanged_{Name} required by module FiM

Upstream requirements: SRS_Fim_04700

⌈

Name	CBFIDStatusChanged_{Name}		
Kind	RequiredPort	***Interface***	FunctionInhibitionCallback
Description	–		
Variation	Name=(({ecuc(FiM/FiMConfigSet/FiMFID/FiMCallbackFIDStatusChanged)} != NULL) && ({ecuc(FiM/FiMConfigSet/FiMFID/FiMCallbackFIDStatusChanged/FiMCallbackFIDStatusChangedFnc)} == NULL)) Name = {ecuc(FiM/FiMConfigSet/FiMFID.SHORT-NAME)}		

⌋

8.4.4 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:

```
Internal Behavior 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 "FiMGetFunctionPermission"

canbeInvokedConcurrently = TRUE

}
```

9 Sequence diagrams

9.1 Initialization sequence of FiM

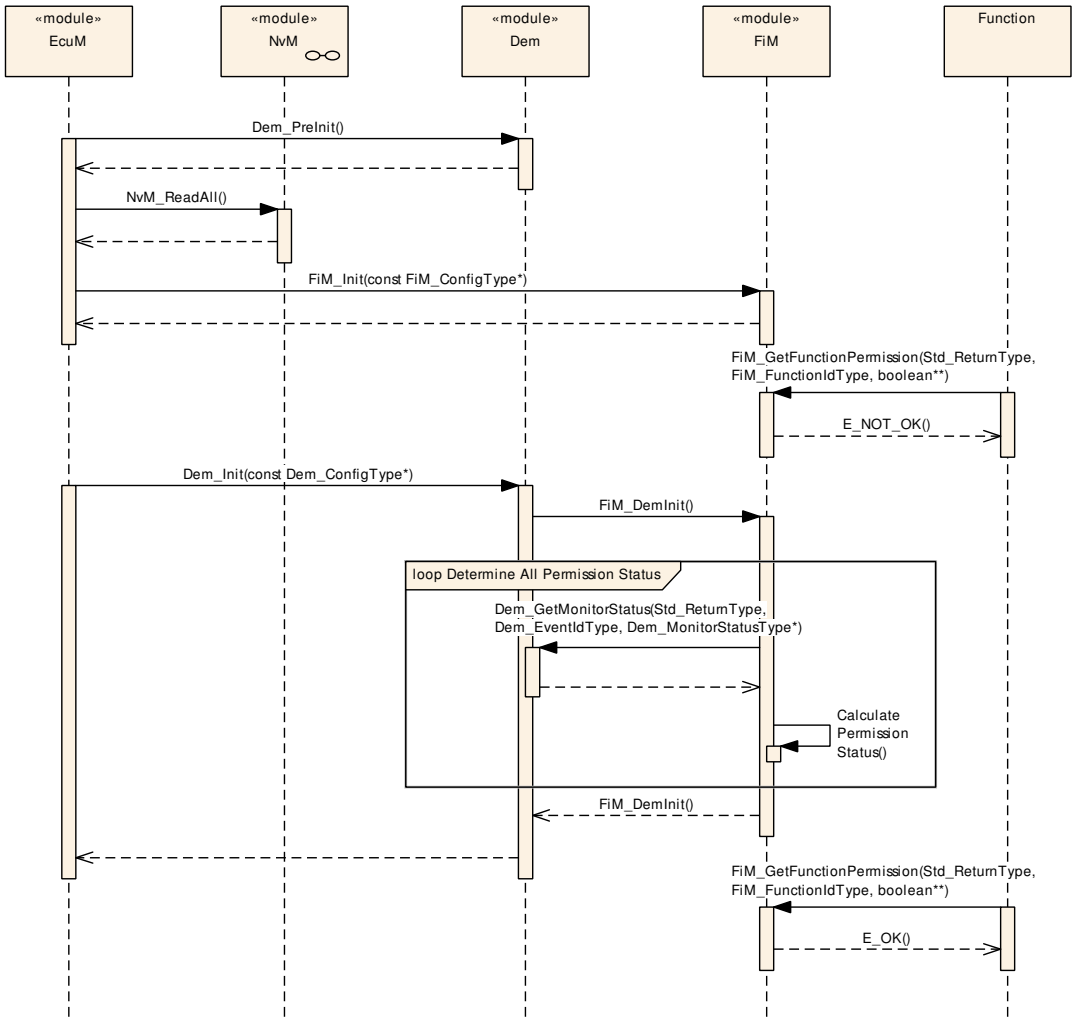

Figure 9.1: Initialization sequence of FiM

9.2 FiM_DemTriggerOnMonitorStatus

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

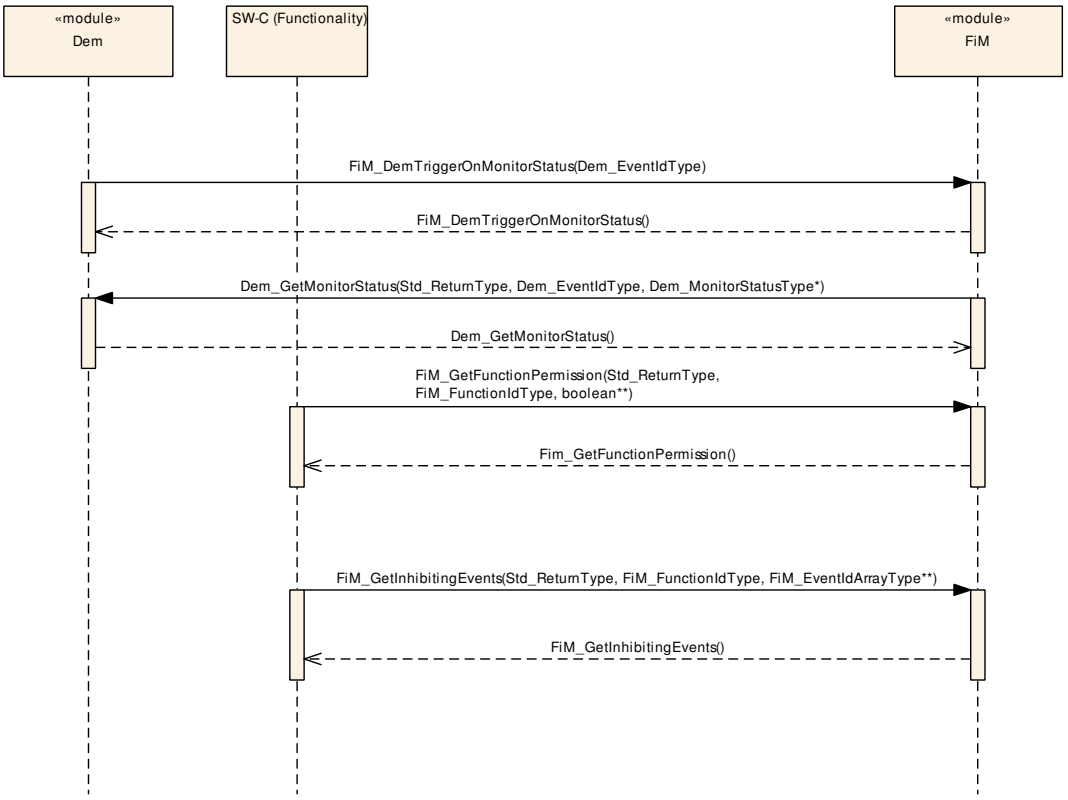

Figure 9.2: FiM_DemTriggerOnMonitorStatus

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 [1] Chapter 10.1 *"Introduction to configuration specification"*.

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 8.

10.2.1 FiM

[ECUC_FiM_00612] Definition of EcucModuleDef FiM ⌈

Module Name	FiM
Description	Configuration of the FiM (Function Inhibition Manager) module.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
FiMConfigSet	1	This container contains the configuration parameters and sub containers of the FiM module supporting multiple configuration sets.
FiMGeneral	1	Contains the general configuration parameters of the module.

