Document Title	Specification of Diagnostics Event Manager
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Identification No	019
Document Classification	Standard

Document Version	3.0.0
Document Status	Final
Part of Release	3.1
Revision	0001

Document Change History

Date	Version	Changed by	Change Description
02.08.2008	3.0.0	AUTOSAR Administration	• Document structure reworked and extended • Add APIs and configuration parameters for OBD support • Improve interaction between DCM and software components • Legal disclaimer revised
28.11.2007	2.3.0	AUTOSAR Administration	• Improvement of RTE compliance • Improvement of configuration part • Improvement of document structure • Rework and tightening of data type useage • New API added • Document meta information extended • Small layout adaptations made

05.12.2006	2.1.0	AUTOSAR Administration	• Complete rework and extension of debouncing part • Dem_ClearGroupOfDTC and Dem_ClearSingleDTC replaced by Dem_SingleDTC • Dem_GetNextFilteredDTCAndFDC, Dem_SetDTCFilterForRecords, Dem_GetSizeOfFreezeFrame, Dem_SetValueByOemId, Dem_SetEnableCondition, Xxx_DemGetFaultDetectionCounter added • DTCTranslationType replaced by DTCKind in several APIs • Chapter "Service DEM" added • Function IDs reworked • File Structure reworked and extended • Configuration chapter reworked and extended • Dem_GetNextFilteredDTC reworked • Legal disclaimer revised
29.06.2006	2.0.0	AUTOSAR Administration	Layout Adaptations
10.07.2005	1.0.0	AUTOSAR Administration	Initial release

Page left intentionally blank

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).
Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and functional overview

The Diagnostic Event Manager DEM is a sub-component, like the Diagnostic Communication Manager (DCM) and Function Inhibition Manager (FIM) of the diagnostic module within AUTOSAR. It is responsible for processing and storing diagnostic events (errors) and associated FreezeFrame data. Further, the DEM provides fault information to the DCM (e.g. read all stored DTCs from the event memory). The DEM offers interfaces to the application layer, the DCM and the FIM. Optional filter services are defined.

The specification document only defines the API calls and roughly describes the internal functionality. For real implementations for OEMs, the OEM related specifications are requirered.

The basic targets of the DEM specification document are:

- Standardization of APIs
- Exchangeability of basic software components
- Definition of optional functions
- Coverage of ´Non end-user-competition related issues'
- Ability for a common approach for automotive manufacturer and component suppliers

This specification defines the API and the configuration of the AUTOSAR Basic Software module Diagnostic Event Manager (DEM). The specification focuses on the description of APIs and not on internal behavior of the DEM. Internal behavior is highly automotive manufacturer specific. Therefore, descriptions of the internal behavior of the DEM inside this document are only examples.

2 Acronyms and abbreviations

Acronym:	*Description:*
N_OK	Not OK
P-Code	Power train code
EcuM	Electronic Control Unit Manager
FreezeFrame	FreezeFrame is defined as a record of data (DIDs/PIDs). FreezeFrames are the same as SnapShotRecords in ISO 14229-1 [12].
Extended Data Record	An Extended Data Record is a record to store specific information assigned to a fault.
Monitor	A diagnostic monitor is an atomic routine determining the proper functionality of a component. This monitoring function identifies a specific fault type (e.g. short to ground, open load) for a monitoring path. A Monitoring Path represents the physical system or a circuit that is being diagnosed (e.g. sensor input).
Operating cycle	An 'Operating cycle' is the base of the event qualifying and also DEM scheduling (e.g. ignition key off-on cycles, driving cycles, …)
Healing	Unlearning/deleting of a no longer failed event/DTC after a defined number of driving/operation cycles from event memory
PossibleErrors	PossibleErrors means the ApplicationErrors as defined in meta model

Abbreviation:	*Description:*
API	Application Programming Interface
BSW	Basic Software
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DCM	Diagnostic Communication Manager
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DID	Data Identifier
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
EMI	Electro Magnetic Interference
ESD	Electro Static Discharge
FDC	Fault Detection Counter
FIM	Function Inhibition Manager
HW	Hardware
ID	Identification/Identifier
ISO	International Standardization Organization
IUMPR	In Use Monitoring Performance Ratio
LPF	Low Pass Filter
MIL	Malfunction Indication Light
NVRAM	Non volatile RAM
OBD	Onboard Diagnostics
OEM	Original Equipment Manufacturer (Automotive Manufacturer)
OS	Operating System
PID	Parameter Identification
PTO	Power Take Off
RAM	Random Access Memory
ROM	Read-only Memory
RTE	Runtime Environment
SW-C	Software Components

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_BasicSoftwareModules.pdf

[2] Specification of ECU Configuration,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_ECU_Configuration.pdf

[3] Layered Software Architecture,
https://svn2.autosar.org/repos2/22_Releases/
AUTOSAR_LayeredSoftwareArchitecture.pdf

[4] General Requirements on Basic Software Modules,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_General.pdf

[5] Specification of Diagnostic Communication Manager
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_DCM.pdf

[6] AUTOSAR Basic Software Module Description Template,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_BSW_Module_Description.pdf

[7] AUTOSAR Basic Software Component Template,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SoftwareComponentTemplate.pdf

[8] Specification of Function Inhibition Manager
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_FIM.pdf

3.2 Related standards and norms

[9] D1.5-General Architecture; ITEA/EAST-EEA, Version 1.0; chapter 3, page 72 et seq.

[10] D2.1-Embedded Basic Software Structure Requirements; ITEA/EAST-EEA, Version 1.0 or higher

[11] D2.2-Description of existing solutions; ITEA/EAST-EEA, Version 1.0 or higher.

[12] ISO 14229-1: Unified diagnostic services (UDS) – Part 1: Specification and requirements (Release 2006-12)

[13] ISO 15031-5: Road vehicles – Communication between vehicle and external equipment for emission-related diagnostic – Part 5: Emission-related diagnostic services.

[14] IEC 7498-1 The Basic Model, IEC Norm, 1994

[15] SAE J1979 Rev May 2007

[16] Title 13, California Code Regulations, Section 1968.2, Malfunction and Diagnostic System Requirements for 2004 and Subsequent Model-Year Passenger Cars, Light-Duty Trucks, and Medium-Duty Vehicles and Engines (OBD II) (Biennial Review MY08-11).

[17] EU III/IV EOBD: Directive 70/220/EEC as last amended with 2003/76/EC

[18] EU 5/5+/6: Regulation (EC) 715/2007 of 20 June 2007 and Implementing part of the Regulation (EC) which is to be finalized by 2 July 2008, REGULATION (EC) No 715/2007 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 20 June 2007

[19] Title 13, California Code Regulations, Section 1971.1, On-Board Diagnostic System Requirements for 2010 and Subsequent Model-Year Heavy-Duty Engines (HD OBD)

4 Constraints and assumptions

Some of the synchronous API calls defined within the DEM might take more time to complete than a software component or basic software component is assigned to run. Thus the calling instance has to ensure that the blocking caused by the execution of the DEM API call is handled appropriately.

Dem126: There shall only be one DEM module available per ECU.

The DEM can have multiple different sections of event memory. The mapping of a DTC to the according section is done with the parameter DTC Origin. A specific ECU's DEM is only accessible by software components located inside the same ECU.

4.1 Limitations

Timing constrains have to be considered for the whole ECU. If there are explicit needs for faster responses from the DEM than the DEM basic cycle time, special measures have to be implemented that are not specified in this AUTOSAR document. This is especially the case in ECUs with many events.
The DEM is able to support additional event memories (Permanent memory, Mirror memory and Secondary memory).

Emission related ECUs shall use the ISO 15031-6 DTC format only. This means also that all DTCs reported on an UDS request have to implement this format

The structure of a specific ExtendedDataRecord identified by its record number is unique per ECU.

This specification does not cover any SAEJ1939 related diagnostic requirements. This means the SAEJ1939 part of the heavy duty OBD regulation cannot be fulfilled applying this document.

4.2 Applicability to car domains

The DEM is designed to fulfill the design demands for ECUs with OBD requirements as well as for ECUs without OBD requirements. The immediate domains of applicability are currently body, chassis and powertrain ECUs. However, there is no reason why the DEM cannot be used to implement ECUs for other car domains like infotainment.

5 Dependencies to other modules

The AUTOSAR **Diagnostic Event Manager (DEM)** has interfaces and dependencies to the following Basic software modules and Software Components:

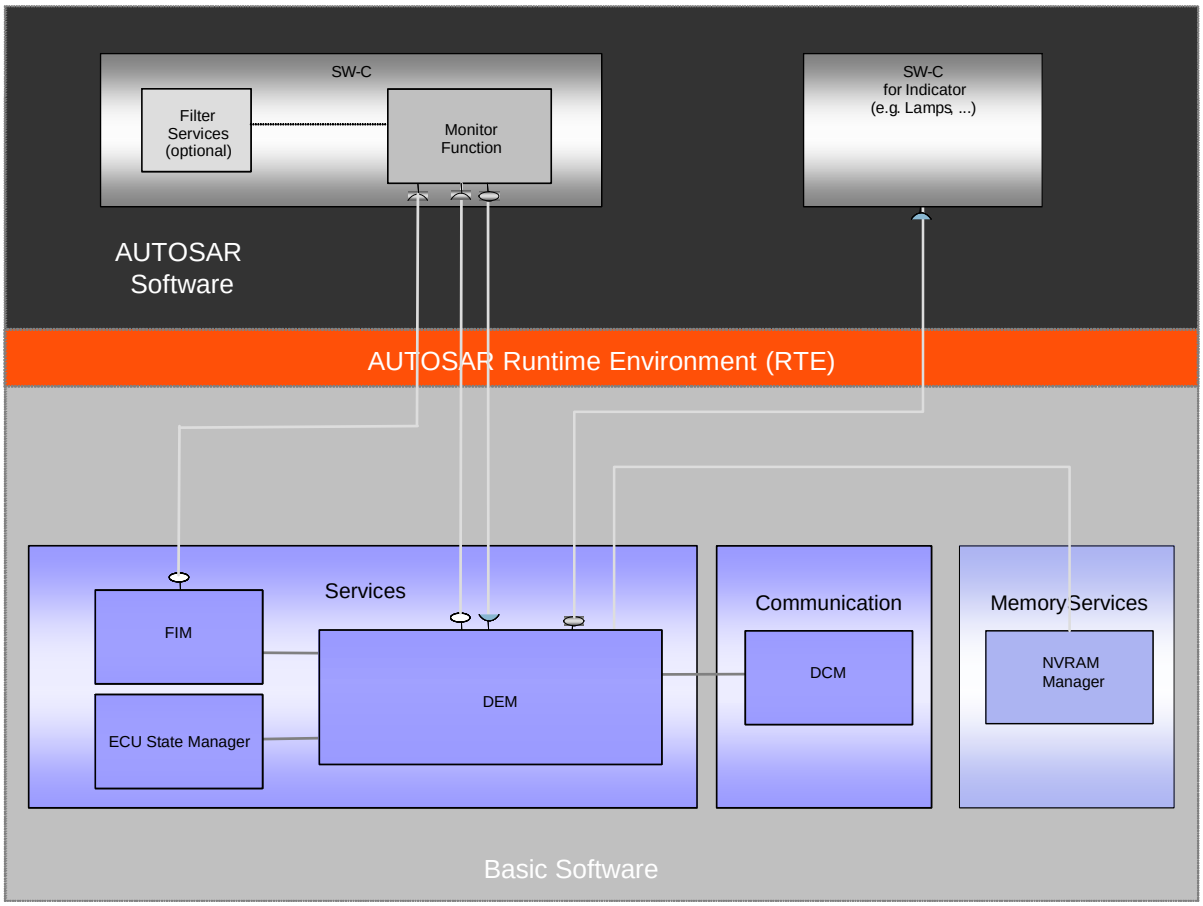


Figure 1 Dependencies of the Diganostic Event Manager (DEM) to other software modules

- The **Function Inhibition Manager** (FIM) stands for the evaluation and assignment of events to the required actions for Software Components (e.g. inhibition of specific “Monitor Functions”). The DEM informs and updates the Function Inhibition Manager (FIM) upon changes of the event status in order to stop or release function entities according to assigned dependencies. An interface to the function entities is defined and supported by the “ECU State Manager”. The FIM is not part of the DEM.

- The **Diagnostic Communication Manager (DCM)** is in charge of the communication path and execution of diagnostic service resulting in the processing of diagnostic requests from an external tester or onboard test system. It forwards requests coming from an external diagnostic scan tool and is further responsible for assembly of response messages (DTC, status information, ...) which will be transferred to the external diagnostic scan tool afterwards.

- **SW-Components (SW-C)** can access the DEM to update and/or retrieve current event status information. SW-Components will also provide data (e.g. event related data). SW-Components can retrieve data from the DEM e.g. to turn the indicator lamps on or off. The **Monitor Function** is a sub-component of a SW-Component.

- **NVRAM** blocks (maximum size is a matter of configuration) are assigned to the DEM and used by the DEM to achieve permanent storage of event status information and associated data (e.g. over power-on reset). The NVRAM manager provides mechanisms to store data blocks in NVRAM.

- The **ECU State Manager** is responsible for the basic initialization and de-initialization of basic SW components including DEM.

5.1 File structure

5.1.1 Code file structure

Dem108: The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files named:

- Dem_Lcfg.c – for link time configurable parameters (in the current version not used by DEM).
- Dem_PBcfg.c – for post build time configurable parameters

These files shall contain all link time and post-build time configurable parameters.

5.1.2 Header file structure

Dem151: The header-file structure shall include the following files named:

- Dem.h – header file of DEM module
- Dem_Types.h – for all DEM data types
- Dem_Lcfg.h – for link time configurable parameters (in the current version not used by Dem)
- Dem_PBcfg.h – for post build time configurable parameters
- Dem_IntErrId.h – for BSW EventId Symbols
- SchM_Dem.h – for Basic Software Module Scheduler symbols
- Fim.h – for Fim Symbols
- Nvm.h – for NVRAM manager Symbols
- MemMap.h – for memory mapping
- Rte_Dem.h – for RTE Symbols
- Std_Types.h – includes all definitions of standard types

Dem152: The module shall include the Dem.h file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

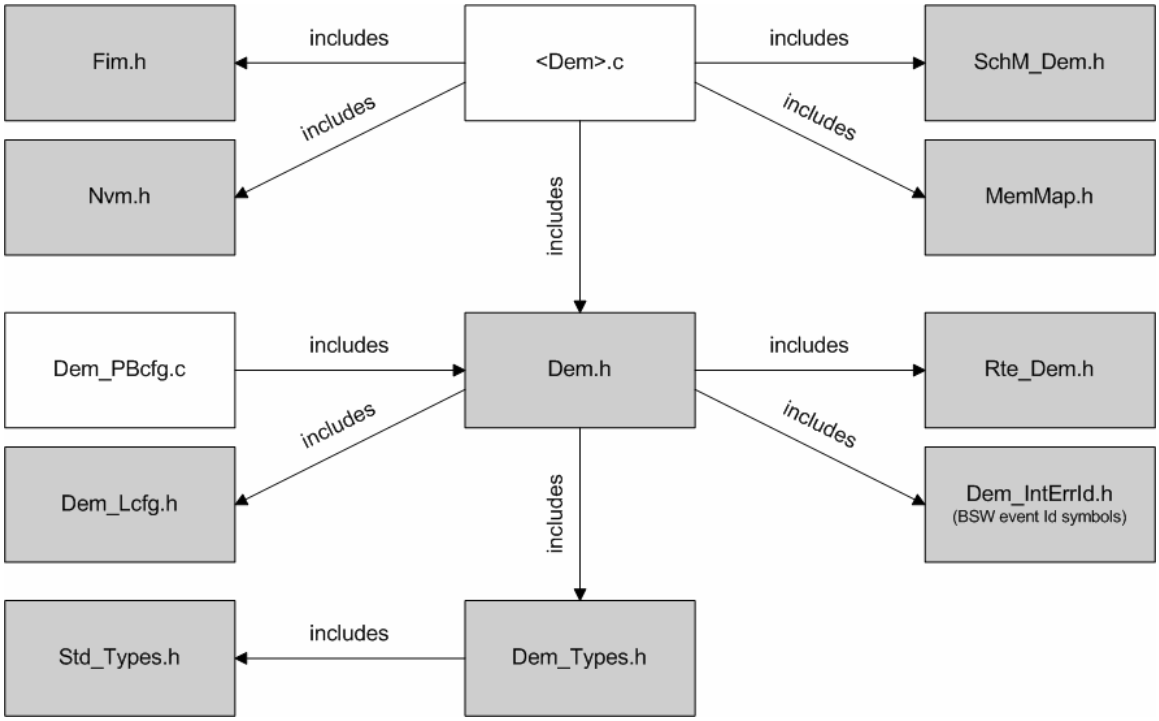


Figure 2 Header file structure

Only the header file Dem.h is used by other BSW modules.

6 Requirements traceability

6.1 Document: AUTOSAR requirements on Basic Software, general

Requirement	*Satisfied by*
[BSW3] Version identification	Dem110, Dem111
[BSW4] Version check	Dem110, Dem111, Dem067
[BSW6] Platform independency	Implementation requirement
[BSW7] HIS MISRA C	Implementation requirement
[BSW5] No hard coded horizontal interfaces within MCAL	Not applicable
[BSW9] Module User Documentation	Documentation requirement
[BSW10] Memory resource documentation	Documentation requirement
[BSW101] Initialization interface	Dem102
[BSW158] Separation of configuration from implementation	Dem108
[BSW159] Tool-based configuration	Ref. to chapter 10. configuration definitions
[BSW160] Human-readable configuration data	Ref. to chapter 10. configuration definitions
[BSW161] Microcontroller abstraction	Not applicable
[BSW162] ECU layout abstraction	Not applicable
[BSW164] Implementation of interrupt service routines	Not applicable
[BSW166] BSW Module interfaces	Dem108
[BSW167] Static configuration checking	See chapter 10. configuration definitions
[BSW168] Diagnostic Interface of SW components	Not applicable
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable
[BSW171] Configurability of optional functionality	Not applicable
[BSW172] Compatibility and documentation of scheduling strategy	Documentation requirement
[BSW300] Module naming convention	Implemented
[BSW301] Limit imported information	Implementation requirement
[BSW302] Limit exported information	Implementation requirement
[BSW304] AUTOSAR integer data types	Implementation requirement
[BSW00305] Self-defined data types naming convention	Chapter 8.2
[BSW306] Avoid direct use of compiler and platform specific keywords	Implementation requirement
[BSW307] Global variables naming convention	Implementation requirement
[BSW308] Definition of global data	Implementation requirement
[BSW309] Global data with read-only constraint	Implementation requirement
[BSW310] API naming convention	Chapter 8.2
[BSW312] Shared code shall be reentrant	See chapter 8.3 function definitions
[BSW314] Separation of interrupt frames and service routines	Implementation requirement
[BSW318] Format of module version numbers	Implemented
[BSW321] Enumeration of module version numbers	Implementation requirement
[BSW323] API parameter checking	Implementation requirement
[BSW324] Do not use HIS I/O Library	Not applicable
[BSW325] Runtime of interrupt service routines	Implementation requirement
[BSW326] Transition from ISRs to OS tasks	Not applicable
[BSW327] Error values naming convention	Not applicable
[BSW328] Avoid duplication of code	Implementation requirement
[BSW329] Avoidance of generic interfaces	Implemented
[BSW330] Usage of macros / inline functions instead of functions	Implementation requirement
[BSW331] Separation of error and status values	Not applicable

Requirement	*Satisfied by*
[BSW333] Documentation of callback function context	Documentation requirement
[BSW334] Provision of XML file	Implementation requirement
[BSW335] Status values naming convention	Implemented
[BSW336] Shutdown interface	Dem182
[BSW337] Classification of errors	Not applicable
[BSW338] Detection and Reporting of development errors	Not applicable
[BSW339] Reporting of production relevant errors and exceptions	Not applicable
[BSW341] Microcontroller compatibility documentation	Not applicable
[BSW342] Usage of source code and object code	Implementation requirement
[BSW343] Specification and configuration of time	Not applicable
[BSW344] Post-Build configuration	Dem267, Dem268
[BSW345] Pre-Build configuration	Dem267, Dem268
[BSW346] Basic set of module files	Dem108
[BSW347] Naming separation of drivers	Not applicable
[BSW348] Standard type header	Not applicable
[BSW350] Development error detection keyword	Not applicable
[BSW353] Platform specific type header	Not applicable
[BSW355] Do not redefine AUTOSAR integer data types	Implementation requirement
[BSW357] Standard API return type	Not applicable
[BSW358] Return type of init() functions	Implemented
[BSW359] Return type of callback functions	Not applicable
[BSW360] Parameters of callback functions	Not applicable
[BSW361] Compiler specific language extension header	Not applicable
[BSW369] Do not return development error codes via API	Not applicable
[BSW370] Separation of callback interface from API	Implementation requirement
[BSW371] Do not pass function pointers via API	Implemented
[BSW373] Main processing function naming convention	No main processing function used
[BSW374] Module vendor identification	Not applicable
[BSW375] Notification of wake-up reason	Not applicable
[BSW376] Return type and parameters of main processing functions	No main processing function used
[BSW377] Module specific API return types	Chapter 8.2
[BSW378] AUTOSAR boolean type	Implementation requirement
[BSW379] Module identification	Not applicable
[BSW00380] Separate C-Files for configuration parameters	Dem108
[BSW00381] Separate configuration header file for pre-compile time parameters	Dem108
[BSW00382] Not-used configuration elements need to be listed	Not applicable
[BSW00383] List dependencies of configuration files	Dem108
[BSW00384] List dependencies to other modules	Dem108
[BSW00385] List possible error notifications	Dem113, Dem114
[BSW00386] Configuration for detecting an error	Dem116
[BSW00387] Specify the configuration class of callback function	Not applicable
[BSW00388] Introduce containers	Ref. to chapter 10. configuration definitions
[BSW00389] Containers shall have names	Ref. to chapter 10. configuration definitions
[BSW00390] Parameter content shall be unique within the module	Chapter 8.2
[BSW00391] Parameter shall have unique names	Chapter 8.2
[BSW00392] Parameters shall have a type	Chapter 8.2
[BSW00393] Parameters shall have a range	Chapter 8.2
[BSW00394] Specify the scope of the parameters	Chapter 8.2
[BSW00395] List the required parameters (per parameter)	Chapter 8.2
[BSW00396] Configuration classes	Dem267, Dem268
[BSW00397] Pre-compile-time parameters	Dem267, Dem268

Requirement	*Satisfied by*
[BSW00398] Link-time parameters	Dem267, Dem268
[BSW00399] Loadable Post-build time parameters	Dem267, Dem268
[BSW00400] Selectable Post-build time parameters	Dem267, Dem268
[BSW00401] Documentation of multiple instances of configuration parameters	Dem267, Dem268
[BSW00402] Published information	Dem112
[BSW00404] Reference to post build time configuration	Dem267, Dem268
[BSW00405] Reference to multiple configuration sets	Dem267, Dem268
[BSW00406] Check module initialization	Dem124, Dem169, Dem170
[BSW00407] Function to read out published parameters	Dem110, Dem111
[BSW00408] Configuration parameter naming convention	Implemented
[BSW00409] Header files for production code error IDs	Dem108
[BSW00410] Compiler switches shall have defined values	Implementation requirement
[BSW00411] Get version info keyword	Dem110, Dem111,Dem112
[BSW00412] Separate H-File for configuration parameters	Dem108
[BSW00413] Accessing instances of BSW modules	Implemented
[BSW00414] Parameter of init function	Implemented
[BSW00415] User dependent include files	Dem108
[BSW00416] Sequence of Initialization	Implemented
[BSW00417] Reporting of Error Events by Non-Basic Software	Dem107
[BSW00418] Allocation of error detection	Dem117
[BSW00419] Separate C-Files for pre-compile time configuration parameters	Dem108
[BSW00420] Production relevant error event rate detection	Dem107
[BSW00421] Reporting of production relevant error events	Dem107
[BSW00422] Debouncing of production relevant error status	Dem004
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Implemented
[BSW00424] BSW main processing function task allocation	Implementation Requirement
[BSW00425] Trigger conditions for schedulable objects	Implementation Requirement
[BSW00426] Exclusive areas in BSW modules	Implementation Requirement
[BSW00427] ISR description for BSW modules	Implementation Requirement
[BSW00428] Execution order dependencies of main processing functions	Implementation Requirement
[BSW00429] Restricted BSW OS functionality access	Implementation Requirement
[BSW00431] The BSW Scheduler module implements task bodies	Implementation Requirement
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Implementation Requirement
[BSW00433] Calling of main processing functions	Not applicable
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable
[BSW00435] Header File Structure for the Basic Software Scheduler	Dem108
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	Dem108

6.2 Document: AUTOSAR requirements on Basic Software, cluster Diagnostic

Requirement	*Satisfied by*
[BSW04010] Interface between Diagnostic service handling and Diagnostic Event (error)	See chapter Function Definition, interface DCM ⇔ DEM (Chapter 8.3.5)

management [approved]	Dem042, Dem020, Dem041
[BSW04002] Basic SW Module for Diagnostic event (error) management [approved]	Defined by AUTOSAR architecture
[BSW04030] Interface between DEM and Monitoring SW Component	refer to [BSW103]
[BSW04031] Interface between DEM and Function Inhibition Manager	Dem029
[BSW04057] Classification of event [approved]	Dem057, Dem058, Dem047, Dem156, Dem157
[BSW04061] Distinction between different function groups [approved]	Dem153
[BSW04063] Single EventId for each monitoring path	Dem153, Dem154, Dem155, Dem006
[BSW04064] Event buffer must be configurable concerning size [approved]	See chapter Configuration specification (Chapters 7.1.2, 10.2)
[BSW04065] Clearing of events and event groups [approved]	See [BSW111], [BSW113]
[BSW04066] Provision of a `Secondary Event Memory [approved]	Dem010
[BSW04058] Support individual deletion and reading services for `Secondary Event Memory [approved]	Dem063
[BSW04067] Counting and evaluation of events according to ISO 14229-1 DTCStatusMask	Dem011, Dem061
[BSW04068] Standardized Event forget/unlearn counting	Dem019
[BSW04069] DEM System status indication [proposed]	Dem046
[BSW04070] Event ‘occurrence order’ definition [approved]	Dem160, Dem161, Dem162, Dem219, Dem221
[BSW04071] Event importance definition [approved]	See [BSW102]
[BSW04072] Event duration definition [approved]	DEM internal
[BSW04073] Event combination and compression [approved]	Dem024, Dem025, Dem026
[BSW04074] Event related ‘environmental data’ [proposed]	Dem039, Dem021, Dem040, Dem070, Dem071, Dem073, Dem074, Dem075, Dem076 (environmental data is called event related data)
[BSW04075] Event and DTC assignment [approved]	Internal calibration/configuration (Chapter 10.2)
[BSW04076] System Cycle definition [approved]	Dem019, Dem047
[BSW04077] Interface between DEM and NVRAM function	Chapter 7.3.9

7 Functional specification

The **Diagnostic Event Manager (DEM)** handles and stores the events, detected by the Software Components using a Monitor Function above the RTE. The stored event information are available via an interface to other Basic software modules and Software Components (SW-C).

7.1 DEM core data

7.1.1 ´Diagnostic Event´ definition

A ´Diagnostic Event´ defines the atomic unit that can be handled by the DEM module. The status of a ´Diagnostic Event´ represents the result of a Monitor Function. The DEM receives the result of a monitor from SW-C via the RTE or other BSW modules.

The DEM uses the EventId to manage the status of the ´Diagnostic Event´ of a system and performs the required actions for individual test results, e.g. store the FreezeFrame.

Dem153: The DEM module shall represent each Diagnostic Event by an EventId and the related EventName.

All monitoring functions and BSW modules use EventId as a symbolic EventName. The DEM configuration tool replaces the symbolic names by numbers.

Dem154: The EventId and the related EventName shall be unique per DEM module represented by the ECU configuration (refer to Dem126).

The DEM is not designed to be able to handle the case where more than one monitor shares a single EventID.

Dem006: The diagnostic event shall have one current extended event status.

The DEM module may use an internal event status. However, when requested by the DCM the extended event status will be reported.

The DEM module supports several attributes for EventIds such as:

- DTC(s) (Diagnostic Trouble Code)
- Event priority (priority of an event)
- Healing cycles (cycles necessary to heal/erase an event)
- Healing allowed (general switch to allow healing or not)
- Identification of the destination of an event (origin)
- Emission relevant (OBD fault definition)
- Indicator request (Indicator to be requested by an EventId)

The values for these attributes are a part of the DEM configuration (see chapter 10 Configuration specification).

7.1.2 ´Event Memory´ description

The ´Event Memory´ is defined as a set of event records located in a dedicated memory block. The event record includes at least the extended event status and event related data.

The size of the ´Event Memory´ is configurable in the DEM configuration.

Dem329: For storing to non-volatile memory the DEM shall use the NVRAM Manager.

Dem010: The DEM module shall support at least the primary event memory and may support additional several event memories (e.g. mirror, permanent).

For the DCM-DEM Interface the parameter DtcOrigin is used to distinguish between the memory areas. The intention is to allow OEM specific operations on the different memory areas (primary, permanent memory and mirror memory). The support of several event memories is not mandatory.

The displacement strategy in the event memory is not specified in AUTOSAR.

7.2 DEM core functionalities

7.2.1 Overview DEM Structure

Figure 3 shows the UML model of a DEM configuration.

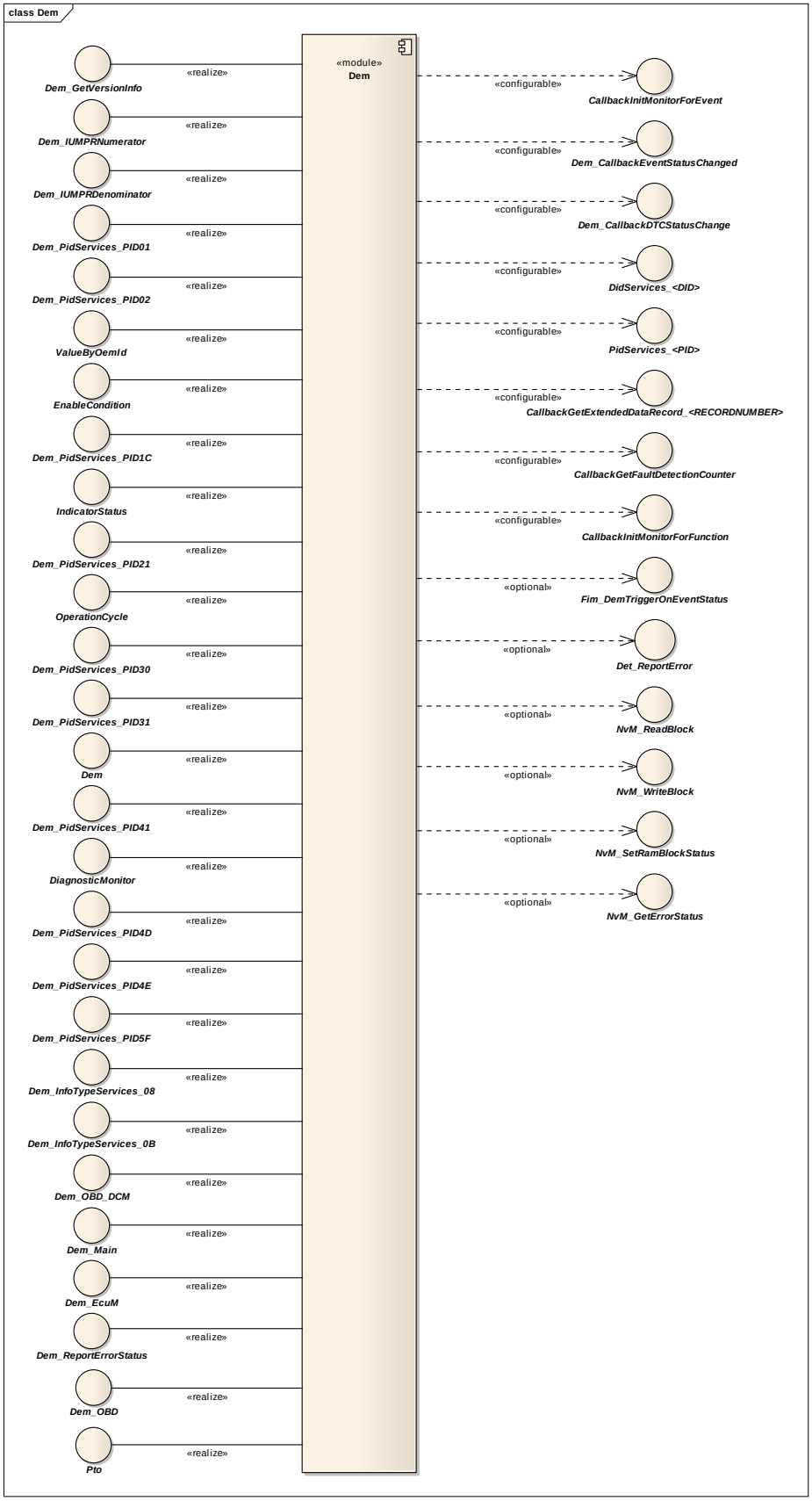


Figure 3 UML model of the DEM configuration

The following figure shows an example of a DEM event memory layout.

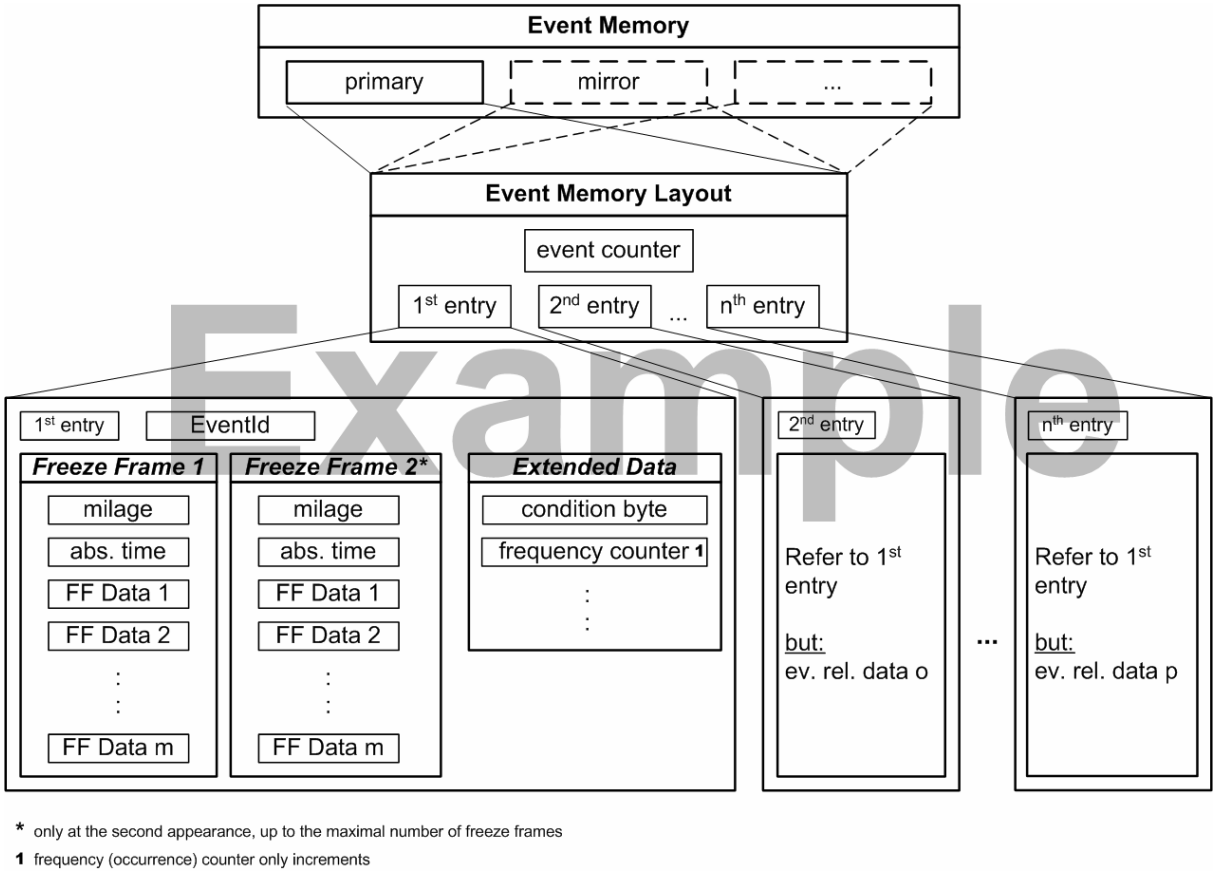


Figure 4 Example of a DEM event memory layout

7.2.2 Event Status Management

The ´Event Status Management´ is the DEMs ability to record and retain events, event status and associated data.

Dem330: The DEM module shall provide the capability to report the status of an event allowing a diagnostic monitor to inform the DEM about the result of the internal diagnostic test (refer to 8.3.3.1 Dem_SetEventStatus).

Dem331: The DEM module shall provide the capability to reset the status of an event (refer to 8.3.3.2 Dem_ResetEventStatus).

Dem332: The DEM module shall provide the capability to retrieve the current status of an event (refer to 8.3.3.6 Dem_GetEventStatus).

Dem333: The DEM module shall provide the functions Dem_GetEventFailed (refer to 8.3.3.7 Dem_GetEventFailed) and Dem_GetEventTested (refer to 8.3.3.8 Dem_GetEventTested).

Dem334: For OBD relevant systems the DEM module shall provide the functions Dem_PrestoreFreezeFrame and Dem_ClearPrestoreFreezeFrame by configuration parameters (refer to 7.2.4 Event related Data and 10.2.3 DemGeneral) and may provide these functions for non OBD relevant systems.

Dem003: For OBD relevant systems the DEM module shall provide the interface InitMonitorForEvent to initalize a diagnostic monitor (refer to 8.4.3.1.1 InitMonitorForEvent) and may provide these function for non OBD relevant systems.

Dem009: The DEM module shall provide the function Dem_ClearDTC (refer to 8.3.5.3.1 Dem_ClearDTC) to the DCM for deleting a single DTC, DTC groups and all DTC from the event memory. This triggers also initalizing of related SW-Cs and BSW modules according to Dem003.

The service ClearDiagnosticInformation (14 hex) of ISO 14229-1 [12] defines and covers the required actions and the deletion of related memory areas like FreezeFrames. A single DTC value is transmitted which can either represents a single DTC or a group of DTCs. The groups are defined in ISO 14229-1 [12] Definition of GroupOfDTC and range of DTC numbers, Annex D1.

Dem335: The DEM module shall provide the interface InitMonitorForFunction (refer to 8.4.3.1.2) to initialize an assigned SW-C function which is not a monitoring function.

Dem011: The DEM module shall control the counting of the number of Diagnostic Event/DTC occurrences.

How these counters are accessed is not specified by AUTOSAR.

Dem013: The DEM module shall support DTC formats according to:

- ISO 14229-1
- ISO 15031-6
- SAEJ1939-73
- ISO 11992-4

The configuration parameter DemTypeOfDTCSupported (refer to 10.2.3 DemGeneral) is used to define the DTC format of an ECU (refer to Dem_GetTranslationType) and determines the DTC format value to be reported for ISO 14229-1 service Read DTC Information (0x19).

Dem336: Emission related ECUs shall use the ISO 15031-6 DTC format only. This means also that all DTCs reported on an UDS request have to implement this format.

For emission and non emission related DTCs exists only one DTC number.

DTCs are configured in the DEM. The DEM uses always a 3 Byte definition with the following representations. For UDS services, the DTC size is 3 bytes (HighByte, MiddleByte and LowByte).

Dem277: The DEM shall report DTC as a uint32 with byte 0 = LowByte, byte 1 = MiddleByte and byte 2 = HighByte. The byte 3 of the uint32 is unused. For OBD services there are only two bytes (HighByte, LowByte) used. The Dem services shall report these DTC as a uint32 with byte 1 = LowByte and byte 2 = HighByte, byte 3 being free and byte 0 = 0x00.

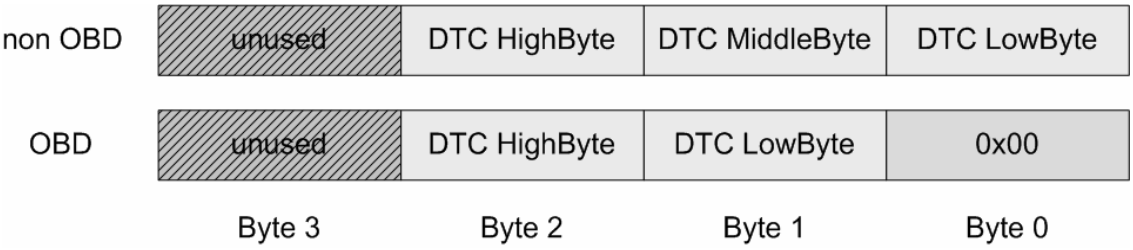


Figure 5 DTC Byte Order

Dem034: The DEM module shall be capable of enabling (Dem_EnableEventStatusUpdate) and disabling (Dem_DisableEventStatusUpdate) the update of all event states.

