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

Document Version	2.2.1
Document Status	Final
Part of Release	3.0
Revision	0002

Document Change History

Date	Version	Changed by	Change Description
23.01.2008	2.2.1	AUTOSAR Administration	Table formatting corrected
28.11.2007	2.2.0	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

Any use of these specifications requires membership within the AUTOSAR Development Partnership or an agreement with the AUTOSAR Development Partnership. The AUTOSAR Development Partnership will not be liable for any use of these specifications.

Following the completion of the development of the AUTOSAR specifications commercial exploitation licenses will be made available to end users by way of written License Agreement only.

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
<Xxx>_	Placeholder for an API provider, like Fim, Rte, Abs, App, Lws,…
ECU-SM	Electronic Control Unit – State Manager
FreezeFrame	FreezeFrame is defined as a record of environmental data used for emission and non-emission relevant faults. FreezeFrames are similar to SnapShotRecords in ISO14229-1 [10]
Extended Data Record	An Extended Data Record is a record to store specific information assigned to a fault.
Monitor Function	• Part of a Software Component or Basic Software Component • Mechanism to monitor and finally to detect a fault of a certain sensor, actuator or plausibility check • Reports states of events to the DEM The ´Monitor Function´ may consist of more than one ´monitoring path´
Monitoring Path	A Monitoring Path represents a circuit or system that is being diagnosed and the type of fault in the circuit or system (e.g. sensor open circuit, sensor shorted to ground, algorithm based failure, etc).
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
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
EMI	Electro Magnetic Interference
EOL	End Of Line
ESD	Electro Static Discharge
ESP	Electronic Stability Program
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
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

3.2 Related standards and norms

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

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

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

[10] ISO14229-1: Unified diagnostic services (UDS) – Part 1: Specification and requirements (ISO14229-1 DIS 26.05.2004), [Note: in additional to the DIS version DCM will support the following feature “handling for service 0x19 subfunction 0x14 reportDTCFaultDetectionCounter”]

[11] ISO15031-5: Road vehicles – Communication between vehicle and external equipment for emission-related diagnostic – Part 5: Emission-related diagnostic services.

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

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 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

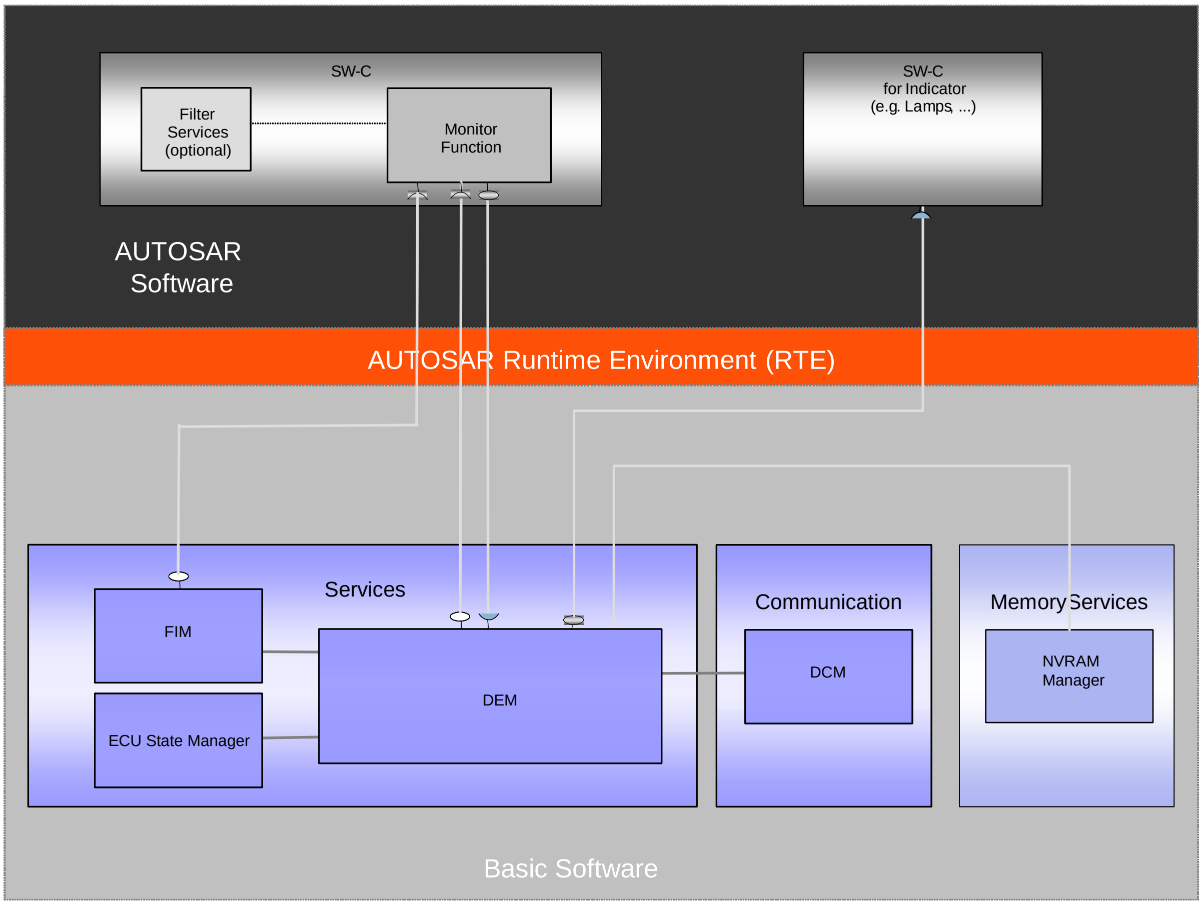
Timing constrains have to be considered for the whole ECU. If there are explicit needs for faster responses from the DEM than the DEM basic cycle time, special measures have to be implemented that are not specified in this AUTOSAR document. This is especially the case in ECUs with many events.
The current version of the DEM SWS doesn’t support OBD requirements.
The DEM is able to support additional event memories (Secondary memory and Mirror memory). Due to restrictions in DCM only the Mirror memory is used by the DCM.