⌋

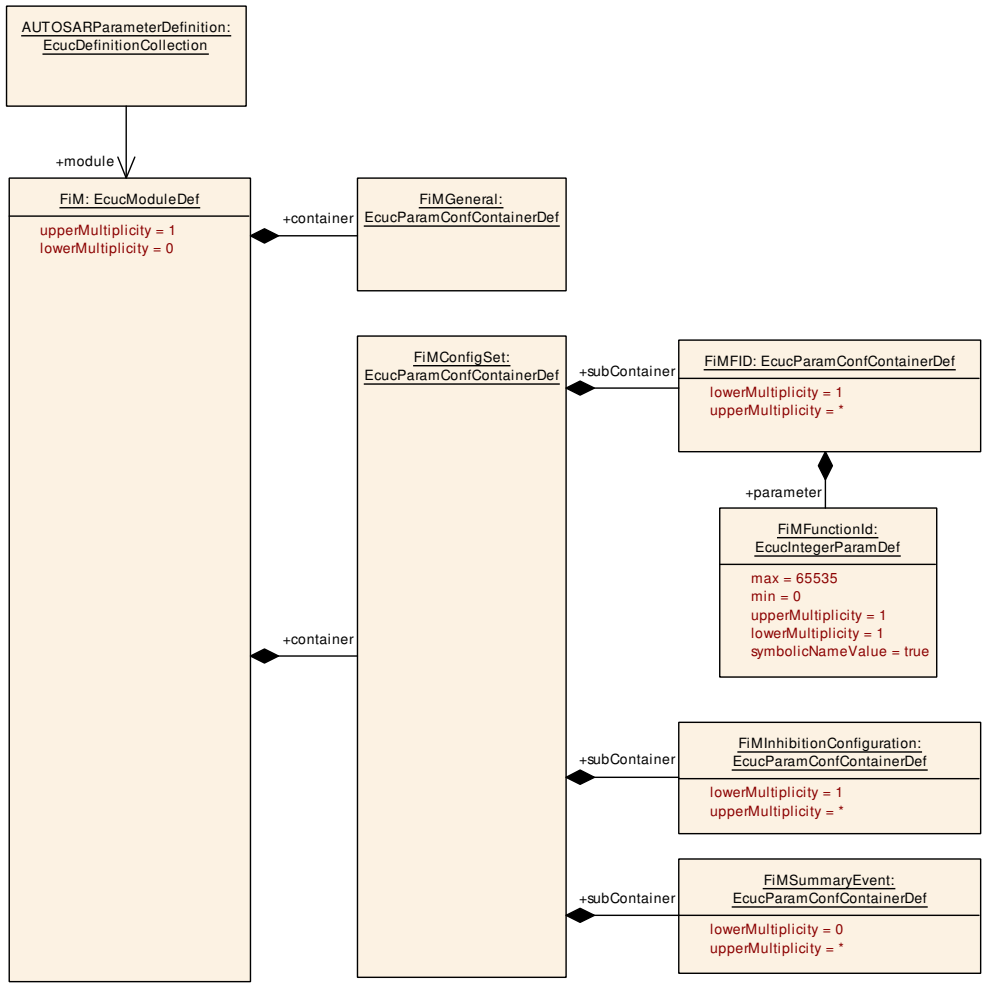

Figure 10.1: Configuration overview for FiM

10.2.2 FiMGeneral

[ECUC_FiM_00040] Definition of EcucParamConfContainerDef FiMGeneral ⌈

Container Name	FiMGeneral
Parent Container	FiM
Upstream mapping defined	–
Description	Contains the general configuration parameters of the module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FiMAvailabilitySupport	1	[ECUC_FiM_00610]
FiMDevErrorDetect	1	[ECUC_FiM_00087]
FiMEventUpdateTriggeredByDem	1	[ECUC_FiM_00086]
FiMInhibitionTrackingDepth	1	[ECUC_FiM_00615]
FiMMainFunctionPeriod	1	[ECUC_FiM_00611]
FiMMaxEventsPerFidInhibitionConfiguration	0..1	[ECUC_FiM_00608]
FiMMaxFiMInhibitionConfigurations	0..1	[ECUC_FiM_00606]
FiMMaxInputEventsPerSummaryEvents	0..1	[ECUC_FiM_00609]
FiMMaxSumEventsPerFidInhibitionConfiguration	0..1	[ECUC_FiM_00607]
FiMMaxSummaryEvents	1	[ECUC_FiM_00091]

No Included Containers

⌋

[ECUC_FiM_00610] Definition of EcucBooleanParamDef FiMAvailabilitySupport ⌈

Parameter Name	FiMAvailabilitySupport
Parent Container	FiMGeneral
Upstream mapping defined	–
Description	This configuration parameter specifies, if the Fim shall support the service to set the Availabity of a Funtionality. true: Service is supported. false: Service is not supported
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_FiM_00087] Definition of EcucBooleanParamDef FiMDevErrorDetect ⌈

Parameter Name	FiMDevErrorDetect
Parent Container	FiMGeneral
Upstream mapping defined	–
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_FiM_00086] Definition of EcucBooleanParamDef FiMEventUpdateTriggeredByDem ⌈

Parameter Name	FiMEventUpdateTriggeredByDem
Parent Container	FiMGeneral
Upstream mapping defined	–
Description	This configuration parameter specifies the way FIM obtains status of EventIds. TRUE: the DEM informs FIM about changes of monitor status, FALSE: the FIM polls monitor status from the DEM module either cyclically or on demand.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_FiM_00615] Definition of EcucIntegerParamDef FiMInhibitionTrackingDepth ⌈

Parameter Name	FiMInhibitionTrackingDepth
Parent Container	FiMGeneral
Upstream mapping defined	–
Description	Defines the maximum number of inhibiting Dem Event IDs that the FiM shall track and store (RAM) for all FIDs • A value of 0 means tracking is disabled • A value > 0 (e.g., x) allocates an array of size x to store the Event IDs.
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 255
Default value	0
Pre-Compile only	true

⌋

[ECUC_FiM_00611] Definition of EcucFloatParamDef FiMMainFunctionPeriod ⌈

Parameter Name	FiMMainFunctionPeriod
Parent Container	FiMGeneral
Upstream mapping defined	–
Description	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. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. FiM configuration tools shall convert this float value to the appropriate value format for the use in the software implementation of FiM.
Multiplicity	1
Type	EcucFloatParamDef
Range	]0 .. INF[
Default value	–
Pre-Compile only	true

⌋

[ECUC_FiM_00608] Definition of EcucIntegerParamDef FiMMaxEventsPerFidInhibitionConfiguration ⌈