Meaning: when the update of event states is disabled, calls to Dem_SetEventStatus, ReportErrorStatus or Dem_ResetEventStatus will not lead to changes in internal states of the DEM module (refer to section 8.3.3.1 Dem_SetEventStatus).

Dem035: The DEM module shall be capable of enabling (Dem_EnableDTCStorage, see 8.3.5.4.2) and disabling (Dem_DisableDTCStorage, see 8.3.5.4.1) the storage of event records.

Meaning: when the storage of event records is disabled, the update of an event status does not result in changes in the event memory (no DTC storage) (ref. to section 8.3.3.1 Dem_SetEventStatus).

Dem019: The DEM module shall support event unlearn counters (e.g. for failure healing). In case of OBD-relevant events they shall be based on the OBD/ISO 15031 defined cycles.

Unlearn counters are configurable per event (refer to 7.2.11 Debouncing of events).

Dem161: The DEM module shall handle the reoccurrence of healed events like new events since they were previously erased from the Event Memory by healing.

Dem014: The DEM module shall be able to return the DTC status availability mask (refer to 8.3.5.1.4 Dem_GetDTCStatusAvailability) in accordance to ISO 14229-1 (see [12]).

It is a system design decision if synchronous or asynchronous event processing is used.

Dem036: In case a qualified diagnostic event (passed / failed) is reported to the DEM module, the DEM shall perform the event status transition immediately for the bits being relevant for fault reactions (does not depend on the design decision if events are processed synchronously or asynchronously):
- Bit 0 TestFailed
- Bit 1 TestFailedThisOperationCycle
- Bit 4 TestNotCompletedSinceLastClear
- Bit 5 TestFailedSinceLastClear
- Bit 6 TestNotCompletedThisOperationCycle

Dem379: Depending on the design decision (synchronous or asynchronous event processing) the status update of the following bits:
- Bit 2 PendingDTC
- Bit 3 ConfirmedDTC
- Bit 7 WarningIndicatorRequested

may occur at a later point in time.

Note: If the status update of the bits described in Dem379 is implemented asynchronously a queuing mechanism is needed which ensures that all the changes applied to Bit 0, 1, 4, 5 and 6 will be considered when calculating Bit 2, 3 and 7. Similarly all processing and data storage associated with Bit 2, 3 and 7 needs to be queued (refer to Figure 6 describing diffenrent approaches of synchronous and asynchronous event processing design).

Dem015: The DEM module may provide the function Dem_GetIndicatorStatus (ref. to 8.3.3.13 Dem_GetIndicatorStatus) in order to retrieve the malfunction indication status (lamps, text message, beep...) for different indicators by a SW-C.

One indicator can be activated by several events and one event can also have more than one indicator assigned to it (refer to 10.2.15 DemIndicatorAttribute).

The indicator status is based on the event qualification provided by the DEM module as well as other contributions such as bulb check

Dem016: The DEM module shall support the execution of the event specific function DemEventStatusChanged (refer to 8.4.3.1.3) upon each event status change (e.g. FIM or ISO 14229-1 service 86hex "ResponseOnEvent Request").

Dem162: The DEM module shall provide enough memory space to store a predefined amount of faults.

Dem033: Severity may be assigned to events regarding the importance of the specific events according to ISO 14229-1, Annex D, DTCSeverityMask and DTCSeverity bit definitions.

ISO 14229-1, Annex D defines the following severity levels: no severity available, Maintenance, Check at next Halt, Check immediately.
The function call “Dem_SetDTCFilter” (Dem057) allows filtering for DTCs with severity information.

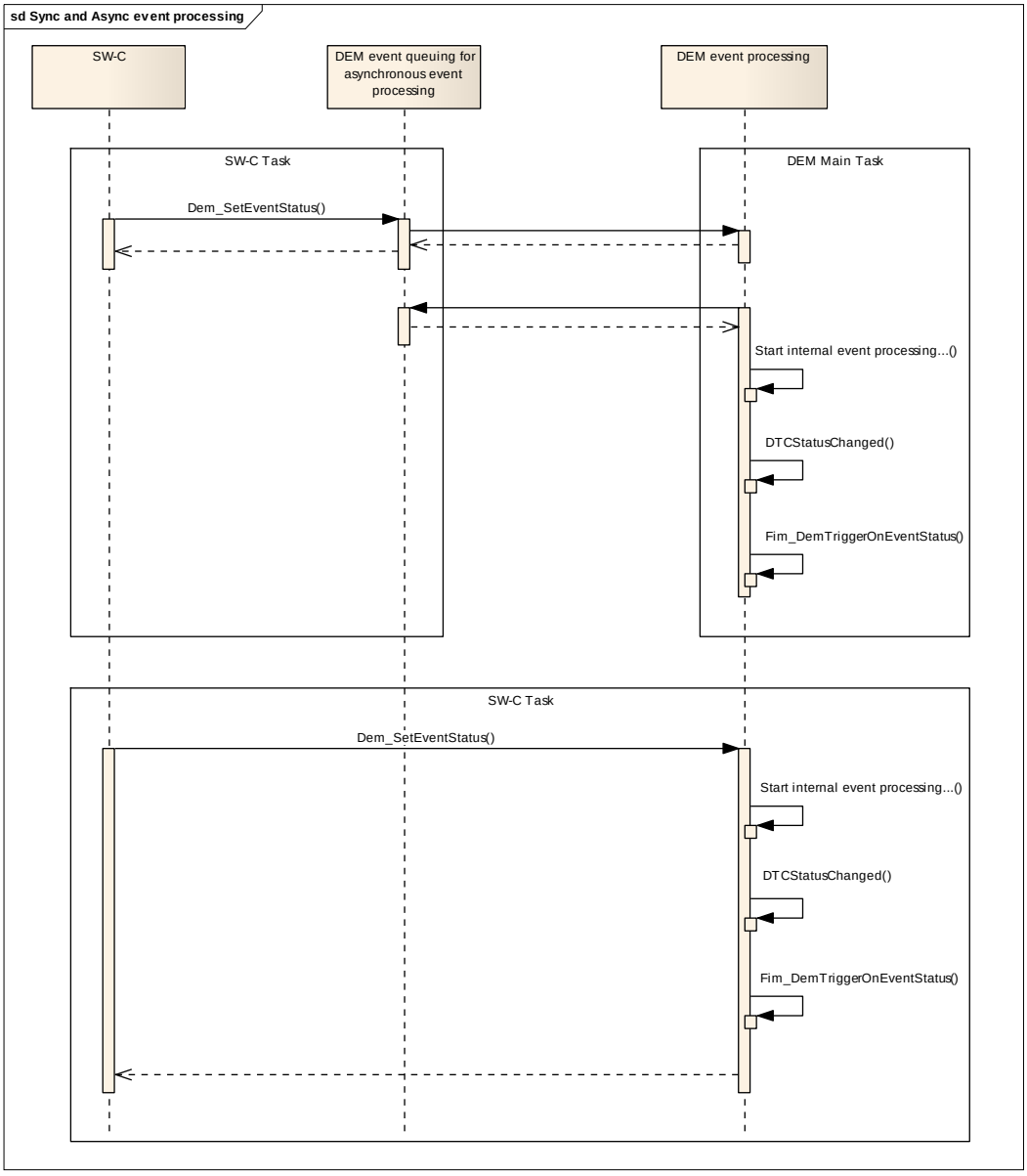


Figure 6 Synchronous and asynchronous event processing

7.2.3 Event combination

Event combination is an optional feature which is implementation-specific. This feature is needed when several monitor results have to be mapped to one specific DTC. The following requirements describe the implementation of this feature.

Dem024: The DEM module may be capable of combining or compressing several individual events to an additional combined event that has its own unique EventId.

Note: This usually implies that only the combined event triggers the storage of a FreezeFrame and updates of additional event specific information (e.g. self-healing, frequency counters, event related data)

Dem163: When the DEM module supports combined events, it shall ensure the consistency between the combined event status and associated Monitor Functions when updating the status or clearing individual events of the combined event or the combined event itself.

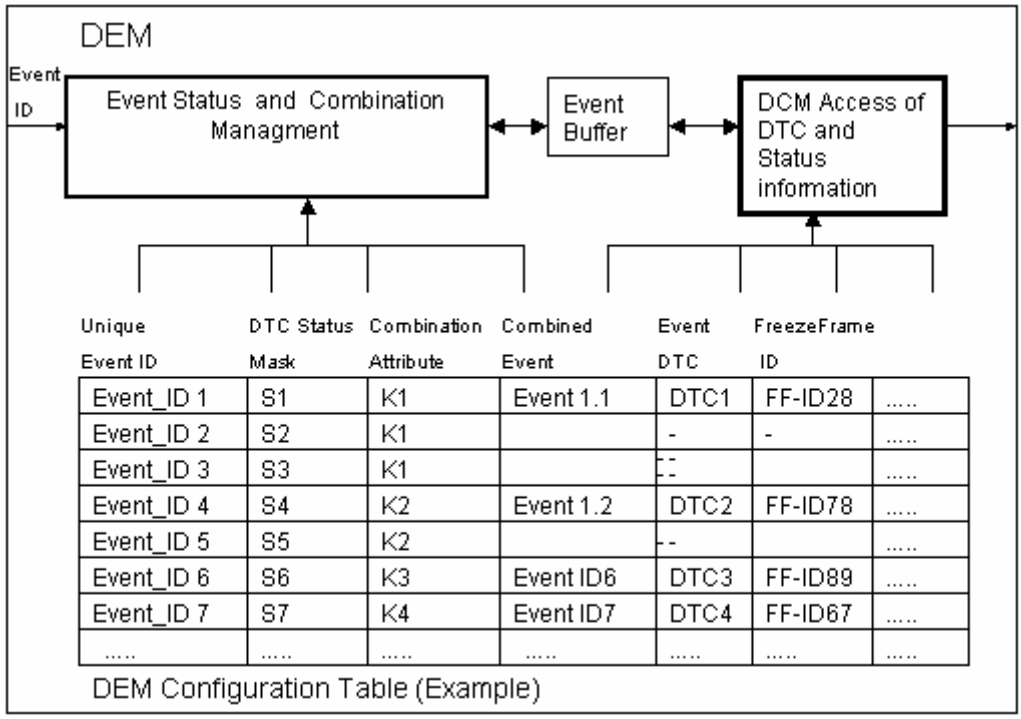
Related to the erasing of the ´combined event´ requested by ISO 14229-1service 14hex “ClearDiagnosticInformation – GroupOfDTC[]” all associated Monitor Functions must be reset.

Based on the event combination the duration of the unlearn mechanism can be extended since several Monitor Functions restart the unlearn counter.

Note: Related to the readiness information all combined events shall consist of the same function group (e.g. electrical Monitor Function of a sensor, plausibility Monitor Function, etc) to have similar test conditions.

Dem025: The configuration of the DEM module may cover the enabling and disabling of “event combinations”.

Dem026: If “combined diagnostic events” are supported, then the configuration of the DEM shall allow the assignment of each “diagnostic event” with an attribute like “combined diagnostic EventId”.



Unique Event ID	DTC Status Mask	Combination Attribute	Combined Event	Event DTC	FreezeFrame ID	
Event_ID 1	S1	K1	Event 1.1	DTC1	FF-ID28	.....
Event_ID 2	S2	K1		-	-	.....
Event_ID 3	S3	K1				.....
Event_ID 4	S4	K2	Event 1.2	DTC2	FF-ID78	.....
Event_ID 5	S5	K2		-		.....
Event_ID 6	S6	K3	Event ID6	DTC3	FF-ID89	.....
Event_ID 7	S7	K4	Event ID7	DTC4	FF-ID67	.....
.....	.....	.....	.....	.....	.....	.....

DEM Configuration Table (Example)

Figure 7 Example of a DEM configuration table

7.2.4 Event related Data

The ´Event related Data´ are additional data, e.g. sensor values or time stamp/milage that are stored in case of an event. ISO 14229-1 defines two different types of event related data: snapshot data (FreezeFrames) and extended data. The number or sets of stored ´Event related Data´ are strongly OEM / failure specific and are therefore configurable.

Dem039: The DEM module shall support several FreezeFrames with different sets of data.

The DEM module is not in charge of validity of event related data. Time related data consistency of event related data is depending on data source and storage time.

Dem021: The DEM module shall support the EventId specific storage of several of the FreezeFrames defined by Dem039:

Dem040: The number and the size of each FreezeFrame that can be stored by the DEM module shall be configurable due to the different domain requirements and ECU complexities (refer to section 10 Configuration specification).

Note: Due to implementation reasons the DEM usually needs to reserve memory for the maximum FreezeFrame size multiplied by the number of FreezeFrames.

Dem337: The DEM module shall be capable to store the possible combination of DTCs and their FreezeFrames depending on vehicle manufacturer specific and legislative requirements.

Dem002: The DEM module may support the pre-storage of a FreezeFrame regardless of the status change of an event.

The pre-storage of FreezeFrames can be processed via API function Dem_PrestoreFreezeFrame.
A pre-stored FreezeFrame is event specific. Upon status change of the event requireing a FreezeFrame to be stored the data from the pre-stored FreezeFrame will be used instead of the current values of the contained parameters.

This feature can be used for time critical events: With the first indication of the appearance of a time critical event – even if the event is not yet de-bounced/qualified – a snap shot of the FreezeFrame is captured. To ensure absence of reaction to stored FreezeFrames of qualified Events an additional FreezeFrame buffer should be used. Due to restrictions in hardware usage, the amount of possible entries can be restricted, therefore a replacement strategy could be required. This replacement strategy is not part of this specification.

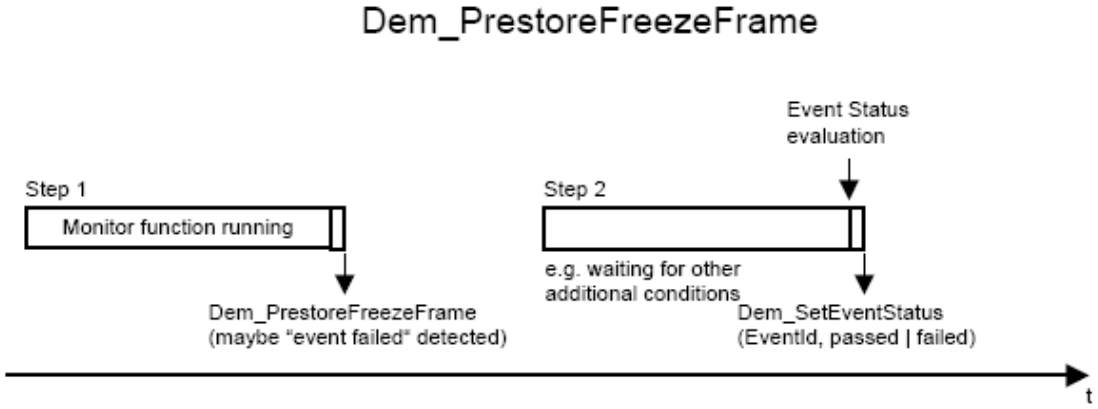


Figure 8 Example to use Dem_PrestoreFreezeFrame to prestore FreezeFrame data

Dem041: The DEM module shall support the storage of ExtendedDataRecords. The calculation and/or data acquisition for ExtendedDataRecords can be handled DEM internally or externally requested.

ExtendedDataRecords contains additional information associated to a specific event that is not contained in a FreezeFrame (extended data, e.g. frequency counters, self-healing counters, etc.). For more information about ExtendedDataRecords see ISO 14229-1.

7.2.5 DEM Cycle Management

Different operation cycles are used by the DEM module (ref. to ISO 14229-1). Those cycles could either be provided by other BSW modules and SW-C or generated by the DEM module itself.

Examples of operation cycles are:
- driving cycle
- engine warm up cycle
- ignition On/Off cycle
- power up/power down cycle
- operation active/passive cycle
- accumulated operating time

The DEM cycle management processes these different types of operation cycle definitions to create DEM specific operation cycle information used for event qualifying.

OBD legislation requires specific implementation of operation cycles. For emission related ECUs it is mandatory to implement these accordingly.

Dem338: For the DEM cycle management the API Dem_SetOperationCycleState shall be used.

7.2.6 NVRAM Manager Access

Non-volatile memory blocks (configurable in size by the NVRAM module) are used by the DEM module to achieve permanent storage of event status information and associated data (e.g. retrieve status at start-up).

Dem339: The DEM module has to verify the validity of its non volatile blocks.

Usually this verification is done in the API Dem_Init (ref. to 8.3.2.2 Dem_Init) for these blocks which are read be the ECU State Manager (ref to API NvM_ReadAll).

Dem340: After the API Dem_Init the DEM shall be fully operational.

Dem102: The API Dem_Shutdown shall finalize all pending operations in the DEM module to prepare the internal states and event data for transfer to the NVRAM. The event memory shall be locked and not modified until the API Dem_Init is recalled.

The ECU State Manager is responsible to initiate the copying process of data from RAM to NVRAM (refer to API NvM_WriteAll).

Dem341: For individual non-volatile blocks the API NvM_WriteBlock shall be called within the API Dem_Shutdown.

.

If the ECU power supply is disconnected before Dem_Shutdown has finished copying the data to NVRAM, data in NVRAM will be incomplete/inconsitent or not stored. At next start up the last operating cycle events could not be found anymore. Therefore the NVRAM Manager configuration provides mechanisms for data consistency, like redundant data blocks.

Dem164: The DEM module shall use the APIs NvM_WriteBlock and NvM_ReadBlock of the NVRAM module if there is the necessity to store and restore data between Dem_Init and Dem_Shutdown.

Dem275: If the call of NvM_ReadBlock after the defined recurrences was not successful, the DEM module may generate a DTC in the RAM area of event memory.

Note: All additional informations, like occurrence counter and FreezeFrames, are not meaningful, because the information could be volatile.

Dem276: If the call of NvM_WriteBlock after the defined recurrences was not successful, the DEM module may generate a DTC in the RAM area of event memory.

Note: The Information will be lost after ECU power down.

7.2.7 Interaction between DEM and Function Inhibition Manager (FIM)

The purpose of the FIM is to control (enable/disable) function entities within SW components based on inhibit conditions such as detected errors.

The DEM contribution to the above functionality is to provide event status information to the FIM.

The Function Inhibition Manager uses the information of dependencies provided by the software components.

Dem029: If the DEM module informs the FIM about a status change of an event, then the DEM module shall use the function Fim_DemTriggerOnEventStatus (ref. to [8]).

The information is also passed to the FIM if Dem_DisableDTCStorage is called.

The DEM module provides the function Dem_GetEventStatus for possible plausibility checks of the FIM, re-building, start-up etc. of inhibition relations by the FIM.

7.2.8 Interaction with BSW or SW-Cs

For certain use-cases the DCM (e.g. DIDs from SW-C or data directly from the NVRAM BSW) and the DEM (e.g. to retrieve PIDs and DIDs from SW-Cs) have to interact with either a BSW or a SW-C. In these cases the DCM and DEM either have to use services (to interact with BSW) or ports (to interact with SW-Cs) to retrieve the desired data.

In order to allow for configuration of whether a specifc data item (e.g. DID) has to be retrieved using service calls or ports the configuration of these items will contain additional parameters to indicate this. The usage of these configuration parameters is explained in chapter 10.2.8 DemPidOrDid.

7.2.9 DEM startup behavior

Dem169: The DEM module shall distinguish between a pre-initialization mode and a full-initialized mode (operation mode).

7.2.10 BSW Error Handling

Beside application software components also Basic Software (BSW) can detect errors (e.g. wrong RAM access), especially during startup. For these errors (only a small number compared to application specific events) some specific requirements apply (ref. to document [4] for further details)

- Errors are detected before DEM is initialized
- Errors can be reported during startup, information is buffered until DEM is fully available
- Errors can be reported between startup and shutdown, information flows directly to event memory
- Error entries in event memory can have a different format (no emphasis on FreezeFrame data for the workshop)
- No emphasis on error reaction (error reaction will be handled by FIM or/and by SW-C/BSW itself)

Dem127: The DEM module shall provide the possibility to unlearn/heal BSW errors.

Note: unlearning/healing of BSW errors can not be triggered by a BSW Monitor Function but by a defined healing cycle (e.g. event not reported for 10 driving cycles).

Dem107: The DEM module shall have the interface Dem_ReportErrorStatus to provide a BSW module the possibility to report errors due to the fact that the DEM module is not available during startup.

The interface Dem_ReportErrorStatus is used by BSW components to report errors after the DEM module is pre-initialized. During normal operation (after full initialization) the API Dem_ReportErrorStatus has the same internal behavior like the interface Dem_SetEventStatus used by SW-Cs.

During initialization of the DEM module the API Dem_ReportErrorStatus does not support debouncing (prefailed and prepassed). Therefore it is assumed, that all reported results are considered as debounced.

Dem207: The size of the buffer queue of the function Dem_ReportErrorStatus is configurable by the configuration parameter: DemBswErrorBufferSize.

A possible implementation could be a buffer of configurable size where all errors reported by BSW are stored until the startup process is finished. During normal operation (startup has been finished) the buffer is processed by a cyclic task of the DEM module and all contained events are reported to the event memory.

Since BSW events are reported and treated as normal application SW-C events in the event memory, they can also be classified (availability in workshop tester) and prioritized (overflow handling).

Dem167: The function Dem_ReportErrorStatus shall pass BSW events directly through the buffer to the Event Memory if the DEM module has been already initialized.

Dem168: The function Dem_ReportErrorStatus shall buffer (FIFO) reported events if the DEM module cannot access the Event Memory during Start Up.

Note: During start up phase there might not be all FreezeFrame data available.
SW-C events can not be stored before complete initialization of the DEM.

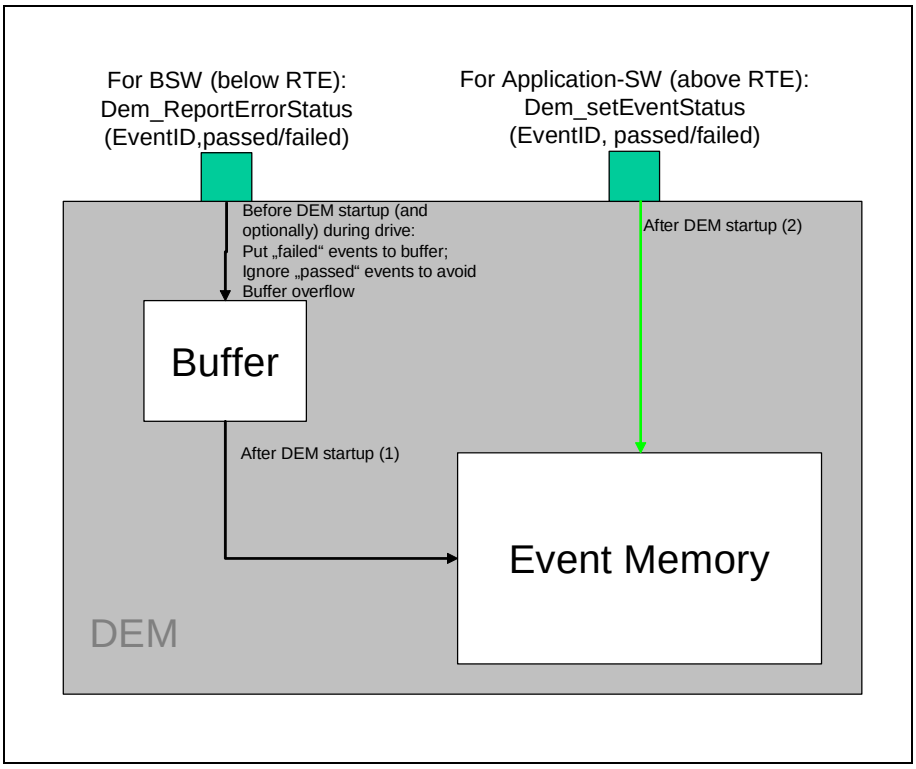


Figure 9 DEM behavior during startup

7.2.11 Debouncing of events

The Diagnostic Events can be qualified (debounced) by the Monitor Function or using a filter, implemented and provided by the DEM, for event qualification.

7.2.11.1 Types of debouncing

There are several types of debouncing:

1. **Signal debouncing** - (i.e. conditioning) is done in Hardware, using measures like EMI- or ESD suppression, low pass filtering etc.
   Signal debouncing in hardware is the responsibility of the ECU-Hardware designer and is not specified in AUTOSAR.

2. **Event debouncing** - can be done by the Monitor Function (SW-C/BSW) or by the DEM. If the Monitor Function debounces the event, the SW-C/BSW reports the diagnostic event statuses
   - DEM_EVENT_STATUS_PASSED
   - DEM_EVENT_STATUS_FAILED

   In case of the event should be debounced by the DEM module, the Monitor Function has to report the diagnostic event statuses
   - DEM_EVENT_STATUS_PREPASSED
   - DEM_EVENT_STATUS_PREFAILED

In addition to event debouncing event qualification is needed to determine the event status. The implementation of event qualification adheres to ISO 14229-1, Annex D [12]. For OBD-units, event qualification according to legal requirements is mandatory.

Dem004: The DEM module may support the debouncing of events.

7.2.11.2 Event debouncing algorithms

Dem342: The DEM module may provide standard algorithms for debouncing of events in order to avoid multiple implementations of the same mechanism in several SW-Cs.

There are various ways of implementing debounce counters e.g. counter based algorithms, time based algorithms and combinations. These algorithms are highly dependent on the diagnostic functionality of a certain monitor and specific to OEM requirements.

The DEM module provides the configuration option for enabling or disabling DEM internal debouncing.

7.2.11.3 Examples of debounce counters

7.2.11.3.1 Counter based

The signal is unqualified until the De-bounce Counter will reach the Maximum value. The De-bounce Counter will increase with Count in step size at every call of Dem_SetEventStatus/Dem_ReportError with status PREFAILED. In case of the occurrence of PREPASSED as the status, the De-bounce Counter will decrease by the count out step size.

7.2.11.3.1.1 Representation the DTCFaultDetectionCounter:

The counter base 1:1 relation with maximum value of 127 and the minimum value of -128. If the pre-debouncing has been finished then the DTCFaultDetectionCounter is either 127 (this means “TestFailed”) or -128 (this means “Passed”). When the debouncing is in progress the counter value can be derived from the internal counter.

Different mechanisms are possible. According to the DTCFaultDetection definition, the counter increments upon a PREFAILED report and decrements upon PREPASSED reports. Additionally, jumps are possible if a PREFAILED report comes in while the DTCFaultDetectionCounter is within the PASSED / PREPASSED range. Hence, the following table should give an overview

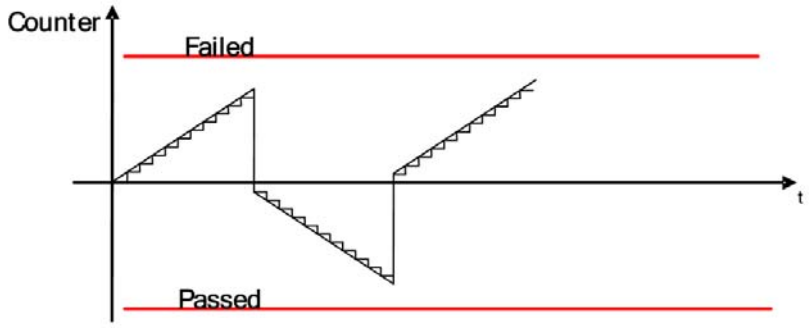
Reported result:	PREFAILED	PREPASSED
Action at continuously and repeated reporting of a result:	Increment by one step	Decrement by one step
Action after changed result being reported:	Jump UP	(Jump down) Only allowed if Jump-UP for PREFAILED reports is also activated! Otherwise, PASSED results could be faster obtained than FAILED results. This is critical from legal point of view.

Table 1 Behavior of debounce counters

Since range of -128 to +127 is fixed, different limits for FAILED and PASSED detection in the internal debouncing can be converted into the 1-byte range via different step sizes. Therefore, it shall be possible to define either a parameter set for step size (assuming equal levels for FAILED and PASSED detection) or a parameter set for the FAILED and PASSED detection (assuming an equal step size for PREFAILED and PREPASSED reports).
It is possible to combine incrementing / decrementing with jumps. If only incrementing / decrementing is used, this yields an “up-down-counter” behavior. If both types of jumps are additionally activated, this yields an “event-in-a-row”-behavior. According to ISO 14229-1 (version of Nov. 2005) the behavior of the

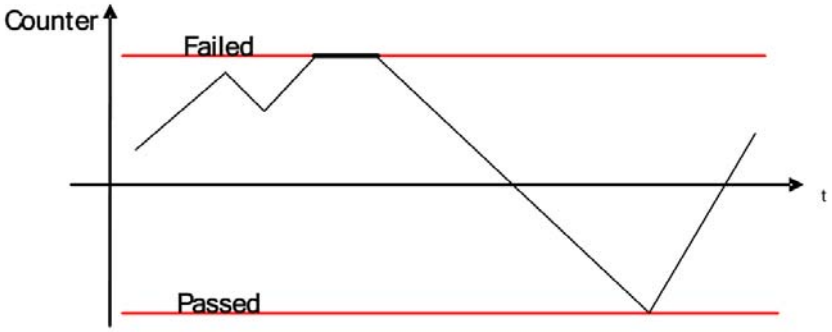
DTCFaultDetectionCounter indicates an asymmetric behavior where the jump is only active upon PREFAILED reports in order to start FAILED detection always from the “0”-level.

- Events in a row counter



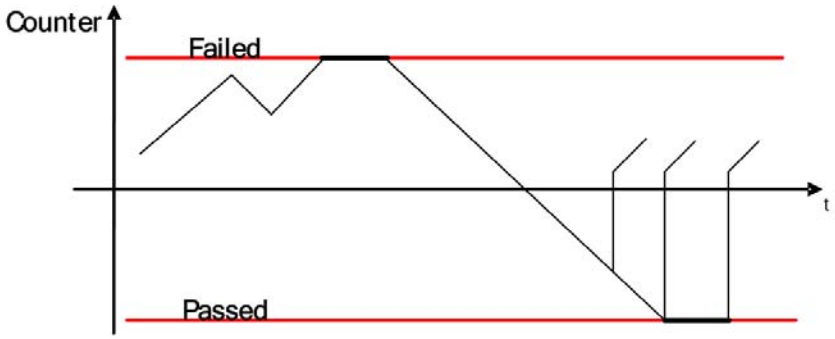
Note, that the steps should indicate individual incrementing and decrementing per report and also to show the combination of a jump and a step upon a change of the reported result.

- Events up-down-counter



This figure shows the up-down-counter behavior, whereas the range is limited by -128 to +127.

- Count-in – Count-out/Jump-in



In this figure the combination of incrementing / decrementing with the jump UP upon a PREFAILED report. This applies – independent of the degree of PREPASSED / PASSED debounding as indicated by the three possible jumps.

7.2.11.3.1.2 Use Cases

- Monitors with cyclic calls, e.g. open load detection

7.2.11.3.1.3 Parameter:

- Step size for incrementation (PREFAILED)
- Step size for decrementation (PREPASSED) result

Alternatively:

- Threshold for FAILED-detection (at this value the event is qualified to DEM_EVENT_STATUS_FAILED)
- Threshold for PASSED detection (at this value the event is qualified to DEM_EVENT_STATUS_PASSED)
- Switch for the activation of Jump-UP (boolean) (upon PREFAILED report within PREPASSED / PASSED range)
- Switch for the activation of Jump-DOWN (boolean) (upon PREFAILED report within PREPASSED / PASSED range) – only in combination with Jump-UP activation.

7.2.11.3.2 Time based

The signal is unqualified until the first call of Dem_SetEventStatus/Dem_ReportErrorStatus. During the call with status PREFAILED or PREPASSED the debounce time out is started until the event is qualified. If the status toggles, the time is restarted and the direction will change. The monitor has to continuously report in order to proceed with debouncing. Thus, the time based debouncing is comparable with event or counter based debouncing. The difference is that here a time increment is added with a size depending on the cyclic process the monitor is called. However, time based debouncing as a second mechanism is still helpful since it enhances the proper determination of the threshold parameters during calibration. Starting of some time based counter and incrementing it without repeating the report is not reasonable because the monitor might leave its physical enable window or it might be inhibited due to a fault. Then the debouncing

should not continue.Therefore, the same description as of Counter-Event based debouncing also applies here.

7.2.11.3.2.1 Representation the DTCFaultDetectionCounter:

For unqualified events and the timer is not running DTCFaultDetectionCounter shall be set to 0. While the timer is running, the DTCFaultDetectionCounter could be set to all other values other than minimum or maximum value. After the event is qualified then the DTCFaultDetectionCounter should be set to minimum or maximum value.

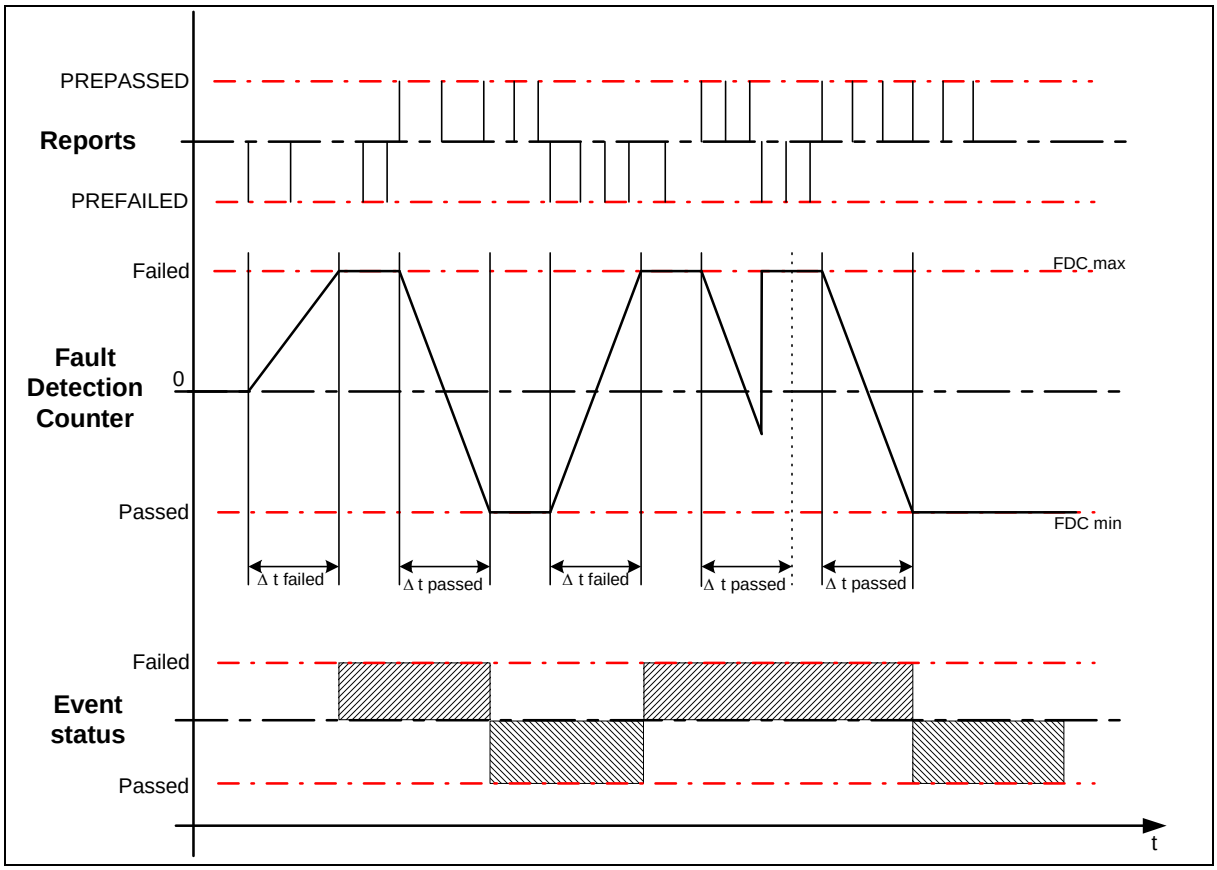


Figure 10 Representation of the DTCFaultDetectionCounter for time based debouncing

7.2.11.3.2.2 Use Cases

- Monitoring of functions with a timeout, e.g. CAN timeout.

7.2.11.3.2.3 Parameter:

- Time threshold for qualification as failed.
- Time threshold for qualification as passed.

7.2.11.3.3 Error occurrence frequency based

An event is unqualified until Dem_SetEventStatus is called. As soon as an event is reported as PreFailed/PrePassed a time window is opened. For the event qualification different counters for PreFailed (PF counts failing events) and PrePassed (PP counts passing events) are used. When one of the two counters reaches the configured threshold and the time window is still open the event is qualified as TestFailed (i.e. PF exceeds it's threshold) or TestPassed (i.e. PP exceeds it's threshold). The qualification of the event is finished as soon as one of the thresholds is reached. Reporting of the next event reopens the time window and a new qualification process starts. If neither threshold is reached within the time window the event is 'unqualified' (readiness is not set). From calibration point of view, it is a critical debouncing mechanism, because if the duration time is calibrated too short, an error would never become debounced even if the fault is constantly reported as PreFailed.

7.2.11.3.3.1 Representation the DTCFaultDetectionCounter:

When the event is 'unqualified' and the time window not open yet DTCFaultDetectionCounter shall be set to 0. While the time window is open, the DTCFaultDetectionCounter could be set to values differing from the minimum or maximum value. After the event is qualified then the DTCFaultDetectionCounter should be set to the minimum or maximum value.

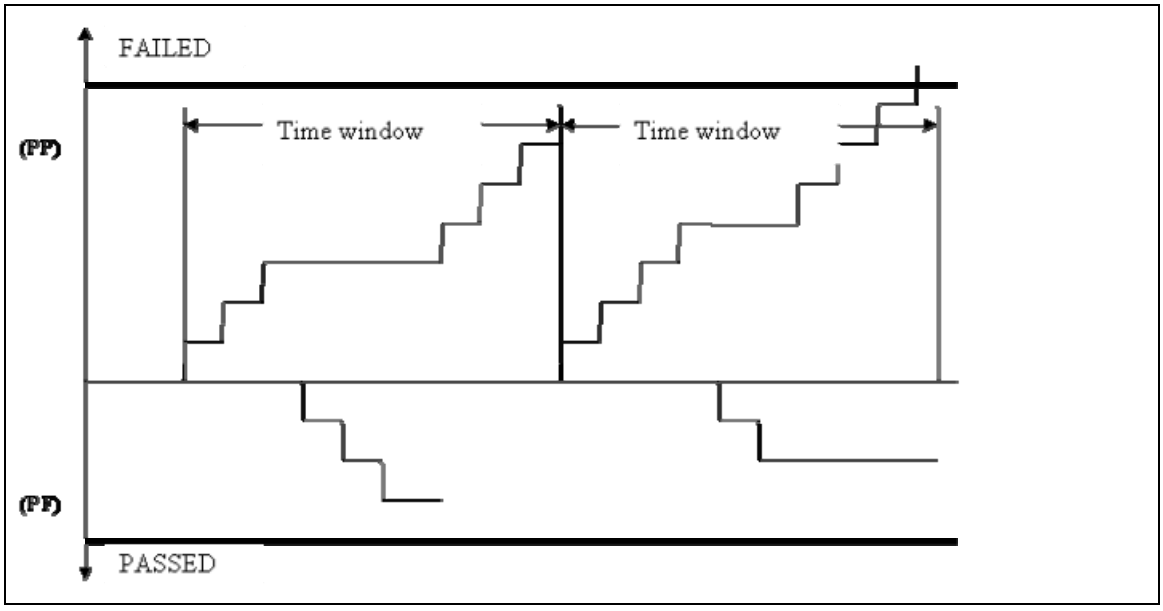


Figure 11 Representation of the DTCFaultDetectionCounter for frequency based debouncing

7.2.11.3.3.2 Use Cases

- Error Messages appear on a CAN bus due to EMC pulses.