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:



- 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.

environmental 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

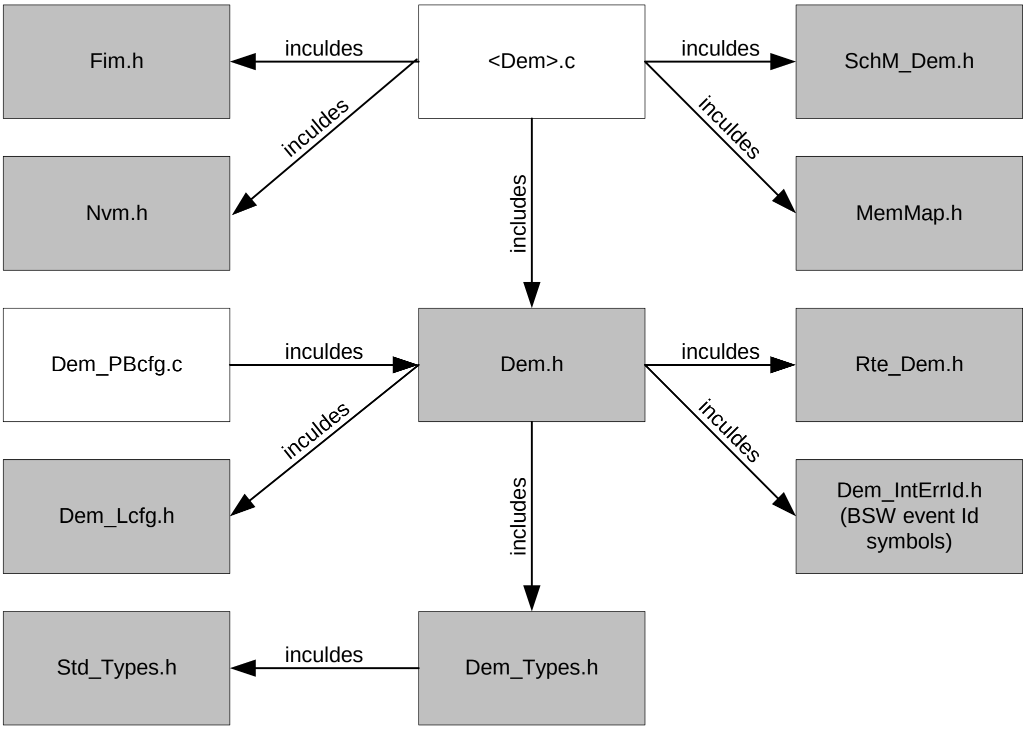
These files shall contain all link time and post-build time configurable parameters.

5.1.2 Header file structure

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

- Dem_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.


Fim.h
inculdes
<Dem>.c
inculdes
SchM_Dem.h
inculdes
inculdes
Nvm.h
includes
MemMap.h
Dem_PBcfg.c
inculdes
Dem.h
inculdes
Rte_Dem.h
inculdes
inculdes
Dem_Lcfg.h
includes
Dem_IntErrId.h
(BSW event Id
symbols)
Std_Types.h
inculdes
Dem_Types.h

6 Requirements traceability

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
[BSW333] Documentation of callback function context	Documentation requirement
[BSW334] Provision of XML file	Implementation requirement
[BSW335] Status values naming convention	Implemented
[BSW336] Shutdown interface	Not applicable

Requirement	***Satisfied by***
[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
[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

Requirement	***Satisfied by***
[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

Document: AUTOSAR requirements on Basic Software, cluster Diagnostic

Requirement	***Satisfied by***
[BSW04010] Interface between Diagnostic service handling and Diagnostic Event (error) management [approved]	See chapter Function Definition, interface DCM ⇔ DEM (Chapter 8.3.5) 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]
[BSW04057] Classification of event [approved]	Dem057, Dem058, Dem047, Dem156, Dem157
[BSW04061] Multiple or parallel usage from different applications of the DEM functionality [approved]	Dem038
[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 ISO14229-1 DTCStatusMask	Dem011, Dem061
[BSW04068] Standardized Event forget/unlearn counting	Dem019
[BSW04069] DEM System status indication [proposed]	Dem016, Dem045, 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
[BSW04075] Event and DTC assignment [approved]	Internal calibration/configuration (Chapter 10.2) See Type definition Dem_DTCTranslationFormatType
[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 variables

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 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.

Dem154: The EventId shall be unique per DEM module.

Dem155: The DEM module shall assign each event to exactly one Monitoring Path. Two Monitoring Paths can never manipulate the same EventId.

Dem006: The EventId shall point directly to the assigned event status information and possible environmental data defined in the DEM module.

EventId and DTC can have a one to one relationship or a one to n relationship. Therefore, Event status and DTC status have to be differentiated.

The DEM module receives via the RTE Diagnostic Events from Monitor Function(s) (SW-Component). The Diagnostic Events can be pre-debounced by the Monitor Function or using a filter, implemented and provided by the DEM for event qualification (ref. to 7.3.12.1).

Dem156: The DEM module shall support the configuration to assign different attributes to each EventId (to achieve a specific behavior).

The assignment of attributes to EventIds can be simplified by grouping sets of attributes to event classes.

Dem157: The DEM module shall support the following attributes for EventIds:

- 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)

7.1.2 ´Event Memory´ description

The ´Event Memory´ is the storage area/array of the events and associated data (example: chapter 7.3.1) in RAM. The events are passed from the Monitor Function via the RTE to the ´Event Memory´. The ´Event Memory´ shall be scalable depending on the number of available events by the Software components, e.g. supporting 30 out of 500 events. For storing to non-volatile memory the NVRAM Manager shall be used.

7.2 Event counting and status management

Dem158: The DEM module shall provide the functions Dem_SetEventStatus, Dem_ResetEventStatus, Dem_GetEventStatus for OBD relevant and non OBD relevant ECUs.

Dem159: The DEM module shall provide the functions Dem_PrestoreFreezeFrame/ Dem_ClearPrestoreFreezeFrame, <Xxx>_DemInitMonitor{EventId}, Dem_GetEventFailed, Dem_GetEventTested for OBD relevant systems and may provide these functions for non OBD relevant systems.

These API functions are the interface functions of the DEM module to control the behavior of the events inside the DEM module. The functions are described in detail in chapter 8.3.

Dem009: The DEM module shall provide the deletion of DTCs to each single event, event groups and all events.

ClearDiagnosticInformation (14 hex) Service of ISO14229-1 [10] defines and covers the required actions and the deletion of related memory areas like FreezeFrames. The groups are defined in ISO14229-1 [10] Definition of GroupOfDTC and range of DTC numbers, Annex D1.

Dem272: The function Dem_ClearDTC shall provide the erase functionality related to a request by the DCM.

According to the ISO14229-1 [10] service, only a DTC is transmitted which can either represent a single DTC or a group of DTCs. This distinction has to be made within the DEM when triggered by the DCM.

Dem003: The DEM module shall use an event specific function to initialize the monitor function (Chapter 7.3). The initialistion function shall be provided by the SW-C who contains the montitoring function.

With the function <Xxx>_DemInitMonitor{EventId} it is possible to initialize the monitor function of the {EventId}. With the parameter InitMonitorKind, the type of initialization is chosen.

With the interface <Xxx>_DemInit{Function} it is possible to initialize a group/set of functions.

Dem010: The DEM module shall support several event memories (the event memories can be physically located in the same address range).

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

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

Dem013: The DEM module shall support the ISO14229-1 format of DTCs (3 bytes) and the ISO 15031-6 format for OBD ECUs (2 bytes) (configuration parameter DemTypeOfDTCSupported).