Parameter Name	FiMMaxEventsPerFidInhibitionConfiguration	
Parent Container	FiMGeneral	
Upstream mapping defined	–	
Description	This configuration parameter specifies the total maximum number of inhibiting events in a FiMInhibitionConfiguration. Its applicable for post build configuration versions only and may be used to allocate the maximum size of memory to store and execute the configuration.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_FiM_00606] Definition of EcucIntegerParamDef FiMMaxFiMInhibitionConfigurations ⌈

Parameter Name	FiMMaxFiMInhibitionConfigurations	
Parent Container	FiMGeneral	
Upstream mapping defined	–	
Description	This configuration parameter specifies the total maximum number of FiMInhibition Configurations. Its applicable for post build configuration versions only and may be used to allocate the maximum size of memory to store and execute the configuration.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_FiM_00609] Definition of EcucIntegerParamDef FiMMaxInputEventsPerSummaryEvents ⌈

Parameter Name	FiMMaxInputEventsPerSummaryEvents	
Parent Container	FiMGeneral	
Upstream mapping defined	–	
Description	This configuration parameter specifies the total maximum number of input events per summary event. Its applicable for post build configuration versions only and may be used to allocate the maximum size of memory to store and execute the configuration.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_FiM_00607] Definition of EcucIntegerParamDef FiMMaxSumEventsPerFidInhibitionConfiguration ⌈

Parameter Name	FiMMaxSumEventsPerFidInhibitionConfiguration	
Parent Container	FiMGeneral	
Upstream mapping defined	–	
Description	This configuration parameter specifies the total maximum number of inhibiting summary events in a FiMInhibitionConfiguration. Its applicable for post build configuration versions only and may be used to allocate the maximum size of memory to store and execute the configuration.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_FiM_00091] Definition of EcucIntegerParamDef FiMMaxSummaryEvents ⌈

Parameter Name	FiMMaxSummaryEvents	
Parent Container	FiMGeneral	
Upstream mapping defined	–	
Description	This configuration parameter specifies the maximum number of summarized events that can be configured.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