- Whenever a message is lost, the counter (PF) increases. Whenever a message is received, the counter (PP) decreases.
- When PF reaches its threshold within the opened time window, the event is ‘qualified’ as ‘TestFailed’. When PP reaches its threshold within the opened time window the event is ‘qualified’ as ‘TestPassed’

7.2.11.3.3.3 Parameter:

- DurationOfTimeWindow in ms
- ThresholdForEventTestedFailed (PP max), threshold for FAILED-detection ((at this value the event is qualified to DEM_EVENT_STATUS_FAILED)

ThresholdForEventTestedPassed (PF max), threshold for PASSED detection (at this value the event is qualified to DEM_EVENT_STATUS_PASSED)

Dem343: After receiving a command for clearing the event memory the according debounced counters shall be initalized with 0 presuming event debouncing is handled DEM internally.

The DTCFaultDetectionCounter represents the Failed/Passed detection together with the Tested detection.

Dem344: If configured the DTCFaultDetectionCounter shall be reset when starting a new monitoring cycle.

The configuration of the reset behavior of the DTCFaultDetectionCounter is OEM specific and not defined in AUTOSAR.

If debouncing is done by SW-C (not handled DEM internally) the SW-C provides the API GetFaultDetectionCounter to access the DTCFaultDetectionCounter.

Dem345: For resetting the DTCFaultDetectionCounter implemented in a SW-C the DEM module shall use the API InitMonitorForEvent (refer to 8.4.3.1.1).

7.3 OBD-Functionality

7.3.1 General overview and restrictions

In the following, a specification of the OBD handling in AUTOSAR is introduced. Herein, “OBD” is used for automotive OBD with respect to different target markets. For SW-sharing and distributed development reasons as well as aspects of packaging and responsibility of releases, the OBD-relevant information / data strutures need to be reported via Standardized AUTOSAR interfaces.

Together with the DCM, this DEM SWS provides standardized AUTOSAR interfaces to support the OBD services $01 - $0A defined in SAE J1979 Rev May 2007 [14]. With these services, Autosar OBD functionality shall be capable of meeting all

light duty OBD regulations worldwide (California OBDII, EOBD, Japan OBD, and all others.)

Some details on the interaction between DEM and specific SW-C might remain open, since they are dependent on the DEM and SW-C implementation. The following functionality is not defined:

- Malfunction Indicator Lamp (MIL)-activation (interface DEM to MIL handler, MIL-bulb check, readiness blinking, blinking in case of catalyst damaging misfire...)
- misfire fault handling (common debouncing, filtering single / multiple misfire faults).

However, this DEM SWS does not prescribe implementation details on how OBD compliance can be achieved within the DEM module, e.g. concerning state handling. Furthermore, the DEM SWS does not prescribe implementation details on the diagnostic algorithms of the SW-C necessary to achieve OBD compliance (how to detect a fault, when to trigger incrementation of IUMPR-numerator...).

7.3.1.1 Service $01, $02 and OBD PID data

In order to retrieve relevant data upon service$01 request or a fault entry, the DEM needs access to current data, adressed via PIDs. For that purpose, a Client (=DCM/DEM)/Server (=SW-C) interface is assigned based on configuration items.

Dem291: The DEM module shall support only the legislative FreezeFrame (record number 0). This will be a single list of PIDs assigned to this FreezeFrame.

This means a request by a generic scan tool for record number one and above will be ignored.

Upon the entry of a fault in the memory, the values of these PIDs/DIDs are requested through the RTE via a client server interface.

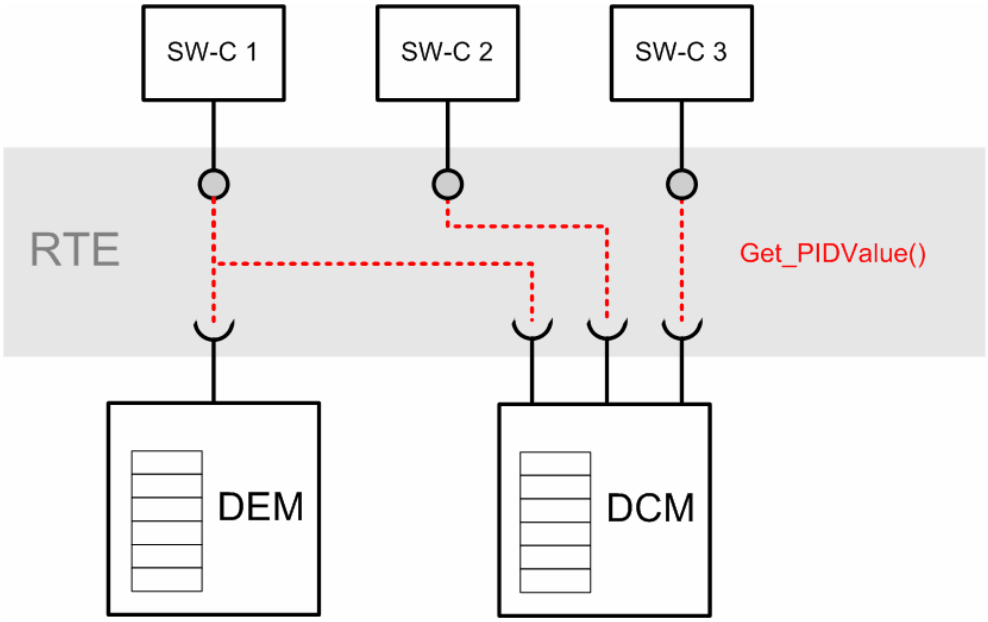


Figure 12 DEM and DCM module requests data of SW-C via Get_PIDValue()

7.3.1.2 PIDs for service$01 computed by DEM

Dem293: There are some PIDs which shall be computed directly within the DEM.

- PID$01 – Monitor status since DTCs cleared (4 byte)
- PID$02 – FreezeFrame DTC (2 byte)
- PID $21 – km with MIL On (2 byte)
- PID $30 – number of Warm Up cycles (WUC) since last fault clear (1 byte)
- PID $31 – km since last fault clear (2 byte)
- PID$41 – Monitor status this driving cycle (4 byte)
- PID $4D – time with MIL On (2 byte)
- PID $4E – time since last fault clear (2 byte)
- PID$1C – OBD requirements to which vehicle or engine is certified. (1 byte)

Any ECU supporting PID $21 and PID $31 is required to support PID $0D (vehicle speed). PID $21 and PID $31 are required for OBD ECUs.

Dem346: The DEM module shall use PID $0D to calculate PID $21 and PID $31.

Dem304: A DEM delivery shall provide function call interfaces to the DCM and the respective ServiceNeeds to declare to the DCM that these PIDs are supported.

Dem347: If PID$1E (auxiliary input status) is supported the PTO (Power Take Off) related event status handling shall implemented inside the DEM module (refer to [19]).

Dem377: The DEM module shall provide an interface Dem_SetPtoStatus allowing a SW-C implementing the PTO functionality to notify the DEM module if PTO is active or inactive (refer to section 8).

Dem378: The DEM module shall support the configuration parameter DemConsiderPtoStatus in DemEventClass indicating that a certain event is affected by the DEM PTO handling.

The DEM module provides the configuration switch [DemPTOSupport] (refer to Dem365) to enable or disable the usage of PID$1E.

A special configuration is applied for the computation of PID$01 and $41:

Dem349: The DEM module shall support DemEventClassExtended for OBD systems.

The DemEventClassExtended allows assigning an individual event to one specific Readiness group.

Dem351: The DEM module shall compute and provide the number of confirmed faults (PID$01, Byte A).

Dem352: The DEM module shall provide the MIL status.

Dem353: The DEM module shall compute the status of an Event (If there is at least on EventId assigned).

Dem354: The DEM module shall compute the ready status (If all Events of a group are reported as OK tested since last clear or Event ID has caused MIL on).

Dem355: The DEM module shall compute the group complete for current driving cycle (If all Events of a group are tested in the current driving cycle.

Dem356: The DEM module shall compute the group disabled (If the disabled status is reported by the diagnostic function for any Event of a group).

Dem348: The DEM module shall provide the disabling of events (refer to Dem_SetEventDisabled).

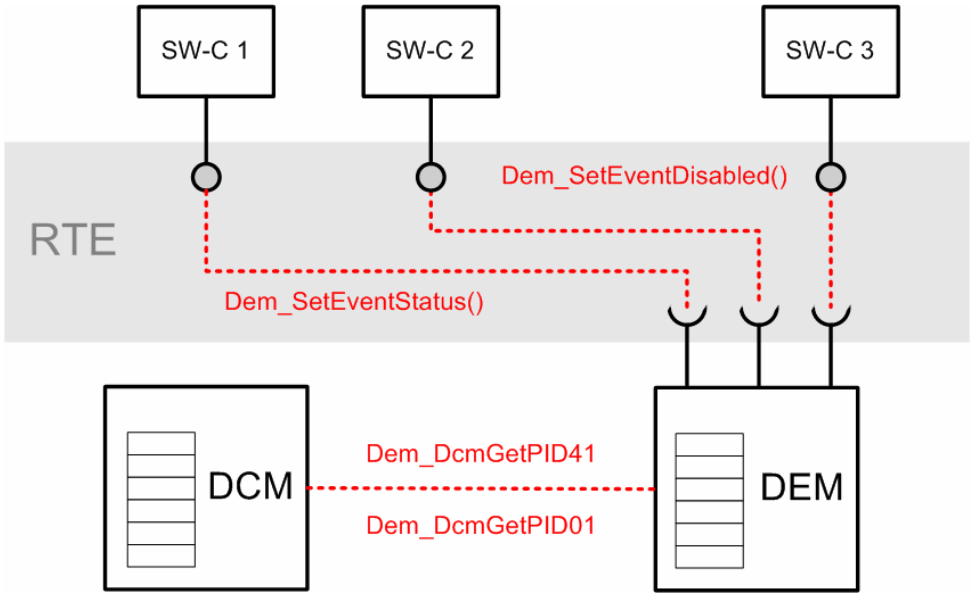


Figure 13 DEM calculates PID$01 and PID$41 data based on specific interface operations

7.3.1.3 In-Use-Monitor Performace Ratio (IUMPR) Support

The IUMPR-data collected needs to be provided upon a service $09 request. For gasoline engines the Info Type $08 is used and for diesel engines the Info Type $0B is used (refer to [13] and [15]).

Dem357: If the OBD system is a spark ignition system the DEM module shall provide the API Dem_GetInfoTypeValue08 for Info Type $08 IUMPR data.

Dem358: If the OBD system is a compression ignition system the DEM module shall provide the API Dem_GetInfoTypeValue0B for Info Type $0B IUMPR data.

The type of OBD system is defined by using the configuration switch DemOBDSupport.

The In-Use-Monitor Performace Ratio (IUMPR) indicates how often the OBD system monitors, particular components, compared to the amount of the vehicle operation. It is defined as the number of times a fault could have been found (=numerator) devided by the number of times the vehicle operation has been fulfilled (=denominator) as defined in the respective OBD regulations.

The relevant data recording is allocated in the DEM based on FIDs and EventIDs. The IUMPR data are recorded for different monitoring groups or components respectively.

Typically, one or more EventIDs serve for the monitoring of these components, e.g. the oxygen sensor. Hence, in order to record the in-use performance of the OBD-system, the test performance of all the relevant EventIDs for all the IUMPR-groups

needs to be recorded. For that purpose, certain data structure needs to be configured by the DEM.

However, in order to stop the incrementing of numerator and denominator in the case, a monitor is stopped due to the occurrence of another event preventing the monitor from running, it is necessary to list all EventIDs that can affect the computation of a particular IUMPR-relevant EventID. However, this information is already embodied in a FID (cf. SWS_FIM) representing a function which serves for the computation e.g. of a particular EvenID. A FID is inhibited in case of certain EventIDs and thus, this relation can also be used to stop the IUMPR record.

This leads to a combination of an EventID to be recorded and a FID representating all the Events stopping the computation of the IUMPR-Event. For the purpose of classifying the EventIDs into the monitors groups, an IUMPR group is also part of this combination.

For the configuration of data structure per EventID / FID / IUMPR-group combination, a new data object is introduced, namely, the RatioId. Thus, the parameter RatioId contains the EventId, FID, IUMPR-group and an interface option to configure “API-use” vs. “observer” whereas the interface option is explained in more detail in the following.

Also for the purpose of the port configuration a ObdRatioServiceNeeds is offered to the SW-C. If a SW-C is IUMPR-relevant, this ServiceNeeds is filled out.

If the diagnostic function is “symmetric”, i.e. upon completion of the test a fault could have been found, even if there is currently no fault in the system, then the numerator can be incremented just by observing the TESTED-status of the assigned EventId.

Dem359: Only for symmetric diagnostic functions the DEM module shall increment the numerator of the corresponding monitor if the SW-C updates the TESTED-status via SetEventStatus.

If the diagnostic is asymmetric and it takes more efforts to detect a malfunction than to detect an OK-status, the diagnostic function needs to call an API in order to report that a malfunction could have been found. Because this may require some simulation within the diagnostic function and can therefore not be purely derived from the TESTED status.

Dem360: For OBD relevant systems the DEM module shall provide the API Dem_RepIUMPRFaultDetect (refer to 8.3.6.2 Dem_RepIUMPRFaultDetect).

Dem361: The DEM module shall provide a configuration parameter to indicate per RatioId if the numerator is calculated based on the TESTED-status or the API call.

For some particular diagnostic functions (e.g. for secondary air system, comprehensive components), there are additional conditions defined on the denominator: Their denominator will only be incremented if certain temperature or acitivity conditions are met. Then, the diagnostic function needs to lock the

denominator per API at the beginning of the driving cycle and will unlock it when the additional conditions on the denominator are met.

Dem362: The DEM module shall provide the APIs for locking (Dem_RepIUMPRDenLock) and unlocking (Dem_RepIUMPRDenRelease) of the denominator under special conditions.

Legislation requires that IUMPR tracking shall be stopped for a specific monitor if it is inhibited by another service $07 visible fault.

Dem299: As long as an event has Pending status the DEM module shall stop increasing the numerator and denominator.

Based on the related FID (FIM access) and RatioId the DEM module can determine which numerator and denominator has to be stopped.

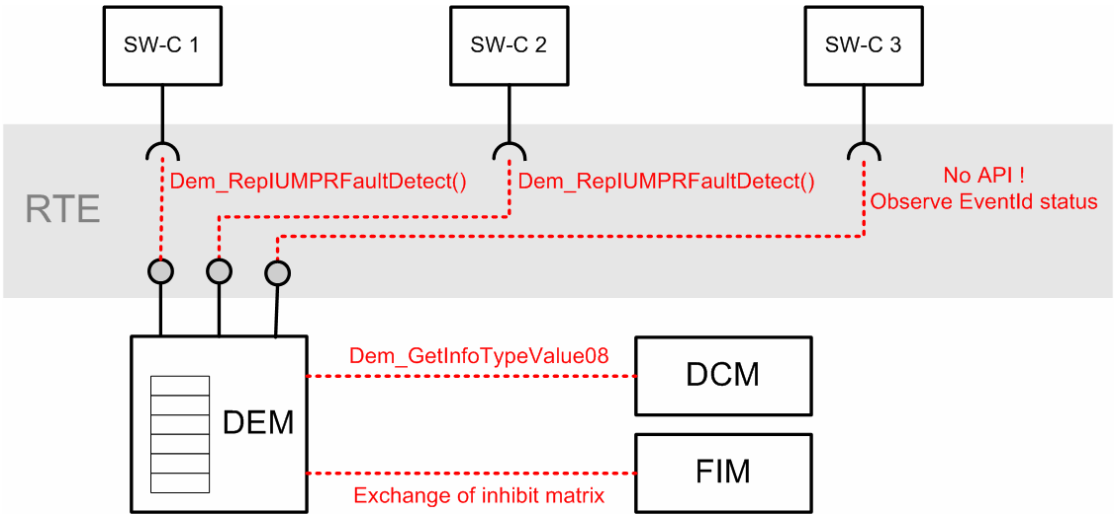


Figure 14 DEM calculates IUMPR data based on specific interface operations

7.3.1.4 Service$0A – Permanent DTC

Dem300: The DEM shall store and provide fault entries as Permanent DTC according to regulations (see ref. [17], [18]).

Dem301: The DEM shall provide the option to access the stored Permanent DTC by filtering for permanent DTCs (refer to 8.3.5.1.1 Dem_SetDTCFilter and 8.3.5.1.6 Dem_GetNextFilteredDTC).

7.3.1.5 Adapter functionality

For a proper state handling and computation of DEM-PIDs and IUMPR conditions, additional input information is required, e.g. engine temperature or vehicle speed.

Dem302: For that purpose, a DEM shall delivery port definition requesting

- engine temperature
- vehicle speed
- distance information
- programming event
- ...

whereas implementation specific extensions are possible.

7.4 Auxiliary explanations and definitions

7.4.1 Requirements on data

7.4.1.1 Data provided for the DEM module

The DEM module requires several input values for computation.

Dem278: The DEM module shall request values of event related data to be stored in FreezeFrames using DidService_<DID> via the data ID configuration table according to ISO 14229-1 [12].

Dem363: The DEM module shall request values of event related data to be stored in ExtendedDataRecords using GetExtendedDataRecord_<RECORDNUMBER> via the record number according to ISO 14229-1.

7.4.1.2 Data returned from the DEM module

Dem171: The DEM module functions with the return code Dem_ReturnGetStatusOfDTCType shall return the value WRONG_DTCORIGIN if the DEM module's environment has requested an unavailable event memory/origin.

Dem172: The DEM module functions with the return code Dem_ReturnGetStatusOfDTCType shall return the value WRONG_DTC if the DEM module's environment has requested a DTC that is available but has a different origin than the requested one.

7.5 Version check

Dem067: The DEM module's implementer shall avoid the integration of incompatible files. Minimum implementation is the version check of the header file.
For included header files:

- DEM_AR_MAJOR_VERSION
- DEM_AR_MINOR_VERSION

shall be identical. For the module internal c and h files:

- DEM_SW_MAJOR_VERSION
- DEM_SW_MINOR_VERSION
- DEM_AR_MAJOR_VERSION
- DEM_AR_MINOR_VERSION
- DEM_AR_PATCH_VERSION

shall be identical.

7.6 Error classification

Dem115: Values for production code EventIds are assigned externally by the configuration of the Dem. They are published in the file Dem_IntErrId.h and included via Dem.h.
Note, that only the BSW reports errors via the EventIDs published by Dem_IntErrId.h whereas the SW-C above the RTE report their errors via eventIDs published by Dem_IntEvtId.h.

Dem116: Development error values are of type uint8.

Dem173: The following errors shall be detectable by the DEM module depending on its configuration (development / production mode):

Type or error	*Relevance*	*Related error code*	*Value [hex]*
API service called with wrong parameter	Development	DEM_E_PARAM_CONFIG DEM_E_PARAM_ADDRESS DEM_E_PARAM_DATA DEM_E_PARAM_LENGTH	0x10 0x11 0x12 0x13
API service called before the DEM module has been full initialized (refer to Dem124, Dem364:)	Development	DEM_E_UNINIT	0x20
No valid data available by the SW-C	Development	DEM_E_NODATAAVAILABLE	0x30

Table 2 Types of errors which can be detected by the DEM module

Dem124: If any instance calls any DEM API, excluding Dem_ReportErrorStatus, before not fully initialized the DEM module shall call DET API Det_ReportError to set the error code DEM_E_UNINIT.

Dem364: If any instance calls DEM_ReportErrorStatus before not pre-initialized the DEM module shall call DET API Det_ReportError to set the error code DEM_E_UNINIT.

Dem370: If development error detection is enabled: If a DEM function returning a standard return type detects a development error, then the DEM function shall return E_NOT_OK.

7.7 Error detection

Dem113: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch *DemDevErrorDetect* (see chapter 10) shall activate or deactivate the detection of all development errors.

Dem114: If the *DemDevErrorDetect* switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.6 and chapter 8.

Dem174: The detection of production code errors cannot be switched off.

7.8 Error notification

Dem117: Detected development errors shall be reported to the *Det_ReportError* service of the Development Error Tracer (DET) if the pre-processor switch *DemDevErrorDetect* is set (see chapter 10).

8 API specification

The graphic below shows the interfaces between DEM and its surrounding software modules. The description of the interface shall give a simple overview of the relation to the DCM, SW-C, BSW and ECUM.

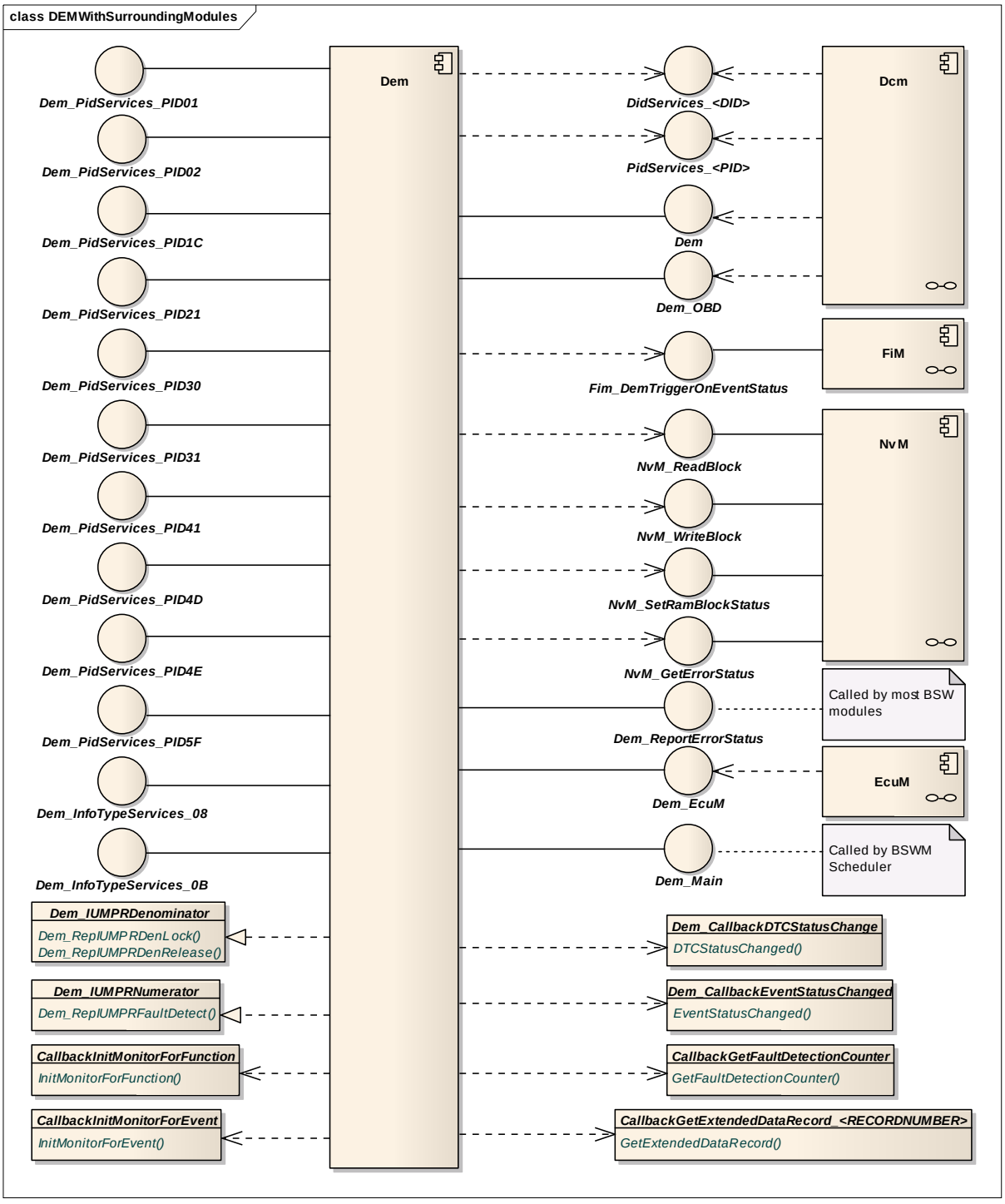


Figure 15 Overview of interfaces between the DEM and other SW modules

8.1 Imported types

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

Dem176:

Header file	*Imported Type*
NvM_Types.h	NvM_BlockIdType
Dcm_Types.h	Dcm_NegativeResponseCodeType
Std_Types.h	Std_VersionInfoType
	Std_ReturnType

8.2 Type definitions

The following Data Types shall be used for the functions defined in this specification.

8.2.1 DEM data types

8.2.1.1 Dem_EventIdType

Name:	`Dem_EventIdType`	
Type:	`uint8,uint16`	
Range:	`1...255,` `1...65535`	Identifier of event Configurable, size depends on system complexity. Remark: 0 is not a valid value
Description:	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Example: 1 refers to Monitor Function x, 2 refers to Monitor Function y, ... Small and encapsulated systems will only use uint8 for EventId definition due to resource optimization. Systems with enough resources shall use uint16. For Monitor Functions using uint8 adaptations might be required to ensure compatibility between different data types.	

8.2.1.2 Dem_DTCGroupType

Name:	Dem_DTCGroupType		
Type:	uint32		
Range:	DEM_DTC_GROUP_POWERTRAIN_DTCS	--	selects group of powertrain DTCs
	DEM_DTC_GROUP_NETWORK_COM_DTCS	--	selects group of network communication DTCs
	DEM_DTC_GROUP_CHASSIS_DTCS	--	selects group of chassis DTCs
	DEM_DTC_GROUP_ALL_DTCS	0xffffffff	selects all DTCs
	DEM_DTC_GROUP_EMISSION_REL_DTCS	0x00000000	selects group of OBD-relevant DTCs
	DEM_DTC_GROUP_BODY_DTCS	--	selects group of body DTCs
Description:	Selects a group of DTCs		

8.2.1.3 Dem_DTCKindType

Name:	Dem_DTCKindType		
Type:	uint8		
Range:	DEM_DTC_KIND_EMISSION_REL_DTCS	0x02	Select OBD-relevant DTCs
	DEM_DTC_KIND_ALL_DTCS	0x01	Select all DTCs
Description:	--		

8.2.1.4 Dem_DTCOriginType

Name:	Dem_DTCOriginType		
Type:	uint8		
Range:	DEM_DTC_ORIGIN_SECONDARY_MEMORY	0x04	Event information located in the secondary memory
	DEM_DTC_ORIGIN_PERMANENT	0x03	The Event information is located in the permanent memory
	DEM_DTC_ORIGIN_MIRROR_MEMORY	0x02	Event information located in the mirror memory,
	DEM_DTC_ORIGIN_PRIMARY_MEMORY	0x01	Event information located in the primary memory,
Description:	This enum is used to define the location of the events. The definition and use of the different memory types is OEM-specific.		

8.2.1.5 Dem_DTCRequestType

Name:	Dem_DTCRequestType		
Type:	uint8		
Range:	DEM_MOST_REC_DET_CONFIRMED_DTC	0x04	most recently detected confirmed DTC requested
	DEM_FIRST_DET_CONFIRMED_DTC	0x03	first detected confirmed DTC requested
	DEM_MOST_RECENT_FAILED_DTC	0x02	most recent failed DTC requested
	DEM_FIRST_FAILED_DTC	0x01	first failed DTC requested
Description:	--		

8.2.1.6 Dem_DTCSeverityType

Name:	Dem_DTCSeverityType		
Type:	uint8		
Range:	DEM_SEVERITY_MAINTENANCE_ONLY	0x20	maintenance required
	DEM_SEVERITY_CHECK_AT_NEXT_HALT	0x40	check at next halt
	DEM_SEVERITY_NO_SEVERITY	0x00	No severity information available
	DEM_SEVERITY_CHECK_IMMEDIATELY	0x80	Check immediately
Description:	Defines the type of a DTCSeverityMask according to ISO14229-1.		

8.2.1.7 Dem_EventStatusExtendedType

Name:	Dem_EventStatusExtendedType		
Type:	uint8		
Range:	warningIndicatorRequested	0x80	--
	testNotCompletedThisOperationCycle	0x40	--
	testFailedSinceLastClear	0x20	--
	testNotCompletedSinceLastClear	0x10	--
	confirmedDTC	0x08	--
	pendingDTC	0x04	--
	testFailedThisOperationCycle	0x02	--
	testFailed	0x01	--
Description:	In this data-type each bit has an individual meaning. The bit is set to 1 when the condition holds. For example, if the 2nd bit (0x02) is set to 1, this means that the test failed this operation cycle. If the bit is set to 0, it has not yet failed this cycle.		

8.2.1.8 Dem_FilterForFDCType

Name:	Dem_FilterForFDCType		
Type:	uint8		
Range:	DEM_FILTER_FOR_FDC_NO	0x01	Fault Detection Counter information not used
	DEM_FILTER_FOR_FDC_YES	0x00	Fault Detection Counter information used
Description:	--		

8.2.1.9 Dem_FilterWithSeverityType

Name:	Dem_FilterWithSeverityType		
Type:	uint8		
Range:	DEM_FILTER_WITH_SEVERITY_NO	0x01	Severity information not used
	DEM_FILTER_WITH_SEVERITY_YES	0x00	Severity information used,
Description:	--		

8.2.1.10 Dem_RatioIdType

Name:	Dem_RatioIdType
Type:	uint8,uint16
Description:	--

8.2.2 DEM return types

8.2.2.1 Dem_ReturnSetDTCFilterType

Name:	Dem_ReturnSetDTCFilterType		
Type:	uint8		
Range:	DEM_WRONG_FILTER	0x01	Wrong filter selected
	DEM_FILTER_ACCEPTED	0x00	Filter was accepted
Description:	Used to return the status of updating the DTC filter.		

8.2.2.2 Dem_ReturnGetStatusOfDTCType

Name:	Dem_ReturnGetStatusOfDTCType		
Type:	uint8		
Range:	DEM_STATUS_WRONG_DTCORIGIN	0x02	Wrong DTC origin
	DEM_STATUS_WRONG_DTC	0x01	Wrong DTC
	DEM_STATUS_FAILED	0x04	DTC failed
	DEM_STATUS_OK	0x00	Status of DTC is OK
	DEM_STATUS_WRONG_DTCKIND	0x03	DTC kind wrong
Description:	Used to return the status of Dem_GetStatusOfDTC.		

8.2.2.3 Dem_ReturnGetNextFilteredDTCType

Name:	Dem_ReturnGetNextFilteredDTCType		
Type:	uint8		
Range:	DEM_FILTERED_PENDING	0x03	The requested value is currently not available. The caller can retry later.
	DEM_FILTERED_WRONG_DTCKIND	0x02	DTC kind wrong
	DEM_FILTERED_NO_MATCHING_DTC	0x01	No DTC matched
	DEM_FILTERED_OK	0x00	Returned next filtered DTC
Description:	Used to return the status of Dem_GetNextFilteredDTC.		

8.2.2.4 Dem_ReturnGetNumberOfFilteredDTCType

Name:	Dem_ReturnGetNumberOfFilteredDTCType		
Type:	uint8		
Range:	DEM_NUMBER_PENDING	0x02	get of number of DTC is pending
	DEM_NUMBER_OK	0x00	get of number of DTC was successful
	DEM_NUMBER_FAILED	0x01	get of number of DTC failed
Description:	Used to return the status of Dem_GetNumberOfFilteredDTC.		

8.2.2.5 Dem_ReturnClearDTCType

Name:	Dem_ReturnClearDTCType		
Type:	uint8		
Range:	DEM_DTC_PENDING	0x05	Clearing of DTC is pending
	DEM_CLEAR_WRONG_DTCORIGIN	0x02	Wrong DTC origin

	DEM_CLEAR_FAILED	0x04	DTC not cleared
	DEM_CLEAR_OK	0x00	DTC successfully cleared
	DEM_CLEAR_WRONG_DTCKIND	0x03	DTC kind wrong
	DEM_CLEAR_WRONG_DTC	0x01	Wrong DTC
Description:	Used to return the status of Dem_ClearDTC.		

8.2.2.6 Dem_ReturnControlDTCStorageType

Name:	Dem_ReturnControlDTCStorageType		
Type:	uint8		
Range:	DEM_CONTROL_DTC_WRONG_DTCGROUP	0x02	DTC storage control not successful because group of DTC was wrong
	DEM_CONTROL_DTC_STORAGE_N_OK	0x01	DTC storage control not successful
	DEM_CONTROL_DTC_STORAGE_OK	0x00	DTC storage control successful
Description:	Used to return the status of Dem_DisableDTCStorage and Dem_EnableDTCStorage.		

8.2.2.7 Dem_ReturnControlEventUpdateType

Name:	Dem_ReturnControlEventUpdateType		
Type:	uint8		
Range:	DEM_CONTROL_EVENT_UPDATE_N_OK	0x01	Event storage control not successful
	DEM_CONTROL_EVENT_WRONG_DTCGROUP	0x02	Event storage control not successful because group of DTC was wrong
	DEM_CONTROL_EVENT_UPDATE_OK	0x00	Event storage control successful
Description:	Used to return the status of Dem_DisableEventStatusUpdate and Dem_EnableEventStatusUpdate.		

8.2.2.8 Dem_ReturnGetDTCOfFreezeFrameRecordType

Name:	Dem_ReturnGetDTCOfFreezeFrameRecordType		
Type:	uint8		
Range:	DEM_GET_DTCOFFF_NO_DTC_FOR_RECORD	0x02	No DTC for record available
	DEM_GET_DTCOFFF_WRONG_DTCKIND	0x03	DTC kind wrong
	DEM_GET_DTCOFFF_OK	0x00	DTC successfully returned
	DEM_GET_DTCOFFF_WRONG_RECORD	0x01	Wrong record

Description:	Used to return the status of Dem_GetDTCOfFreezeFrameRecord.

8.2.2.9 Dem_ReturnGetFreezeFrameDataIdentifierByDTCType

Name:	Dem_ReturnGetFreezeFrameDataIdentifierByDTCType		
Type:	uint8		
Range:	DEM_GET_ID_WRONG_FF_TYPE	0x04	FreezeFrame type wrong
	DEM_GET_ID_WRONG_DTCKIND	0x03	DTC kind wrong
	DEM_GET_ID_WRONG_DTCORIGIN	0x02	Wrong DTC origin
	DEM_GET_ID_WRONG_DTC	0x01	Wrong DTC
	DEM_GET_ID_OK	0x00	FreezeFrame data identifier successfully returned
Description:	Used to return the status of Dem_GetFreezeFrameDataIdentifierByDTC.		

8.2.2.10 Dem_ReturnGetExtendedDataRecordByDTCType

Name:	Dem_ReturnGetExtendedDataRecordByDTCType		
Type:	uint8		
Range:	DEM_RECORD_PENDING	0x06	The requested value is currently not available. The caller can retry later.
	DEM_RECORD_OK	0x00	Extended data record successfully returned
	DEM_RECORD_WRONG_DTC	0x01	Wrong DTC
	DEM_RECORD_WRONG_DTCKIND	0x03	DTC kind wrong
	DEM_RECORD_WRONG_DTCORIGIN	0x02	Origin wrong
	DEM_RECORD_WRONG_BUFFERSIZE	0x05	Provided buffer to small
	DEM_RECORD_WRONG_NUMBER	0x04	Record number wrong
Description:	Used to return the status of Dem_GetExtendedDataRecordByDTC.		

8.2.2.11 Dem_ReturnGetDTCByOccurrenceTimeType

Name:	Dem_ReturnGetDTCByOccurrenceTimeType		
Type:	uint8		
Range:	DEM_OCCURR_WRONG_DTCKIND	0x01	DTC kind wrong
	DEM_OCCURR_OK	0x00	Status of DTC was OK
	DEM_OCCURR_FAILED	0x02	DTC failed
Description:	Status of the operation of type Dem_ReturnGetDTCByOccurrenceTime.		

8.2.2.12 Dem_ReturnGetFreezeFrameDataByDTCType

Name:	Dem_ReturnGetFreezeFrameDataByDTCType		
Type:	uint8		
Range:	DEM_GET_ID_PENDING	0x07	The requested value is currently not available. The caller can retry later.
	DEM_GET_FFDATABYDTC_WRONG_BUFFERSIZE	0x06	provided buffer size to small
	DEM_GET_FFDATABYDTC_WRONG_DATAID	0x05	Wrong DataID
	DEM_GET_FFDATABYDTC_OK	0x00	Size successfully returned.
	DEM_GET_FFDATABYDTC_WRONG_DTC	0x01	Wrong DTC
	DEM_GET_FFDATABYDTC_WRONG_DTCKIND	0x03	DTC kind wrong
	DEM_GET_FFDATABYDTC_WRONG_RECORDNUMBER	0x04	Wrong Record Number
	DEM_GET_FFDATABYDTC_WRONG_DTCORIGIN	0x02	Wrong DTC origin
Description:	Used to return the status of Dem_GetFreezeFrameDataByDTC.		

8.2.2.13 Dem_ReturnGetSizeOfExtendedDataRecordByDTCType

Name:	Dem_ReturnGetSizeOfExtendedDataRecordByDTCType		
Type:	uint8		
Range:	DEM_GET_SIZEOFEDRBYDTC_PENDING	0x05	The requested value is currently not available. The caller can retry later.
	DEM_GET_SIZEOFEDRBYDTC_W_DTC	0x01	Wrong DTC
	DEM_GET_SIZEOFEDRBYDTC_OK	0x00	Size successfully returned.
	DEM_GET_SIZEOFEDRBYDTC_W_DTCKI	0x03	DTC kind wrong
	DEM_GET_SIZEOFEDRBYDTC_W_RNUM	0x04	Wrong Record Number
	DEM_GET_SIZEOFEDRBYDTC_W_DTCOR	0x02	Wrong DTC origin
Description:	Used to return the status of Dem_GetSizeOfExtendedDataRecordByDTC.		

8.2.2.14 Dem_ReturnGetSizeOfFreezeFrameType

Name:	Dem_ReturnGetSizeOfFreezeFrameType		
Type:	uint8		
Range:	DEM_GET_SIZEOFFF_PENDING	0x05	The requested value is currently not available. The caller can retry later.
	DEM_GET_SIZEOFFF_WRONG_DTCOR	0x02	Wrong DTC origin
	DEM_GET_SIZEOFFF_WRONG_DTCKIND	0x03	DTC kind wrong

	DEM_GET_SIZEOFFF_OK	0x00	Size successfully returned.
	DEM_GET_SIZEOFFF_WRONG_RNUM	0x04	Wrong Record Number
	DEM_GET_SIZEOFFF_WRONG_DTC	0x01	Wrong DTC
Description:	Used to return the status of Dem_GetSizeOfFreezeFrame.		

8.2.2.15 Dem_ReturnGetSeverityOfDTCType

Name:	Dem_ReturnGetSeverityOfDTCType		
Type:	uint8		
Range:	DEM_GET_SEVERITYOFDTC_NOSEVERITY	0x03	Severity information is not available
	DEM_GET_SEVERITYOFDTC_WRONG_DTCORIGIN	0x02	Wrong DTC origin
	DEM_GET_SEVERITYOFDTC_WRONG_DTC	0x01	Wrong DTC
	DEM_GET_SEVERITYOFDTC_OK	0x00	Severity successfully returned.
Description:	Used to return the status of Dem_GetSeverityOfDTC.		

8.3 Function definitions

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

8.3.1 Dem_GetVersionInfo

Dem177:

Service name:	Dem_GetVersionInfo	
Syntax:	void Dem_GetVersionInfo(Std_VersionInfoType* versioninfo)	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Returns the version information of this module.	

Dem110: The function Dem_GetVersionInfo shall return the version information of this module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers (BSW00407).

Dem111: The function Dem_GetVersionInfo shall be precompile time configurable (ON/OFF) by the configuration parameter DemVersionInfoApi

Dem178: If source code for caller and callee of Dem_GetVersionInfo is available, the DEM module should realize Dem_GetVersionInfo as a macro, defined in the module's header file.

8.3.2 Interface ECU State Manager ⇔ DEM

8.3.2.1 Dem_PreInit

Dem179:

Service name:	Dem_PreInit
Syntax:	`void Dem_PreInit( )`
Service ID[hex]:	0x01
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Initializes the internal states necessary to process events reported by BSW-modules

Dem180: The function Dem_PreInit shall initialize the internal states of the DEM module necessary to process events reported by BSW modules by using Dem_ReportErrorStatus.

The function DEM_PreInit is called by the ECU State Manager during the startup phase of the ECU before the NVRAM Manager is initialized.

8.3.2.2 Dem_Init

Dem181:

Service name:	Dem_Init
Syntax:	`void Dem_Init( )`
Service ID[hex]:	0x02
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Initializes this module.

The function Dem_Init is used during the startup phase of the ECU after the NVRAM Manager has finished the restore of NVRAM data. SW-Components including Monitor Functions are initialized afterwards. The function Dem_Init is also used to reinitialize the DEM module after the Dem_Shutdown was called.

Caveats of Dem_Init: The DEM module is not functional until the DEM module's environment has called the function Dem_Init.