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 services shall report DTC as a uint32 with byte 0 = LowByte, byte 1 = MiddleByte and byte 2 = HighByte. The byte 3 of the uint32 is free. 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.

DTC Byte Order

non OBD	free	DTC HighByte	DTC MiddleByte	DTC LowByte
OBD	free	DTC HighByte	DTC LowByte	0x00
	Byte 3	Byte 2	Byte 1	Byte 0

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 all event states is disabled, calls to Dem_SetEventStatus or Dem_ResetEventStatus will not lead to changes in internal states of the DEM module (ref. to section 8.3.3.1).

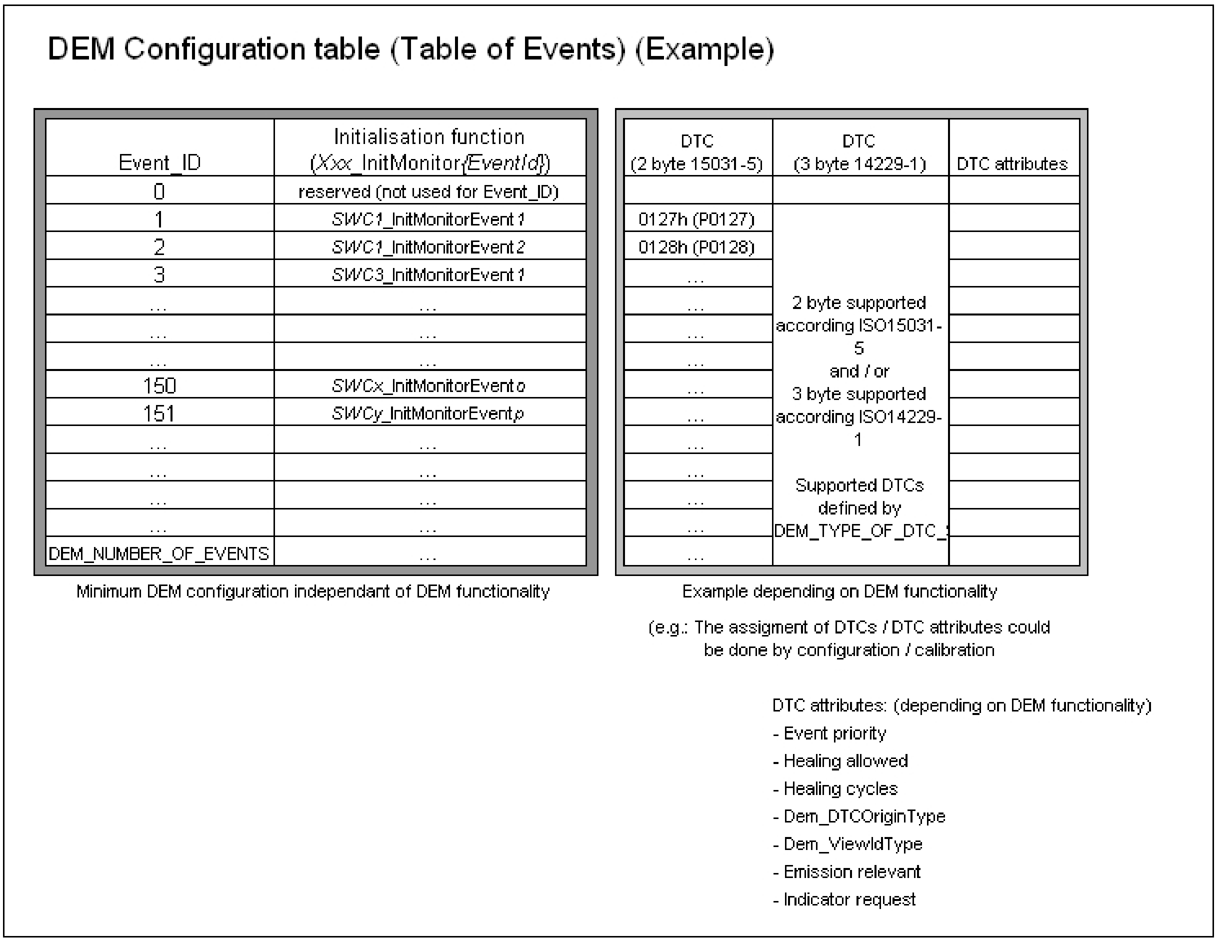
Dem035: The DEM module shall be capable of enabling (Dem_EnableDTCStorage) and disabling (Dem_DisableDTCStorage) the storage of all event records.

Meaning: when the storage of all 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).

7.3 DEM core functionalities

7.3.1 Overview DEM Structure

The figure below shows a possible DEM structure consisting of a configuration table and DEM layout.



DEM Configuration table (Table of Events) (Example)

Event_ID	Initialisation function (*Xxx*_InitMonitor*{EventId}*)
0	reserved (not used for Event_ID)
1	*SWC1*_InitMonitorEvent*1*
2	*SWC1*_InitMonitorEvent*2*
3	*SWC3*_InitMonitorEvent*1*
…	…
…	…
…	…
150	*SWCx*_InitMonitorEvent*o*
151	*SWCy*_InitMonitorEvent*p*
…	…
…	…
…	…
…	…
DEM_NUMBER_OF_EVENTS	…