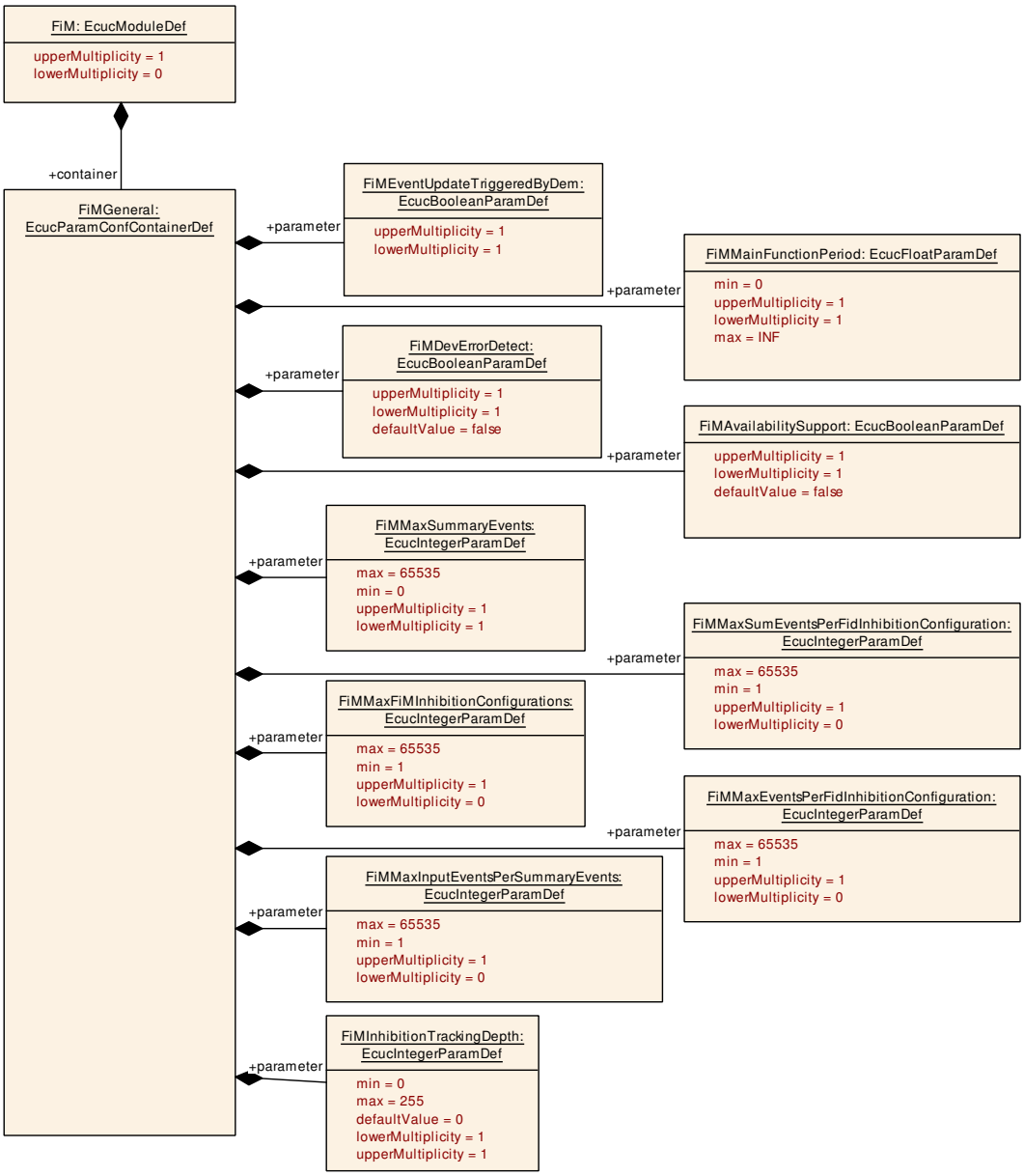

Figure 10.2: Configuration overview for FiMGeneral

10.2.3 FiMConfigSet

[ECUC_FiM_00601] Definition of EcucParamConfContainerDef FiMConfigSet ⌈

Container Name	FiMConfigSet
Parent Container	FiM
Upstream mapping defined	–
Description	This container contains the configuration parameters and sub containers of the FiM module supporting multiple configuration sets.

▽

△

Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
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.
FiMSummaryEvent	0..*	The summarized EventId definition record consists of a summarized event ID and specific Dem Events. 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.

⌋

10.2.4 FiMFID

[ECUC_FiM_00039] Definition of EcucParamConfContainerDef FiMFID ⌈

Container Name	FiMFID
Parent Container	FiMConfigSet
Upstream mapping defined	[7, CP_TPS_BSWModuleDescriptionTemplate], [8, CP_TPS_DiagnosticExtract Template]
Description	This container includes symbolic names of all FIDs.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FiMFunctionId	1	[ECUC_FiM_00085]
FiMInhibitionTracking	1	[ECUC_FiM_00616]

Included Containers		
Container Name	**Multiplicity**	**Description**
FiMCallbackFIDStatusChanged	0..1	The presence of this container sets up the callout function FIDStatusChanged for this FID.

⌋

[ECUC_FiM_00085] Definition of EcucIntegerParamDef FiMFunctionId ⌈

Parameter Name	FiMFunctionId
Parent Container	FiMFID
Upstream mapping defined	–
Description	Unique identifier of a FimFunctionId. This parameter should not be changeable by user, because the Id should be generated by Fim itself to prevent gaps and multiple use of an Id. Note: The implementer can add the attribute 'withAuto' to the parameter definition which indicates that the value can be calculated by the generator automatically. When 'withAuto' is set to 'true' for this parameter definition the 'isAutoValue' can be set to 'true'. If 'isAutoValue' is set to 'true' the actual value will not be considered during ECU Configuration but will be (re-)calculated by the code generator and stored in the value attribute afterwards.