8.3.2.3 Dem_Shutdown

Dem182:

Service name:	Dem_Shutdown
Syntax:	`void Dem_Shutdown( )`
Service ID[hex]:	0x03
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Shutdowns this module.

Caveats of Dem_Shutdown: Once this function has been executed, no further updates are applied to the DEM module internal event data. Further requirements are depending on OEM specific needs.

Dem368: After Dem_Shutdown has been called the DEM module shall ignore all function calls until Dem_Init is called again.

8.3.3 Interface SW-Components via RTE ⇔ DEM

8.3.3.1 Dem_SetEventStatus

Dem183:

Service name:	Dem_SetEventStatus	
Syntax:	`Std_ReturnType Dem_SetEventStatus( Dem_EventIdType EventId, uint8 EventStatus )`	
Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event) Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)
	EventStatus	uint8 {DEM_EVENT_STATUS_PASSED, DEM_EVENT_STATUS_FAILED, DEM_EVENT_STATUS_PREPASSED, DEM_EVENT_STATUS_PREFAILED [, Custom]}
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: set of event status was successful E_NOT_OK: set of event status failed or could not be accepted (e.g.: the operation cycle configured for this event has not been started, the event status update has been disabled)
Description:	Set the status of an event.	

Dem184: The function Dem_SetEventStatus shall store the event related data to the Event Memory.

Dem091: The function Dem_SetEventStatus may directly change the status of events (DEM_EVENT_STATUS_PASSED, DEM_EVENT_STATUS_FAILED).

Dem190: The function Dem_SetEventStatus shall trigger the FreezeFrame storage.

Dem192: If the the function Dem_SetEventStatus is used with a pre-debounced status (DEM_EVENT_STATUS_PASSED, DEM_EVENT_STATUS_FAILED), then a pre-stored FreezeFrame of the corresponding event shall be discarded (same behaviour like Dem_ClearPrestoredFreezeFrame).

Caveats of Dem_SetEventStatus: DEM configuration during integration of Monitor Functions is system specific.

The DEM module's environment shall use the function Dem_SetEventStatus to report an event status as soon as a new test result is available.

The function Dem_SetEventStatus will be called by a Monitor Function. [ref. to Dem330: , Dem334:]

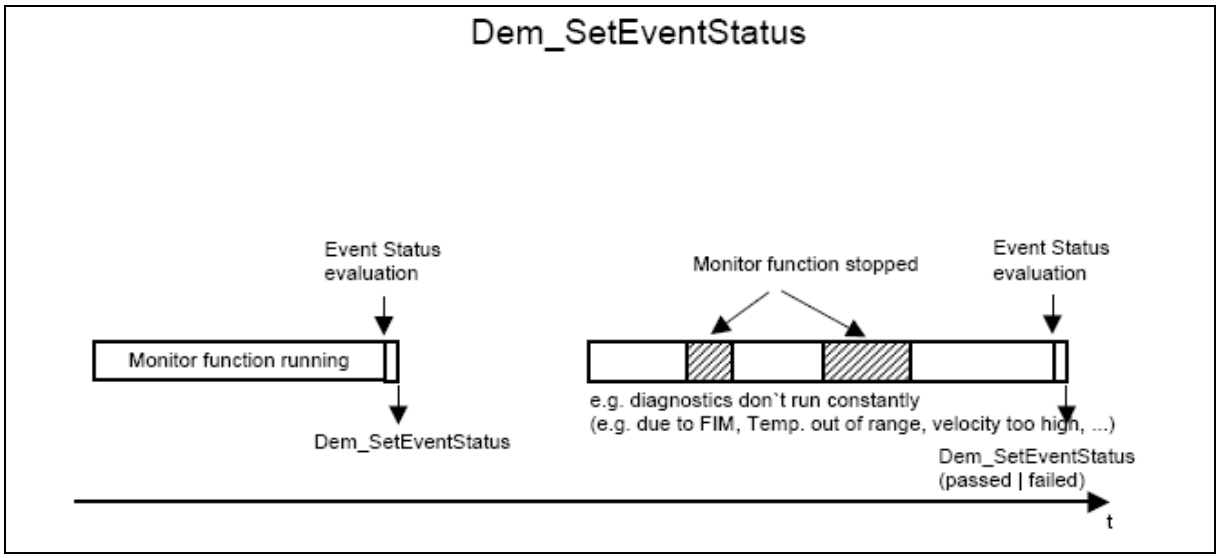


Figure 16 Example for using Dem_SetEventStatus

8.3.3.2 Dem_ResetEventStatus

Dem185:

Service name:	Dem_ResetEventStatus	
Syntax:	`Std_ReturnType Dem_ResetEventStatus( Dem_EventIdType EventId )`	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: reset of event status was successful E_NOT_OK: reset of event status failed or was not accepted (e.g. event status update has been disabled)

Description:	Resets the Event Status stored in the Event Memory in the DEM.

Dem186: The function Dem_ResetEventStatus may reset the Event Status stored in the Event Memory in the DEM module without the usage of the function Dem_SetEventStatus, because no new test result is available at this time.

Dem187: The function Dem_ResetEventStatus shall set the status bit “Failed”/ Bit 0 defined by StatusOfDTC (ISO 14229-1 [12]) to 0.

The function Dem_ResetEventStatus will be called by a Monitor Function in order to deactivate a potential default mode of operation (limp home) thus a new test can be executed.

The function Dem_ResetEventStatus does not influence the status bit 6 (“TestNotCompletedThisMonitoringCycle”). [ref. to Dem331: , Dem334:] and the pre-stored FreezeFrame. [ref. to Dem331: , Dem334:].

Refer to ISO 14229: DTC Status Bit Definition, Table D.14, Bit0 Test failed and Bit6 TestNotCompletedThisMonitoringCycle.

Caveats of Dem_ResetEventStatus: DEM configuration during integration of Monitor Functions is system specific.

8.3.3.3 Dem_PrestoreFreezeFrame

Dem188:

Service name:	Dem_PrestoreFreezeFrame	
Syntax:	`Std_ReturnType Dem_PrestoreFreezeFrame( Dem_EventIdType EventId )`	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.: Result of configuration of EventId’s in DEM (Max is either 255 or 65535)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK PreStoreFreezeFrame was successful E_NOT_OK PreStoreFreezeFrame failed
Description:	Captures the FreezeFrame data for a specific EventId.	

Dem189: The function Dem_PrestoreFreezeFrame shall capture the FreezeFrame data for a specific EventId before the Monitor Function reports the event status DEM_EVENT_STATUS_FAILED to the DEM module by calling Dem_SetEventStatus (e.g. rapid changing of event related data during running failure monitoring phase).

Dem191: The capture of FreezeFrames shall be linked to the function Dem_SetEventStatus if the DEM module does not receive any request to pre-store a FreezeFrame.

The function Dem_PrestoreFreezeFrame will be called by a Monitor Function.

Cavetas of Dem_PrestoreFreezeFrame: DEM configuration during integration of Monitor Functions is system specific.

Configuration of Dem_PrestoreFreezeFrame: During the configuration of the DEM module the capability of pre-store functionality for the required event has to be defined.

8.3.3.4 Dem_ClearPrestoredFreezeFrame

Dem193:

Service name:	Dem_ClearPrestoredFreezeFrame	
Syntax:	`Std_ReturnType Dem_ClearPrestoredFreezeFrame( Dem_EventIdType EventId )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: ClearPreStoreFreezeFrame was successful E_NOT_OK: ClearPreStoreFreezeFrame failed
Description:	Clears a prestored FreezeFrame	

Dem050: The function Dem_ClearPrestoredFreezeFrame shall delete or release the pre-stored FreezeFrame for specific EventId, if the affiliated EventID is configured as capable of pre-store functionality.

The function Dem_ClearPrestoredFreezeFrame has the same effect like the function call Dem_SetEventStatus (DEM_EVENT_STATUS_PASSED|DEM_EVENT_STATUS_FAILED) – that means it's not necessary to call the function Dem_ClearPrestoredFreezeFrame directly after Dem_SetEventStatus.

Caveats of Dem_ClearPrestoredFreezeFrame: DEM configuration during integration of Monitor Functions is system specific.

Configuration of Dem_ClearPrestoredFreezeFrame: During configuration of the DEM module the capability of pre-store functionality for the required event has to be defined.

8.3.3.5 Dem_SetOperationCycleState

Dem194:

Service name:	Dem_SetOperationCycleState	
Syntax:	`Std_ReturnType Dem_SetOperationCycleState( uint8 OperationCycleId, uint8 CycleState )`	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OperationCycleId	Identification of operation cycle, like power cycle, driving cycle.
	CycleState	DEM_CYCLE_STATE_END 0x02 End of operation cycle, DEM_CYCLE_STATE_START 0x01 Start of operation cycle
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: set of operation cycle was successful E_NOT_OK: set of operation cycle failed
Description:	Sets an operation cycle state.	

Dem047: DEM function Dem_SetOperationCycleState shall be called by the SW-Component as soon as it detects the status change of the CycleState for the Operation Cycle.

The functionality Operation Cycle State Handling can be DEM module internal for DEM module self calculated operation cycles.

Configuration of Dem_SetOperationCycleState: The OperationCycleId shall be configured in view of sender receiver communication.

8.3.3.6 Dem_GetEventStatus

Dem195:

Service name:	Dem_GetEventStatus	
Syntax:	`Std_ReturnType Dem_GetEventStatus( Dem_EventIdType EventId, Dem_EventStatusExtendedType* EventStatusExtended )`	
Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event) Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)
Parameters (inout):	None	
Parameters (out):	EventStatusExtended	Bit 0 TestFailed is set to 1 if the last event status update by the function Dem_SetEventStatus(Passed | Failed) was called with failed. The status is set to 0 if Dem_SetEventStatus is called with passed, on tester clear command and by API Dem_ResetEventStatus. Bit 0 and 6 is intended to set/reset monitor inhibit or default. Bit 1 TestFailedThisOperationCycle is set if at least one time the function Dem_SetEventStatus (passed | failed) is called with failed this cycle. Intended to be used for defaults reset only at next key on. Bit 2 PendingDTC is set when associated DTC becomes available in Mode07 (currently corresponds to ISO pending bit 2 [9] Intended to be used for the control of IUMPR counters. Bit 3 ConfirmedDTC is set when associated DTC becomes available in Mode03 (currently corresponds to ISO confirmed bit 3 [9] Could be used to set e.g. service request message. Bit 4 TestNotCompletedSinceLastClear is set to 0 if at least one time the function Dem_SetEventStatus (passed | failed) is called after last ClearDTC. Bit 5 testFailedSinceLastClear is set to 0 if at least one time the function Dem_SetEventStatus is caled with failed this cycle. Bit 6 TestNotCompletedThisOperationCycle is set if at least one time the function Dem_SetEventStatus (passed | failed) is called within this cycle (the usage of different cycles is application-specific, if only one cycle is used, the differentiation is obsolete). Bit 7 WarningIndicatorRequested reports the status of any warning indicators associated with a particular DTC.
Return value:	Std_ReturnType	E_OK: get of event status was successful E_NOT_OK: get of event status failed
Description:	Gets the current extended event status of an event.	

Dem051: The function Dem_GetEventStatus shall read the extended event status from the DEM module for a specific event.

The function Dem_GetEventStatus is provided to be used by SW-Components or other basic software modules e.g. FIM.

For the DCM, the DEM module's environment shall use the function Dem_GetStatusOfDTC instead of the function Dem_GetEventStatus.

8.3.3.7 Dem_GetEventFailed

Dem196:

Service name:	Dem_GetEventFailed	
Syntax:	`Std_ReturnType Dem_GetEventFailed( Dem_EventIdType EventId, boolean* EventFailed )`	
Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event) Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)
Parameters (inout):	None	
Parameters (out):	EventFailed	TRUE - Last Failed FALSE - not Last Failed
Return value:	Std_ReturnType	E_OK: get of "EventFailed" was successful E_NOT_OK: get of "EventFailed" was not successful
Description:	Gets the event failed status of an event.	

Dem052: The function Dem_GetEventFailed shall report the status of TestFailed of the requested Diagnostic Event.

For the DCM, the DEM module's environment shall use the function Dem_GetStatusOfDTC instead of the function Dem_GetEventFailed.

8.3.3.8 Dem_GetEventTested

Dem197:

Service name:	Dem_GetEventTested

Syntax:	`Std_ReturnType Dem_GetEventTested( Dem_EventIdType EventId, boolean* EventTested )`	
Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event) Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)
Parameters (inout):	None	
Parameters (out):	EventTested	TRUE - event tested this cycle FALSE - event not tested this cycle
Return value:	Std_ReturnType	E_OK: get of event state "tested" successful E_NOT_OK: get of event state "tested" failed
Description:	Gets the event tested status of an event.	

Dem053: The function Dem_GetEventTested shall read the negated TestNotCompletedThisOperationCycle status of the requested Diagnostic Event.

For the DCM, the DEM module's environment shall use the function Dem_GetStatusOfDTC instead of the function Dem_GetEventTested.

8.3.3.9 Dem_GetDTCOfEvent

Dem198:

Service name:	Dem_GetDTCOfEvent	
Syntax:	`Std_ReturnType Dem_GetDTCOfEvent( Dem_EventIdType EventId, Dem_DTCKindType DTCKind, uint32* DTCOfEvent )`	
Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)
	DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs
Parameters (inout):	None	
Parameters (out):	DTCOfEvent	Receives the DTC value returned by the function. If the return value of the function is other than E_OK this parameter does not contain valid data.
Return value:	Std_ReturnType	E_OK: get of DTC was successful

		E_NOT_OK: the call was not successful E_NO_DTC_AVAILABLE: there is no DTC
Description:	Gets the DTC of an event.	

Dem269: The function Dem_GetDTCOfEvent shall get the DTC which is mapped to EventId by the DEM Configuration.

8.3.3.10 Dem_SetValueByOemId

Dem199:

Service name:	Dem_SetValueByOemId	
Syntax:	`Std_ReturnType Dem_SetValueByOemId( uint16 OemID, uint8* DataValue, uint8 BufferLength )`	
Service ID[hex]:	0x38	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OemID	This OEM specific parameter identifies a data value the DEM requires for internal processing, e.g. vehicle speed or mileage.
	BufferLength	Data length of the value to be set
Parameters (inout):	None	
Parameters (out):	DataValue	Pointer to the buffer with the value to be set
Return value:	Std_ReturnType	In case the data value could be set successfully the API call returns E_OK. If the setting of the data value failed the return value of the function is E_NOT_OK.
Description:	Sets a data value assigned to a specific data identifier	

Dem200: The function Dem_SetValueByOemId shall set a data value assigned to a specific data identifier.

The list of data identifiers is OEM specific and has to be fixed at configuration time. Only simple data types (uint8… uint32; sint8…sint32) are allowed. Structured data types (struct, array) are not allowed.

Configuration of Dem_SetValueByOemId: OemID and real name of the assigned value

8.3.3.11 Dem_SetEnableCondition

Dem201:

Service name:	Dem_SetEnableCondition
Syntax:	`Std_ReturnType Dem_SetEnableCondition( uint8 EnableConditionID, boolean ConditionFulfilled`

	)	
Service ID[hex]:	0x39	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	EnableConditionID	This parameter identifies the enable condition.
	ConditionFulfilled	This parameter specifies whether the enable condition assigned to the EnableConditionID is fulfilled (TRUE) or not fulfilled (FALSE).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	In case the enable condition could be set successfully the API call returns E_OK. If the setting of the enable condition failed the return value of the function is E_NOT_OK.
Description:	Sets the enable condition.	

Dem202: The function Dem_SetEnableCondition may set the enable condition.

For each event an enable condition value is assigned to. An enable condition specifies a certain number of checks (e.g. correct voltage range) for an event before the event can be qualified as confirmed.

The function Dem_SetEnableCondition is **optional** and depends on the automotive manufacturer.

Required configuration of Dem_SetEnableCondition parameters per event:

- EnableConditionID
- EnableConditionStatus

8.3.3.12 Dem_GetFaultDetectionCounter

Dem203:

Service name:	Dem_GetFaultDetectionCounter	
Syntax:	`Std_ReturnType Dem_GetFaultDetectionCounter( Dem_EventIdType EventId, sint8* EventIdFaultDetectionCounter )`	
Service ID[hex]:	0x3e	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	EventId	Provide the EventId value the fault detection counter is requested for. If the return value of the function is other than OK this parameter does not contain valid data.
Parameters (inout):	None	
Parameters (out):	EventIdFaultDetectionCounter	This parameter receives the Fault Detection Counter information of the requested EventId. If the return value of the function call is other than E_OK

		this parameter does not contain valid data. -128dec...127dec PASSED... FAILED according to ISO 14229-1
Return value:	Std_ReturnType	E_OK: request of severity was successful E_NOT_OK: request of severity failed
Description:	Gets the fault detection counter of an event.	

Dem204: The function Dem_GetFaultDetectionCounter shall request the current Fault Detection Counter for a given EventID.

8.3.3.13 Dem_GetIndicatorStatus

Dem205:

Service name:	Dem_GetIndicatorStatus	
Syntax:	`Std_ReturnType Dem_GetIndicatorStatus( uint8 IndicatorId, uint8* IndicatorStatus )`	
Service ID[hex]:	0x29	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	IndicatorId	Number of indicator
Parameters (inout):	None	
Parameters (out):	IndicatorStatus	Status of the indicator, like on, off, blinking. DEM_INDICATOR_BLINK_CONT 0x03 Continuous and blinking mode, DEM_INDICATOR_CONTINUOUS 0x01 Continuous on, DEM_INDICATOR_OFF 0x00 Indicator off, DEM_INDICATOR_BLINKING 0x02 blinking mode
Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Description:	Gets the indicator status derived from the event status.	

Dem046: The function Dem_GetIndicatorStatus shall read the indicator-status derived from the event status as a summary of all assigned events.

Configuration of Dem_GetIndicatorstatus: The assignment for the Dem_IndicatorId to indicator has to be done. Examples for indicators: lamps, different text messages, icons, …

8.3.4 Interface BSW-Components ⇔ DEM

8.3.4.1 Dem_ReportErrorStatus

Dem206:

Service name:	Dem_ReportErrorStatus	
Syntax:	`void Dem_ReportErrorStatus( Dem_EventIdType EventId, uint8 EventStatus )`	
Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an Event by assigned Event ID. The Event ID is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.: Result of configuration of Event IDs in DEM (Max is either 255 or 65535)
	EventStatus	uint8 {DEM_EVENT_STATUS_PASSED, DEM_EVENT_STATUS_FAILED, DEM_EVENT_STATUS_PREPASSED, DEM_EVENT_STATUS_PREFAILED [, Custom]}
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Reports errors to the DEM.	

8.3.5 Interface DCM ⇔ DEM

A further description of the usage of the interface between DCM and DEM can be found in chapter 7.3.3.5 of the DCM SWS document. Here, especially the handling of FreezeFrame data is described.

8.3.5.1 Access DTCs and Status Information

The following chapter defines the API calls that shall be used to access the number of DTCs, DTCs matching specific filter criteria and the associated status information.

8.3.5.1.1 Dem_SetDTCFilter

Dem208:

Service name:	Dem_SetDTCFilter	
Syntax:	`Dem_ReturnSetDTCFilterType Dem_SetDTCFilter( uint8 DTCStatusMask, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, Dem_FilterWithSeverityType FilterWithSeverity, Dem_DTCSeverityType DTCSeverityMask, Dem_FilterForFDCType FilterForFaultDetectionCounter )`	
Service ID[hex]:	0x13	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTCStatusMask	According ISO14229-1 StatusOfDTC Values: 0x00: Report all supported DTCs 0x01...0xFF: Match DTCStatusMask as defined in ISO14229-1
	DTCKind	Defines the functional group of DTCs to be reported (e.g. all DTC, OBD-relevant DTC)
	DTCOrigin	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.
	FilterWithSeverity	This flag defines whether severity information (ref. to parameter below) shall be used for filtering. This is to allow for coexistence of DTCs with and without severity information.
	DTCSeverityMask	This parameter contains the DTCSeverityMask according to ISO14229-1 (see for example Service 0x19, subfunction 0x08)
	FilterForFaultDetectionCounter	This flag defines whether Fault Detection Counter information shall be used for filtering. This is to allow for coexistence of DTCs with and without Fault Detection Counter information. If Fault Detection Counter information is filter criteria, only those DTCs with a Fault Detection Counter value between 1 and 0x7E shall be reported. Remark: If the event does not uses the debouncing inside DEM, then the DEM must request this information via GetFaultDetectionCounter.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_ReturnSetDTCFilterType	Status of the operation of type Dem_ReturnSetDTCFilterType
Description:	Sets the filter mask over all DTCs. The server shall perform a bit-wise logical AND-ing operation between the mask specified in the client's request and the actual status associated with each DTC supported by the server. In addition to the DTCStatusAvailabilityMask, the server shall return all DTCs for which the result of the AND-ing operation is non-zero [i.e. (statusOfDTC & DTCStatusMask) != 0]. If the client specifies a status mask that contains bits that the server does not support, then the server shall process the DTC information using only the bits that it does support. If no DTCs within the server	

	match the masking criteria specified in the client's request, no DTC or status information shall be provided following the DTCStatusAvailabilityMask byte in the positive response message. ((statusOfDTC & DTCStatusMask) & (severity & DTCSeverityMask)) != 0

Dem057: The function Dem_SetDTCFilter shall set the filter mask attributes to be used for the sub-sequent calls of Dem_GetNextFilteredDTC, Dem_GetNextFilteredDTCAndFDC as well as Dem_GetNextFilteredDTCAndSeverity and reset the internal counter to the first event. The filter mask attributes shall be used until the next call of Dem_SetDTCFilter or DEM_init.

When paged buffering is used by the transport protocol, the total length of the reponse is required to be contained in the first frame. Thus the DCM must set the DTC filter status in DEM, then request and count the DTCs which matched the requested filter mask and calculate the total response length. Afterwards a second run is needed to actually send the data as a diagnostic response. Since DTC status changes may occur between these two runs, the actual response may be of a different length and would result in a response error. Thus certain diagnostic services should not be used with paged buffering. Refer to the DCM SWS specification, chapter “Limitations” for a list of diagnostic services and sub-functions which should not be used with paged buffering.

8.3.5.1.2 Dem_SetDTCFilterForRecords

Dem209:

Service name:	Dem_SetDTCFilterForRecords	
Syntax:	`Dem_ReturnSetDTCFilterType Dem_SetDTCFilterForRecords( uint16* NumberOfFilteredRecords )`	
Service ID[hex]:	0x3f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	NumberOfFilteredRecords	Number of snapshot records currently stored in the event memory.
Return value:	Dem_ReturnSetDTCFilterType	Status of the operation of type Dem_ReturnSetDTCFilterType
Description:	Sets DTC Filter for records.	

Dem210: The function Dem_SetDTCFilterForRecords shall retrieve the filtered snapshot records. This filter always belongs to primary memory.

8.3.5.1.3 Dem_GetStatusOfDTC

Dem212:

Service name:	Dem_GetStatusOfDTC	
Syntax:	`Dem_ReturnGetStatusOfDTCType Dem_GetStatusOfDTC( uint32 DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, uint8* DTCStatus )`	
Service ID[hex]:	0x15	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	For this DTC its status is requested
	DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs
	DTCOrigin	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.
Parameters (inout):	None	
Parameters (out):	DTCStatus	This parameter receives the status information of the requested DTC. If the return value of the function call is other than DEM_STATUS_OK this parameter does not contain valid data. 0x00...0xFF match DTCStatusMask as defined in ISO14229-1
Return value:	Dem_ReturnGetStatusOfDTCType	Status of the operation of type Dem_ReturnGetStatusOfDTCType.
Description:	Gets the status of a DTC	

Dem059: The function Dem_GetStatusOfDTC shall read the status of a DTC to the parameter DTCStatus according to ISO 14229-1 [12].

If the DTC is not stored in one of the available event memories, the parameter DTCOrigin of the function Dem_GetStatusOfDTC is neglected e.g., when DTC status is pending.
It is possible that a DTC with different states depending on the location exists. If the secondary memory is used as a kind of protocol stack that gives information which services have been performed on the primary memory different DTC states might appear (e.g. DTC has been deleted from the primary memory and is written to the secondary memory with its latest status).

8.3.5.1.4 Dem_GetDTCStatusAvailabilityMask

Dem213:

Service name:	Dem_GetDTCStatusAvailabilityMask

Syntax:	`Std_ReturnType Dem_GetDTCStatusAvailabilityMask( uint8* DTCStatusMask )`	
Service ID[hex]:	0x16	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	DTCStatusMask	The value DTCStatusMask indicates the supported DTC status bits from the DEM. All supported information is indicated by setting the corresponding status bit to 1. See ISO14229-1
Return value:	Std_ReturnType	E_OK: get of DTC status mask was successful E_NOT_OK: get of DTC status mask failed
Description:	Gets the DTC Status availability mask	

Dem060: The function Dem_GetDTCStatusAvailabilityMask shall get the DTC Status availability mask that means the DTC status information (according to ISO 14229-1 [12]) supported by the DEM module.

The function Dem_SetDTCFilter can only use supported bits as filter parameters.

8.3.5.1.5 Dem_GetNumberOfFilteredDTC

Dem214:

Service name:	Dem_GetNumberOfFilteredDTC	
Syntax:	`Dem_ReturnGetNumberOfFilteredDTCType Dem_GetNumberOfFilteredDTC( uint16* NumberOfFilteredDTC )`	
Service ID[hex]:	0x17	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	NumberOfFilteredDTC	The number of DTCs matching the defined status mask.
Return value:	Dem_ReturnGetNumberOfFilteredDTCType	Status of the operation to retrieve a number of DTC from the DEM
Description:	Gets the number of a filtered DTC	

Dem061: The function Dem_GetNumberOfFilteredDTC shall get the number of DTC matching the defined status mask.

The function Dem_SetDTCFilter will set the DTC Status mask filter.

Caveats of Dem_GetNumberOfFilteredDTC: DTC filter has been set up properly before function call (Dem_SetDTCFilter).

8.3.5.1.6 Dem_GetNextFilteredDTC

Dem215:

Service name:	Dem_GetNextFilteredDTC	
Syntax:	`Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredDTC( uint32* DTC, uint8* DTCStatus )`	
Service ID[hex]:	0x18	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	DTC	Receives the DTC value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.
	DTCStatus	This parameter receives the status information of the requested DTC. It follows the format as defined in ISO14229-1 If the return value of the function call is other than DEM_FILTERED_OK this parameter does not contain valid data.
Return value:	Dem_ReturnGetNextFilteredDTCType	Status of the operation to retrieve a DTC from the DEM.
Description:	Gets the next filtered DTC.	

Dem216: The function Dem_GetNextFilteredDTC shall return the current DTC and its associated status from the DEM module matching the filter criteria defined by the function call of Dem_SetDTCFilter.

Dem217: The function Dem_GetNextFilteredDTC shall skip to the next DTC matching the filter criteria after having returned the requested data.

The DEM module's environment shall call the function Dem_GetNextFilteredDTC continuously until the return value of the function is DEM_FILTERED_NO_MATCHING_DTC to receive all DTCs matching the filter criteria.

The chronological order shall be reported if the DTC status mask parameter is set to "pending" and/or "confirmed" (no other status bits are allowed to be set). The function shall start with the most recent DTC. The chronological order may vary with the customer specific attributes used by the algorithm for sorting the DTC records (e.g. pre-sorted records or time-stamp attributes of the records).

8.3.5.1.7 Dem_GetDTCByOccurrenceTime

Dem218:

Service name:	Dem_GetDTCByOccurrenceTime	
Syntax:	`Dem_ReturnGetDTCByOccurrenceTimeType Dem_GetDTCByOccurrenceTime( Dem_DTCRequestType DTCRequest, Dem_DTCKindType DTCKind, uint32* DTC )`	
Service ID[hex]:	0x19	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTCRequest	This parameter defines the request type of the DTC.
	DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs
Parameters (inout):	None	
Parameters (out):	DTC	Receives the DTC value returned by the function. If the return value of the function is other than DEM_OCCURR_OK this parameter does not contain valid data.
Return value:	Dem_ReturnGetDTCByOccurrenceTimeType	Status of the operation of type Dem_ReturnGetDTCByOccurrenceTimeType.
Description:	Gets the DTC by occurrence time. There is no explicit parameter for the DTC-origin as the origin always is DEM_DTC_ORIGIN_PRIMARY_MEMORY.	

Dem219: The function Dem_GetDTCByOccurrenceTime shall provide the capability to get one DTC from stored event data sets according to the parameter DTCRequest, which specifies the relevant occurrence time.

Dem221: The function Dem_GetDTCByOccurrenceTime shall return the appropriate operation status (ref. to 8.2.2.11) and set the DTC value to zero (0) if no DTC is matching the requested point in time.

If this API is implemented the DEM is supposed to provide an OEM specific ordering scheme.

8.3.5.1.8 Dem_GetNextFilteredRecord

Dem224:

Service name:	Dem_GetNextFilteredRecord

Syntax:	`Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredRecord( uint32* DTC, uint8* SnapshotRecord )`	
Service ID[hex]:	0x3a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	DTC	Receives the DTC value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.
	SnapshotRecord	Snapshot Record Number for the reported DTC.
Return value:	Dem_ReturnGetNextFilteredDTCType	Status of the operation to retrieve a DTC from the DEM.
Description:	Gets the current DTC and its associated snapshot record numbers from the DEM.	

Dem225: The function Dem_GetNextFilteredRecord shall return the current DTC and its associated Snapshot Record numbers from the DEM module matching the filter criteria defined by the function call Dem_SetDTCFilterForRecords.

Dem226: After having returned the data the function Dem_GetNextFilteredRecord shall skip to the next Record matching the filter criteria.

The DEM module's environment shall call the function Dem_GetNextFilteredRecord continuously until the return value of the function is DEM_FILTERED_NO_MATCHING_DTC to receive all records matching the filter criteria.

8.3.5.1.9 Dem_GetNextFilteredDTCAndFDC

Dem227:

Service name:	Dem_GetNextFilteredDTCAndFDC	
Syntax:	`Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredDTCAndFDC( uint32* DTC, sint8* DTCFaultDetectionCounter )`	
Service ID[hex]:	0x3b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	DTC	Receives the DTC value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this

		parameter does not contain valid data.
	DTCFaultDetectionCounter	This parameter receives the Fault Detection Counter information of the requested DTC. If the return value of the function call is other than DEM_FILTERED_OK this parameter does not contain valid data. -128dec...127dec PASSED...FAILED according to ISO 14229-1
Return value:	Dem_ReturnGetNextFilteredDTCType	Status of the operation to retrieve a DTC from the DEM.
Description:	Gets the current DTC and its associated Fault Detection Counter (FDC) from the DEM.	

Dem228: The function Dem_GetNextFilteredDTCAndFDC shall return the current DTC and its associated Fault Detection Counter (FDC) from the DEM matching the filter criteria defined by the function call Dem_SetDTCFilter.

Dem229: After having returned the data the function Dem_GetNextFilteredDTCAndFDC shall skip to the next DTC matching the filter criteria.

The DEM module's environment shall call the function Dem_GetNextFilteredDTCAndFDC continuously until the return value of the function is DEM_FILTERED_NO_MATCHING_DTC to receive all DTCs matching the filter criteria

8.3.5.1.10 Dem_GetNextFilterdDTCAndSeverity

Dem281:

Service name:	Dem_GetNextFilteredDTCAndSeverity	
Syntax:	`Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredDTCAndSeverity( uint32* DTC, uint8* DTCStatus, Dem_DTCSeverityType* DTCSeverity, uint8* DTCFunctionalUnit )`	
Service ID[hex]:	0x3d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	DTC	Receives the DTC value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.
	DTCStatus	Receives the status value returned by the

		function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.
	DTCSeverity	Receives the severity value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.
	DTCFunctionalUnit	Receives the functional unit value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.
Return value:	Dem_ReturnGetNextFilteredDTCType	Status of the operation to retrieve a DTC from the DEM.
Description:	Gets the current DTC and its Severity from the DEM.	

Dem287: The function Dem_GetNextFilteredDTCAndSeverity shall return the current DTC and its associated Fault Severity from the DEM matching the filter criteria defined by the function call Dem_SetDTCFilter.

Dem288: After having returned the data the function Dem_GetNextFilteredDTCAndSeverity shall skip to the next DTC matching the filter criteria.

The DEM module's environment shall call the function Dem_GetNextFilteredDTCAndSeverity continuously until the return value of the function is DEM_FILTERED_NO_MATCHING_DTC to receive all DTCs matching the filter criteria.

8.3.5.1.11 Dem_GetTranslationType

Dem230:

Service name:	Dem_GetTranslationType				
Syntax:	`uint8 Dem_GetTranslationType( )`				
Service ID[hex]:	0x3c				
Sync/Async:	Synchronous				
Reentrancy:	Non Reentrant				
Parameters (in):	None				
Parameters (inout):	None				
Parameters (out):	None				
Return value:	uint8	The TranslationFormat provides the configured translation formats according ISO14229-1 Service 0x19.			
		0x00	2	byte	ISO15031-6DTCFormat
		0x01	3	byte	ISO14229-1DTCFormat
		0x02			SAEJ1939-73DTCFormat
		0x03			ISO11992-4DTCFormat

		Combinations of different DTC formats are not possible. The associated configuration value shall be the parameter DEM_TYPE_OF_DTC_SUPPORTED.
Description:	Gets the supported DTC formats of the ECU. The supported formats are configured via DemTypeOfDTCSupported.	

Dem231: The function Dem_GetTranslationType shall provide the capability to get the configured translation format of the ECU.

8.3.5.1.12 Dem_GetSeverityOfDTC

Dem232:

Service name:	Dem_GetSeverityOfDTC	
Syntax:	`Dem_ReturnGetSeverityOfDTCType Dem_GetSeverityOfDTC( uint32 DTC, Dem_DTCSeverityType* DTCSeverity )`	
Service ID[hex]:	0x0e	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	The Severity assigned to this DTC should be returned
Parameters (inout):	None	
Parameters (out):	DTCSeverity	This parameter contains the DTCSeverityMask according to ISO14229-1. DEM_SEVERITY_CHECK_IMMEDIATELY 0x80 Check immediately, DEM_SEVERITY_NO_SEVERITY 0x00 No severity information available, DEM_SEVERITY_CHECK_AT_NEXT_HALT 0x40 check at next halt, DEM_SEVERITY_MAINTENANCE_ONLY 0x20 maintenance required
Return value:	Dem_ReturnGetSeverityOfDTCType	Status of the operation of type Dem_ReturnGetSeverityOfDTCType.
Description:	Gets the severity of the requested DTC.	

Caveats of Dem_GetSeverityOfDTC: DTCKind not needed, because Severity is only available for ISO 14229-1 DTCs

8.3.5.2 Access extended data records and FreezeFrame data

This section defines the API-calls to be used to get access to the event related data stored with the DTCs in the records of the DEM. Furthermore access to OBD-

relevant PIDs stored in a FreezeFrame is made available. The FreezeFrames can be addressed either with absolute numbers or relative numbers. If absolute addressing is used (emission relevant ECUs) a unique number for a FreezeFrame exists throughout the whole ECU. In case of relative addressing the FreezeFrames are unique to a DTC. Inside an ECU only absolute or relative addressing can be used not both addressing modes in parallel. The implementation of two different addressing modes is OEM-specific. Details concerning FreezeFrame handling can be found in ISO 14229-1 and ISO 15031-5. The usage of the following API calls is illustrated in chapter “Error Manager Interface” of the DCM SWS document [5].

8.3.5.2.1 Dem_DisableDTCRecordUpdate

Dem233:

Service name:	Dem_DisableDTCRecordUpdate	
Syntax:	`Std_ReturnType Dem_DisableDTCRecordUpdate( )`	
Service ID[hex]:	0x1a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	Disables the DTC record update.	

The DEM module’s environment shall use the function Dem_DisableDTCRecordUpdate if the FreezeFrame or extended data record are about to be accessed by subsequent API-calls. It is done to ensure that the data contained in this record is not changed while the FreezeFrame or extended data record are accessed by the external application, e.g. DCM.

Dem270: The function Dem_DisableDTCRecordUpdate shall prevent the DEM module from manipulating, overwriting or deleting any existing DTC, associated FreezeFrame and/or extended data records.

New DTCs and associated FreezeFrames and extended data records can still be added to the fault record storage as long as memory is available.

The function Dem_DisableDTCRecordUpdate does not affect the DTC status information update.

8.3.5.2.2 Dem_EnableDTCRecordUpdate

Dem234:

Service name:	Dem_EnableDTCRecordUpdate	
Syntax:	`Std_ReturnType Dem_EnableDTCRecordUpdate( )`	
Service ID[hex]:	0x1b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	Enables the DTC record update	

Dem271: The function Dem_EnableDTCRecordUpdate shall release the data contained in the record that has been protected by the function Dem_DisableDTCRecordUpdate so that the data can be accessed or manipulated by the external application, e.g. the DCM module, again.

The function Dem_EnableDTCRecordUpdate is the counterpart to the function Dem_DisableDTCRecordUpdate.

The DEM module's environment shall call the function Dem_EnableDTCRecordUpdate after the FreezeFrame and extended data record were protected by the function Dem_DisableDTCRecordUpdate and after the access by subsequent API-calls is finished.

8.3.5.2.3 Dem_GetDTCOfFreezeFrameRecord

Dem235:

Service name:	Dem_GetDTCOfFreezeFrameRecord
Syntax:	`Dem_ReturnGetDTCOfFreezeFrameRecordType Dem_GetDTCOfFreezeFrameRecord( uint8 RecordNumber, Dem_DTCOriginType DTCOrigin, Dem_DTCKindType DTCKind, uint32* DTC )`
Service ID[hex]:	0x1c
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	RecordNumber	This parameter is a unique identifier for a FreezeFrame record as defined in ISO15031-5 and ISO14229-1. This parameter cannot be 0xFF.
	DTCOrigin	This parameter selects the source memory the DTCs shall be read from.
	DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs
Parameters (inout):	None	
Parameters (out):	DTC	Receives the DTC value returned by the function. If the return value of the function is other than DEM_GET_DTCOFFF_OK this parameter does not contain valid data.
Return value:	Dem_ReturnGetDTCOfFreezeFrameRecordType	Status of the operation of type Dem_ReturnGetDTCOfFreezeFrameRecordType.
Description:	Gets a DTC associated with a FreezeFrame.	

Dem070: The function Dem_GetDTCOfFreezeFrameRecord shall return the DTC associated with the FreezeFrame selected via its absolute record number.

Caveats of Dem_GetDTCOfFreezeFrameRecord: The record number has to be unique throughout the whole ECU. The function Dem_GetDTCOfFreezeFrameRecord is only required for OBD-relevant ECUs.

8.3.5.2.4 Dem_GetFreezeFrameDataByDTC

Dem236:

Service name:	Dem_GetFreezeFrameDataByDTC	
Syntax:	`Dem_ReturnGetFreezeFrameDataByDTCType Dem_GetFreezeFrameDataByDTC( uint32 DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, uint8 RecordNumber, uint8* DestBuffer, uint8* BufSize )`	
Service ID[hex]:	0x1d	
Sync/Asynch:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	This is the DTC the FreezeFrame is assigned to.
	DTCKind	This parameter defines the requested DTC,

		either only OBD-relevant DTCs or all DTCs
	DTCOrigin	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.
	RecordNumber	This parameter is an identifier for a FreezeFrame record as defined in ISO15031-5 and ISO14229-1. The value 0xff is not allowed.
Parameters (inout):	BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Parameters (out):	DestBuffer	This parameter contains a byte pointer that points to the buffer to which the FreezeFrame data shall be written.
Return value:	Dem_ReturnGetFreezeFrameDataByDTCType	Status of the operation of type Dem_ReturnGetFreezeFrameDataByDTCType.
Description:	Gets a FreezeFrame Data by DTC. The function stores the data in the provided DestBuffer. The data returned includes the DTCSnapshotNumberOfIdentifiers and DTCSnapshotRecord as defined by UDS for the response message to Service 0x19 subfunction 0x05 or subfunction 0x04.	

Dem071: The function Dem_GetFreezeFrameDataByDTC shall copy a specific PID/DataId of a FreezeFrame selected via the associated DTC number and an optional FreezeFrame RecordNumber to the destination buffer. The function Dem_GetFreezeFrameDataByDTC shall transmit it as a complete record with format PID followed by data, PID – data, ...