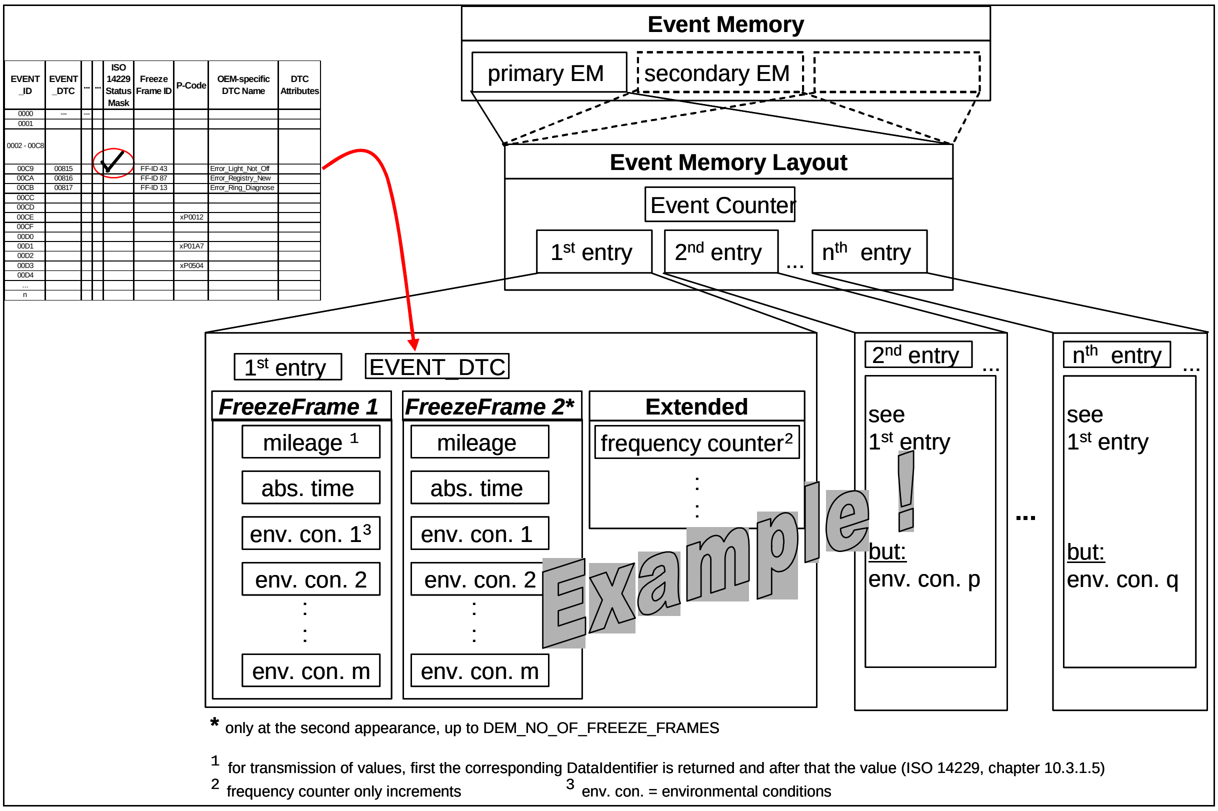
Minimum DEM configuration independant of DEM functionality

DTC (2 byte 15031-5)	DTC (3 byte 14229-1)	DTC attributes
0127h (P0127)	2 byte supported according ISO15031-5 and / or 3 byte supported according ISO14229-1 Supported DTCs defined by DEM_TYPE_OF_DTC_	
0128h (P0128)		
…		
…		
…		
…		
…		
…		
…		
…		
…		
…		
…		

Example depending on DEM functionality

(e.g.: The assigment of DTCs / DTC attributes could be done by configuration / calibration

DTC attributes: (depending on DEM functionality)
- Event priority
- Healing allowed
- Healing cycles
- Dem_DTCOriginType
- Dem_ViewIdType
- Emission relevant
- Indicator request



EVENT _ID	EVENT _DTC	...	...	ISO 14229 Status Mask	Freeze Frame ID	P-Code	OEM-specific DTC Name	DTC Attributes
0000	---	---						
0001								
0002 - 00C8								
00C9	00815			✓	FF-ID 43		Error_Light_Not_Off	
00CA	00816				FF-ID 87		Error_Registry_New	
00CB	00817				FF-ID 13		Error_Ring_Diagnose	
00CC								
00CD								
00CE						xP0012		
00CF								
00D0								
00D1						xP01A7		
00D2								
00D3						xP0504		
00D4								
...								
n								

* only at the second appearance, up to DEM_NO_OF_FREEZE_FRAMES

[1] for transmission of values, first the corresponding DataIdentifier is returned and after that the value (ISO 14229, chapter 10.3.1.5)

[2] frequency counter only increments

[3] env. con. = environmental conditions

7.3.2 Event Status Management

The functionality of the ´Event Status Management´ mainly consists of the storage timing, the order and additional status information of the events.

Dem019: The DEM module shall support event unlearn counters (e.g. for failure healing) which are event specific. In case of OBD-relevant events they shall be based on the OBD/ISO15031 defined cycles.

Dem014: The DEM module shall support the DTC Status mask in accordance to ISO14229-1 (see [10]).

The DCM can check the availability of specific status mask information by using the function Dem_GetDTCStatusAvailabilityMask (Dem060) before calling the Status mask filter function (Dem_SetDTCFilter, Dem057). The DCM shall set the Status mask as required (Dem_SetDTCFilter, Dem057).

Dem042: The DEM module shall use the internal event status information to meet the DCM-requirement of a DTC-status request.

Note: Some DTC status information exists for all events (e.g. “failedDTC”) independent whether or not they have been stored, while other DTC status information only exists for events already stored in the event memory (“pending”).

Dem036: In case the SW-C has already reported a qualified Diagnostic Event to the DEM module, the DEM module shall perform the event status transition immediately for each status supported for that event when requested via DCM API call, like Dem_GetStatusOfDTC .

Dem038: The DEM module may provide DTC information only of events of specific function groups (wiper, seat, climate control), which have been selected within the function Dem_SetViewFilter (Dem058).

Identification of views are assigned by identification numbers. The view ID is configured in the DEM. The ID describes a functional group in the car, like a wiper system or a window lift for the access of corresponding DTCs and related data. Example:

1 refers to functionality x,
2 refers to functionality y

DEM only has to support a limited number of views according to configuration of DEM_NUMBER_OF_VIEWS

Dem015: The DEM module may provide the malfunction indication status (lamps, text message, beep...) for different indicators.

The indication status is defined depending on event status information available in the DEM module. An IndicatorId is assigned to an EventId by configuration or calibration.

Dem016: The DEM module shall support the execution of the event specific function <Xxx>_DemTriggerOnEventStatus (ref. to Chapter 8.4.3.1.3) upon each event status change (e.g. FIM or ISO14229-1service 86hex "ResponseOnEvent Request").

Dem160: The DEM module shall recognize the occurrence order of events by e.g. mileage, time stamps and/or age.

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

Dem162: The DEM module shall provide enough memory space to store all high priority faults.

Dem033: Severity may be assigned to events regarding the importance of the specific events according to ISO14229, Annex D, DTCSeverityMask and DTCSeverity bit definitions.

ISO14229-1, 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.

7.3.3 DEM Statistics

Dem020: (required for OBD-relevant diagnostics)
The DEM module may provide statistical data information, e.g. according ISO15031-5. Examples of statistical data information:

- distance traveled while MIL activated (Dem_GetDistanceMIL,Dem084)
- time since DTC cleared (Dem_GetDistanceDTCclear, Dem086).

7.3.4 Event combination

Event combination is an optional feature which is implementation-specific. A possible handling of event combination is included in Dem024, Dem025 and Dem026.

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, environmental 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.