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	–
Pre-Compile only	true

⌋

[ECUC_FiM_00616] Definition of EcucBooleanParamDef FiMInhibitionTracking ⌈

Parameter Name	FiMInhibitionTracking
Parent Container	FiMFID
Upstream mapping defined	–
Description	Defines if inhibition tracking is required for this FID • True: inhibition tracking is enabled • False: inhibition tracking is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

10.2.5 FiMInhibitionConfiguration

[ECUC_FiM_00038] Definition of EcucParamConfContainerDef FiMInhibitionConfiguration ⌈

Container Name	FiMInhibitionConfiguration
Parent Container	FiMConfigSet
Upstream mapping defined	[8, CP_TPS_DiagnosticExtractTemplate]
Description	This container includes all configuration parameters concerning the relationship between event and FID.

▽

△

Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FiMInhInhibitionMask	1	[ECUC_FiM_00096]
FiMInhComponentRef	0..*	[ECUC_FiM_00605]
FiMInhEventRef	0..*	[ECUC_FiM_00100]
FiMInhFunctionIdRef	1	[ECUC_FiM_00095]
FiMInhSumRef	0..*	[ECUC_FiM_00102]

No Included Containers

⌋

[ECUC_FiM_00096] Definition of EcucEnumerationParamDef FiMInhInhibitionMask ⌈

Parameter Name	FiMInhInhibitionMask	
Parent Container	FiMInhibitionConfiguration	
Upstream mapping defined	[8, CP_TPS_DiagnosticExtractTemplate]	
Description	The configuration parameter is used to specify the inhibition mask for an event - FID relation.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	FIM_LAST_FAILED	Last Failed - The monitor Status DEM_MONITOR_STATUS_TF is set to 1.
	FIM_NOT_TESTED	Not Tested this cycle - DEM_MONITOR_STATUS_TNCTOC flag is set.
	FIM_TESTED	Tested - DEM_MONITOR_STATUS_TNCTOC flag is not set.
	FIM_TESTED_AND_FAILED	Tested and Failed - DEM_MONITOR_STATUS_TF flag is set and DEM_MONITOR_STATUS_TNCTOC flag is not set.
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FiM_00605] Definition of EcucReferenceDef FiMInhComponentRef ⌈