8.3.5.2.5 Dem_GetFreezeFrameDataIdentifierByDTC

Dem237:

Service name:	Dem_GetFreezeFrameDataIdentifierByDTC
Syntax:	`Dem_ReturnGetFreezeFrameDataIdentifierByDTCType Dem_GetFreezeFrameDataIdentifierByDTC( uint32 DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, uint8 RecordNumber, uint8* ArraySize, const uint16** DataId )`
Service ID[hex]:	0x1e
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	DTC	This is the DTC the FreezeFrame is assigned to.
	DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs
	DTCOrigin	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from. The parameter is simlar to an enum. 0x01 DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory, 0x02 DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, 0x03 DEM_DTC_ORIGIN_PERMANENT The Event information is located in the permanent memory The definition and use of the different memory types is OEM specific.
	RecordNumber	This parameter is a unique identifier for a FreezeFrame record as defined in ISO15031-5 and ISO14229-1.
Parameters (inout):	None	
Parameters (out):	ArraySize	This parameter specifies the number of data identifiers for the selected RecordNumber.
	DataId	Pointer to an array with the supported data identifier for the selected RecordNumber and DTC.
Return value:	Dem_ReturnGetFreezeFrameDataIdentifierByDTCType	Status of the operation of type Dem_ReturnGetFreezeFrameDataIdentifierByDTCType.
Description:	Gets a FreezeFrame Data identifier by DTC	

Dem073: The function Dem_GetFreezeFrameDataIdentifierByDTC shall return the data identifiers and the number of data identifiers of a FreezeFrame which belongs to a specific DTC.

8.3.5.2.6 Dem_GetSizeOfFreezeFrame

Dem238:

Service name:	Dem_GetSizeOfFreezeFrame
Syntax:	`Dem_ReturnGetSizeOfFreezeFrameType Dem_GetSizeOfFreezeFrame( uint32 DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin,`

	uint8 RecordNumber, uint16* SizeOfFreezeFrame)	
Service ID[hex]:	0x1f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	This is the DTC the FreezeFrame is assigned to.
	DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs
	DTCOrigin	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.
	RecordNumber	This parameter is a unique identifier for a FreezeFrame record as defined in ISO15031-5 and ISO14229-1.
Parameters (inout):	None	
Parameters (out):	SizeOfFreezeFrame	Number of bytes in the requested FreezeFrame.
Return value:	Dem_ReturnGetSizeOfFreezeFrameType	Status of the operation of type Dem_ReturnGetSizeOfFreezeFrameType
Description:	Gets the size of a FreezeFrame	

Dem074: The function Dem_GetSizeOfFreezeFrame shall return the size of the requested FreezeFrame.

The return value of the function Dem_GetSizeOfFreezeFrame represents only the number of user data bytes (pure FreezeFrame data) and does not contain any FreezeFrame structure information.

8.3.5.2.7 Dem_GetExtendedDataRecordByDTC

Dem239:

Service name:	Dem_GetExtendedDataRecordByDTC
Syntax:	Dem_ReturnGetExtendedDataRecordByDTCType Dem_GetExtendedDataRecordByDTC(uint32 DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, uint8 ExtendedDataNumber, uint8* DestBuffer, uint8* BufSize)
Service ID[hex]:	0x20
Sync/Asynch:	Synchronous

Reentrancy:	Non Reentrant	
Parameters (in):	DTC	This is the DTC the 'Extended Data Record' is assigned to.
	DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs DEM_DTC_KIND_ALL_DTCS Select all DTCs DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs DEM_DTC_KIND_ALL_DTCS Select all DTCs DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs
	DTCOrigin	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from. The parameter is simlar to an enum. 0x01 DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory, 0x02 DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, 0x03 DEM_DTC_ORIGIN_PERMANENT The Event information is located in the permanent memory The definition and use of the different memory types is OEM specific.
	ExtendedDataNumber	Identification of requested Extended data record. Valid values are between 0x01 and 0xEF.
Parameters (inout):	BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Parameters (out):	DestBuffer	This parameter contains a byte pointer that points to the buffer to which the Extended Data shall be written.
Return value:	Dem_ReturnGetExtendedDataRecordByDTCType	Status of the operation of type Dem_ReturnGetExtendedDataRecordByDTCType
Description:	Returns the DTCExtendedDataRecord as defined in UDS Service 0x19 subfunction 0x06, 0x10.	

Dem075: The function Dem_GetExtendedDataRecordByDTC shall return the complete Extended Data Record for the requested DTC.

The format of the data referenced by the pointer DestBuffer of the function Dem_GetExtendedDataRecordByDTC is raw hexadecimal values and is not standardized to comply with predefined scaling methods.

Configuration of Dem_GetExtendedDataRecordByDTC: Values of 'Extended Data Record' have to be defined.

8.3.5.2.8 Dem_GetSizeOfExtendedDataRecordByDTC

Dem240:

Service name:	Dem_GetSizeOfExtendedDataRecordByDTC	
Syntax:	`Dem_ReturnGetSizeOfExtendedDataRecordByDTCType Dem_GetSizeOfExtendedDataRecordByDTC( uint32 DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, uint8 ExtendedDataNumber, uint16* SizeOfExtendedDataRecord )`	
Service ID[hex]:	0x21	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	This is the DTC the 'Extended Data Record' is assigned to.
	DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs
	DTCOrigin	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from. The parameter is simlar to an enum. 0x01 DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory, 0x02 DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, 0x03 DEM_DTC_ORIGIN_PERMANENT The Event information is located in the permanent memory The definition and use of the different memory types is OEM specific.
	ExtendedDataNumber	Identification of requested Extended data record. The requested record is copied to the destination buffer.

Parameters (inout):	None	
Parameters (out):	SizeOfExtendedDataRecord	Pointer to Size of the requested data record
Return value:	Dem_ReturnGetSizeOfExtendedDataRecordByDTCType	Status of the operation of type Dem_ReturnGetSizeOfExtendedDataRecordByDTCType
Description:	Gets the size of an extended data record by DTC	

Dem076: The function Dem_GetSizeOfExtendedDataRecordByDTC shall return the size of the requested 'Extended Data Record' frame, which only represents the number of user data bytes stored in the 'Extended Data Record'.

Configuration of Dem_GetSizeOfExtendedDataRecordByDTC: Values of 'Extended Data Record' have to be defined.

8.3.5.3 Clear DTC information

The next sections define the usage of the function calls to delete single DTCs as well as groups of DTCs from the records of the DEM module.

8.3.5.3.1 Dem_ClearDTC

Dem241:

Service name:	Dem_ClearDTC	
Syntax:	`Dem_ReturnClearDTCType Dem_ClearDTC( uint32 DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin )`	
Service ID[hex]:	0x22	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	Defines the DTC that shall be cleared from the event memory. If the DTC fits to a DTC group number, all DTCs of the group shall be cleared.
	DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs
	DTCOrigin	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_ReturnClearDTCType	Status of the operation of type Dem_ReturnClearDTCType.
Description:	Clears a DTC	

Dem077: The function Dem_ClearDTC shall clear all event status related to the specified DTC and all associated event memory entries for these events (event related and/or FreezeFrame data, …).

Configuration of Dem_ClearDTC: The initialization of the corresponding Monitor Function (DTC → EventId → Monitor for specific event) is managed by DemInitMonitorForEvent.

8.3.5.4 Control DTC storage

This section defines the function calls to enable and disable DTC storage in the DEM module.

8.3.5.4.1 Dem_DisableDTCStorage

Dem242:

Service name:	Dem_DisableDTCStorage	
Syntax:	`Dem_ReturnControlDTCStorageType Dem_DisableDTCStorage( Dem_DTCGroupType DTCGroup, Dem_DTCKindType DTCKind )`	
Service ID[hex]:	0x24	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTCGroup	Defines the group of DTC that shall be disabled to store in event memory.
	DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_ReturnControlDTCStorageType	Returns status of the operation
Description:	Disables the storage of a DTC group	

Dem079: The function Dem_DisableDTCStorage shall disable the storage of a DTC group in the event memory (derived from Dem035).

The function Dem_DisableDTCStorage does not affect the DTC status information update.

The function Dem_DisableDTCStorage is only for preventing DTCs from being stored in case of an induced failure situations in a system, e.g. during flash-reprogramming of one ECU in a network. In that case all the ECUs are commanded via diagnostic request (linked to the above diagnostic request) to suppress storage of a DTC while maintaining correct fail-safe behavior as the flashed ECU is not participating in the

normal communication anymore. If one of the other networked ECUs needs one of the signals which are now missing, this will lead to a failsafe-reaction of the ECU as by the AUTOSAR concept the fail-safe reaction of an ECU is triggered by certain event-status updates or a FIM-command which is itself triggered by an event-status update.

8.3.5.4.2 Dem_EnableDTCStorage

Dem243:

Service name:	Dem_EnableDTCStorage	
Syntax:	`Dem_ReturnControlDTCStorageType Dem_EnableDTCStorage( Dem_DTCGroupType DTCGroup, Dem_DTCKindType DTCKind )`	
Service ID[hex]:	0x25	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTCGroup	Defines the group of DTC that shall be enabled to store in event memory.
	DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_ReturnControlDTCStorageType	Returns the status of the operation
Description:	Enables the storage of a DTC group	

Dem080: The function Dem_EnableDTCStorage shall enable the storage of a DTC group in the event memory (derived from Dem035).

See also Dem_DisableDTCStorage.

8.3.5.4.3 Dem_DisableEventStatusUpdate

Dem244:

Service name:	Dem_DisableEventStatusUpdate	
Syntax:	`Dem_ReturnControlEventUpdateType Dem_DisableEventStatusUpdate( Dem_DTCGroupType DTCGroup, Dem_DTCKindType DTCKind )`	
Service ID[hex]:	0x26	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTCGroup	Defines the group of DTC that shall be disabled to store in event memory.
	DTCKind	This parameter defines the requested DTC,

		either only OBD-relevant DTCs or all DTCs
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_ReturnControlEventUpdateType	Status of the operation of type Status of the operation of type Dem_ReturnControlDTCStorageType
Description:	Disables the event status update of a DTC group	

Dem081: The function Dem_DisableEventStatusUpdate shall disable the update of the event status of a DTC group.

The function Dem_DisableEventStatusUpdate influences only the execution of the functions Dem_SetEventStatus, Dem_ReportErrorStatus and Dem_ResetEventStatus that will be defined within the configuration of the DEM module (Dem034).
In this case, both, the event status update and consequently the storage of DTCs, are suppressed. Thereby any fail-safe reaction of the ECU which is tight to certain event-status-updates will be suppressed as well, leaving the system in an unpredictable or even self-destructive condition if failures are not correctly handled anymore.

The function Dem_DisableEventStatusUpdate may be used for engineering purposes or during manufacturing in a controlled environment to suppress failsafe-reaction (e.g. prevent headlamps on, windshield wiper on, etc.).

Configuration of Dem_DisableEventStatusUpdate: Depending on configuration within the function Dem_DisableEventStatusUpdate, the reaction on the event status is defined. For example: the execution of Dem_ResetEventStatus is possible also during this phase.

8.3.5.4.4 Dem_EnableEventStatusUpdate

Dem245:

Service name:	Dem_EnableEventStatusUpdate	
Syntax:	`Dem_ReturnControlEventUpdateType Dem_EnableEventStatusUpdate( Dem_DTCGroupType DTCGroup, Dem_DTCKindType DTCKind )`	
Service ID[hex]:	0x27	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTCGroup	Defines the group of DTC that shall be disabled to store in event memory.
	DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs
Parameters (inout):	None	

Parameters (out):	None	
Return value:	Dem_ReturnControlEventUpdateType	Status of the operation of type Status of the operation of type Dem_ReturnControlDTCStorageType
Description:	Enables the event status update of a DTC group	

Dem082: The function Dem_EnableEventStatusUpdate shall enable the update of the event status of a DTC group (derived from Dem034).

See also Dem_DisableEventStatusUpdate.

8.3.6 OBD-specific Interfaces

8.3.6.1 Dem_SetEventDisabled

Dem312:

Service name:	Dem_SetEventDisabled	
Syntax:	`Std_ReturnType Dem_SetEventDisabled( Dem_EventIdType EventId )`	
Service ID[hex]:	0x51	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event) Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK set of event to disabled was successfull. E_NOT_OK set of event disabled failed
Description:	Service for reporting the Event as disabled to the DEM	

Dem294: In order to allow a diagnostic function to report that the EventId cannot be computed in the driving cycle (aborted e.g. due to physical reasons), the DEM shall provide an API Dem_SetEventDisabled(EventId).

Dem305: For the computation of PID$41, the diagnostic function has to report its Event as disabled if the test cannot be carried out anymore until the end of this driving cycle.

8.3.6.2 Dem_RepIUMPRFaultDetect

Dem313:

Service name:	Dem_RepIUMPRFaultDetect	
Syntax:	`Std_ReturnType Dem_RepIUMPRFaultDetect( Dem_RatioIdType RatioID )`	
Service ID[hex]:	0x73	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	RatioID	Ratio Identifier reporting that a respective diagnostic function could have found a fault - only used when interface option "API" is selected
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK report of IUMPR result was successfully reported
Description:	Service for reporting that faults are possibly found because are all conditions are fullfilled.	

Dem296: The DEM shall provide an API for the asymmetric diagnostic functions to report that a malfunction could have been found. The API to be used by the diagnostic function is Dem_RepIUMPRFaultDetect (RatioId).

Dem306: This service DemRepIUMPRFaultDetect shall be used to report that a fault could have been found - even if there is currently no fault at all - according to the IUMPR regulations that all conditions are met for the detection of a malfunction

8.3.6.3 Dem_RepIUMPRDenLock

Dem314:

Service name:	Dem_RepIUMPRDenLock	
Syntax:	`Std_ReturnType Dem_RepIUMPRDenLock( Dem_RatioIdType RatioID )`	
Service ID[hex]:	0x71	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	RatioID	Ratio Identifier reporting that specific denominator is locked (for physical reasons - e.g. temperature conditions or minimum activity)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK report of IUMPR denominator status was successfully reported E_NOK report of IUMPR denominator status was not successfully reported

Description:	Service is used to lock a denominator of a specific monitor.

Dem297: The DEM shall provide the API Dem_RepIUMPRDenLock (RatioId) to IUMPR-relevant SW-C to control the denominator specific to the respective RatioId).

Dem307: This service shall be used to report that a denominator of a specific monitor (represented by FID and EventID) is locked for physical reasons.

8.3.6.4 Dem_RepIUMPRDenRelease

Dem315:

Service name:	Dem_RepIUMPRDenRelease	
Syntax:	`Std_ReturnType Dem_RepIUMPRDenRelease( Dem_RatioIdType RatioID )`	
Service ID[hex]:	0x72	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	RatioID	Ratio Identifier reporting that specific denominator is released (for physical reasons - e.g. temperature conditions or minimum activity)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK report of IUMPR denominator status was successfully reported E_NOK report of IUMPR denominator status was not successfully reported
Description:	Service is used to release a denominator of a specific monitor.	

Dem308: The DEM shall provide the API Dem_RepIUMPRDenRelease (RatioId) to IUMPR-relevant SW-C to control the denominator specific to the respective RatioId.

Dem309: This service DemRepIUMPRDenRelease shall be used to report that a denominator of a specific monitor (represented by FID and EventID) is released for physical reasons.

8.3.6.5 Dem_GetInfoTypeValue08

Dem316:

Service name:	Dem_GetInfoTypeValue08
Syntax:	`Std_ReturnType Dem_GetInfoTypeValue08( uint8* Iumprdata08 )`

Service ID[hex]:	0x6b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	Iumprdata08	Buffer containing the contents of InfoType $08. The buffer is provided by the DCM.
Return value:	Std_ReturnType	E_OK IUMPR data was successfully reported E_NOK IUMPR data was not successfully reported
Description:	Service is used to request for IUMPR data accoirding InfoType $08.	

Dem298: In order to support the data requests in service $09 as described above, the Dem shall provide the API Dem_GetInfoTypeValue08 or Dem_GetInfoTypeValue0B to the DCM.

Dem310: The service Dem_GetInfoTypeValue08 shall be used by the DCM to request for the IUMPR-data according to the InfoType $08 data format used for the output to Service$09.

8.3.6.6 Dem_GetInfoTypeValue0B

Dem317:

Service name:	Dem_GetInfoTypeValue0B	
Syntax:	`Std_ReturnType Dem_GetInfoTypeValue0B( uint8* Iumprdata0B )`	
Service ID[hex]:	0x6c	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	Iumprdata0B	Buffer containing the contents of InfoType $0B. The buffer is provided by the DCM.
Return value:	Std_ReturnType	E_OK IUMPR data was successfully reported E_NOK IUMPR data was not successfully reported
Description:	Service is used to request for IUMPR data accoirding InfoType $0B.	

Dem311: The service Dem_GetInfoTypeValue0B shall be used by the DCM to request for the IUMPR-data according to the InfoType $0B data format used for the output to Service$09.

8.3.6.7 Dem_DcmGetPID01

Dem318:

Service name:	Dem_DcmGetPID01	
Syntax:	`Std_ReturnType Dem_DcmGetPID01( uint8* PID01value )`	
Service ID[hex]:	0x61	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID01value	Buffer containing the contents of PID$01 computed by the DEM. The buffer is provided by the DCM with the appropriate size, i.e. during configuration, the DCM identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	E_OK PID data was successfully reported E_NOK PID data was not successfully reported
Description:	Service to report the value of PID $01 computed by the DEM.	

Since the BSW modules DEM and DCM do not communicate via RTE a direct interface, direct function calls are assigned to report the value of these PIDs computed by the DEM. See also Dem293: **.**

8.3.6.8 Dem_DcmGetPID21

Dem319:

Service name:	Dem_DcmGetPID21	
Syntax:	`Std_ReturnType Dem_DcmGetPID21( uint8* PID21value )`	
Service ID[hex]:	0x64	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID21value	Buffer containing the contents of PID$21 computed by the DEM. The buffer is provided by the DCM with the appropriate size, i.e. during configuration, the DCM identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	E_OK PID data was successfully reported E_NOK PID data was not successfully reported
Description:	Service to report the value of PID $21 computed by the DEM.	

Since the BSW modules DEM and DCM do not communicate via RTE a direct interface, direct function calls are assigned to report the value of these PIDs computed by the DEM. See also Dem293: .

8.3.6.9 Dem_DcmGetPID30

Dem320:

Service name:	Dem_DcmGetPID30	
Syntax:	`Std_ReturnType Dem_DcmGetPID30( uint8* PID30value )`	
Service ID[hex]:	0x65	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID30value	Buffer containing the contents of PID$30 computed by the DEM. The buffer is provided by the DCM with the appropriate size, i.e. during configuration, the DCM identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	E_OK PID data was successfully reported E_NOK PID data was not successfully reported
Description:	Service to report the value of PID $30 computed by the DEM.	

Since the BSW modules DEM and DCM do not communicate via RTE a direct interface, direct function calls are assigned to report the value of these PIDs computed by the DEM. See also Dem293: .

8.3.6.10 Dem_DcmGetPID31

Dem321:

Service name:	Dem_DcmGetPID31	
Syntax:	`Std_ReturnType Dem_DcmGetPID31( uint8* PID31value )`	
Service ID[hex]:	0x66	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID31value	Buffer containing the contents of PID$31 computed by the DEM.

		The buffer is provided by the DCM with the appropriate size, i.e. during configuration, the DCM identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	E_OK PID data was successfully reported E_NOK PID data was not successfully reported
Description:	Service to report the value of PID $31 computed by the DEM.	

Since the BSW modules DEM and DCM do not communicate via RTE a direct interface, direct function calls are assigned to report the value of these PIDs computed by the DEM. See also Dem293: **.**

8.3.6.11 Dem_DcmGetPID41

Dem322:

Service name:	Dem_DcmGetPID41	
Syntax:	`Std_ReturnType Dem_DcmGetPID41( uint8* PID41value )`	
Service ID[hex]:	0x67	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID41value	Buffer containing the contents of PID$41 computed by the DEM. The buffer is provided by the DCM with the appropriate size, i.e. during configuration, the DCM identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	E_OK PID data was successfully reported E_NOK PID data was not successfully reported
Description:	Service to report the value of PID $41 computed by the DEM.	

Since the BSW modules DEM and DCM do not communicate via RTE a direct interface, direct function calls are assigned to report the value of these PIDs computed by the DEM. See also Dem293: **.**

8.3.6.12 Dem_DcmGetPID4D

Dem323:

Service name:	Dem_DcmGetPID4D
Syntax:	`Std_ReturnType Dem_DcmGetPID4D(`

	uint8* PID4Dvalue)	
Service ID[hex]:	0x68	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID4Dvalue	Buffer containing the contents of PID$4D computed by the DEM. The buffer is provided by the DCM with the appropriate size, i.e. during configuration, the DCM identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	E_OK PID data was successfully reported E_NOK PID data was not successfully reported
Description:	Service to report the value of PID $4D computed by the DEM.	

Since the BSW modules DEM and DCM do not communicate via RTE a direct interface, direct function calls are assigned to report the value of these PIDs computed by the DEM. See also Dem293: **.**

8.3.6.13 Dem_DcmGetPID4E

Dem324:

Service name:	Dem_DcmGetPID4E	
Syntax:	Std_ReturnType Dem_DcmGetPID4E(uint8* PID4Dvalue)	
Service ID[hex]:	0x69	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID4Dvalue	--
Return value:	Std_ReturnType	--
Description:	--	

Since the BSW modules DEM and DCM do not communicate via RTE a direct interface, direct function calls are assigned to report the value of these PIDs computed by the DEM. See also Dem293: **.**

8.3.6.14 Dem_DcmGetPID1C

Dem325:

Service name:	Dem_DcmGetPID1C	
Syntax:	`Std_ReturnType Dem_DcmGetPID1C( uint8* PID1Cvalue )`	
Service ID[hex]:	0x63	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID1Cvalue	Buffer containing the contents of PID$1C computed by the DEM. The buffer is provided by the DCM with the appropriate size, i.e. during configuration, the DCM identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	E_OK PID data was successfully reported E_NOK PID data was not successfully reported
Description:	Service to report the value of PID $1C computed by the DEM.	

Since the BSW modules DEM and DCM do not communicate via RTE a direct interface, direct function calls are assigned to report the value of these PIDs computed by the DEM. See also Dem293: .

8.3.6.15 Dem_GetOBDFreezeFrameData

Dem327:

Service name:	Dem_GetOBDFreezeFrameData	
Syntax:	`Std_ReturnType Dem_GetOBDFreezeFrameData( uint8 PID, uint8* DestBuffer, uint8* BufSize )`	
Service ID[hex]:	0x52	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	PID	This parameter is a identifier for a PID as defined in ISO15031-5.
Parameters (inout):	DestBuffer	This parameter contains a byte pointer that points to the buffer to which the FreezeFrame data shall be written.
	BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK FreezeFrame data was successfully reported E_NOK FreezeFrame data was not successfully reported
Description:	--	

Since the BSW modules DEM and DCM do not communicate via RTE a direct interface, direct function calls are assigned to report the FreezeFrame data of the requested RecordNumber computed by the DEM.

8.3.6.16 Dem_SetPtoStatus

Service name:	Dem_SetPtoStatus	
Syntax:	`Std_ReturnType Dem_SetPtoStatus( boolean PtoStatus )`	
Service ID[hex]:	0x79	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	PtoStatus	sets the status of the PTO (TRUE==active; FALSE==inactive)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	returns E_OK when the new PTO-status has been adopted by the DEM; returns E_NOT_OK in all other cases.
Description:	--	

8.4 Expected Interfaces

In this chapter, all interfaces required from other modules are listed.

8.4.1 Mandatory Interfaces

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

API function	***Description***

8.4.2 Optional Interfaces

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

Dem255:

API function	*Description*
Det_ReportError	Service to report development errors.
NvM_SetRamBlockStatus	Service for setting the RAM block status of an NVRAM block.
NvM_WriteBlock	Service to copy the data of the RAM block to its corresponding NV block.
Fim_DemTriggerOnEventStatus	This service to be provided to the DEM in order to call FIM upon status changes.
NvM_ReadBlock	Service to copy the data of the NV block to its corresponding RAM block.
NvM_GetErrorStatus	Service to read the block dependent error/status information.

8.4.3 Configurable interfaces

In this chapter, all interfaces are listed where the target function could be configured. The target function is usually a call-back function. The names of these kind of interfaces is not fixed because they are configurable. The figure below gives an overview of the class DemConfigurationInterfaces.

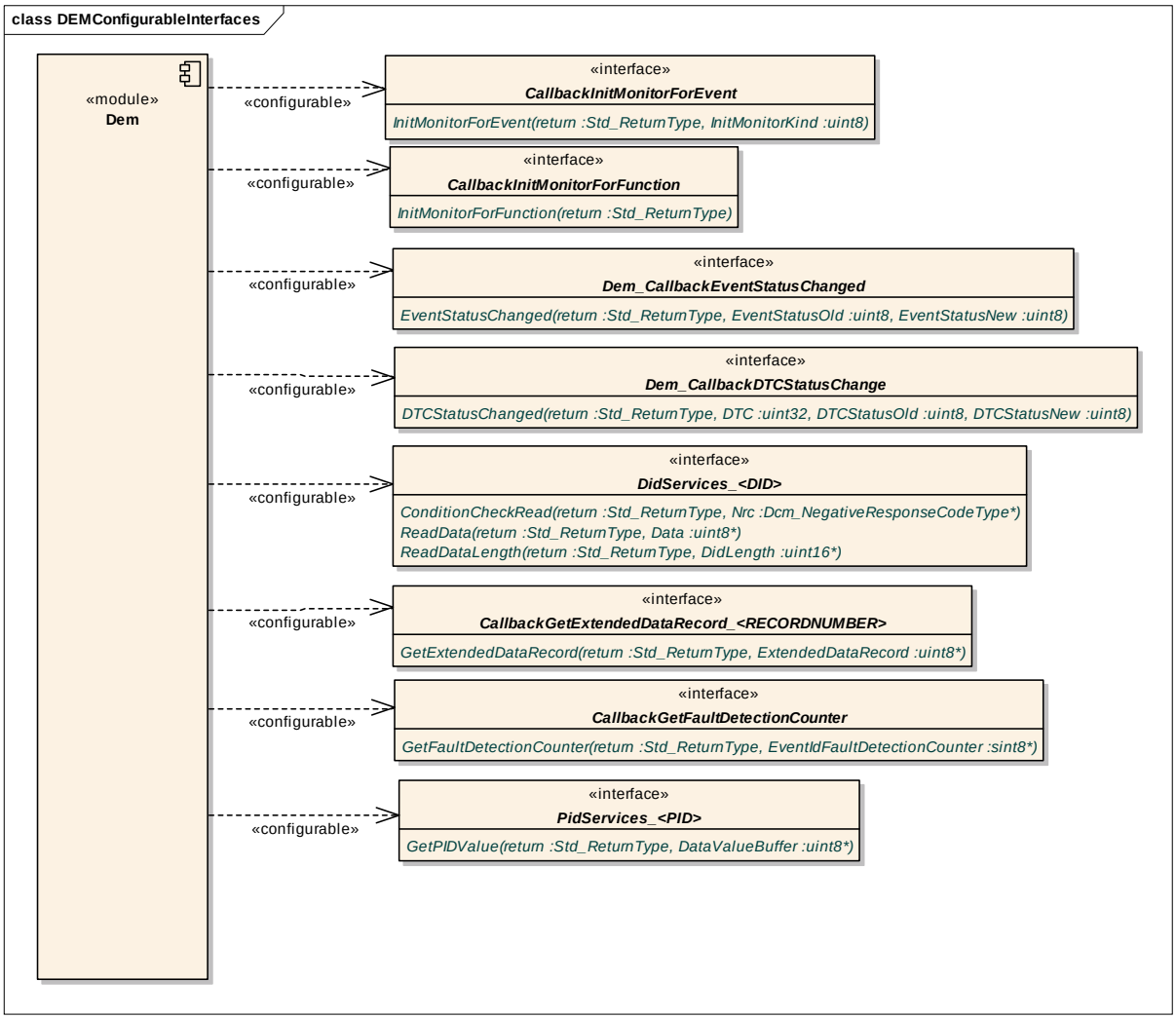


Figure 17 Configuration interfaces of the DEM module

8.4.3.1 RTE-Interface SW-Components ⇔ DEM

The callback interface from DEM to SW-Components is realized via RTE port interfaces.

8.4.3.1.1 InitMonitorForEvent

Dem256:

Service name:	InitMonitorForEvent
Syntax:	`Std_ReturnType InitMonitorForEvent( uint8 InitMonitorKind )`
Sync/Async:	Synchronous

Reentrancy:	Reentrant	
Parameters (in):	InitMonitorKind	uint8 {Clear, Restart} Identification of the type of init function to be called from Monitor Function. DEM_INIT_MONITOR_RESTART 0x02 Monitor Function of the EventId is requested to restart, DEM_INIT_MONITOR_CLEAR 0x01 Monitor Function of the EventId is cleared and all internal values and states are reseted.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	Inits the Monitor Function of a specific Event {EventId}.	

Dem376: The function InitMonitorForEvent shall init the Monitor Function of a specific Event (EventId).

The parameter InitMonitorKind is used to choose the type of initialisation.

The function InitMonitorForEvent is called from the DEM module and has to be provided by SW-C (refer to Dem003).

The function InitMonitorForEvent does not require API parameter checks.

Caveats of InitMonitorForEvent: The configuration of the DEM module during integration of Monitor Functions is system specific.

Configuration of InitMonitorForEvent: The link between the EventId and the corresponding function of the event (InitMonitorForEvent) is configured within the DEM module.

The following figures show an example of the function name after configuration:

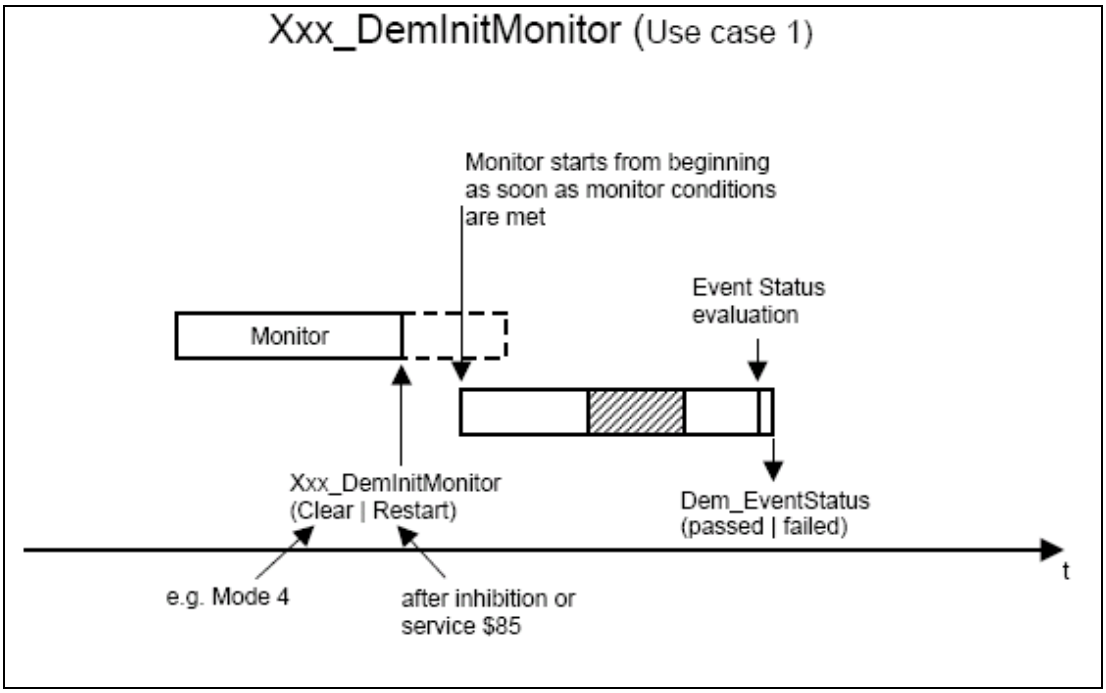


Figure 18 Describes a use-case of the interface InitMonitorForEvent for a specific event

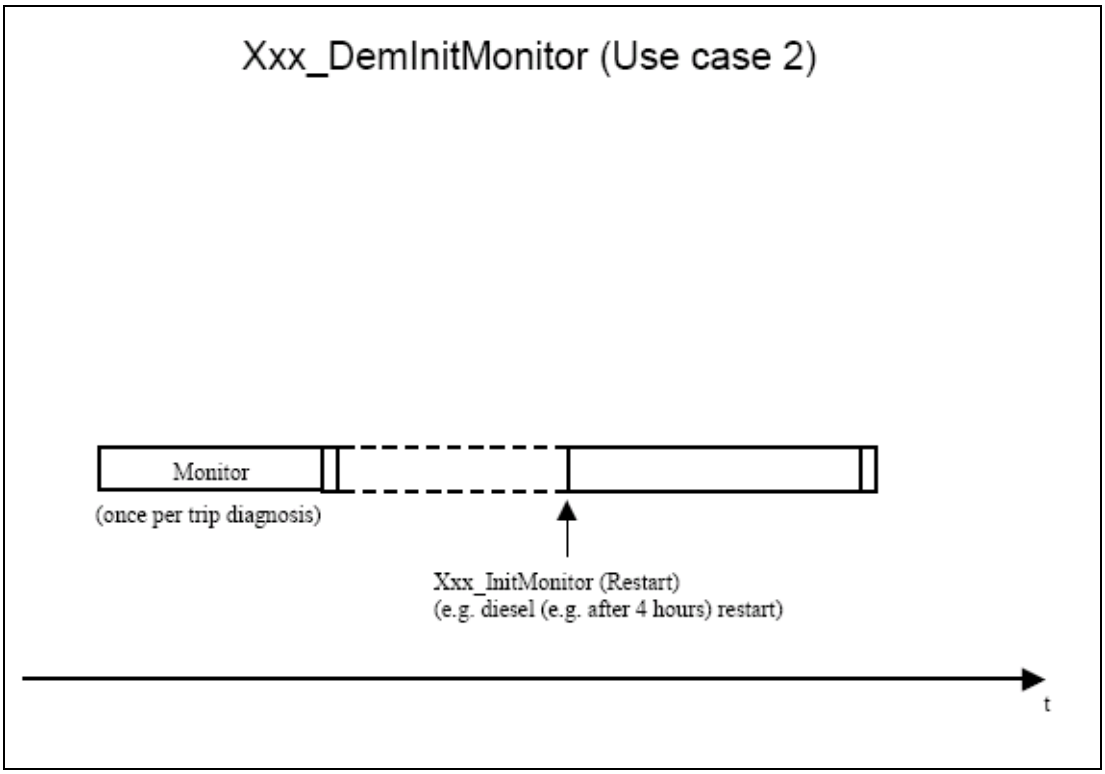


Figure 19 Describes a use-case of the interface InitMonitorForEvent for a specific event

8.4.3.1.2 InitMonitorForFunction

Dem258:

Service name:	InitMonitorForFunction	
Syntax:	`Std_ReturnType InitMonitorForFunction( )`	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	Resets the {Function} of the Module Xxx.	

Dem049: The DEM module shall call the function InitMonitorForFunction to initialize the {Function} of a specific SW-C.

Example: Adaptations may be initialized in case of clearing the DEM module (on service 04/ISO15031-5 request).

Caveats of InitMonitorForFunction: DEM module configuration during integration of Monitor Functions is system specific.

Configuration of InitMonitorForFunction: During the configuration of a system, one list has to be created to assign functions to be initialized. If different clearing processes have to be distinguished (only powertrain, wiper system, …) then several task lists have to be created.

The term {Function} is a placeholder for a real unique function name, provided by the SW-C.

8.4.3.1.3 EventStatusChanged

Dem259:

Service name:	EventStatusChanged	
Syntax:	`Std_ReturnType EventStatusChanged( uint8 EventStatusOld, uint8 EventStatusNew )`	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	EventStatusOld	Event staus before change Bit 0 TestFailed is set to 1 if the last event status update by the

		function Dem_SetEventStatus(Passed | Failed) was called with failed. The status is set to 0 if Dem_SetEventStatus is called with passed, on tester clear command and by API Dem_ResetEventStatus. Bit 0 and 6 is intended to set/reset monitor inhibit or default. Bit 1 TestFailedThisOperationCycle is set if at least one time the function Dem_SetEventStatus (passed | failed) is called with failed this cycle. Intended to be used for defaults reset only at next key on. Bit 2 PendingDTC is set when associated DTC becomes available in Mode07 (currently corresponds to ISO pending bit ? [9] Intended to be used for the control of IUMPR counters. Bit 3 ConfirmedDTC is set when associated DTC becomes available in Mode03 (currently corresponds to ISO confirmed bit ? [9] Could be used to set e.g. service request message. Bit 4 TestNotCompletedSinceLastClear is set to 0 if at least one time the function Dem_SetEventStatus (passed | failed) is called after last ClearDTC. Bit 5 testFailedSinceLastClear is set to 0 if at least one time the function Dem_SetEventStatus is caled with failed this cycle. Bit 6 TestNotCompletedThisOperationCycle is set if at least one time the function Dem_SetEventStatus (passed | failed) is called within this cycle (the usage of different cycles is application-specific, if only one cycle is used, the differentiation is obsolete). Bit 7 WarningIndicatorRequested reports the status of any warning indicators associated with a particular DTC.
	EventStatusNew	Event status after change Bit 0 TestFailed is set to 1 if the last event status update by the function Dem_SetEventStatus(Passed | Failed) was called with failed. The status is set to 0 if Dem_SetEventStatus is called with passed, on tester clear command and by API Dem_ResetEventStatus. Bit 0 and 6 is intended to set/reset monitor inhibit or default. Bit 1 TestFailedThisOperationCycle is set if at least one time the function Dem_SetEventStatus (passed | failed) is called with failed this cycle. Intended to be used for defaults reset only at next key on. Bit 2 PendingDTC is set when associated DTC becomes available in Mode07 (currently corresponds to ISO pending bit ? [9] Intended to be used for the control of IUMPR counters. Bit 3 ConfirmedDTC is set when associated DTC becomes available in Mode03 (currently corresponds to ISO confirmed bit ? [9] Could be used to set e.g. service request message.

		Bit 4 TestNotCompletedSinceLastClear is set to 0 if at least one time the function Dem_SetEventStatus (passed | failed) is called after last ClearDTC. Bit 5 testFailedSinceLastClear is set to 0 if at least one time the function Dem_SetEventStatus is caled with failed this cycle. Bit 6 TestNotCompletedThisOperationCycle is set if at least one time the function Dem_SetEventStatus (passed | failed) is called within this cycle (the usage of different cycles is application-specific, if only one cycle is used, the differentiation is obsolete). Bit 7 WarningIndicatorRequested reports the status of any warning indicators associated with a particular DTC.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	Triggers on changes of the extended Event status	

Dem285: SW-Components shall provide the function EventStatusChanged that will be triggered by the DEM module on changes of the extended Event status.

Caveats of EventStatusChanged: In case of disabling the event status update the function EventStatusChanged does not need to be called because no status change can be reported.

Configuration of EventStatusChanged: During system configuration, lists have to be created to assign functions to the required event status triggers, e.g. event status change from not tested to tested in this cycle.

8.4.3.1.4 DTCStatusChanged

Dem260:

Service name:	DTCStatusChanged	
Syntax:	`Std_ReturnType DTCStatusChanged( uint32 DTC, uint8 DTCStatusOld, uint8 DTCStatusNew )`	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	This is the DTC the change trigger is assigned to.
	DTCStatusOld	DTC status before change
	DTCStatusNew	DTC status after change
Parameters (inout):	None	
Parameters (out):	None	

Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	Triggers on changes of the DTC status.	

Dem284: SW-Components shall provide the function DTCStatusChanged that will be triggered by the DEM module on changes of the DTC status.

Configuration of DTCStatusChanged: During system configuration, lists have to be created to assign functions to the required event status triggers, e.g. event status change from not tested to tested in this cycle.

8.4.3.1.5 DidServices_<DID>

The interface DidServices_<DID> is provided by the DCM module and returns the associated data value for a requested data identifier. This interface is used by SW-Cs and DEM module.

Dem261: The DEM module shall use the following opereations of the interface DidServices_<DID> described below: ConditionCheckRead, ReadData and ReadDataLength.

8.4.3.1.5.1 ConditionCheckRead

Service name:	ConditionCheckRead	
Syntax:	`Std_ReturnType ConditionCheckRead( Dcm_NegativeResponseCodeType* Nrc )`	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	Nrc	--
Parameters (out):	None	
Return value:	Std_ReturnType	--
Description:	--	

Dem380: The DEM module shall return the default value 0x22 for the parameter Dcm_NegativeResponseCodeType.