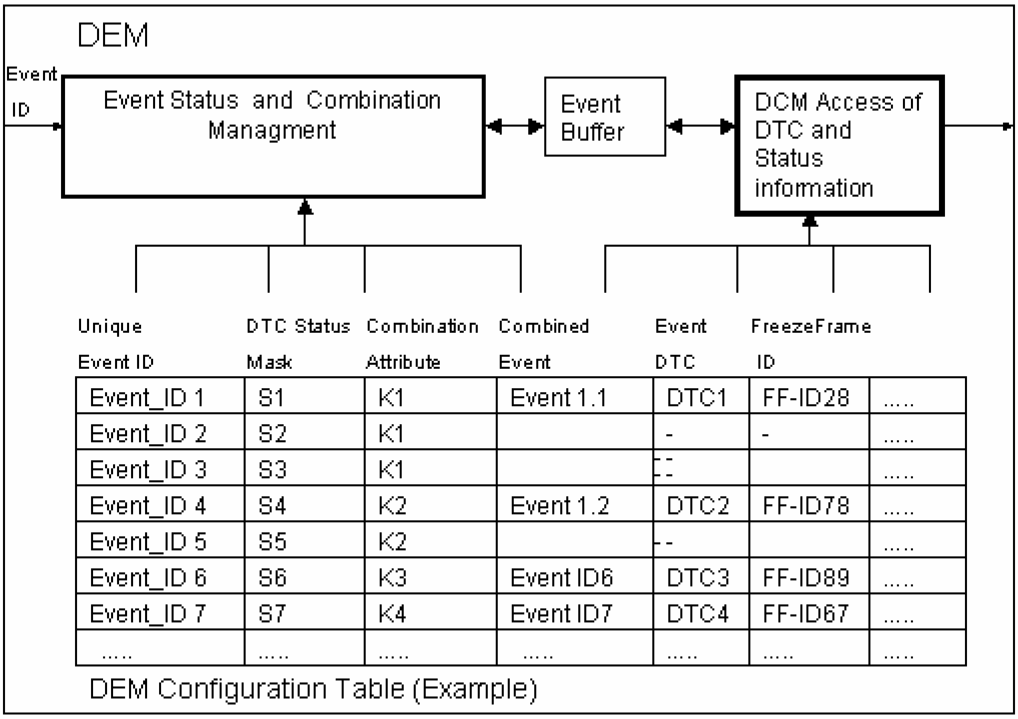
Related to the erasing of the ´combined event´ requested by ISO14229-1service 14hex "ClearDiagnosticInformation – GroupOfDTC[]" all associated Monitor Functions must be reset.

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".


DEM
Event ID
Event Status and Combination Managment
Event Buffer
DCM Access of DTC and Status information

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)

7.3.5 Environmental Data

The ´Environmental Data´ are additional data, mostly sensor values that are stored in case of an event. The number or sets of stored ´Environmental Data´ are strongly OEM / failure specific and are therefore configurable.

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

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

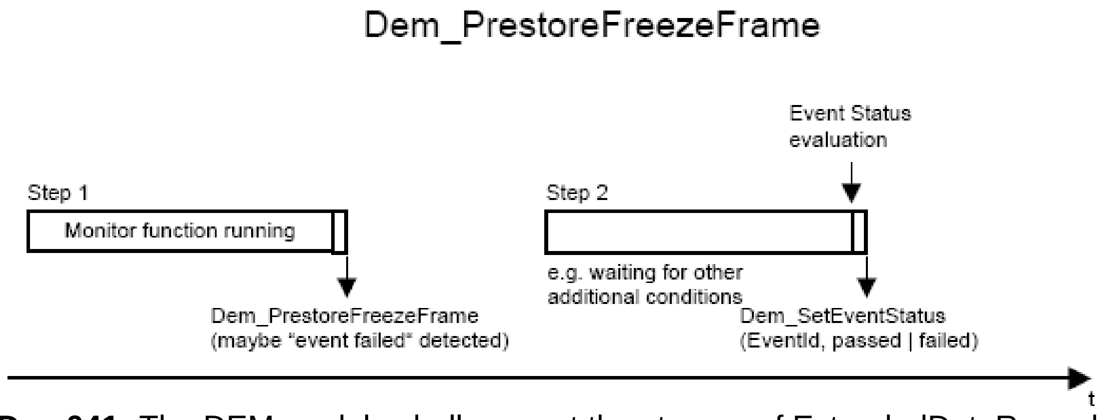
Dem021: The DEM module shall support the EventId specific storage of several of the FreezeFrames defined by Dem039:

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.

Note: Due to implementation reasons the DEM usually needs to reserve memory for the maximum FreezeFrame size multiplied by the number of FreezeFrames it shall be capable to store to cover the “worst case”.

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

The pre-storeage of FreezeFrames can be processed via API function Dem_PrestoreFreezeFrame.
A pre-stored FreezeFrame is event specific. Upon status change of the event requiring 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.



Dem041: The DEM module shall support the storage of ExtendedDataRecords.

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 ISO14229-1.

Dem090: To support events set before error is established as well as system-events, it may be possible to select DTC storage that does not use pending status at all.

Dem273: Its shall be possible to calibrate DTCs with 0x68 in the least significant bytes (i.e. 0x<nnmm>68 according to (ISO_DIS_15031-6.4_(E)(dated Jan 14th 2004)).

Dem274: DTCs calibrated with 0x68 in the least significant bytes shall be accessible through the same service as other ISO15031 DTCs (i.e. service 0x19 sub mode 0x02).

DTCs calibrated with 0x68 in the least significant bytes shall be treated as separate entities by the DEM, i.e. supporting ISO14229-1 status bit register and FreezeFrame data response etc.
None of these events are needed as input to FIM.
OBD access is not needed for these events.

7.3.6 DTC Management

The goal of DTC Management is to assign car manufacturer specific or standardized numbering and naming conventions of DTCs (e.g. P-codes for the powertrain according to ISO15031 standard [11]) to the internal EventId's.
It is possible to have more than one translation for the internal EventId (ref. to Dem006:).

7.3.7 DEM Cycle Management

Different operation cycles are used by the DEM module (ref. to ISO14229-1). 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.
For DEM specific cycles, and in accordance to the different conditions and circumstances in ECU's, operating cycles are created independently of the event in order to define the time base for qualifying the event (e.g. DTC goes from pending to confirmed state according to OBD legal definition).

Broadcast of operation cycles, in case domain or system wise synchronization is needed, which is not defined in this document. It is automotive manufacturer specific.

This shall be managed by Dem047: API Dem_SetOperationCycleState.

7.3.8 NVRAM Manager Access

The 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).

During startup before Dem_Init is called the ECU State Manager has to initiate the copying process from NVRAM to RAM. After that the DEM is operational.

When the ECU shall be shut down, the DEM module shall finish all operations on the event memory by Dem_Shutdown. The event memory shall be locked afterwards. After that, the ECU State Manager is able to initiate the copying process of data from RAM to NVRAM.

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 functions 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.

Dem165: The DEM module shall retry to access the NVRAM Manager for a configurable number of times (configuration parameter `DemNvmAccessRetry`) if the DEM module retrieves an error message from the NVRAM Manager while using the function NvM_WriteBlock or NvM_ReadBlock.