Parameter Name	FiMInhComponentRef	
Parent Container	FiMInhibitionConfiguration	
Upstream mapping defined	–	
Description	Reference to a DemComponent which is necessary for function permission.	
Multiplicity	0..*	
Type	Reference to DemComponent	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FiM_00100] Definition of EcucReferenceDef FiMInhEventRef ⌈

Parameter Name	FiMInhEventRef	
Parent Container	FiMInhibitionConfiguration	
Upstream mapping defined	[8, CP_TPS_DiagnosticExtractTemplate]	
Description	Selection of an single DEM Event.	
Multiplicity	0..*	
Type	Symbolic name reference to DemEventParameter	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FiM_00095] Definition of EcucReferenceDef FiMInhFunctionIdRef ⌈

Parameter Name	FiMInhFunctionIdRef	
Parent Container	FiMInhibitionConfiguration	
Upstream mapping defined	–	
Description	Reference to the FID for an inhibit function	
Multiplicity	1	
Type	Reference to FiMFID	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FiM_00102] Definition of EcucReferenceDef FiMInhSumRef ⌈

Parameter Name	FiMInhSumRef	
Parent Container	FiMInhibitionConfiguration	
Upstream mapping defined	–	
Description	Selection of a summarized Event.	
Multiplicity	0..*	
Type	Reference to FiMSummaryEvent	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