8.4.3.1.5.2 ReadData

Service name:	ReadData	
Syntax:	`Std_ReturnType ReadData( uint8* Data )`	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	Data	--
Return value:	Std_ReturnType	--
Description:	--	

8.4.3.1.5.3 ReadDataLength

Service name:	ReadDataLength	
Syntax:	`Std_ReturnType ReadDataLength( uint16* DidLength )`	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DidLength	--
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	--
Description:	--	

Furthermore, the interface DidServices_<DID> contains the functions ConditionCheckWrite, FreezeCurrentState, GetScalingInformation, ResetToDefault, ReturnControlToECU, ShortTermAdjustment and WriteData, used by the DCM module. All these functions, including ConditionCheckRead, ReadData and ReadDataLength, are provided by SW components.

Dem283: The DEM module shall use the interface DidServices_<DID> when collecting data from SW-Cs.

This functionality is needed when the DEM module is required to store FreezeFrame data associated with a specifc diagnostic event. The software component which is responsible for providing and updating the data values (e.g. vehicle speed, RPM…) has to provide the interface DidServices_<DID>, too.

If the SW-C cannot not povide the requested data (ConditionCheckRead, ReadData and ReadDataLength return E_NOT_OK or ConditionCheckRead did not pass) the

DEM fills the missing data with the padding value 0xFF, reports the development error DEM_E_NODATAAVAILABLE to the DET and continuous his normal operation.

8.4.3.1.6 GetExtendedDataRecord_<RECORDNUMBER>

Dem262:

Service name:	GetExtendedDataRecord	
Syntax:	`Std_ReturnType GetExtendedDataRecord( uint8* ExtendedDataRecord )`	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	ExtendedDataRecord	Pointer to the buffer in the DEM which should be filled with the requested value
Return value:	Std_ReturnType	The return value specified whether the API call has been successful (Extended Data Record available) or not (no Extended Data Record available) according to Std_ReturnType. In case of E_NOT_OK the DEM fills the missing Data with the padding value 0xFF, reports the development error DEM_E_NODATAAVAILABLE to the DET and continuous his normal operation.
Description:	Gets an extended data record	

Dem282: The function GetExtendedDataRecord_<RECORDNUMBER> (provided by a SW-C) may be used in case the Extended Data Record is provided by the application.

The function GetExtendedDataRecord_<RECORDNUMBER> (provided by a SW-C) returns the associated Extended Data Record for a requested ExtendedDataRecordNumber.

The DEM module will use the function GetExtendedDataRecord_<RECORDNUMBER> as soon as it requires this ExtendedDataRecord. When using this interface the SW-C is responsible for providing and updating the data values contained in the ExtendedDataRecord (e.g. vehicle speed, RPM, …).

Configuration of GetExtendedDataRecord_<RECORDNUMBER>: The configuration has to provide one or several ExtendedDataRecordNumber(s) assigned to one DTC because the DEM module might store different ExtendedDataRecords depending on the point in time when the event is stored in the event memory (example: first and last occurrence of fault).

8.4.3.1.7 GetFaultDetectionCounter

Dem263:

Service name:	GetFaultDetectionCounter	
Syntax:	`Std_ReturnType GetFaultDetectionCounter( sint8* EventIdFaultDetectionCounter )`	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	EventIdFaultDetectionCounter	This parameter receives the Fault Detection Counter information of the requested EventId. If the return value of the function call is other than E_OK this parameter does not contain valid data. -128dec...127dec PASSED...FAILED according to ISO 14229-1
Return value:	Std_ReturnType	E_OK: request of severity was successful E_NOT_OK: request of severity failed
Description:	Gets the current Fault Detection Counter for a given EventID	

Dem264: The DEM module shall use the function GetFaultDetectionCounter to request the current Fault Detection Counter for a given EventID.

Dem265: The DEM module shall use the function GetFaultDetectionCounter only if debouncing is not done by the DEM module itself.

8.4.3.2 OBD-specific Interfaces

The OBD-specific callback interfaces from DEM to SW-Components are also realized via RTE port interfaces.

8.4.3.2.1 GetPIDValue

Dem326:

Service name:	GetPIDValue	
Syntax:	`Std_ReturnType GetPIDValue( uint8* DataValueBuffer )`	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	DataValueBuffer	Buffer provided by the caller and filled by the callee. The correct size of the buffer is part of the configuration,
Parameters (out):	None	
Return value:	Std_ReturnType	See the definition of the corresponding AUTOSAR ClientServerInterface

Description:	Fills the provided DataValueBuffer with the value of certain PID

8.5 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.5.1 Dem_MainFunction

Dem266:

Service name:	Dem_MainFunction
Syntax:	`void Dem_MainFunction( )`
Service ID[hex]:	0x55
Timing:	FIXED_CYCLIC
Description:	Processes all not event based DEM internal functions.

Dem125: The function Dem_MainFunction shall process all not event based DEM module internal functions.

Dem286: The DEM module's environment (e.g. by operating system) shall call the function Dem_MainFunction periodically as cyclic task.

Configuration of Dem_MainFunction: The cyclic time for the main function has to be defined as an operating system task or run able entity.

Terms and definitions:
Fixed cyclic: Fixed cyclic means that one cycle time is defined at configuration and shall not be changed because functionality is requiring that fixed timing (e.g. filters).
Variable cyclic: Variable cyclic means that the cycle times are defined at configuration, but might be mode dependent and therefore vary during runtime.
On pre condition: On pre-condition means that no cycle time can be defined. The function will be called when conditions are fulfilled. Alternatively, the function may be called cyclically however the cycle time will be assigned dynamically during runtime by other modules.

9 Sequence diagrams

9.1 ControlDTCStorage

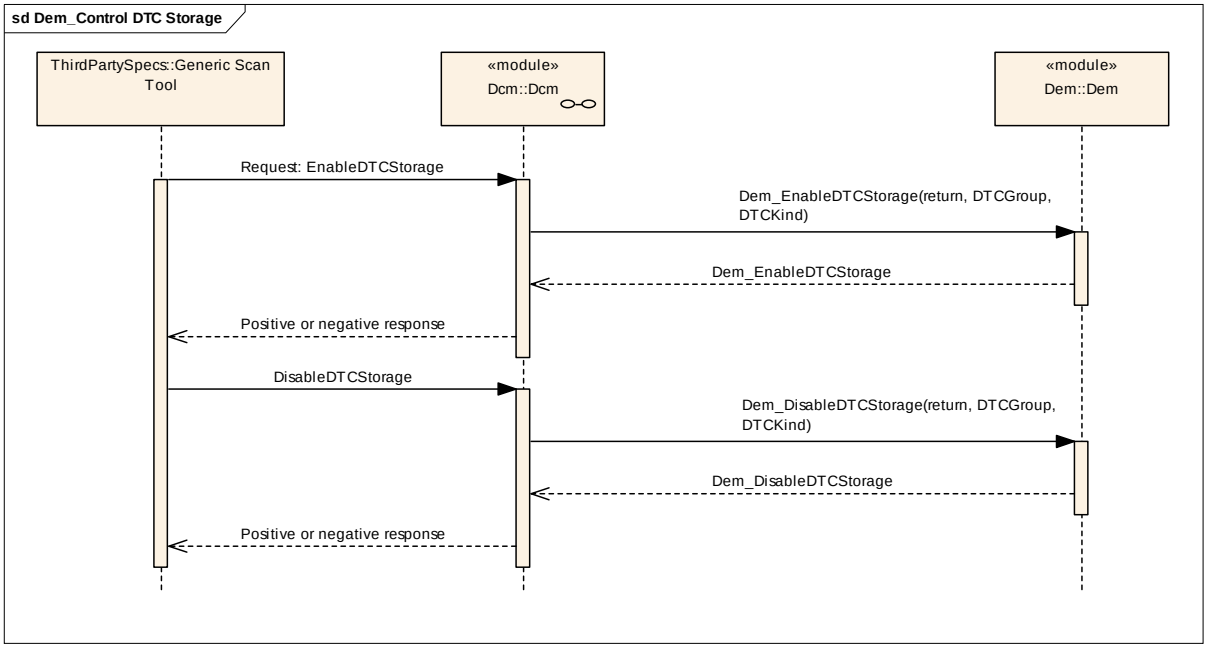


Figure 20 Sequence diagram of Dem_ControlDTCStorage

9.2 Dem_ClearDTC

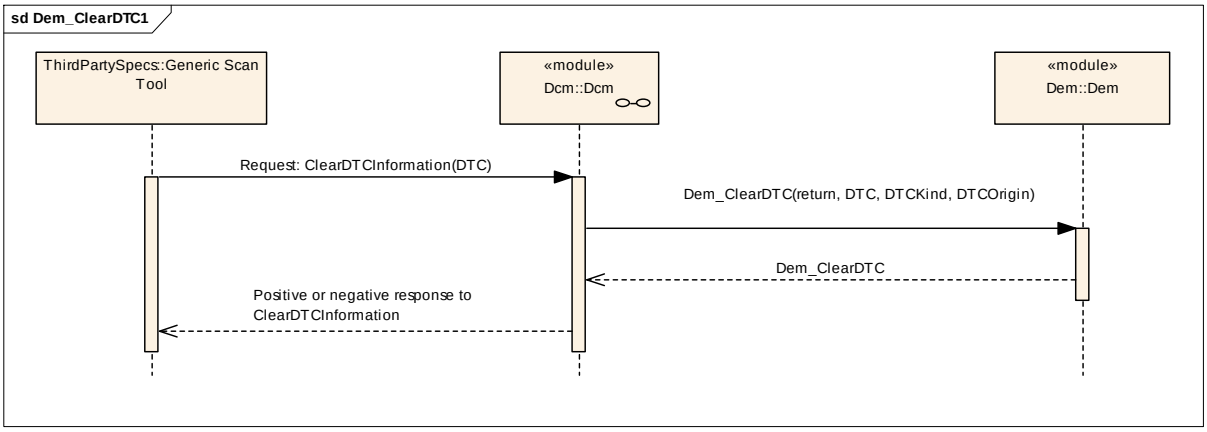


Figure 21 Sequence diagram of DEM_ClearDTC

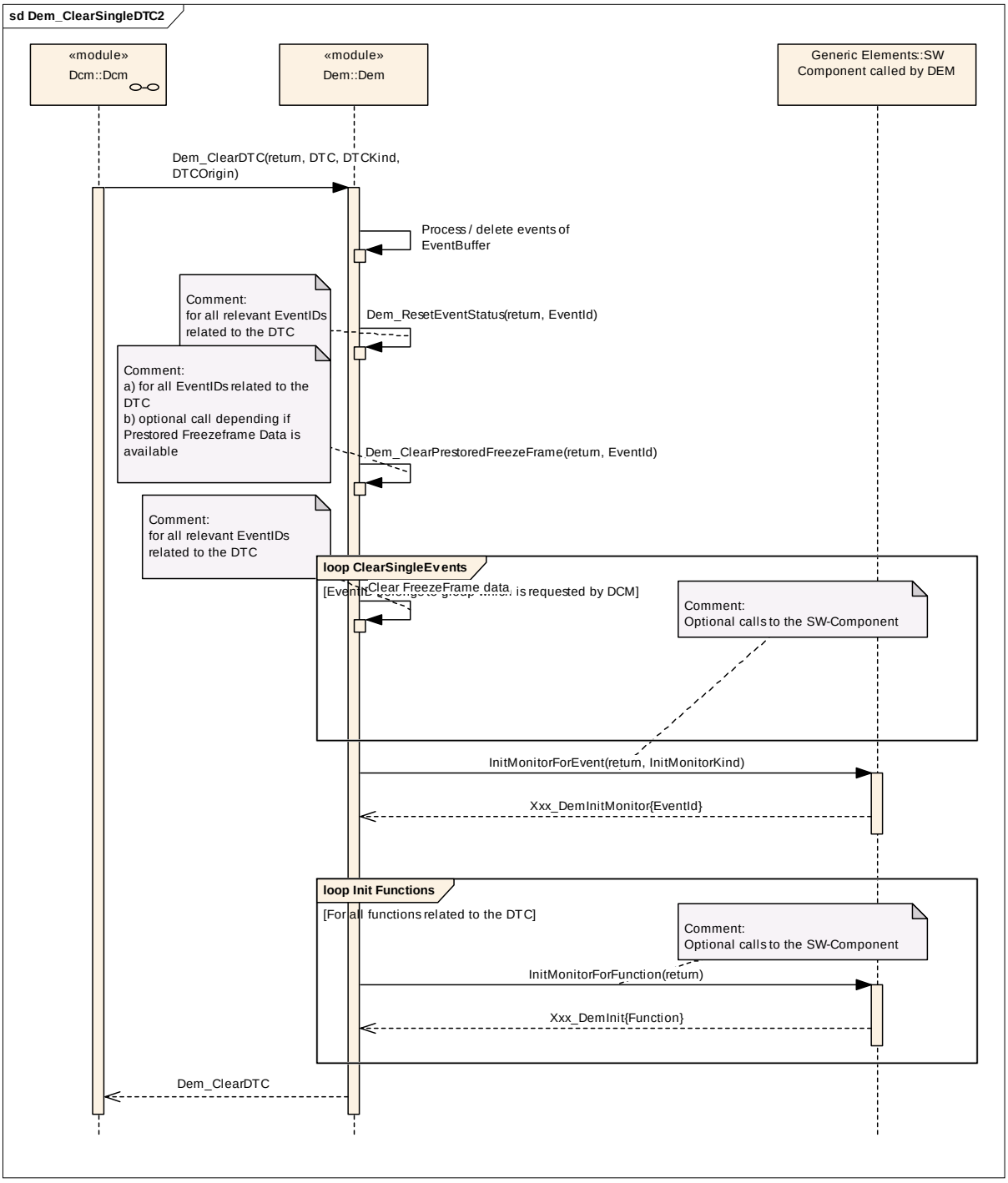


Figure 22 Sequence diagram of Dem_ClearSingleDTC

9.3 Dem_DisableEventStatusUpdate

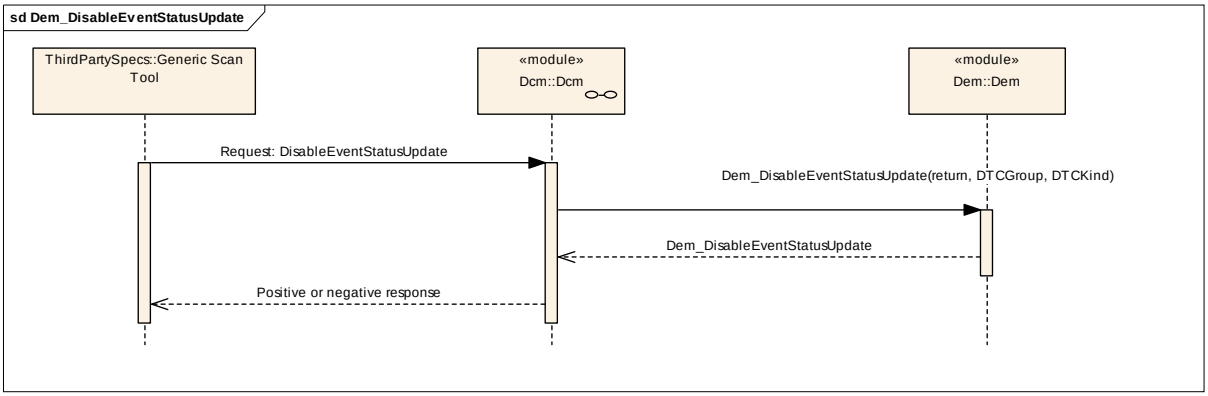


Figure 23 Sequence diagram of Dem_DisableEventStatusUpdate

9.4 Dem_EnableEventStatusUpdate

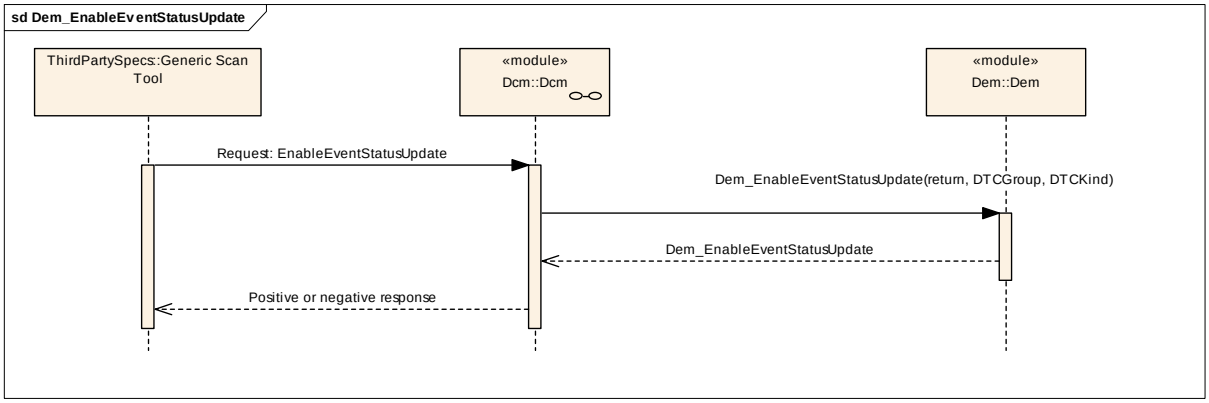


Figure 24 Sequence diagram of Dem_EnableEventStatusUpdate

9.5 Dem_GetDTCByOccurrenceTime

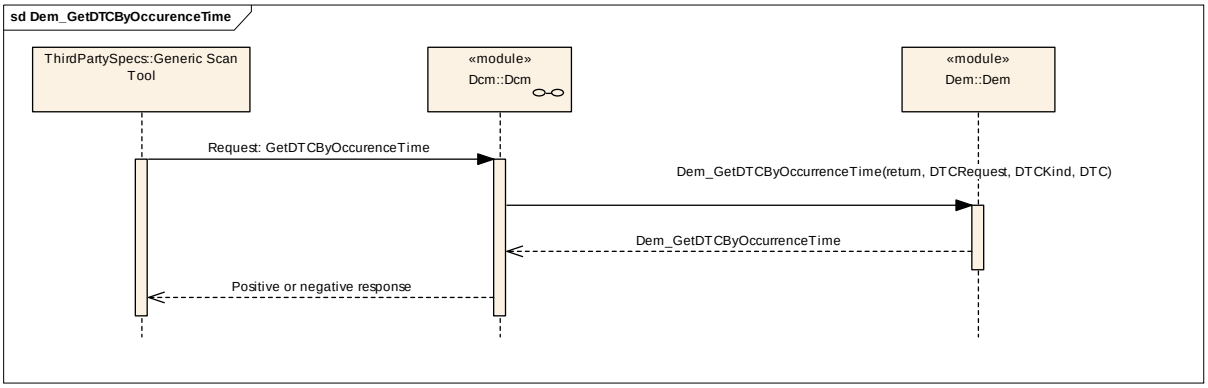


Figure 25 Sequence diagram of Dem_GetDTCByOccurenceTime

9.6 Dem_GetExtendedDataRecordByDTC

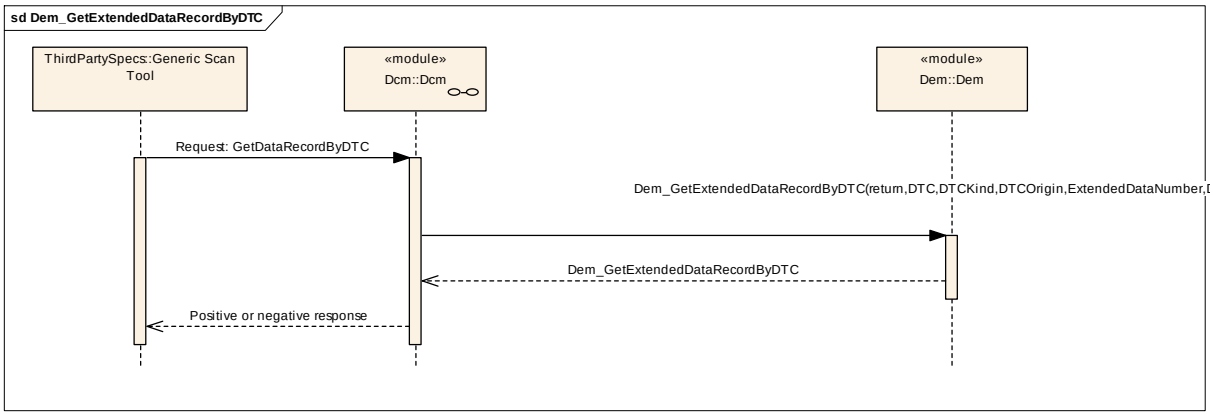


Figure 26 Sequence diagram of Dem_GetExtendedDataRecordByDTC

9.7 Dem_GetOBDReadiness

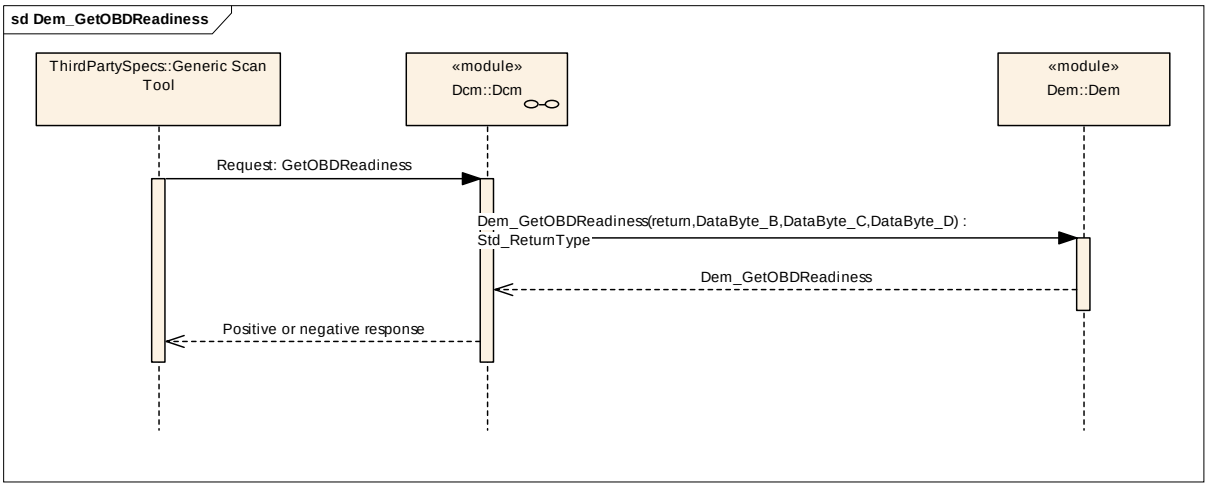


Figure 27 Sequence diagram of Dem_GetOBDReadiness

9.8 Dem_GetStatusOfDTC

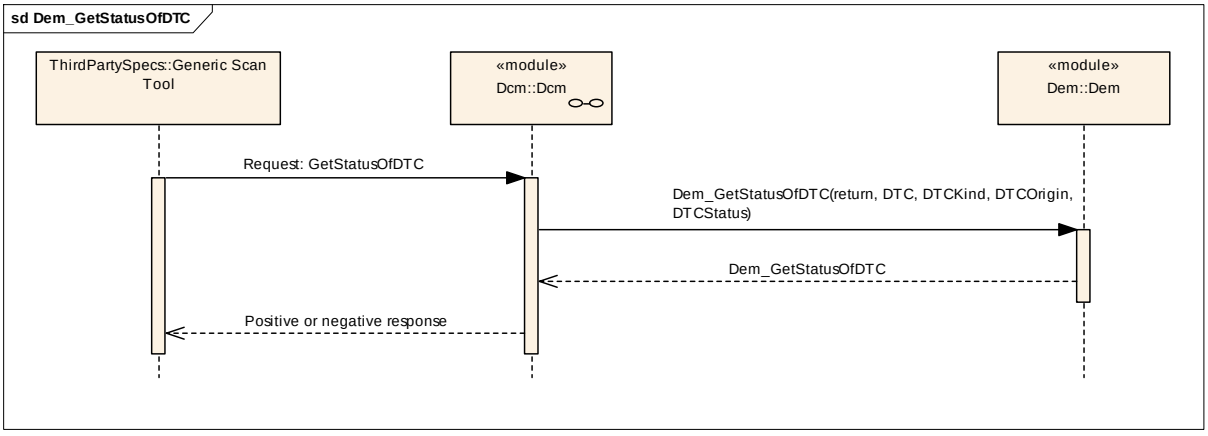


Figure 28 Sequence diagram of Dem_GetStatusOfDTC

9.9 Dem_GetSizeOfFreezeFrame

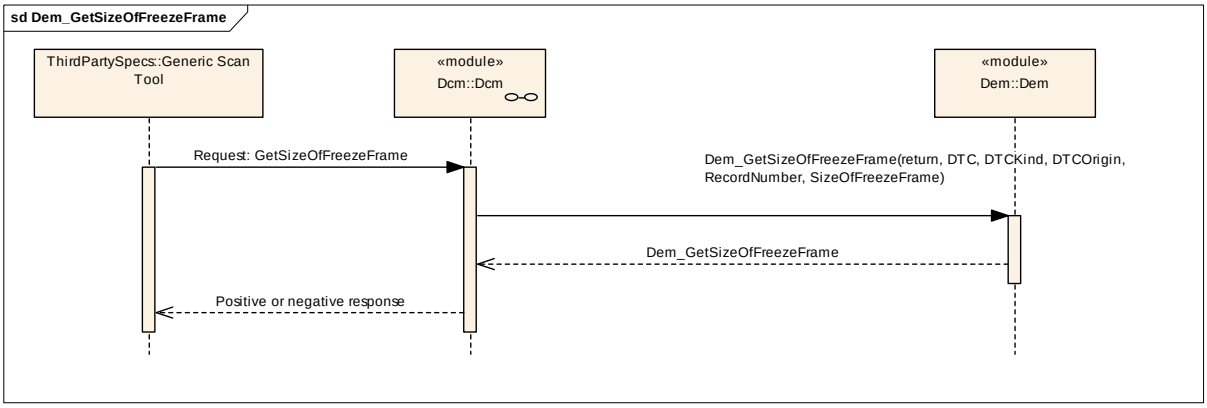


Figure 29 Sequence diagram of Dem_GetSizeOfFreezeFrame

9.10 GetOBDFaultInformation

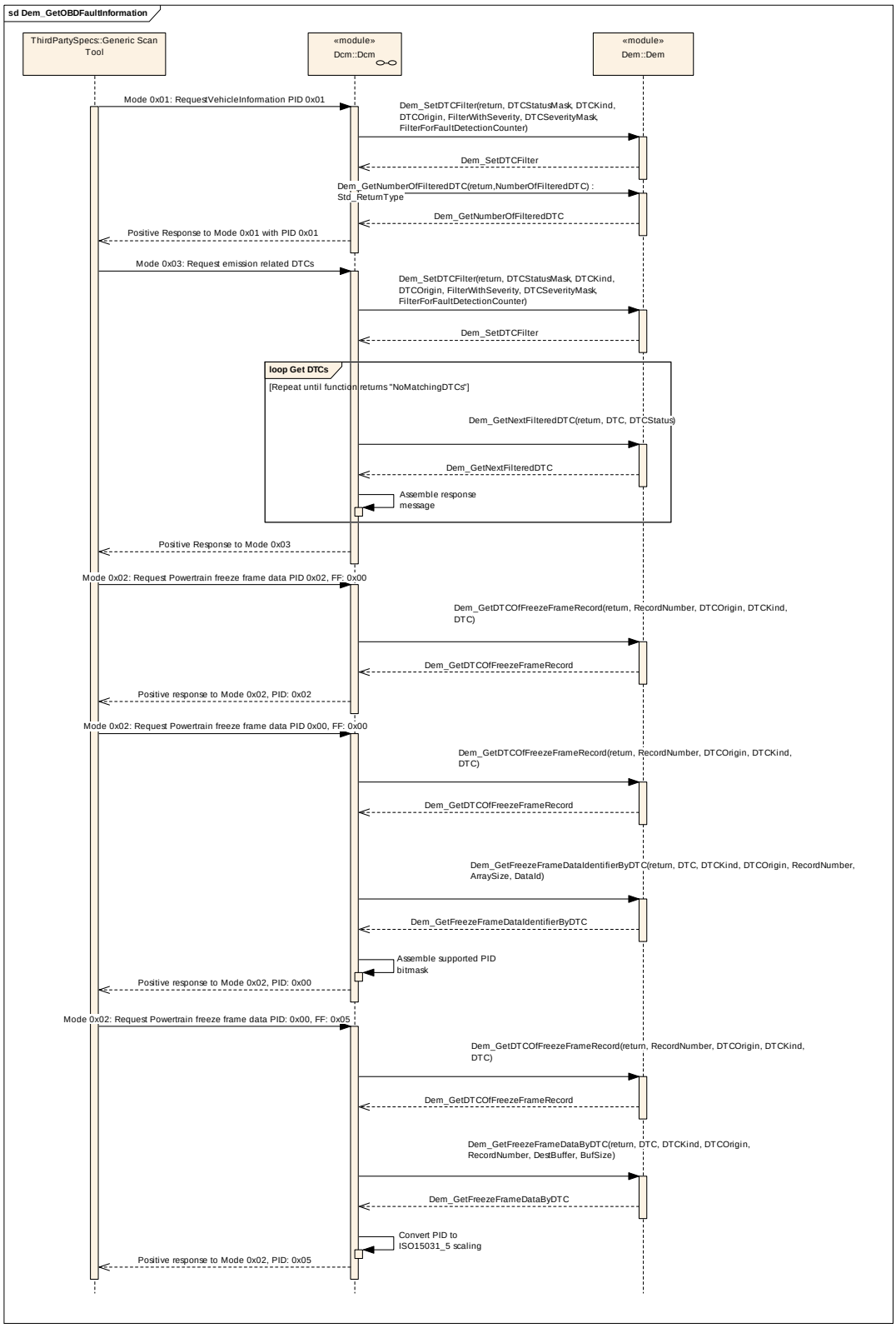


Figure 30 Sequence diagram of GetOBDFaultInformation

9.11 ReportDTCByStatusMask

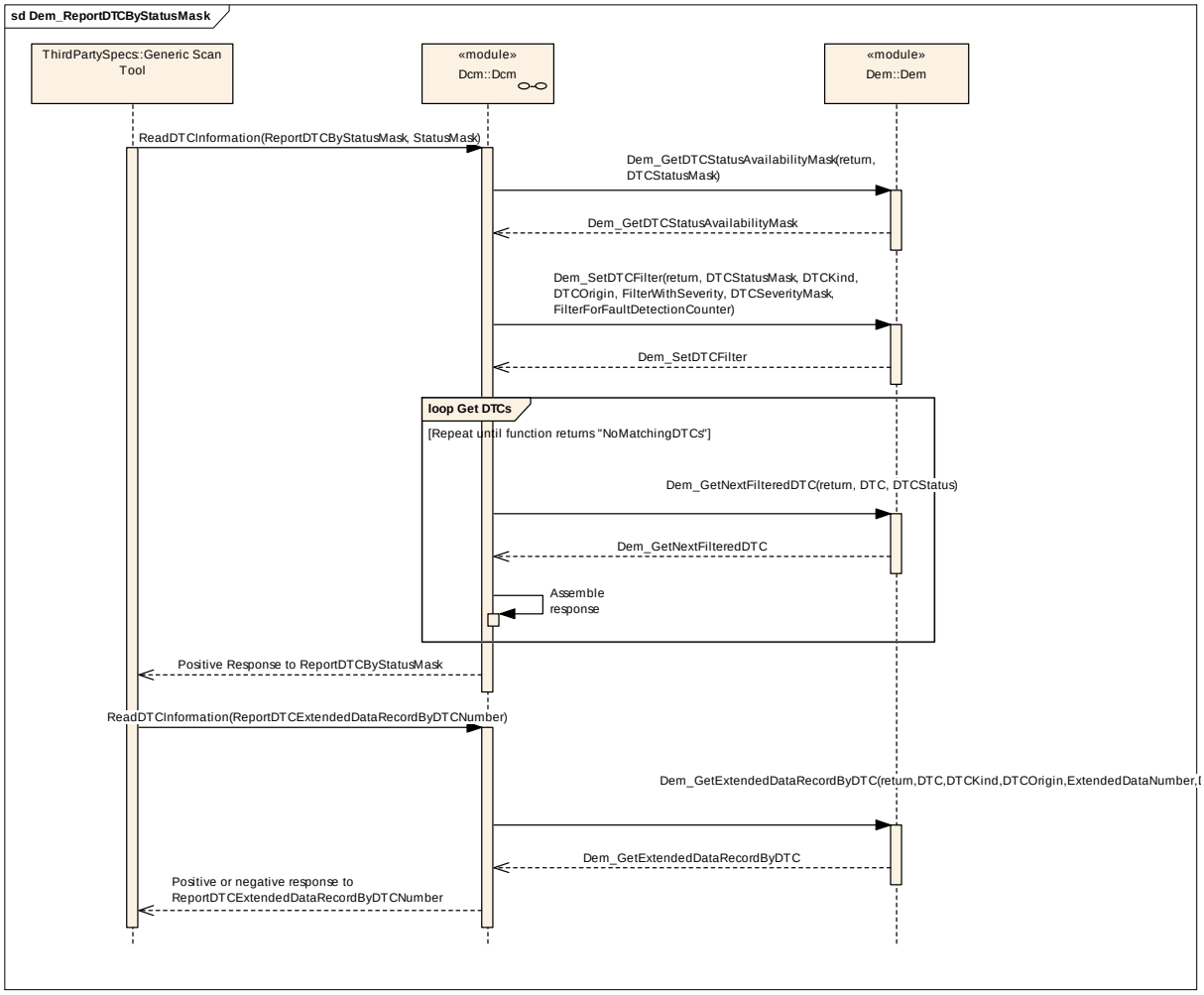


Figure 31 Sequence diagram of ReportDTCStatusMask

9.12 Fim_DemTriggerOnEventStatus

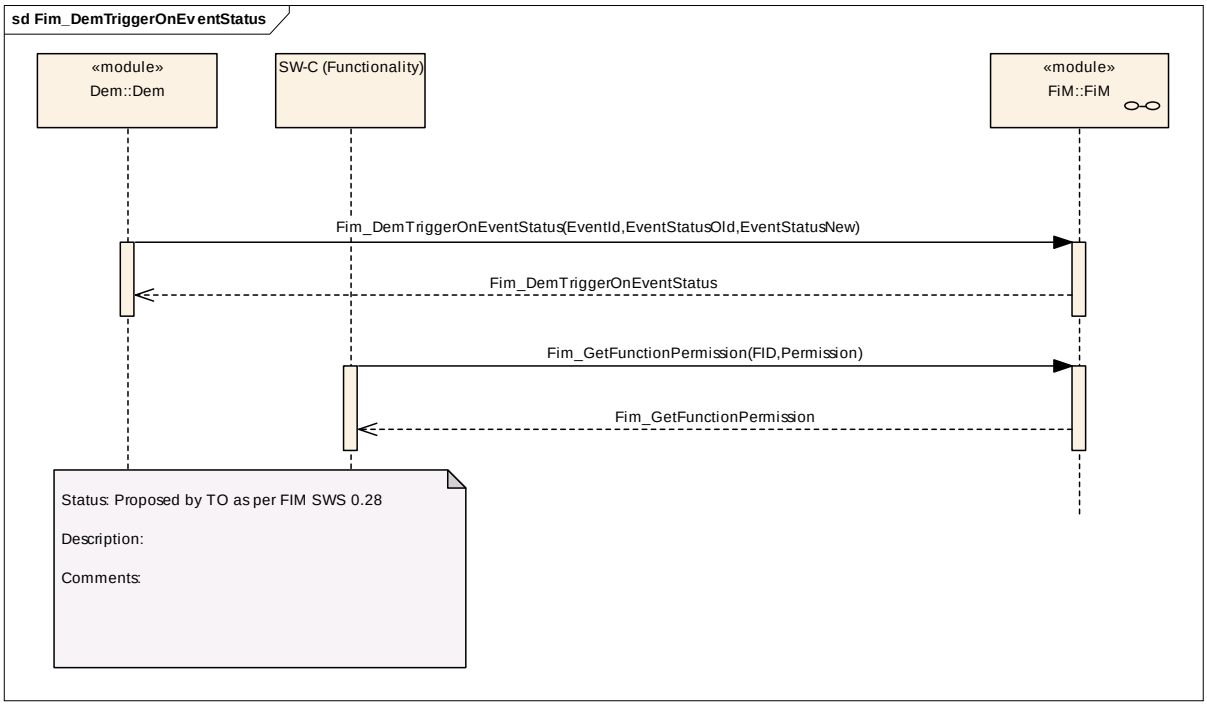


Figure 32 Sequence diagram of Fim_DemTriggerOnEventStatus

9.13 ProcessEvent (Example)

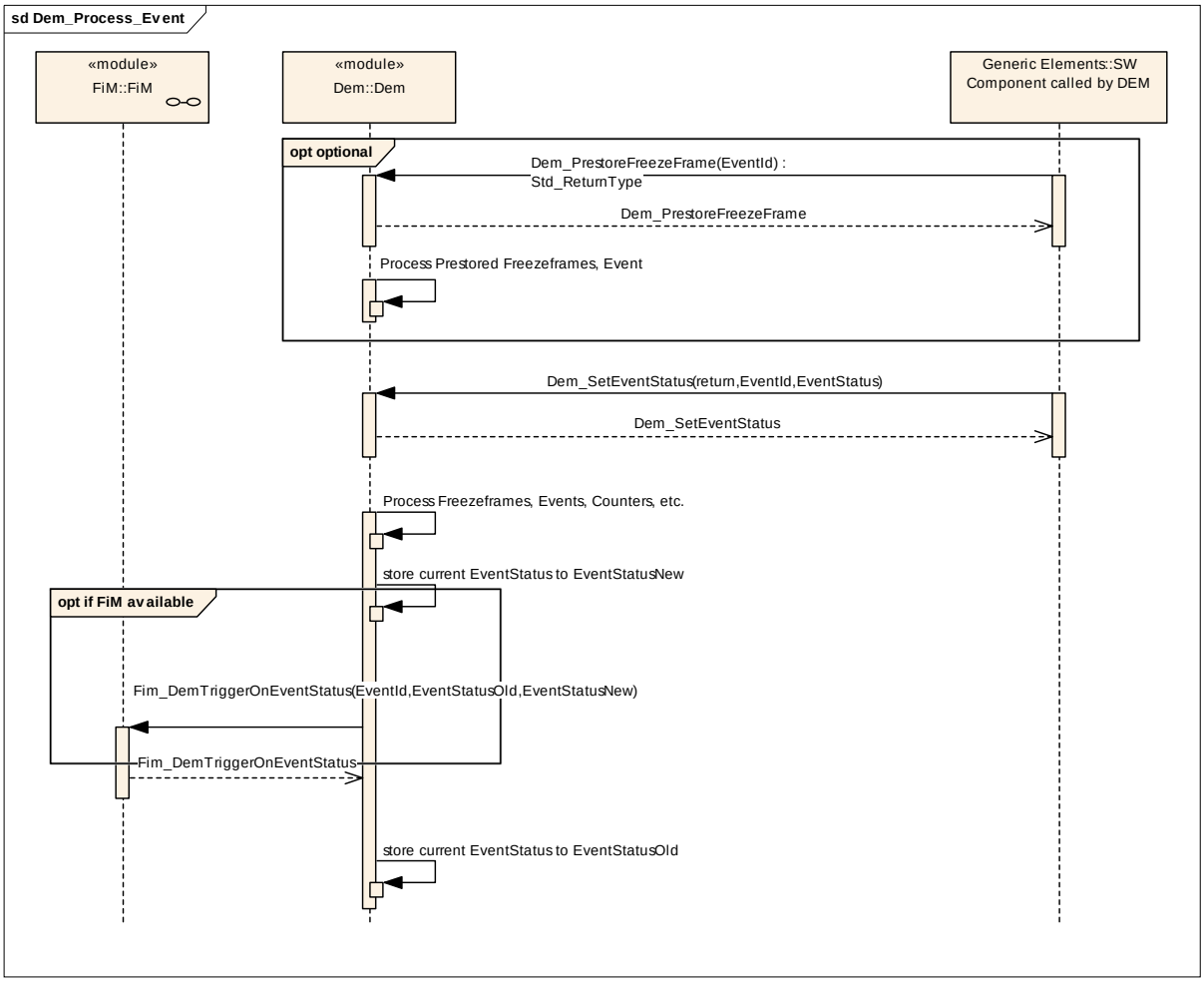


Figure 33 Sequence diagram for an example of ProcessEvent

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 DEM.

Chapter 10.2.2 specifies published information of the module DEM.

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR ECU Configuration Specification [2]
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.
- AUTOSAR Layered Software Architecture [3]

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Variants

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

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.

- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 8.

10.2.1 Variants

The following configuration parameters shall be available:

- **Dem267:** variant 1: only pre-compile time configuration parameters
- **Dem268:** variant 2: mix of pre-compile- and post build time-configuration parameters.