Dem166: The DEM module shall retry to access the NVRAM Manager after a configurable time as expired (configuration parameter `DemNvmAccessRetryTimeDelay`) if the DEM module retrieves an error message from the NVRAM Manager while using the function NvM_WriteBlock or NvM_ReadBlock.

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.

7.3.9 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 shall use the information of dependencies provided by the software components.

Dem029: The DEM module may inform the FIM about new event states using the function Fim_DemTriggerOnEventStatus according to the prototype <Xxx>_DemTriggerOnEventStatus (ref. to Chapter 8.4.3.1.3).

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

Dem031: The inhibition relations between events and application software depends on the event status. However, the DEM module shall report this information mapped onto an extended type of event status (not dependent on ISO14229-1) in order to use the extended range of possible states (e.g. pending or confirmed) and to keep the FIM functionality robust to changes in ISO14229-1 [10].
By this extension, function entities for complex application, e.g. long expression, can be stopped upon pending status.

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.

7.3.10 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)
- Reports on the same EventId by different modules in preemptive tasks shall be supported

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.

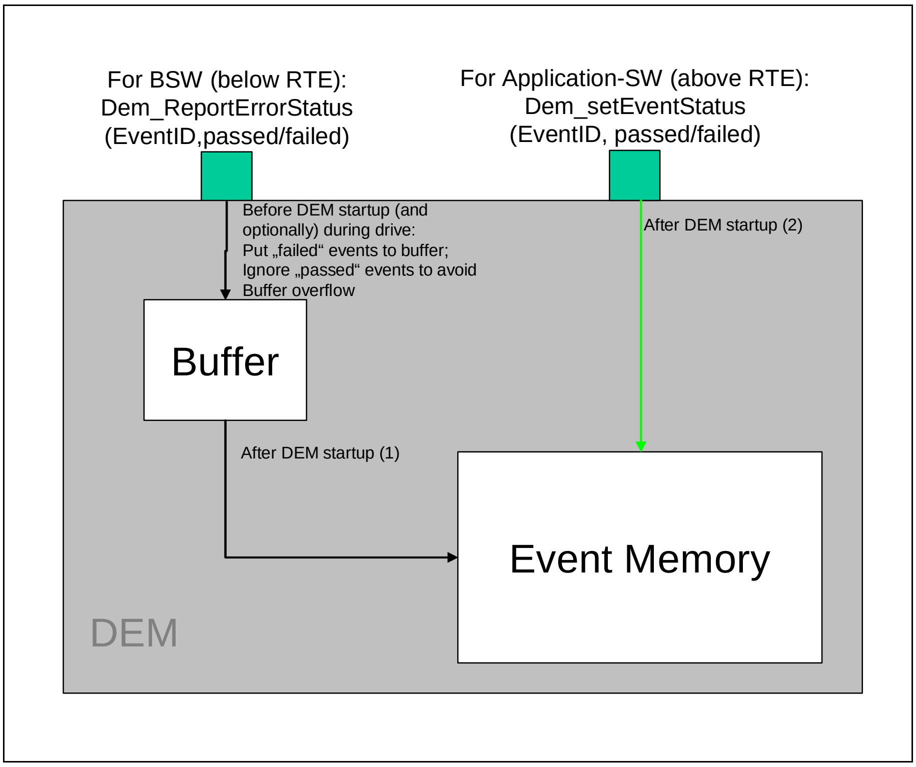
A possible implementation could be a buffer of configurable size where all errors reported by BSW are stored until the startup process is finished. During normal operation (startup has been finished) the buffer is processed by a cyclic task of the DEM module and all contained events are reported to the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



7.3.11 DEM startup behavior

Dem169: The DEM module shall distinguish between an initialization mode and an operation mode.

Dem124: If the DEM module environment is calling a function of the DEM module (excepting the function Dem_ReportErrorStatus) before the module has been initialized and if the DET is activated, the DEM module shall call the DET API Det_ReportError to set the error code DEM_E_UNINIT.

7.3.12 Debouncing of events

7.3.12.1 Kinds of de-bouncing

Dem004: De-bouncing
There are 3 levels of signal improvement:

1. **Signal debouncing** - (i.e. conditioning) is done in Hardware, using measures like EMI- or ESD suppression, low pass filtering etc.
   Signal de-bouncing in hardware is the responsibility of the ECU-Hardware designer and is not part of specification work in this document.

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

3. **Event qualification** - is defined in accordance to the *statusOfDTC* bit definition in ISO14229-1, Annex D [10].
   Event qualification is processed inside DEM if the event de-bouncing is done inside DEM. Otherwise the Event qualification is done by a Monitor Function.
   For OBD-units, event qualification according to legal requirements is mandatory.

7.3.12.2 Event de-bouncing algorithms

The DEM offers some standard algorithm for debouncing events in order to avoid multiple implementations of the same mechanism in several monitoring modules. Below the defined algorithms are described. Important to note is that a debounce mechanism can be configured per EventID (for details see Chap. 10.2) This enables to select, whether the debouncing takes place within the monitoring function or inside the DEM.
It is common for all mechanisms, that the counters are reset to 0 upon fault memory clearing. Furthermore, it is implied that the DTCFaultDetectionCounter represents the Failed / Passed detection together with the Tested detection. Therefore, the DTCFaultDetectionCounter always starts the monitoring cycle with 0.
In case of debouncing is done by monitoring function, the SW-C shall provide the services <Xxx>_DemGetFaultDetectionCounter to deliver the DTCFaultDetectionCounter and <Xxx>_DemInitMonitor{EventName} to reset the DTCFaultDetectionCounter after the DTC was cleared.

7.3.12.2.1 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.

7.3.12.2.1.1 Representation the DTCFaultDetectionCounter:

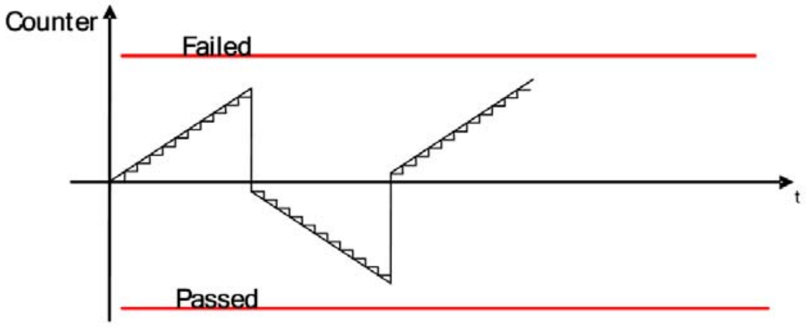
The counter base 1:1 relation with maximum value of 127 and the minimum value of -128. If the pre-debouncing has been finished then the DTCFaultDetectionCounter is either 127 (this means “TestFailed”) or -128 (this means “Passed”). When the debouncing is in progress the counter value can be derived from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