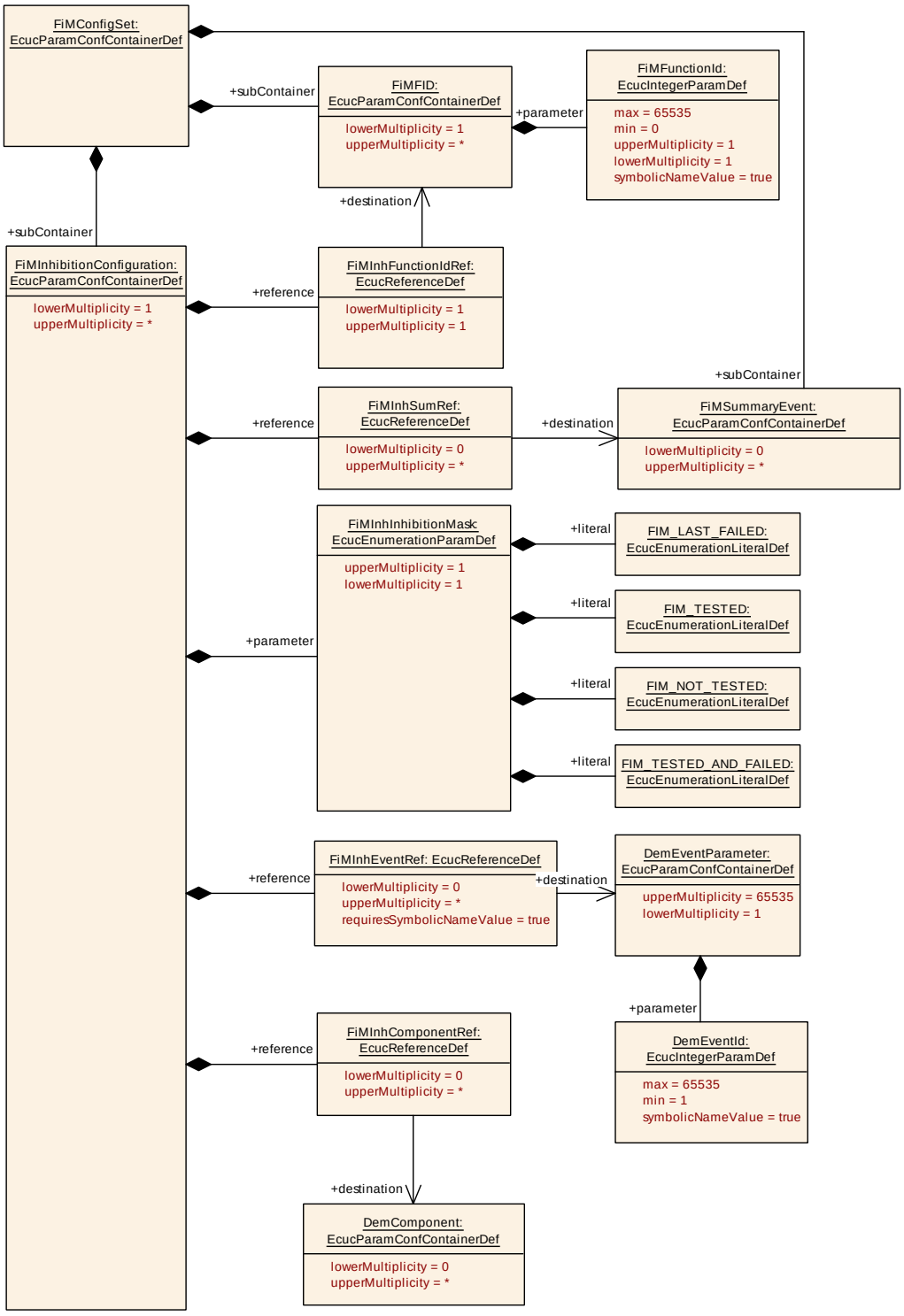

Figure 10.3: Configuration overview for FiMInhibitionConfiguration

10.2.6 FiMSummaryEvent

[ECUC_FiM_00603] Definition of EcucParamConfContainerDef FiMSummary Event ⌈

Container Name	FiMSummaryEvent	
Parent Container	FiMConfigSet	
Upstream mapping defined	[8, CP_TPS_DiagnosticExtractTemplate]	
Description	The summarized EventId definition record consists of a summarized event ID and specific Dem Events. 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.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FiMInputEventRef	1..*	[ECUC_FiM_00604]

No Included Containers

⌋

[ECUC_FiM_00604] Definition of EcucReferenceDef FiMInputEventRef ⌈

Parameter Name	FiMInputEventRef	
Parent Container	FiMSummaryEvent	
Upstream mapping defined	–	
Description	Reference to DemEventParameters combined to this summarized event.	
Multiplicity	1..*	
Type	Symbolic name reference to DemEventParameter	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

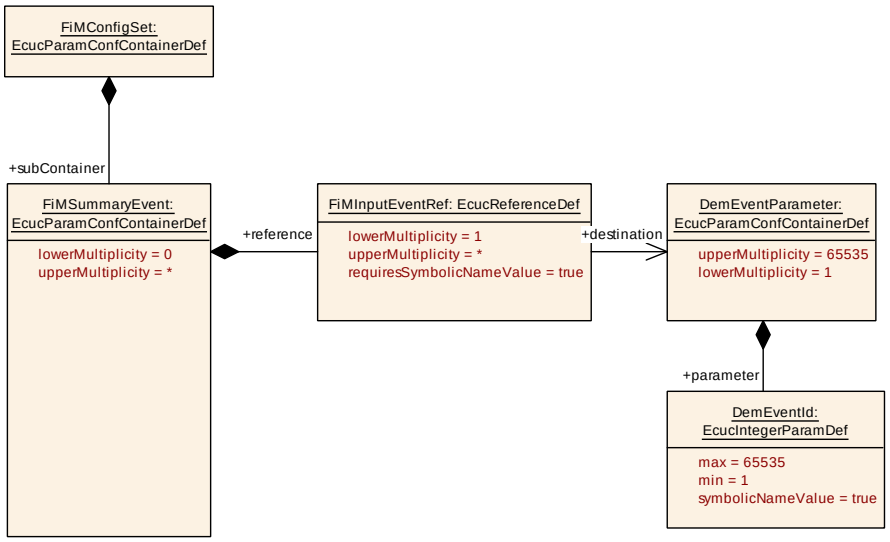

Figure 10.4: Configuration overview for FiMSummaryEvent

10.2.7 FiMCallbackFIDStatusChanged

[ECUC_FiM_00613] Definition of EcucParamConfContainerDef FiMCallbackFIDStatusChanged ⌈

Container Name	FiMCallbackFIDStatusChanged
Parent Container	FiMFID
Upstream mapping defined	–
Description	The presence of this container sets up the callout function FIDStatusChanged for this FID.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FiMCallbackFIDStatusChangedFnc	0..1	[ECUC_FiM_00614]

No Included Containers

⌋

[ECUC_FiM_00614] Definition of EcucFunctionNameDef FiMCallbackFIDStatusChangedFnc ⌈

Parameter Name	FiMCallbackFIDStatusChangedFnc
Parent Container	FiMCallbackFIDStatusChanged
Upstream mapping defined	–

▽

△

Description	If this parameter is configured, it defines the c callout function name that is called when the FID state changes. If this parameter is not configured, the operation CBFidStatus Changed on the C/S interface FiM_FunctionInhibition is called.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

10.3 Published Information

For details refer to [1] Chapter 10.3 *“Published Information”*.