Link time configurable parameters are not used in this specification.

10.2.2 Dem

Module Name	*Dem*
Module Description	Configuration of the Dem (Diagnostic Event Manager) module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemConfigSet	1	This container contains the configuration parameters and sub containers of the DEM module supporting multiple configuration sets. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
DemGeneral	1	This container contains the configuration (parameters) of the BSW DEM

10.2.3 DemGeneral

SWS Item	**Dem128 :**
Container Name	DemGeneral{DemConfiguation}
Description	This container contains the configuration (parameters) of the BSW DEM
Configuration Parameters	

SWS Item	**Dem128, Dem107 :**	
Name	DemBswErrorBufferSize {DEM_BSW_ERROR_BUFFER_SIZE}	
Description	Maximum number of elements in buffer for handling of BSW errors (ref. to Dem107).	
Multiplicity	1	
Type	IntegerParamDef	
Range	0 .. 255	

Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem128 :**		
Name	DemDevErrorDetect {DEM_DEV_ERROR_DETECT}		
Description	Activate/Deactivate the Development Error Detection and Notification. true: Development Error Detection and Notification activated false: Development Error Detection and Notification deactivated		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**Dem128 :**		
Name	DemDtcStatusAvailabilityMask {DEM_DTC_STATUS_AVAILABILITY_MASK}		
Description	Mask for the supported DTC status bits by the DEM. This mask is used by UDS service 0x19.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem128 :**		
Name	DemFFDataIDLength {DEM_FF_DID_LENGTH}		
Description	Length of the DID and PID of FreezeFrames in Bytes.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 4		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem128 :**		
Name	DemMaxNumberEventEntryMirror {DEM_MAX_NUMBER_EVENT_ENTRY_MIR}		
Description	Maximum number of events which can be stored in the mirror memory		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem128 :**		
Name	DemMaxNumberEventEntryPermanent {DEM_MAX_NUMBER_EVENT_ENTRY_PER}		
Description	Maximum number of events which can be stored in the permanent memory		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem128 :**		
Name	DemMaxNumberEventEntryPrimary {DEM_MAX_NUMBER_EVENT_ENTRY_PRI}		
Description	Maximum number of events which can be stored in the primary memory		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem367 :**		
Name	DemMaxNumberEventEntrySecondary {DEM_MAX_NUMBER_EVENT_ENTRY_SEC}		
Description	Maximum number of events which can be stored in the secondary memory		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem128 :**		
Name	DemMaxNumberPrestoredFF {DEM_MAX_NUMBER_PRESTORED_FF}		
Description	Defines the maximum number for prestored FreezeFrames. If set to 0, then prestorage is not supported by the ECU.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem366 :**		
Name	DemOBDSupport		
Description	This configuration switch defines whether OBD is supported or not		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem365 :**		
Name	DemPTOSupport		
Description	Defines whether PTO support is enabled or not		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**--**		
Name	DemTaskTime		
Description	Allow to configure the time for the periodic cyclic task (in ms). Please note: This configuration value shall be equal to the value in the ScheduleManager module. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. DEM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DEM. min: A negative value is not allowed. upperMultiplicity: Exactly one TaskTime must be specified per configuration. lowerMultiplicity: Exactly one TaskTime must be specified per configuration.		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem128 :**		
Name	DemTypeOfDTCSupported {DEM_TYPE_OF_DTC_SUPPORTED}		
Description	This parameter defines the format returned by Dem_GetTranslationType.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	DEM_ISO11992_4		
	DEM_ISO14229_1		
	DEM_ISO15031_6		
	DEM_SAEJ1939_73		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem128 :**		
Name	DemVersionInfoApi {DEM_VERSION_INFO_API}		
Description	Activate/Deactivate the version information API. true: version information activated false: version information deactivated		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemEnableCondition	0..255	This container contains the configuration (parameters) for Enable Conditions.
DemExtendedDataClass	0..*	This class contains the combinations of extended data records (ExtendedDataClassRec).
DemExtendedDataRecordClass	0..253	This container contains the configuration (parameters) for ExtendedDataClassRecords
DemFreezeFrameClass	0..255	This container contains the configuration (parameters) for FreezeFrameClass.
DemIndicator	0..255	This container contains the configuaration (parameters) for Indicators. Note hat this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the INDICATOR_NAME.
DemNvramBlockId	0..*	This container contains the configuaration (parameters) for Dem_OperationCycleList. Note hat this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the VALUE_NAME.
DemOperationCycleTgt	0..*	Note hat this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the OperationCycleName.
DemPidOrDid	0..255	This container defines how the DEM can obtain the value of a PID or a DID from either a SWC or another BSWM. Whether a port or a direct function-call is used is defined by DemPidOrDidUsePort. Whether the value is a PID or a DID is indicated through the presence of either a DemDidIdentifier OR a DemPidIdentifier. The name of the container is the name of the R-Port used to obtain the PID or DID data (when DemPidOrDidUsePort is true).

10.2.4 DemIndicator

SWS Item	**Dem129 :**
Container Name	DemIndicator{IndicatorList}

Description	This container contains the configuaration (parameters) for Indicators. Note hat this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the INDICATOR_NAME.
Configuration Parameters	

SWS Item	**Dem129 :**		
Name	DemIndicatorID {IndicatorID}		
Description	Unique identifier of an indicator.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.5 DemCallbackEventStatusChanged

SWS Item	**Dem140 :**
Container Name	DemCallbackEventStatusChanged
Description	The presence of this container indicates that the DEM has access to an "EventStatusChanged" callback. In case there is a DemCallbackEvenStatusChangedFnc, this parameter contains the name of the function that the DEM will call. In case there is no DemCallbackEvenStatusChangedFnc, the DEM will have an R-Port requiring the interface CallbackEventStatusChanged whose name is the name of this container.
Configuration Parameters	

SWS Item	**Dem139 :**		
Name	DemCallbackEventStatusChangedFnc		
Description	Function name of prototype "EventStatusChanged"		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.6 DemCallbackDTCStatusChanged

SWS Item	**Dem140 :**
Container Name	DemCallbackDTCStatusChanged
Description	The presence of this container indicates that the DEM has access to an "DTCStatusChanged" callback, which the DEM will call to notify other components about the change in the status of a DTC. In case there is a DemCallbackDTCStatusChangedFnc, this parameter contains the name of the function that the DEM will call. In case there is no DemCallbackDTCStatusChangedFnc, the DEM will have an R-Port requiring the interface CallbackDTCStatusChanged whose name is the name of this container.
Configuration Parameters	

SWS Item	**Dem139 :**		
Name	DemCallbackDTCStatusChangedFnc		
Description	Function name of prototype "DTCStatusChanged"		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.7 DemCallbackGetExtDataRecord

SWS Item	**Dem139 :**
Container Name	DemCallbackGetExtDataRecord
Description	The presence of this container indicates that the DEM has access to an "GetExtendedDataRecord" callback, which the DEM will call to obtain an extended data record. In case there is a DemCallbackGetExtDataRecordFnc, this parameter contains the name of the function that the DEM will call. In case there is no DemCallbackGetExtDataRecordFnc, the DEM will have an R-Port requiring tthe interface CallbackGetExtendedDataRecord whose name is the name of this container
Configuration Parameters	

SWS Item	**Dem139 :**		
Name	DemCallbackGetExtDataRecordFnc		
Description	Function name of prototype "GetExtendedDataRecord"		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.8 DemPidOrDid

SWS Item	**Dem136 :**
Container Name	DemPidOrDid{FreezeFrameClass}
Description	This container defines how the DEM can obtain the value of a PID or a DID from either a SWC or another BSWM. Whether a port or a direct function-call is used is defined by DemPidOrDidUsePort. Whether the value is a PID or a DID is indicated through the presence of either a DemDidIdentifier OR a DemPidIdentifier. The name of the container is the name of the R-Port used to obtain the PID or DID data (when DemPidOrDidUsePort is true).
Configuration Parameters	

SWS Item	**--**		
Name	DemDidConditionCheckReadFnc		
Description	Name of a function of prototype "ConditionCheckRead" used to check whether the DID can be read		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	DemDidIdentifier		
Description	Identifier of the DID		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	DemDidReadDataLengthFnc		
Description	Name of a function of prototype "ReadDataLength" used to obtain the lenght of a dynamically sized DID. In case this container is present, it indicates that the DID has a dynamic length. In case DemPidOrDidUsePort=true, the actual function-name is not used as the information is obtained through an R-Port.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	DemDidReadFnc		
Description	In case of a DID and if DemPidOrDidUserPort is false, this defines the function of prototype "ReadData" used to get the value of the DID.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem136 :**		
Name	DemPidIdentifier {PID}		
Description	identifier of the PID		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem136 :**		
Name	DemPidOrDidSize {DID}		
Description	This defines the size of the PID or DID		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**--**		
Name	DemPidOrDidUsePort		
Description	True when an R-Port is used to obtain the PID (interface PIDServices) or DID (interface DIDServices). False when the information is obtained when calling functions on another BSWM.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**
Name	DemPidReadFnc
Description	Name of a function of prototype "GetPIDValue". This function is used to obtain the value of a PID for the case that DemPidOrDidUsePort is false.

Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.9 DemConfigSet

SWS Item	**Dem130 :**
Container Name	DemConfigSet{DEMConfigSet} [Multi Config Container]
Description	This container contains the configuration parameters and sub containers of the DEM module supporting multiple configuration sets. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemDTCClass	1..16777214	This container contains the configuration (parameters) for DTCClass.
DemEventParameter	0..65535	This container contains the configuaration (parameters) for events. Note that this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the DEM_EVENT_NAME.
DemGroupOfDTC	1..16777214	This container contains the configuaration (parameters) for DTC Groups.
DemOemIdClass	0..*	This container contains the configuaration (parameters) for OEMIdClass. Note hat this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the VALUE_NAME.

10.2.10 DemOemIdClass

SWS Item	**Dem141 :**
Container Name	DemOemIdClass{OEMIdClass}
Description	This container contains the configuaration (parameters) for OEMIdClass. Note hat this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the VALUE_NAME.
Configuration Parameters	

SWS Item	**Dem141 :**		
Name	DemOemID {OemID}		
Description	Defines a unique ID of a data value.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.11 DemOperationCycleTgt

SWS Item	**Dem142 :**
Container Name	DemOperationCycleTgt{Dem_OperationCycleList}
Description	Note hat this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the OperationCycleName.
Configuration Parameters	

SWS Item	**Dem142 :**		
Name	DemOperationCycle {OperationCycleName}		
Description	List of cycles for the DEM to be supported by API Dem_SetOperationCycleState in SW-C. Therein, only the symbolic names shall be used. The declaration is given via Dem.h. Further cycles can be specified as part of the DEM delivery.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	DEM_IGNITION	Ignition ON / OFF Cycle	
	DEM_OBD_DCY	OBD Driving Cycle	
	DEM_POWER	Power ON / OFF Cycle	
	DEM_WARMUP	OBD OBD Warm up Cycle	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.12 DemEventParameter

SWS Item	**Dem130 :**
Container Name	DemEventParameter{EventParameter}
Description	This container contains the configuaration (parameters) for events. Note that this container definition does not explicitly define a symbolic

	name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the DEM_EVENT_NAME.
Configuration Parameters	

SWS Item	**Dem130 :**		
Name	DemEventID {EventId}		
Description	Unique identifier of an EVENT, this parameter should not be changeable by user, because the EventId should be generated by DEM itself to prevent gaps and multiple use of an Id. max = 255 For small ECUs with < 255 different events and a limited RAM, the events should be sequentially ordered beginning with 1 and no gaps in between. max = 65535 For ECUs with > 255 different events, the events should be sequentially ordered beginning with 1 and no gaps in between.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	1 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem131 :**		
Name	DemEventKind {EventKind}		
Description	This container contains the configuaration (parameters) for DemEventType. This parameter is used to distinguish between SW-C and BSW events. SW-C events are for Dem_SetEventStatus API and BSW events are for Dem_ReportErrorStatus API.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	DEM_EVENT_KIND_BSW	event is a assigned to a BSW modul	
	DEM_EVENT_KIND_SWC	event is a assigned to a SW-C	
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem132 :**		
Name	DemDTCClassRef {DTCClass}		
Description	This defines the DTC's associated with the Event. It is allowed to have Events without a DTC.		
Multiplicity	0..1		
Type	Reference to DemDTCClass		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem135 :**		
Name	DemExtendedDataRef {ExtendedDataClassRef}		
Description	This reference defines the link to a ExtendedDataClass sampler.		
Multiplicity	0..1		
Type	Reference to DemExtendedDataClass		
ConfigurationClass	***Pre-compile time***	--	

	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem136 :**		
Name	DemFreezeFrameClassRef {FreezeFrameClassRef}		
Description	These references define the links to the FreezeFrameClass sampler. The number of linked classes complies to the possible captured freeze frames, which are also reported from UDS service 0x19.		
Multiplicity	0..255		
Type	Reference to DemFreezeFrameClass		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemCallbackEventStatusChanged	0..*	The presence of this container indicates that the DEM has access to an "EventStatusChanged" callback. In case there is a DemCallbackEvenStatusChangedFnc, this parameter contains the name of the function that the DEM will call. In case there is no DemCallbackEvenStatusChangedFnc, the DEM will have an R-Port requiring the interface CallbackEventStatusChanged whose name is the name of this container.
DemCallbackInitMForE	0..1	Monitor function which has to be initialized for the event. This is either provided by a BSWM (when the parameter DemCallbackInitMForEFnc is present) or through an R-Port requiring a CallbackInitMonitorForEvent interface whose name is the name of this container.
DemEventClass	1	This container contains the configuaration (parameters) for EventClass

10.2.13 DemExtendedDataClass

SWS Item	**Dem135 :**
Container Name	DemExtendedDataClass{ExtendedDataClassRef}
Description	This class contains the combinations of extended data records (ExtendedDataClassRec).
Configuration Parameters	

SWS Item	**Dem135 :**		
Name	DemExtendedDataClassRef {ExtendedDataClassRef}		
Description	This reference contains the link to a ExtendedDataClassRecord.		
Multiplicity	1..253		
Type	Reference to DemExtendedDataRecordClass		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.14 DemEventClass

SWS Item	**Dem131 :**
Container Name	DemEventClass{EventClass}
Description	This container contains the configuaration (parameters) for EventClass
Configuration Parameters	

SWS Item	**--**		
Name	DemConsiderPtoStatus		
Description	This is TRUE when the event is affected by the DEM PTO handling		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem131 :**		
Name	DemEventDestination {EventDestination}		
Description	The event destination assigns events to none, one or multiple origins. If no event destination is assigned to a specific event, the event is handled internally and is not visible externally to the DCM. If more than one event destination is assigned to a specific event, the event can be present in the corresponding origins. ImplementationType:		
Multiplicity	0..4		
Type	EnumerationParamDef		
Range	DEM_DTC_ORIGIN_MIRROR_MEMORY		Event information located in the mirror memory.
	DEM_DTC_ORIGIN_PERMANENT_MEMORY		Event information located in the permanent memory.
	DEM_DTC_ORIGIN_PRIMARY_MEMORY		Event information located in the primary memory.
	DEM_DTC_ORIGIN_SECONDARY_MEMORY		Event information located in the secondary memory.
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem131 :**		
Name	DemEventPriority {EventPriority}		
Description	Priority of an event, in view of full event buffer (ref. to Dem104).		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: ECU

SWS Item	**Dem131 :**		
Name	DemFFPrestorageSupported {FF_Prestorage_Supported}		
Description	If this parameter is set to true, then the Prestorage of FreezeFrames is supported by the assigned event. This parameter is useful to calculate the buffer size.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem131 :**		
Name	DemHealingAllowed {HealingAllowed}		
Description	(Dem104) general switch to allow healing/unlearning or not. true = healing/unlearning allowed false = healing/unlearning not allowed		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem131, Dem104 :**		
Name	DemHealingCycleCounter {HealingCycleCounter}		
Description	cycles needed to heal/erase event (ref. Dem104). This parameter is optional (depends on OEM).		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	1 .. 256		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	DemEnableConditionRef		
Description	Defines a Enable Condition. This parameter is optional and depends on manufacturer.		
Multiplicity	0..*		
Type	Reference to DemEnableCondition		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem131 :**
Name	DemHealingCycleRef {OperationCycle}
Description	optional reference to the healing cycle
Multiplicity	0..1

Type	Reference to DemOperationCycleTgt		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem131 :**		
Name	DemOperationCycleRef {OperationCycle}		
Description	Kind of operation cycle for the event (e.g. power cycle, driving cycle, ...)		
Multiplicity	1		
Type	Reference to DemOperationCycleTgt		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemIndicatorAttribute	0..255	This container contains the event specific configuration of Indicators.
DemOEMSpecific	0..1	This container contains the configuaration for OEM specific additional parameter.
DemPredebounceAlgorithmClass	0..255	Used algorithm class (Dem_PredebounceMonitorInternal, Dem_PredebounceFrequencyBased, Dem_PredebounceCounterBased, Dem_PredebounceTimeBased) depends on parameter EventClass.PredebounceAlgorithm It is possible to assign more then one algorithm to one event. This is useful if the behaviour of debouncing depends on other things, like status of DTC. Example: If the event doesn't occurs before, Debounce Algorithm A with paramater set A is used. If the event occurs again, then Debounce Algorithm B with paratmeter set B is used.

10.2.15 DemIndicatorAttribute

SWS Item	**Dem133 :**
Container Name	DemIndicatorAttribute{IndicatorAttribute}
Description	This container contains the event specific configuration of Indicators.
Configuration Parameters	

SWS Item	**Dem133 :**		
Name	DemIndicatorBehaviour {IndicatorBehaviour}		
Description	Behaviour of the linked indicator		
Multiplicity	1		
Type	EnumerationParamDef		
Range	DEM_INDICATOR_BLINKING	the indicator blinks when the event has status FAILED	
	DEM_INDICATOR_BLINK_CONT	the indicator is active and blinks when the event has status FAILED	
	DEM_INDICATOR_CONTINUOUS	The indicator is active when the even has status FAILED	
ConfigurationClass	***Pre-compile time***	X	All Variants

	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem133 :**		
Name	DemLinkedIndicator {LinkedIndicator}		
Description	indicator name of the used indicator		
Multiplicity	1		
Type	Reference to DemIndicator		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.16 DemOEMSpecific

SWS Item	**Dem134 :**
Container Name	DemOEMSpecific{OEMSpecific}
Description	This container contains the configuaration for OEM specific additional parameter.
Configuration Parameters	

No Included Containers

10.2.17 DemDTCClass

SWS Item	**Dem132 :**
Container Name	DemDTCClass{DTCClass}
Description	This container contains the configuration (parameters) for DTCClass.
Configuration Parameters	

SWS Item	**Dem132 :**		
Name	DemDTC {DTC}		
Description	Diagnostic Trouble Code		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 16777215		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem132 :**
Name	DemDTCFunctionalUnit {DEM_DTC_FUNCTIONALUNIT}
Description	DTCFuncitonalUnit is a 1-byte value which identifies the corresponding basic vehicle / system function which reports the DTC. This parameter is

	necessary for the report of severity informations.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem132 :**		
Name	DemDTCKind {DTCKind}		
Description	Defines whether the DTC is OBD-relevant or not		
Multiplicity	1		
Type	EnumerationParamDef		
Range	DEM_DTC_KIND_ALL_DTCS	Select all DTCs	
	DEM_DTC_KIND_EMISSION_REL_DTCS	Select OBD-relevant DTCs	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem132 :**		
Name	DemDTCSeverity {Severity}		
Description	DTC severity This parameter depends on automotive manufacturer and is optional.		
Multiplicity	0..1		
Type	EnumerationParamDef		
Range	DEM_DTC_SEV_CHECK_AT_NEXT_HALT	Check at next halt	
	DEM_DTC_SEV_IMMEDIATELY	Check immediately	
	DEM_DTC_SEV_MAINTENANCE_ONLY	Maintenance required	
	DEM_DTC_SEV_NO_SEVERITY	No severity information available	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemCallbackDTCStatusChanged	0..*	The presence of this container indicates that the DEM has access to an "DTCStatusChanged" callback, which the DEM will call to notify other components about the change in the status of a DTC. In case there is a DemCallbackDTCStatusChangedFnc, this parameter contains the name of the function that the DEM will call. In case there is no DemCallbackDTCStatusChangedFnc, the DEM will have an R-Port requiring the interface CallbackDTCStatusChanged whose name is the name of this container.
DemCallbackInitMForF	0..*	The presence of this container means that the DEM will call an InitMonitorForFunction callback provided by either a SW-C or another BSWM. In case the container has a DemCallbackInitMForFFnc, this parameter defines the name of the function that the DEM will use. If there is no such parameter, the name of the container is the name of an R-Port through which the DEM requires the interface

		CallbackInitMonitorForFunction.

10.2.18 DemFreezeFrameClass

SWS Item	**Dem136 :**
Container Name	DemFreezeFrameClass{FreezeFrameClass}
Description	This container contains the configuration (parameters) for FreezeFrameClass.
Configuration Parameters	

SWS Item	**Dem136 :**		
Name	DemFreezeFrameKind {FFKind}		
Description	Defines whether the freeze-frame is OBD relevant or not		
Multiplicity	1		
Type	EnumerationParamDef		
Range	DEM_FREEZE_FRAME_NON_OBD	No severity information available	
	DEM_FREEZE_FRAME_OBD	No severity information available	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem136 :**		
Name	DemFreezeFrameRecordNum {FFRecordNumber}		
Description	For OBD relevant data Multiple PIDs can be relevant per Freezeframe. This parameter is optional!		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem136 :**		
Name	DemFreezeFrameIdClassRef {DemFFIDClassRef}		
Description	For OBD relevant data Multiple PIDs can be relevant per Freezeframe. This parameter is optional!		
Multiplicity	1..255		
Type	Reference to DemPidOrDid		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.19 DemGroupOfDTC

SWS Item	**Dem137 :**
Container Name	DemGroupOfDTC{GroupOfDTC}
Description	This container contains the configuaration (parameters) for DTC Groups.
Configuration Parameters	

SWS Item	**Dem137 :**		
Name	DemGroupDTCs {DTCGroup}		
Description	DTC of the selected group of DTC (according to ISO14229-1[9] Annex D1)		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	1 .. 16777214		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Vehicle		

No Included Containers

10.2.20 DemPredebounceAlgorithmClass

SWS Item	**--**
Choice Container Name	DemPredebounceAlgorithmClass
Description	Used algorithm class (Dem_PredebounceMonitorInternal, Dem_PredebounceFrequencyBased, Dem_PredebounceCounterBased, Dem_PredebounceTimeBased) depends on parameter EventClass.PredebounceAlgorithm It is possible to assign more then one algorithm to one event. This is useful if the behaviour of debouncing depends on other things, like status of DTC. Example: If the event doesn't occurs before, Debounce Algorithm A with paramater set A is used. If the event occurs again, then Debounce Algorithm B with paratmeter set B is used.

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
DemPreDebounceCounterBased	0..1	This container contains the configuration (parameters) for DemPredebounceCounterBased
DemPreDebounceFrequencyBased	0..1	This container contains the configuration (parameters) for DemPredebounceFrequencyBased .
DemPreDebounceMonitorInternal	0..1	This container contains the configuration (parameters) for DemPredebounceMonitorInternal
DemPreDebounceTimeBase	0..1	This container contains the configuration (parameters) for DemPredebounceTimeBased.

10.2.21 DemPreDebounceCounterBased

SWS Item	**Dem144 :**
Container Name	DemPreDebounceCounterBased{PreDebounceCounterBased}

Description	This container contains the configuration (parameters) for DemPredebounceCounterBased
Configuration Parameters	

SWS Item	**Dem144 :**		
Name	DemCountInStepSize {CountOutStepSize}		
Description	Defines the Step size for incrementation of FDC (PREFAILED)		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 127		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem144 :**		
Name	DemCountOutStepSize {CountOutStepSize}		
Description	Defines the Step size for decrementation of FDC (PREPASSED)		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 127		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem144 :**		
Name	DemJumpDown {JumpDown}		
Description	Switch for the activation of Jump-Down – only in combination with Jump-UP activation. true: JumpDown activated false: JumpDown not activated		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DemJumpUp		

SWS Item	**Dem144 :**		
Name	DemJumpUp {JumpUp}		
Description	Switch for the activation of Jump-UP true: JumpUp activated false: JumpUp not activated		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem143 :**

Name	DemPreDebounceName {PreDebounceName}		
Description	Defines the selected debounce algorithm		
Multiplicity	1		
Type	EnumerationParamDef		
Range	DEM_PRE_DEBOUNCE_COUNTER_BASED	Dem_PredebounceCounterBased	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.22 DemPreDebounceFrequencyBased

SWS Item	**Dem145 :**
Container Name	DemPreDebounceFrequencyBased{PreDebounceFrequencyBased}
Description	This container contains the configuration (parameters) for DemPredebounceFrequencyBased .
Configuration Parameters	

SWS Item	**Dem145 :**		
Name	DemDurationOfTimeWindow {DurationOfTimeWindow}		
Description	Defines duration of the Time Window. Range defined in the DEM SWS as 0 .. 2^32, this parameter contains a value in milliseconds. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. DEM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DEM.		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem143 :**		
Name	DemPreDebounceName {PreDebounceName}		
Description	Defines the selected debounce algorithm		
Multiplicity	1		
Type	EnumerationParamDef		
Range	DEM_PRE_DEBOUNCE_FREQUENCY_BASED	Dem_PredebounceFrequencyBased	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem145 :**
Name	DemThresholdForEventTestedFailed {ThresholdForEventTestedFailed}

Description	Defines the threshold for FAILED-detection		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem145 :**		
Name	DemThresholdForEventTestedPassed {ThresholdForEventTestedPassed}		
Description	Defines the threshold for PASSED-detection		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.23 DemPreDebounceMonitorInternal

SWS Item	**Dem146 :**
Container Name	DemPreDebounceMonitorInternal{PreDebounceMonitorInternal}
Description	This container contains the configuration (parameters) for DemPredebounceMonitorInternal
Configuration Parameters	

SWS Item	**--**		
Name	DemPreDebounceName {PreDebounceName}		
Description	Defines the selected debounce algorithm		
Multiplicity	1		
Type	EnumerationParamDef		
Range	DEM_NO_PRE_DEBOUNCE	No Predebouncing, DemPreDebounceMonitorInternal is active and predebouncing is controlled by Monitor	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemCallbackGetFDCnt	1	This container defines the function that the OEM uses to obtain the value of the fault detection counter. In case the container has a parameter DemCallbackGetFDCntFnc, the function is provided through this name by another BSW. In case this parameter is not present, the name of this container

		is the name of a port requiring the interface CallbackGetFaultDetectionCounter

10.2.24 DemPreDebounceTimeBase

SWS Item	**Dem143 :**
Container Name	DemPreDebounceTimeBase{PreDebounceTimeBased}
Description	This container contains the configuration (parameters) for DemPredebounceTimeBased.
Configuration Parameters	

SWS Item	**Dem143 :**			
Name	DemPreDebounceName {PreDebounceName}			
Description	Defines the selected debounce algorithm			
Multiplicity	1			
Type	EnumerationParamDef			
Range	DEM_PRE_DEBOUNCE_TIME_BASED	Dem_PredebounceTimeBased		
ConfigurationClass	***Pre-compile time***		X	All Variants
	Link time		--	
	Post-build time		--	
Scope / Dependency	scope: ECU			

SWS Item	**Dem143 :**		
Name	DemTimeFailedThreshold {TimeFailedThreshold}		
Description	Defines the time out duration in ms for "Event Failed" qualification. Range defined in the DEM SWS as 0 .. 2^32, this parameter contains a value in milliseconds. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. DEM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DEM.		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem143 :**		
Name	DemTimePassedThreshold {TimePassedThreshold}		
Description	Defines the time out duration in ms for "Event Passed" qualification. Range defined in the DEM SWS as 0 .. 2^32, this parameter contains a value in milliseconds. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. DEM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DEM.		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	

	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.25 DemEnableCondition

SWS Item	**--**
Container Name	DemEnableCondition
Description	This container contains the configuration (parameters) for Enable Conditions.
Configuration Parameters	

SWS Item	**Dem131 :**		
Name	DemEnableConditionID {EnableConditionID}		
Description	Defines a condition ID. This parameter is optional and depends on manufacturer.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem131 :**		
Name	DemEnableConditionStatus {EnableConditionStatus}		
Description	Defines a status for enable or disable of storage of a event. The value is the initialization after power up. (TRUE=enabled FALSE=disabled)		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.26 DemCallbackInitMForF

SWS Item	**--**
Container Name	DemCallbackInitMForF{DEM_INIT_FUNC}
Description	The presence of this container means that the DEM will call an InitMonitorForFunction callback provided by either a SW-C or another BSWM. In case the container has a DemCallbackInitMForFFnc, this parameter defines the name of the function that the DEM will use. If there is no such parameter, the name of the container is the name of an R-Port

	through which the DEM requires the interface CallbackInitMonitorForFunction.
Configuration Parameters	

SWS Item	**Dem130 :**		
Name	DemCallbackInitMForFFnc		
Description	Name of a function of prototype "InitMonitorForFunction"		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.27 DemCallbackInitMForE

SWS Item	**Dem130 :**
Container Name	DemCallbackInitMForE
Description	Monitor function which has to be initialized for the event. This is either provided by a BSWM (when the parameter DemCallbackInitMForEFnc is present) or through an R-Port requiring a CallbackInitMonitorForEvent interface whose name is the name of this container.
Configuration Parameters	

SWS Item	**--**		
Name	DemCallbackInitMForEFnc		
Description	Name of a function with prototype "InitMonitorForEvent"		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.28 DemNvramBlockId

SWS Item	**Dem147 :**
Container Name	DemNvramBlockId{NVRAMBlockIDList}
Description	This container contains the configuaration (parameters) for Dem_OperationCycleList. Note hat this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the

	VALUE_NAME.
Configuration Parameters	

SWS Item	**FIM083 :**		
Name	DemNvramBlockIdRef {FIM_INPUT_SUMMARIZED_EVENT}		
Description			
Multiplicity	1		
Type	Reference to NvmBlockDescriptor		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.29 DemCallbackGetFDCnt

SWS Item	**--**
Container Name	DemCallbackGetFDCnt
Description	This container defines the function that the OEM uses to obtain the value of the fault detection counter. In case the container has a parameter DemCallbackGetFDCntFnc, the function is provided through this name by another BSW. In case this parameter is not present, the name of this container is the name of a port requiring the interface CallbackGetFaultDetectionCounter
Configuration Parameters	

SWS Item	**Dem146 :**		
Name	DemCallbackGetFDCntFnc {Prefix_DemGetFaultDetectionCounter}		
Description	The name of a function of prototype "GetFaultDetectionCounter"		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.30 DemExtendedDataRecordClass

SWS Item	**Dem135 :**
Container Name	DemExtendedDataRecordClass{ExtendedDataClass}
Description	This container contains the configuration (parameters) for ExtendedDataClassRecords
Configuration Parameters	

SWS Item	**Dem135 :**

Name	DemExtendedDataRecordDataSize {DataSize}		
Description	Defines the size of the extended Data Record in Bytes.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 256		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem135 :**		
Name	DemExtendedDataRecordNumber {ExtendedDataRecordNumber}		
Description	This configuration parameter specifies an unique identifier for an ExtendedDataRecord. One or more ExtendedDataRecords can be assigned to one DTC. max = 253 because 0xFF and 0xFE are resereved by ISO		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 253		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemCallbackGetExtDataRecord	1	The presence of this container indicates that the DEM has access to an "GetExtendedDataRecord" callback, which the DEM will call to obtain an extended data record. In case there is a DemCallbackGetExtDataRecordFnc, this parameter contains the name of the function that the DEM will call. In case there is no DemCallbackGetExtDataRecordFnc, the DEM will have an R-Port requiring tthe interface CallbackGetExtendedDataRecord whose name is the name of this container

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

The standard common published information like

vendorId (<Module>_VENDOR_ID),
moduleId (<Module>_MODULE_ID),
arMajorVersion (<Module>_AR_MAJOR_VERSION),
arMinorVersion (<Module>_ AR_MINOR_VERSION),

arPatchVersion (<Module>_ AR_PATCH_VERSION),
swMajorVersion (<Module>_SW_MAJOR_VERSION),
swMinorVersion (<Module>_ SW_MINOR_VERSION),
swPatchVersion (<Module>_ SW_PATCH_VERSION),
vendorApiInfix (<Module>_VENDOR_API_INFIX)

is provided in the BSW Module Description Template (see [6] Figure 4.1 and Figure 7.1).

Additional published parameters are listed below if applicable for this module.

11 Service Diagnostic Event Manager (DEM)

11.1 Scope of this Chapter

This chapter is an addition to the specification of the DEM. That specification currently defines the behavior and the C-interfaces of the corresponding basic software module. Based on this, this chapter formally specifies the corresponding AUTOSAR Service, which will be visible on the VFB.

11.2 Overview

11.2.1 Architecture

In the AUTOSAR ECU architecture the Diagnostic Event Manager implements an AUTOSAR Service. The DEM communicates with other BSW modules and via the RTE with SW-C.

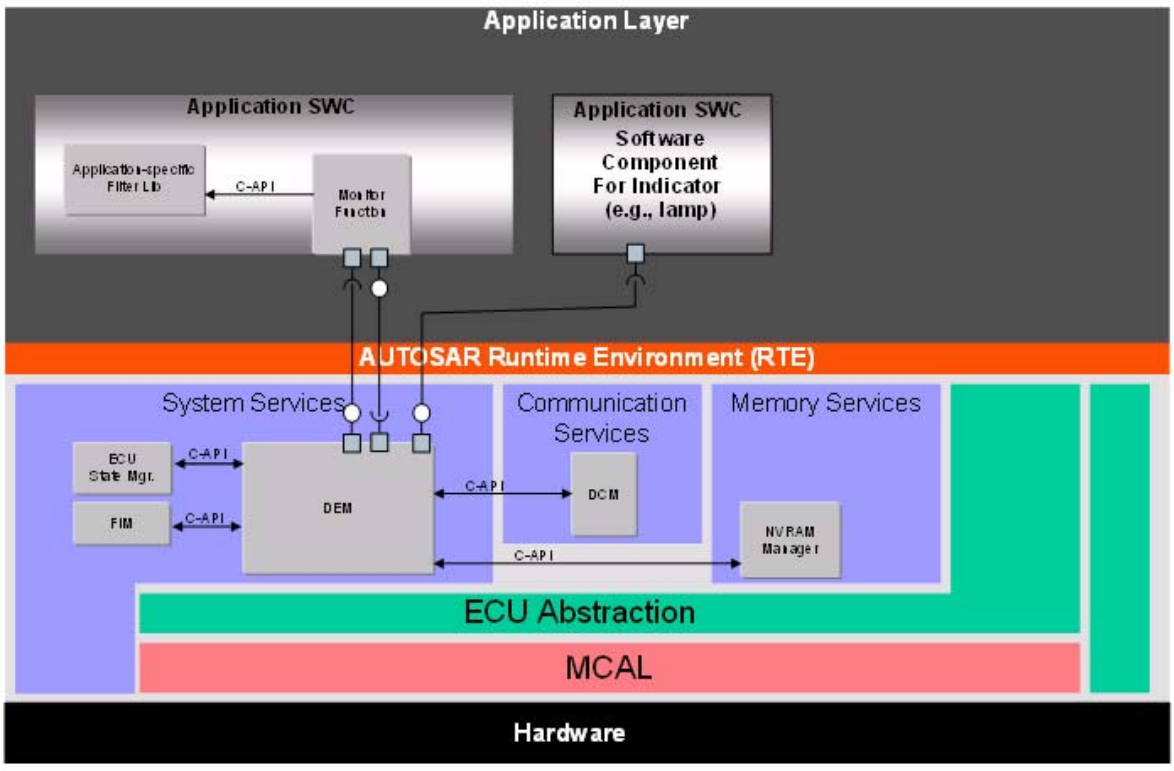


Figure 34: DEM in the ECU software architecture.

From the viewpoint of the basic software C-module “DEM” there are three kinds of dependencies between the Service and the AUTOSAR Software Components above the RTE :

- the application accesses the API (implemented as C-functions) of the DEM
- the application is optionally notified upon the outcome of requested asynchronous activity (via callback-C-functions by the DEM),
- an initialization function of the SW-C is invoked by the DEM.

These dependencies must be described in terms of the AUTOSAR meta-model, which will contribute to the SW-C Description of the application component as well as to the SW-C Description of the DEM Service.

11.2.2 Requirements

The requirements for the functionality of the DEM service are specified in chapter 7 Functional specification of this document.

11.2.3 Use Cases

On each ECU we have typically one instance of the DEM Service and several Atomic Software Component instances, named “clients” further on in this document, which are using this Service. In addition, there are parts of the basic software which communicate with the DEM.

The Monitor part of the SW-C is responsible for detecting a fault. It is expected to run periodically. To avoid the generation of a DTC for transient or intermittent faults, the faults can be filtered.
The DEM maintains counters per event.
The DEM supports a healing mechanism. For each event a number of healing cycles can be defined.

11.2.3.1 Initialization of event-specific part of the monitor

The initialization of the event-specific part of the monitor can be triggered by the DEM.

11.2.3.2 Initialization of function-specific part of the monitor

The initialization of the function-specific part of the monitor can be triggered by the DEM.

11.2.3.3 Notification of the DEM about status change of a diagnostic event

A SW-C monitor sets the status of the diagnostic event.

11.2.3.4 Notification of the Monitor about status change of a diagnostic event or diagnostic trouble code

The DEM module informs the monitor about the status change of the event or DTC.

11.2.3.5 Notification of an indicator SW-C about the status change of an event

The DEM can notify an indicator SW-C about the status change of a diagnostic event.

11.3 Data types that are relevant to RTE-Communication

The following types are contained in the Rte_Type.h header file, which is generated by the RTE generator.

```
IntegerType Dem_EventStatusType {
LOWER-LIMIT=0;
UPPER-LIMIT=255;
0 -> DEM_EVENT_STATUS_PASSED
1 -> DEM_EVENT_STATUS_FAILED
2 -> DEM_EVENT_STATUS_PREPASSED
3 -> DEM_EVENT_STATUS_PREFAILED
// 32..255 -> custom status values
}

IntegerType Dem_EventStatusExtendedType {
LOWER-LIMIT = 0;
UPPER-LIMIT = 255;
}

IntegerType Dem_DTCKindType {
LOWER-LIMIT=1;
UPPER-LIMIT=2;
1 -> DEM_DTC_KIND_ALL_DTCS
2 -> DEM_DTC_KIND_EMISSION_REL_DTCS
}

IntegerType Dem_DTCType {
LOWER-LIMIT = 0;
UPPER-LIMIT = 16777215; // 0xFFFFFF
}

IntegerType Dem_InitMonitorKindType {
LOWER-LIMIT=1;
UPPER-LIMIT=2;
1 -> DEM_INIT_MONITOR_CLEAR
2 -> DEM_INIT_MONITOR_RESTART
}
```

```
IntegerType Dem_OperationCycleStateType {
LOWER-LIMIT=1;
UPPER-LIMIT=2;
1 -> DEM_CYCLE_STATE_START
2 -> DEM_CYCLE_STATE_END
}

IntegerType Dem_FaultDetectionCounterType {
LOWER-LIMIT = -128;
UPPER-LIMIT =  127;
}

IntegerType Dem_IndicatorStatusType {
LOWER-LIMIT=0;
UPPER-LIMIT=3;
0 -> DEM_INDICATOR_OFF
1 -> DEM_INDICATOR_CONTINUOUS
2 -> DEM_INDICATOR_BLINKING
3 -> DEM_INDICATOR_BLINK_CONT
}
```

11.4 Specification of the Ports and Port Interfaces

This chapter specifies the ports and port interfaces which are needed in order to operate the DEM functionality over the VFB. Note that there are ports on both sides of the RTE: The SW-C description of the DEM Service will define the ports below the RTE. Each SW-C component, which uses the Service, must contain “service ports” in its own SW-C description, which will be typed by the same interfaces and must be connected to the ports of the DEM, so that the RTE can be generated.

11.4.1 Description of the Interfaces

The following pseudo code defines the interfaces between the SW-Cs and the DEM. The corresponding figures show the related API functions.

The *DiagnosticMonitor* interface provides the capability to obtain and modify the event information. One port of this interface type is provided per EventId by the *DEM Service Component*. It has EventId as a port-defined argument.