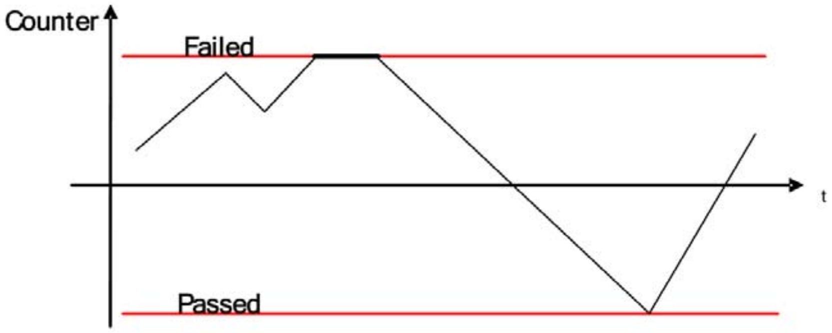
Since range of -128 to +127 is fixed, different limits for FAILED and PASSED detection in the internal debouncing can be converted into the 1-byte range via different step sizes. Therefore, it shall be possible to define either a parameter set for step size (assuming equal levels for FAILED and PASSED 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 ISO14229-1 (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



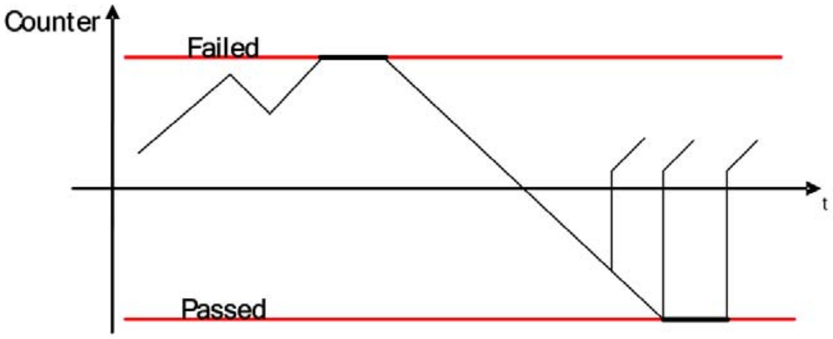
Note, that the steps should indicate individual incrementing and decrementing per report and also to show the combination of a jump and a step upon a change of the reported result.

- 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.

7.3.12.2.1.2 Use Cases

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

7.3.12.2.1.3 Parameter:

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

Alternatively:

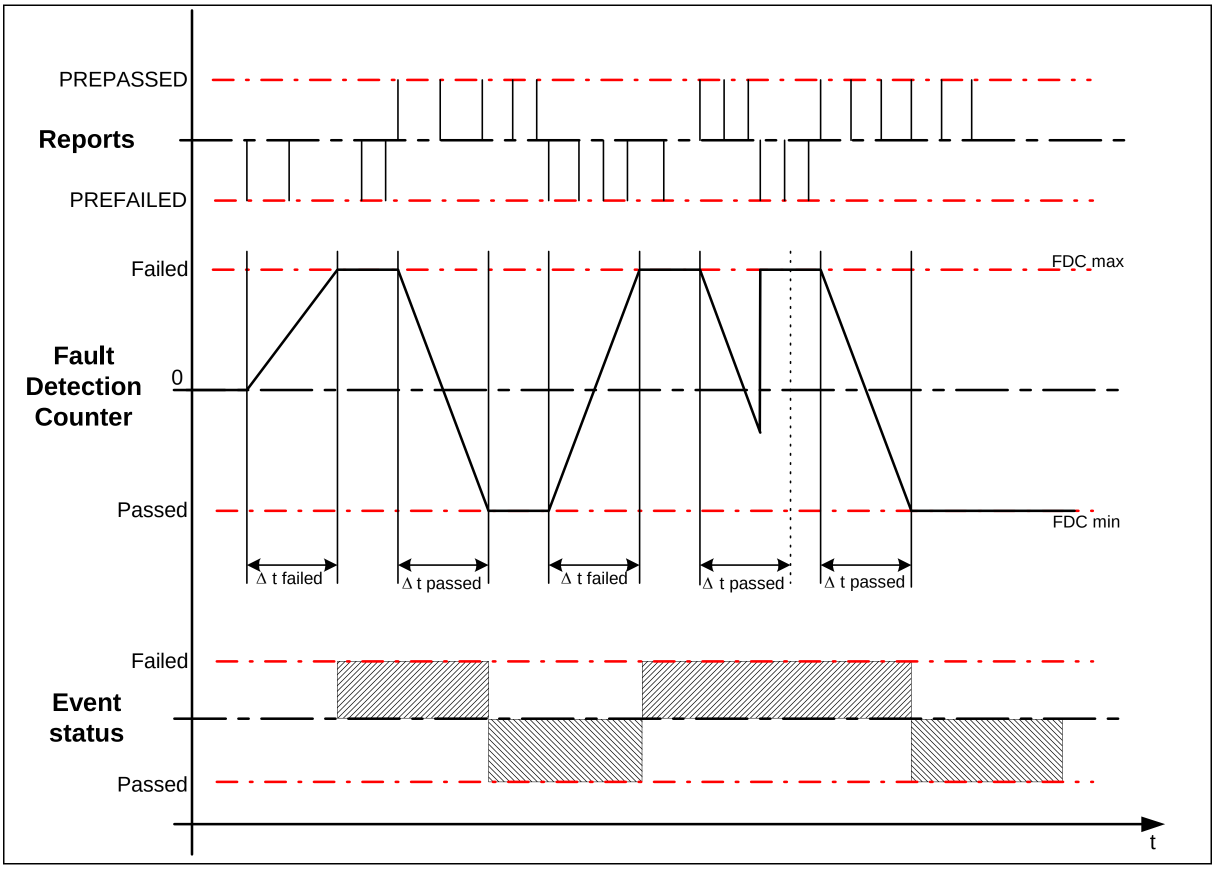
- Threshold for FAILED-detection (at this value the event is qualified to DEM_EVENT_STATUS_FAILED)
- Threshold for PASSED detection (at this value the event is qualified to DEM_EVENT_STATUS_PASSED)
- Switch for the activation of Jump-UP (boolean) (upon PREFAILED report within PREPASSED / PASSED range)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

7.3.12.2.2 Time based

The signal is unqualified until the first call of Dem_SetEventStatus/Dem_ReportError. 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 monitoring function 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 monitoring function 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 monitoring function 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 deboucing also applies here.

7.3.12.2.2.1 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.