A Not applicable requirements

[SWS_Fim_NA_00999]

Upstream requirements: SRS_BSW_00004, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00323, SRS_BSW_00336, SRS_BSW_00369, SRS_BSW_00375, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00403, SRS_BSW_00409, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00469, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00478, SRS_BSW_00490, SRS_BSW_00491, SRS_Fim_04721

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R26-11

B.1.1 Added Specification Items in R26-11

Number	Heading
[ECUC_FiM_00615]	Definition of EcucIntegerParamDef FiMInhibitionTrackingDepth
[ECUC_FiM_00616]	Definition of EcucBooleanParamDef FiMInhibitionTracking
[SWS_FiM_91005]	Definition of API function FiM_GetInhibitingEvents
[SWS_FiM_91006]	Definition of ImplementationDataType FiM_EventIdArrayType
[SWS_Fim_00120]	Storing the inhibiting EventId
[SWS_Fim_00121]	Tracking Array Depth
[SWS_Fim_00122]	Displacement of the inhibiting EventId
[SWS_Fim_00123]	Initial values of the tracking array
[SWS_Fim_00124]	Initial state of FiM_GetInhibitingEvents
[SWS_Fim_00125]	Usage of EventIdBuffer
[SWS_Fim_00126]	Development error detection of FiM_GetInhibitingEvents

Table B.1: Added Specification Items in R26-11

B.1.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_FiM_00039]	Definition of EcucParamConfContainerDef FiMFID
[ECUC_FiM_00040]	Definition of EcucParamConfContainerDef FiMGeneral
[SWS_Fim_00081]	Definition of imported datatypes of module FiM
[SWS_Fim_00090]	Definition of ClientServerInterface FunctionInhibition
[SWS_Fim_91001]	Definition of ImplementationDataType FiM_FidStatusChangeType
[SWS_Fim_91004]	Definition of ClientServerInterface FunctionInhibitionCallback

Table B.2: Changed Specification Items in R26-11

B.1.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_FiM_00094]	Definition of EcucBooleanParamDef FiMVersionInfoApi
[SWS_Fim_00078]	Definition of API function FiM_GetVersionInfo

Table B.3: Deleted Specification Items in R26-11

B.1.4 Added Constraints in R26-11

Number	Heading
[SWS_Fim_CONSTR_00002]	Parameter FiMEventUpdateTriggeredByDem
[SWS_Fim_CONSTR_00003]	Multiple inhibition masks

Table B.4: Added Constraints in R26-11

B.1.5 Changed Constraints in R26-11

none

B.1.6 Deleted Constraints in R26-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R25-11

B.2.1 Added Specification Items in R25-11

none

B.2.2 Changed Specification Items in R25-11

none

B.2.3 Deleted Specification Items in R25-11

none

B.2.4 Added Constraints in R25-11

none

B.2.5 Changed Constraints in R25-11

none

B.2.6 Deleted Constraints in R25-11

none

B.3 Traceable item history of this document according to AUTOSAR Release R24-11

B.3.1 Added Specification Items in R24-11

Number	Heading
[ECUC_FiM_00613]	Definition of EcucParamConfContainerDef FiMCallbackFIDStatusChanged
[ECUC_FiM_00614]	Definition of EcucFunctionNameDef FiMCallbackFIDStatusChangedFnc
[SWS_Fim_00109]	Initial FID state information callback
[SWS_Fim_00110]	FID state change callback
[SWS_Fim_91001]	Definition of datatype FiM_FidStatusChangeType
[SWS_Fim_91002]	Definition of configurable interface CBFidStatusChanged
[SWS_Fim_91003]	Definition of Port CBFIDStatusChanged_{Name} required by module FiM
[SWS_Fim_91004]	Definition of ClientServerInterface FunctionInhibitionCallback

Table B.5: Added Specification Items in R24-11

B.3.2 Changed Specification Items in R24-11

Number	Heading
[ECUC_FiM_00039]	Definition of EcucParamConfContainerDef FiMFID
[ECUC_FiM_00096]	Definition of EcucEnumerationParamDef FiMInhInhibitionMask
[SWS_Fim_00025]	
[SWS_Fim_00066]	
[SWS_Fim_00099]	
[SWS_Fim_00101]	Definition of API function FiM_DemTriggerOnComponentStatus

Table B.6: Changed Specification Items in R24-11

B.3.3 Deleted Specification Items in R24-11

none

B.3.4 Added Constraints in R24-11

none

B.3.5 Changed Constraints in R24-11

none

B.3.6 Deleted Constraints in R24-11

none

B.4 Traceable item history of this document according to AUTOSAR Release R23-11

B.4.1 Added Specification Items in R23-11

none

B.4.2 Changed Specification Items in R23-11

none

B.4.3 Deleted Specification Items in R23-11

none

B.5 Traceable item history of this document according to AUTOSAR Release R22-11

B.5.1 Added Specification Items in R22-11

[SWS_Fim_NA_00999]

B.5.2 Changed Specification Items in R22-11

[SWS_Fim_00006] [SWS_Fim_00011] [SWS_Fim_00021] [SWS_Fim_00027] [SWS_Fim_00060] [SWS_Fim_00076] [SWS_Fim_00077] [SWS_Fim_00078] [SWS_Fim_00079] [SWS_Fim_00080] [SWS_Fim_00081] [SWS_Fim_00090] [SWS_Fim_00092] [SWS_Fim_00094] [SWS_Fim_00101] [SWS_Fim_00106] [SWS_Fim_00107] [SWS_Fim_00108]

B.5.3 Deleted Specification Items in R22-11

[SWS_Fim_00999]