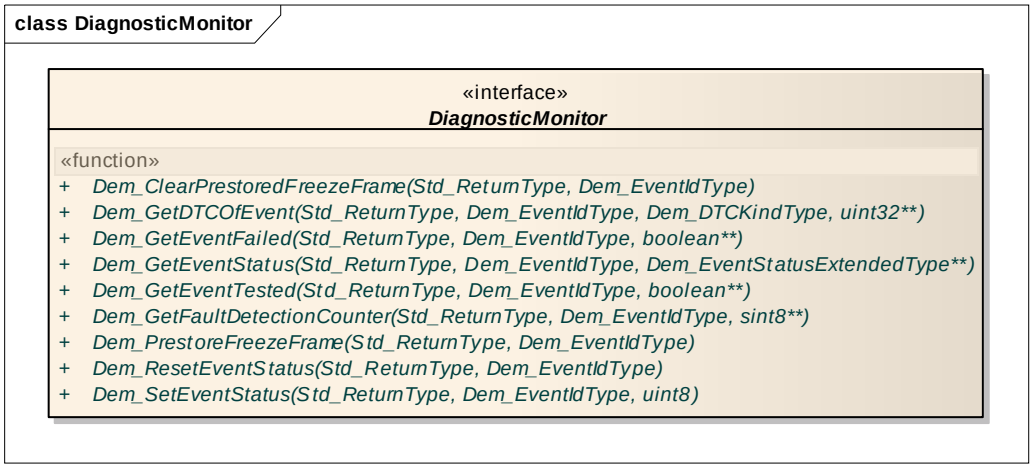


Figure 35 Interface: DiagnosticMonitor

```
ClientServerInterface DiagnosticMonitor {
    PossibleErrors {
        E_NOT_OK = 1,
        E_NO_DTC_AVAILABLE = 2
    }
    SetEventStatus(IN Dem_EventStatusType EventStatus,
        ERR{E_NOT_OK});
    ResetEventStatus(ERR{E_NOT_OK});
    GetEventStatus(OUT Dem_EventStatusExtendedType* EventStatusExtended,
        ERR{E_NOT_OK});
    GetEventFailed (OUT Boolean EventFailed,
        ERR{E_NOT_OK});
    GetEventTested (OUT Boolean EventTested,
        ERR{E_NOT_OK});
    GetDTCOfEvent (IN Dem_DTCKindType DTCKind,
        OUT UInt32 DTCofEvent,
        ERR{E_NOT_OK,
        E_NO_DTC_AVAILABLE});
    PrestoreFreezeFrame(
        ERR{E_NOT_OK}); // OPTIONAL for non-OBD DEM
    ClearPrestoredFreezeFrame(
        ERR{E_NOT_OK}); // OPTIONAL for non-OBD DEM
    GetFaultDetectionCounter(
        OUT sint8* EventIdFaultDetectionCounter,
        ERR{E_NOT_OK});
    SetEventDisable(
        ERR{E_NOT_OK}); // OPTIONAL, only if DEM uses OBD
}
```

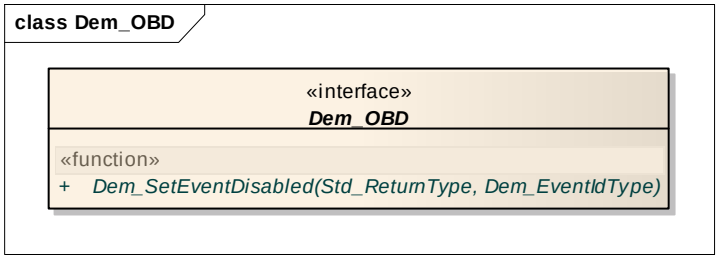


Figure 36 Interface: Dem_OBD

The API of the interface Dem_OBD is used by the interface DiagnosticMonitor if the configuration switch DemOBDSupport defines an OBD relevant system. Therefore no addional port per event is necessary.

The *OperationCycle* interface provides the capability to set a operation cycle state. One port of this interface type is provided per OperationCycleId by the *DEM Service Component*. It has OperationCycleId as a port-defined argument.

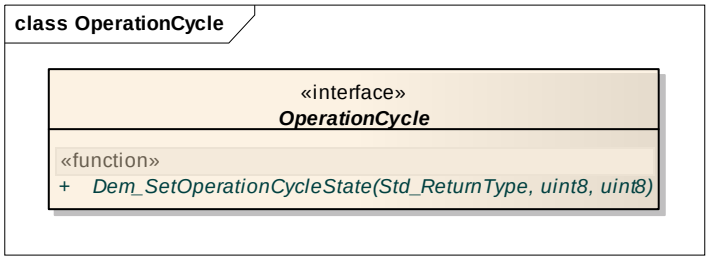


Figure 37 Interface: OperationCycle

```
ClientServerInterface OperationCycle {
    PossibleErrors {
        E_NOT_OK = 1
    }
    SetOperationCycleState(
        IN uint8 CycleState,
        ERR{E_NOT_OK});
}
```

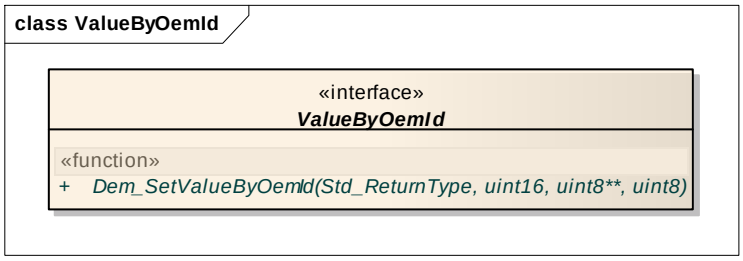


Figure 38 Interface: ValueByOemId

```
// optional interface
ClientServerInterface ValueByOemId {
    PossibleErrors {
        E_NOT_OK = 1
    }
    SetValueByOemId(IN UInt16 OemID,
        OUT UInt8 DataValue,
        IN UInt8 BufferLength,
```

```
        ERR{E_NOT_OK});
}
```

The *EnableCondition* interface provides the capability to define an enable condition. One port of this interface type is provided per EnableConditionId by the *DEM Service Component*. It has EnableConditionId as a port-defined argument.

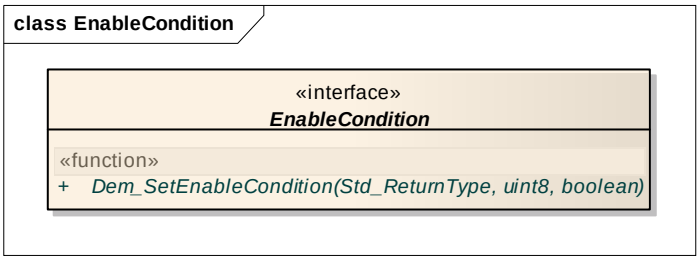


Figure 39 Interface: EnableCondition

```
// optional interface
ClientServerInterface EnableCondition {
  PossibleErrors {
    E_NOT_OK = 1
  }
  SetEnableCondition(
    IN Boolean ConditionFulfilled,
    ERR{E_NOT_OK});
}
```

The *IndicatorStatus* interface provides the capability to set the status of an indicator. One port of this interface type is provided per IndicatorId by the *DEM Service Component*. It has IndicatorId as a port-defined argument.

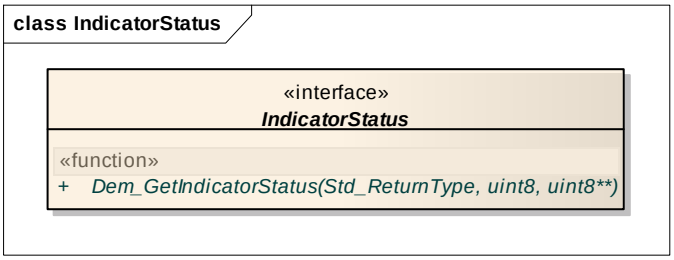


Figure 40 Interface: IndicatorStatus

```
// optional interface
ClientServerInterface IndicatorStatus {
  PossibleErrors {
    E_NOT_OK = 1
  }
```

```
  GetIndicatorStatus (
    OUT IndicatorStatusType IndicatorStatus,
    ERR{E_NOT_OK});
}
```

11.4.2 OBD-specific Interfaces

The DEM SWS defines an API to obtain the PID content from the SW-C. This API is also used by the DCM. Thus, DEM and DCM provide a common interface PidSevices_<PID> containing the function GetPIDValue.

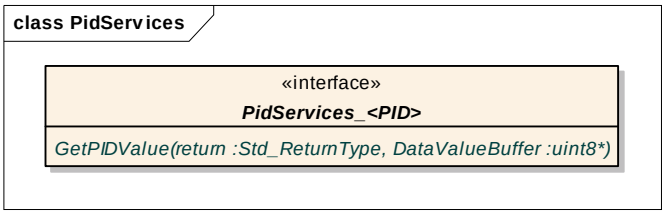


Figure 41 OBD specific interface: PidServices_<PID>

```
ClientServerInterface PidServices_<PID>
{
PossibleErrors {
  E_OK = 0,
  E_NOT_OK = 1,
  E_PENDING = 10
};
GetPIDValue(
  INOUT UInt8 DataValueBuffer[size of largest PID],
  ERR{E_NOT_OK, E_PENDING})
}
```

Since IUMPR can be connected either via “API-use” or as “observer”, a port interface is introduced. This is to report that a fault could have been found by “API-use”.

The *IUMPRNumerator* interface provides the capability to defined the number of times a fault could have been found. One port of this interface type is provided per RatioID by the *DEM Service Component*. It has RatioID as a port-defined argument.

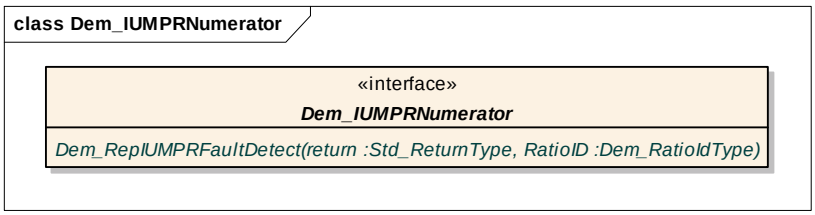


Figure 42 OBD specific interface: Dem_IUMPRNumerator

```
ClientServerInterface IUMPRNumerator {
  PossibleErrors {
    E_NOT_OK = 1
  }
  RepIUMPRFaultDetect (
    ERR{E_NOT_OK});
}
```

The following interface is needed if additional conditions apply when increasing the denominator. This interface allows a SW-C to lock or release denominator of a specific RatioId.

The *IUMPRDenominator* interface provides the capability to define the number of times the vehicle operation has been fulfilled. One port of this interface type is provided per RatioID by the *DEM Service Component*. It has RatioID as a port-defined argument.

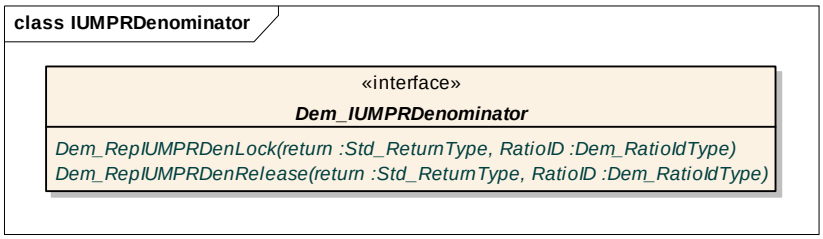


Figure 43 OBD specific interface: Dem_IUMPRDenominator

```
ClientServerInterface IUMPRDenominator {
  PossibleErrors {
    E_NOT_OK = 1
  }
  RepIUMPRDenLock  (ERR{E_NOT_OK});
  RepIUMPRDenRelease (ERR{E_NOT_OK});
}
```

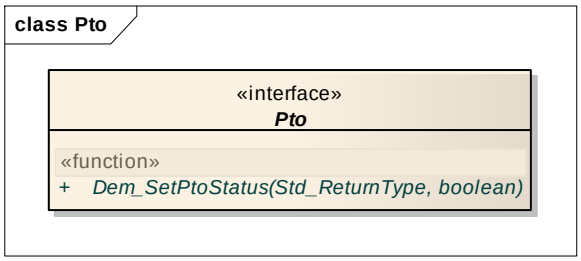


Figure 44 OBD specific interface: Pto

```
ClientServerInterface Pto {
  PossibleErrors {
    E_NOT_OK = 1
  }
  SetPtoStatus(Boolean PtoStatus, ERR{E_NOT_OK});
}
```

11.4.3 Callback Interfaces

The DEM SWS defines a number of callback functions from the DEM to the monitor.

The callbacks do not use the mechanism of the port-defined arguments. Instead, the DEM configuration mechanism must ensure that the callback is delivered to the configured port and invokes the correct operation at this port using an RTE (direct or indirect) API call. The EventId must **not** be passed as the first argument of the operation, because the monitor does not cope with EventIds explicitly.

The following interfaces *CallbackInitMonitorForEvent and CallbackInitMonitorForFunction* allow an event-specific and function-specific initialization of the Monitor part of the SW-C. For each SW-C there is one initialization port per EventID. The parameter *InitMonitorKind* has the value Clear or Restart (see 8.4.3.1.1 of DEM SWS) and tells the initialization function the reason for the initialization call.

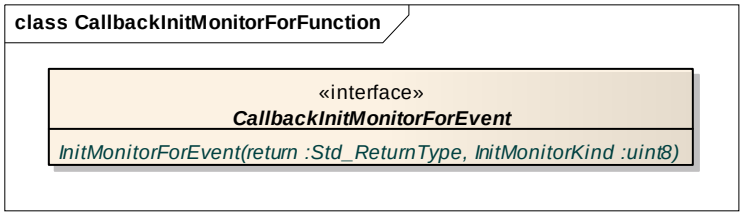


Figure 45 Callback: InitMonitorForEvent

```
ClientServerInterface CallbackInitMonitorForEvent {
    PossibleErrors {
        E_NOT_OK = 1
    }
    // Init functions are used to notify the monitor from the DEM (from DEM
SWS 8.4.3.1)
    InitMonitorForEvent{EventName}(IN InitMonitorKind,
        ERR{E_NOT_OK});
}
```

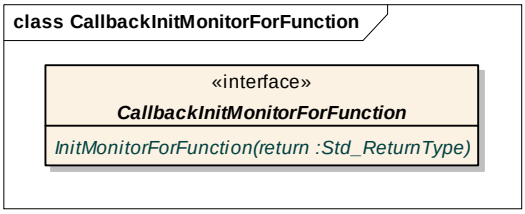


Figure 46 Callback: InitMonitorForFunction

```
ClientServerInterface CallbackInitMonitorForFunction {
    PossibleErrors {
        E_NOT_OK = 1
    }
    // Init functions are used to notify the monitor from the DEM (from DEM
SWS 8.4.3.2)
    InitMonitorForFunction{Function}(ERR{E_NOT_OK});
}
```

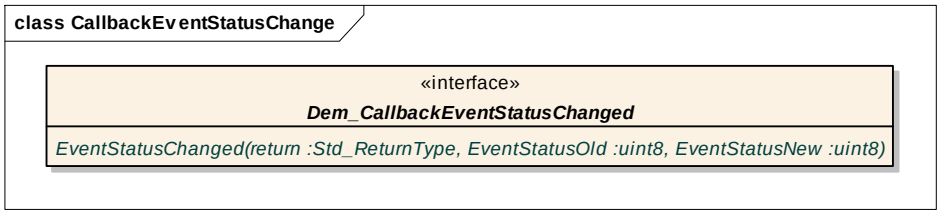


Figure 47 Callback: EventStatusChanged

```
ClientServerInterface CallbackEventStatusChange {
    PossibleErrors {
        E_NOT_OK = 1
    }
    // used to notify the monitor from the DEM (from DEM SWS 8.4.3.1.3)
    EventStatusChanged(IN Dem_EventStatusExtendedType EventStatusOld,
        IN Dem_EventStatusExtendedType EventStatusNew,
        ERR{E_NOT_OK});
    // this operation was called TriggerOnEventStatus
}
```

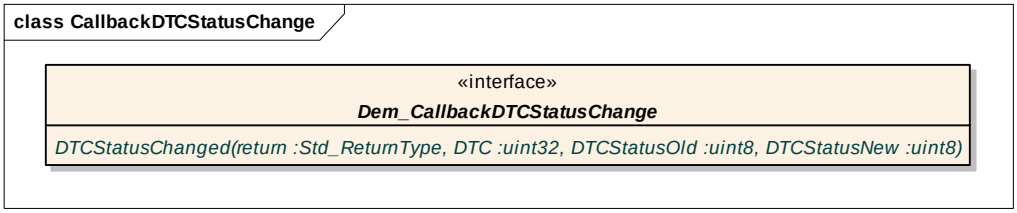


Figure 48 Callback: DTCStatusChanged

```
ClientServerInterface CallbackDTCStatusChange {
    PossibleErrors {
        E_NOT_OK = 1
    }
    // used to notify the monitor from the DEM (from DEM SWS 8.4.3.1.4)
    DTCStatusChanged(IN Dem_DTCType DTC,
        IN Dem_DTCStatusMaskType DTCStatusOld,
        IN Dem_DTCStatusMaskType DTCStatusNew,
        ERR{E_NOT_OK});
    // this operation was called TriggerOnDTCStatus
}
```

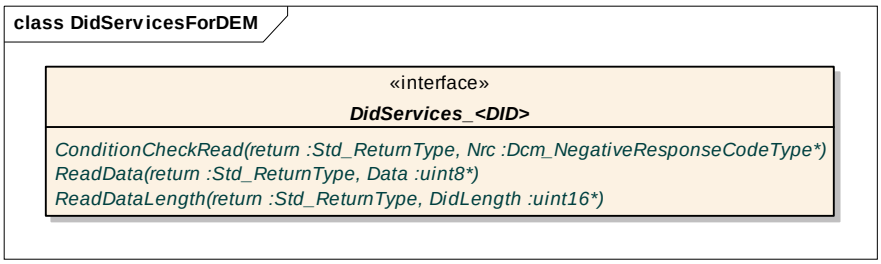


Figure 49 Callback: DidServices_<DID>

```
ClientServerInterface DidServices_<DID> {
  PossibleErrors {
    E_NOT_OK = 1,
    E_PENDING = 10
  }
  // used to get FreezeFrame data from SW-C
  ConditionCheckRead(INOUT Dcm_NegativeResponseCodeType Nrc,
    ERR{E_NOT_OK, E_PENDING})
  ReadData(OUT UInt8 Data[size of this DID],
    ERR{E_PENDING}))
  ReadDataLength(OUT UInt16 DidLength,
    ERR{E_PENDING}));
  }
}
```

The type Dcm_NegativeResponseCodeType is referenced from the DCM service component.

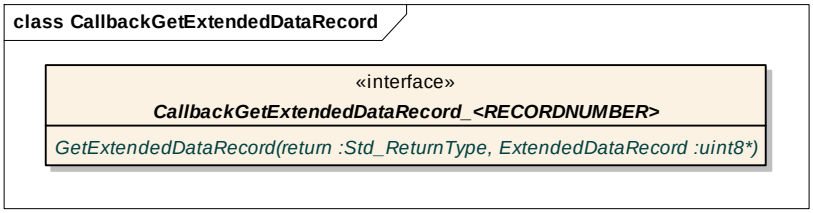


Figure 50 Callback: GetExtendedDataRecord

```
ClientServerInterface CallbackGetExtendedDataRecord_<RecordNumber> {
    PossibleErrors {
        E_NOT_OK = 1
    }
    // used to get extended data from SW-C
    GetExtendedDataRecord(
INOUT UInt8 ExtendedDataRecord[size of this Record],
ERR{E_NOT_OK});
}
```

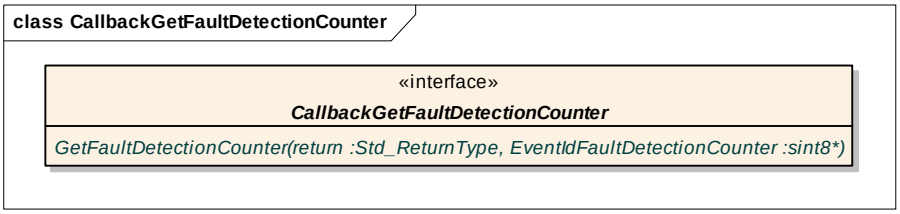


Figure 51 Callback: GetFaultDetectionCounter

```
ClientServerInterface CallbackGetFaultDetectCounter {
    PossibleErrors {
        E_NOT_OK = 1
    }
    // used to get fault detection counter from SW-C
    GetFaultDetectionCounter (OUT Dem_FaultDetectionCounterType
        EventIdFaultDetectionCounter, ERR{E_NOT_OK});
}
```

11.4.4 Unused APIs

The DEM SWS defines an API to obtain the version of the DEM. This API is not part of the service interface, because the version information can be obtained using other mechanisms.

11.4.5 Definition of the Service DEM

The following types are not shown up in the service ports of the client components, because they are implemented as port defined argument values, which are part of the internal behaviour of the DEM Service. So, the ECU dependency of Dem_EventIdType is not visible for the clients.

```
IntegerType Dem_EventIdType  {
            LOWER-LIMIT = 0;
            UPPER-LIMIT = <N>;
      };
```

```
IntegerType Dem_OperationCycleIdType {
            LOWER-LIMIT = 0;
            UPPER-LIMIT = <N - 1>;
      };

IntegerType Dem_IndicatorIdType {
            LOWER-LIMIT = 0;
            UPPER-LIMIT = <N - 1>;
      };

IntegerType Dem_RatioIdType {
            LOWER-LIMIT = 0;
            UPPER-LIMIT = <N - 1>;
      };


ServiceComponent Dem {

ProvidePort DiagnosticMonitor Event_<EventName>;
ProvidePort DiagnosticMonitor Event_<EventName>;
…

RequirePort CallbackInitMonitorForEvent CBInitEvt_<EventName>;
RequirePort CallbackInitMonitorForEvent CBInitEvt_<EventName>;
…

RequirePort CallbackInitMonitorForFunction CBInitFct_1;
RequirePort CallbackInitMonitorForFunction CBInitFct_2;
…
RequirePort CallbackInitMonitorForFunction CBInitFct_n;

RequirePort CallbackEventStatusChange CBStatusEvt_<EventName>_<SWC>;
RequirePort CallbackEventStatusChange CBStatusEvt_<EventName>_<SWC>;
…

RequirePort CallbackDTCStatusChange CBStatusDTC_<EventName>_<SWC>;
RequirePort CallbackDTCStatusChange CBStatusDTC_<EventName>_<SWC>;
…

ProvidePort OperationCycle OpCycle_<CycleName>;
ProvidePort OperationCycle OpCycle_<CycleName>;
…

ProvidePort ValueByOemId ValByOemId;

ProvidePort EnableCondition EnableCond_<ConditionName>;
ProvidePort EnableCondition EnableCond_<ConditionName>;
…

ProvidePort IndicatorStatus IndStatus_<IndicatorName>;
ProvidePort IndicatorStatus IndStatus_<IndicatorName>;
…

ProvidePort IUMPRNumerator IUMPRNumerator_<RatioName>;
ProvidePort IUMPRNumerator IUMPRNumerator_<RatioName>;
…

ProvidePort IUMPRDenominator IUMPRDenominator_<RatioName>;
ProvidePort IUMPRDenominator IUMPRDenominator_<RatioName>;
…
```

```
ProvidePort Pto PtoStatus;


// the <DID> has to be in the format '0xNNNN' (e.g. '0x0200')
RequirePort DidServices_<DID> CBValByDID_<DID>;
RequirePort DidServices_<DID> CBValByDID_<DID>;
…

// the <PID> has to be in the format '0xNNNN' (e.g. '0x0200')
RequirePort PidServices_<PID> CBValByPID_<PID>;
RequirePort PidServices_<PID> CBValByPID_<PID>;
…

// the <RecordNumber > has to be in the format '0xNN' (e.g. '0x05')
RequirePort CallbackGetExtendedDataRecord_<RecordNumber>
CBExtDataRec_<RecordNumber>;
RequirePort CallbackGetExtendedDataRecord_<RecordNumber>
CBExtDataRec_<RecordNumber>;
…

RequirePort CallbackGetFaultDetectCounter CBFaultDetectCtr_<EventName>;
RequirePort CallbackGetFaultDetectCounter CBFaultDetectCtr_<EventName>;
…

};


/* This is the inside description of the DEM. This detailed description is
only needed for the configuration of the local RTE */
InternalBehavior DEM {

      // Runnable entities of the DEM
RunnableEntity SetEventStatus
            symbol “Dem_SetEventStatus”
            canbeInvokedConcurrently = TRUE
RunnableEntity ResetEventStatus
            symbol “Dem_ResetEventStatus”
            canbeInvokedConcurrently = TRUE
RunnableEntity GetEventStatus
            symbol “Dem_GetEventStatus”
            canbeInvokedConcurrently = TRUE
RunnableEntity GetEventFailed
            symbol “Dem_GetEventFailed”
            canbeInvokedConcurrently = TRUE
RunnableEntity GetEventTested
            symbol “Dem_GetEventTested”
            canbeInvokedConcurrently = TRUE
RunnableEntity GetDTCOfEvent
            symbol “Dem_GetDTCOfEvent”
            canbeInvokedConcurrently = TRUE
RunnableEntity PrestoreFreezeFrame
            symbol “Dem_PrestoreFreezeFrame”
            canbeInvokedConcurrently = TRUE
RunnableEntity ClearPrestoredFreezeFrame
            symbol “Dem_ClearPrestoredFreezeFrame”
            canbeInvokedConcurrently = TRUE
RunnableEntity GetFaultDetectionCounter
            symbol “Dem_GetFaultDetectionCounter”
            canbeInvokedConcurrently = TRUE
RunnableEntity SetEventDisabled
            symbol “Dem_SetEventDisabled”
```

```
            canbeInvokedConcurrently = TRUE
RunnableEntity SetOperationCycleState
            symbol “Dem_SetOperationCycleState”
            canbeInvokedConcurrently = TRUE
RunnableEntity SetValueByOemId
            symbol “Dem_SetValueByOemId”
            canbeInvokedConcurrently = FALSE
RunnableEntity SetEnableCondition
            symbol “Dem_SetEnableCondition”
            canbeInvokedConcurrently = TRUE
RunnableEntity GetIndicatorStatus
            symbol “Dem_GetIndicatorStatus”
            canbeInvokedConcurrently = TRUE

RunnableEntity RepIUMPRFaultDetect
            symbol “Dem_RepIUMPRFaultDetect”
            canbeInvokedConcurrently = TRUE
RunnableEntity RepIUMPRDenLock
            symbol “Dem_RepIUMPRDenLock”
            canbeInvokedConcurrently = TRUE
RunnableEntity RepIUMPRDenRelease
            symbol “Dem_RepIUMPRDenRelease”
            canbeInvokedConcurrently = TRUE
RunnableEntity SetPtoStatus
            symbol “Dem_SetPtoStatus”
            canbeInvokedConcurrently = TRUE


// for each port providing the DiagnosticMonitor Interface:
PortArgument {port=Event_<EventName>, value.type=Dem_EventIdType,
              value.value=<n>, where <n> = 1..<N>}

// for each port providing the OperationCycle Interface:
PortArgument {port=OpCycle_<CycleName>,
              value.type=Dem_OperationCycleIdType,
              value.value=<n>, where <n> = 0..<N - 1>}

// for each port providing the EnableCondition Interface:
PortArgument {port=EnableCond_<ConditionName>, value.type=UInt8,
              value.value=<n>, where <n> = 0..<N - 1>}

// for each port providing the IndicatorStatus Interface:
PortArgument {port=IndStatus_<IndicatorName>,
              value.type=Dem_IndicatorIdType,
              value.value=<n>, where <n> = 0..<N - 1>}
};

// for each port providing the IUMPRNumerator Interface:
PortArgument {port=IUMPRNumerator_<RatioName>,
              value.type=Dem_RatioIdType,
              value.value=<n>, where <n> = 0..<N - 1>}

// for each port providing the IUMPRDenominator Interface:
PortArgument {port= IUMPRDenominator_<RatioName>,
              value.type=Dem_RatioIdType,
              value.value=<n>, where <n> = 0..<N - 1>}
```

12 Changes to Release 1

12.1 Deleted SWS Items

SWS Item	*Rationale*
Dem101	Obsolete requirement
Dem008	Obsolete requirement
Dem012	Obsolete requirement
Dem030	Obsolete requirement
Dem272	Obsolete requirement, see Dem009 (1)
Dem042	Obsolete requirement (1)
Dem038	Obsolete requirement (ViewId is not needed) (1)
Dem058	Obsolete requirement, see Dem038 (1)
Dem160	Obsolete requirement (1)
Dem020	Obsolete requirement, covered by Dem293 (1)
Dem090	Obsolete requirement (1)
Dem031	Obsolete requirement (1)
Dem292	Obsolete requirement (1)
Dem223	Obsolete requirement (ViewId was removed) (1)
Dem211	Obsolete requirement (ViewId was removed) (1)
Dem222	Obsolete requirement (ViewId was removed) (1)
Dem170	Obsolete requirement (1)

(1) Changes to Release 3.0

12.2 Replaced SWS Items

SWS Item	*replaced by SWS Item*	*Rationale*
Dem158	Dem330, Dem331, Dem332, Dem333	Made requirement atomic (1)
Dem159	Dem334	Made requirement atomic (1)
Dem350	Dem351, Dem352, Dem353, Dem354, Dem355, Dem356	Made requirement atomic (1)
Dem295	Dem361	New requirement atomic (RatioId) (1)

(1) Changes to Release 3.0

12.3 Changed SWS Items

SWS Item	*Rationale*
Dem006	Refinement of requirement
Dem003	Refinement of requirement
Dem010	Requirement not mandatory now
Dem034	Refinement of requirement
Dem035	Refinement of requirement
Dem019	Obsolete parts of requirements are deleted
Dem036	Clarification of requirement
Dem029	Extension of requirement due to FIM
Dem058	Clarification of requirement
Dem079	Clarification of requirement
Dem081	Clarification of requirement

Dem112	Extension of requirement
Dem009	Rewording of requirement (1)
Dem154	Rewording of requirement, change EventId to EventName (1)
Dem153	Rewording of requirement (1)
Dem013	Rewording of requirement (1)
Dem019	Rewording of requirement (1)
Dem014	Rewording of requirement (1)
Dem036	Rewording of requirement, update of status bits (1)
Dem015	Rewording of requirement (1)
Dem040	Rewording of requirement (1)
Dem041	Rewording of requirement (1)
Dem029	Rewording of requirement, add reference (1)
Dem291	Rewording of requirement (1)
Dem283	Rewording of requirement (clarify the collection of FreezeFrame data) (1)

(1) Changes to Release 3.0

12.4 Added SWS Items

SWS Item	***Rationale***
Dem126	Extension and refinement of specification
Dem108	Extension and refinement of specification, file structure added
Dem127	New requirement due to BSW error handling
Dem107	New requirement due to BSW error handling
Dem123	New requirement due to startup behavior
Dem124	New requirement due to startup behavior
Dem115	New requirement due to error classification
Dem116	New requirement due to error classification
Dem113	New requirement due to error detection
Dem114	New requirement due to error detection
Dem117	New requirement due to error notification
Dem118	New requirement for data types
Dem111	New requirement for version information
Dem125	New requirement for cyclic function
Dem120	New requirement for configuration container
Dem119	New requirement for variants
Dem329	New requirement for event memory (1)
Dem335	New requirement to proprive an API (1)
Dem336	New requirement to support 15031-6 DTC format (1)
Dem337	New requirement for storage of DTCs/FreezeFrame (1)
Dem338	New requirement to support cycle management (1)
Dem339	New requirement to verify blocks (1)
Dem340	New requirement for startup (1)
Dem341	New requirement for shutdown (1)
Dem342	New requirement for debouncing (1)
Dem343	New requirement to reset debounce counters (1)
Dem344	New requirement to reset debounce counters (1)
Dem345	New requirement to reset debounce counter via SW-C, add API (1)
Dem346	New requirement for calculation of PID$21 and PID$31 (1)
Dem347	New requirement to support PTO (1)
Dem348	New requirement for disabling events (1)
Dem349	New requirement to support DemClassExtended (1)
Dem350	New requirement for internal calculations (1)
Dem357	New requirement to support IUMPR (1)
Dem358	New requirement to support IUMPR (1)
Dem359	New requirement to increment the numerator (1)
Dem360	New requirement to provide the API Dem_RepIUMPRFaultDetect (1)

Dem362	New requirement to provide API for lock/unlock IUMPR (1)
Dem363	New requirement to provide GetExtendedDataRecord (1)
Dem364	New requirement (1)
Dem365	New requirement, add switch for PTO support (1)
Dem366	New requirement, add configuration parameter for OBD support (1)
Dem368	New requirement to describe the DEM shutdown behavior (1)
Dem370	New requirement to define response of development errors (1)
Dem376	Requirement on DEM module, refer to API InitMonitorForEvent (1)
Dem377	New requirement for PTO support (interface) (1)
Dem378	New requirement for PTO support (configuration parameter) (1)
Dem379	Made requirement atomic (refer to Dem036) (1)
Dem380	New requirement for Dcm_NegativeResponseCodeType (1)

(1) Changes to Release 3.0

13 Changes during SWS Improvements by Technical Office

13.1 Deleted SWS Items

SWS Item	*Rationale*
Dem023	No requirement on the module.
Dem044	No requirement on the module.
Dem065	Requirement on other module.
Dem068	No requirement on the module.
Dem069	No requirement on the module.
Dem106	No requirement on the module.
Dem118	No requirement on the module.
Dem119	No requirement on the module.
Dem120	No requirement on the module but standard text.
Dem155	No requirement on the module but standard text.
Dem257	No requirement on the module.

13.2 Replaced SWS Items

SWS Item	*replaced by SWS Item*	*Rationale*
Dem001	Dem158, Dem159	Made requirement atomic
Dem005	Dem153, Dem154, Dem155	Made requirement atomic.
Dem017	Dem160, Dem161, Dem162	Made requirement atomic
Dem043	Dem219, Dem221	Made requirement atomic
Dem062	Dem216, Dem217	Made requirement atomic
Dem104	Dem156, Dem157	Made requirement atomic
Dem123	Dem169, Dem170	Made requirement atomic

13.3 Changed SWS Items

Many requirements have been changed to improve understandability without changing the technical contents.

13.4 Added SWS Items

SWS Item	*Rationale*
Dem151	Requirement on the header file structure
Dem152	Standard requirement on the header file structure
Dem163	Requirement on the Dem module
Dem164	Requirement on the Dem module
Dem165	Requirement on the Dem module
Dem166	Requirement on the Dem module
Dem167	Requirement on Dem-ReportErrorStatus
Dem168	Requirement on Dem-ReportErrorStatus
Dem171	Requirement on the Dem module

Dem172	Requirement on the Dem module
Dem173	Requirement on the Dem module
Dem174	Standard requirement on error detection
Dem175	Standard requirement on error notification
Dem176	UML model linking of imported types
Dem177	UML Model linking of Dem_GetVersionInfo
Dem178	Requirement on Dem_GetVersionInfo
Dem179	UML Model linking of Dem_PreInit
Dem180	Requirement on Dem_PreInit
Dem181	UML Model linking of Dem_Init
Dem182	UML Model linking of Dem_Shutdown
Dem183	UML Model linking of Dem_SetEventStatus
Dem184	Requirement on Dem_SetEventStatus
Dem185	UML Model linking of Dem_ResetEventStatus
Dem186	Requirement on Dem_ResetEventStatus
Dem187	Requirement on Dem_ResetEventStatus
Dem188	UML Model linking of Dem_PrestoreFreezeFrame
Dem189	Requirement on Dem_PrestoreFreezeFrame
Dem190	Requirement on Dem_SetEventStatus
Dem191	Requirement on Dem_PrestoreFreezeFrame
Dem192	Requirement on Dem_SetEventStatus
Dem193	UML Model linking of Dem_ClearPrestoredFreezeFrame
Dem194	UML Model linking of Dem_SetOperationCycleState
Dem195	UML Model linking of Dem_GetEventStatus
Dem196	UML Model linking of Dem_GetEventFailed
Dem197	UML Model linking of Dem_GetEventTested
Dem198	UML Model linking of Dem_GetDTCOfEvent
Dem199	UML Model linking of Dem_SetValueByOemId
Dem200	Requirement on Dem_SetValueByOemId
Dem201	UML Model linking of Dem_SetEnableCondition
Dem202	Requirement on Dem_SetEnableCondition
Dem203	UML Model linking of Dem_GetFaultDetectionCounter
Dem204	Requirement on Dem_GetFaultDetectionCounter
Dem205	UML Model linking of Dem_GetIndicatorStatus
Dem206	UML Model linking of Dem_ReportErrorStatus
Dem207	Requirement on Dem_ReportErrorStatus
Dem208	UML Model linking of Dem_SetDTCFilter
Dem209	UML Model linking of Dem_SetDTCFilterForRecords
Dem210	Requirement on Dem_SetDTCFilterForRecords
Dem211	UML Model linking of Dem_SetViewFilter
Dem212	UML Model linking of Dem_GetStatusOfDTC
Dem213	UML Model linking of Dem_GetDTCStatusAvailabilityMask
Dem214	UML Model linking of Dem_GetNumberOfFilteredDTC
Dem215	UML Model linking of Dem_GetNextFilteredDTC
Dem218	UML Model linking of Dem_GetDTCByOccurrenceTime
Dem222	UML Model linking of Dem_GetViewIDOfDTC
Dem223	Requirement on Dem_GetViewIDOfDTC
Dem224	UML Model linking of Dem_GetNextFilteredRecord
Dem225	Requirement on Dem_GetNextFilteredRecord
Dem226	Requirement on Dem_GetNextFilteredRecord
Dem227	UML Model linking of Dem_GetNextFilteredDTCAndFDC
Dem228	Requirement on Dem_GetNextFilteredDTCAndFDC
Dem229	Requirement on Dem_GetNextFilteredDTCAndFDC
Dem230	UML Model linking of Dem_GetTranslationType
Dem231	Requirement on Dem_GetTranslationType
Dem232	UML Model linking of Dem_GetSeverityOfDTC
Dem233	UML Model linking of Dem_DisableDTCRecordUpdate

Dem234	UML Model linking of Dem_EnableDTCRecordUpdate
Dem235	UML Model linking of Dem_GetDTCOfFreezeFrameRecord
Dem236	UML Model linking of Dem_GetFreezeFrameDataByDTC
Dem237	UML Model linking of Dem_GetFreezeFrameDataIdentifierByDTC
Dem238	UML Model linking of Dem_GetSizeOfFreezeFrame
Dem239	UML Model linking of Dem_GetExtendedDataRecordByDTC
Dem240	UML Model linking of Dem_GetSizeOfExtendedDataRecordByDTC
Dem241	UML Model linking of Dem_ClearDTC
Dem242	UML Model linking of Dem_DisableDTCStorage
Dem243	UML Model linking of Dem_EnableDTCStorage
Dem244	UML Model linking of Dem_DisableEventStatusUpdate
Dem245	UML Model linking of Dem_EnableEventStatusUpdate
Dem246	UML Model linking of Dem_GetMILStatus
Dem247	UML Model linking of Dem_GetOBDReadiness
Dem248	UML Model linking of Dem_GetDistanceMIL
Dem249	UML Model linking of Dem_GetWarmupCycleDTCclear
Dem250	UML Model linking of Dem_GetDistanceDTCclear
Dem251	UML Model linking of Dem_GetMonitorStatus
Dem252	UML Model linking of Dem_GetTimeMIL
Dem253	UML Model linking of Dem_GetTimeDTCclear
Dem254	UML Model linking of mandatory interfaces
Dem255	UML Model linking of optional interfaces
Dem256	UML Model linking of <Xxx>_DemInitMonitor{EventId}
Dem258	UML Model linking of <Xxx>_DemInit{Function}
Dem259	UML Model linking of <Xxx>_DemTriggerOnEventStatus
Dem260	UML Model linking of <Xxx>_DemTriggerOnDTCStatus
Dem261	UML Model linking of <Xxx>_DemGetDataValueByDataIdentifier
Dem262	UML Model linking of <Xxx>_DemGetExtendedDataRecord
Dem263	UML Model linking of <Xxx>_DemGetFaultDetectionCounter
Dem264	Requirement on the DEM module
Dem265	Requirement on the DEM module
Dem266	UML Model linking of Dem_MainFunction
Dem267	Definition of configuration variant needs an ID
Dem268	Definition of configuration variant needs an ID
Dem269	Requirement on Dem_GetDTCOfEvent
Dem270	Requirement on Dem_DisableDTCRecordUpdate
Dem271	Requirement on Dem_EnableDTCRecordUpdate
Dem272	Requirement on the DEM module
Dem273	Requirement on the DEM module
Dem274	Requirement on the DEM module
Dem275	Requirement on the DEM module
Dem276	Requirement on the DEM module