7.3.12.2.2.2 Use Cases

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

7.3.12.2.2.3 Parameter:

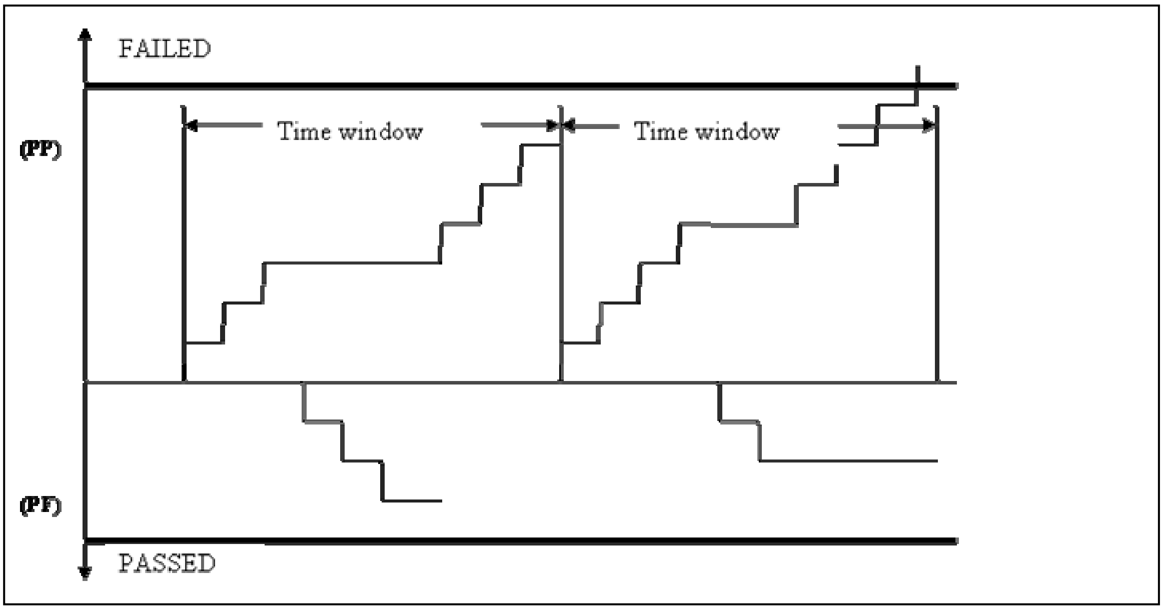
- Time threshold for qualification as failed.
- Time threshold for qualification as passed.

7.3.12.2.3 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.

7.3.12.2.3.1 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.



7.3.12.2.3.2 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'

7.3.12.2.3.3 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)

7.4 Auxiliary explanations and definitions

7.4.1 Requirements on variables

7.4.1.1 Variables provided for the DEM module

The DEM module requires several input values for computation.

Dem278: The DEM module shall request values of environmental to be stored in FreezeFrames or Extended Data Records <Xxx>_GetDataValueByIdentifier via the data ID configuration table according to ISO14229-1.

Dem279: The DEM module shall support a data ID configuration table containing Data IDs according to ISO14229-1 [10] and parameters required as PIDs according to ISO15031-5 [11].

The PIDs are necessary to fulfil OBD mode 2 of ISO15031-5.

Dem280: The DEM module shall request data for the calculation of operation cycles or statistical data via normal RTE interface.

Examples for such data are engine speed or ambient temperature.

7.4.1.2 Variables 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 initialized	Development	`DEM_E_UNINIT`	`0x20`
No valid data available by the SW-C	Development	`DEM_E_NODATAAVAILABLE`	`0x30`

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.

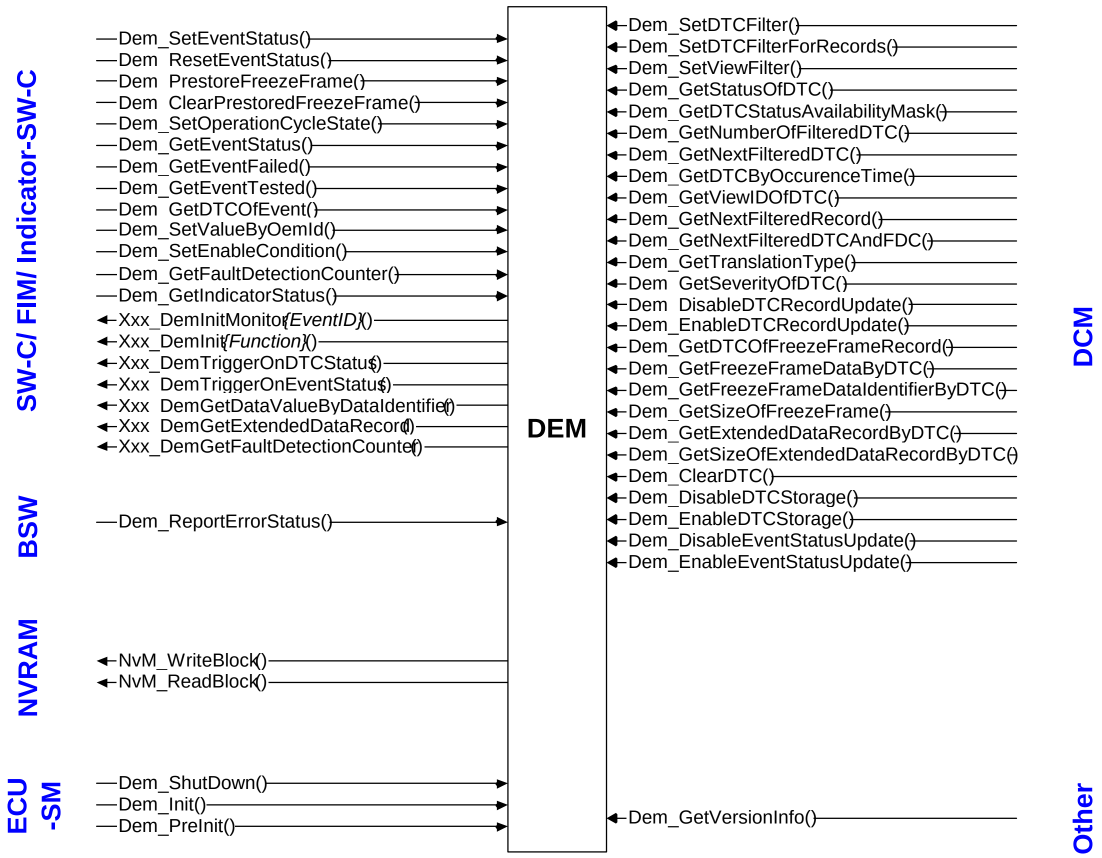
Dem174: The detection of production code errors cannot be switched off.

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 ECU-SM.


SW-C/ FIM/ Indicator-SW-C
Dem_SetEventStatus()
Dem ResetEventStatus()
Dem PrestoreFreezeFrame()
Dem ClearPrestoredFreezeFrame()
Dem_SetOperationCycleState()
Dem_GetEventStatus()
Dem_GetEventFailed()
Dem_GetEventTested()
Dem GetDTCOfEvent()
Dem_SetValueByOemId()
Dem_SetEnableCondition()
Dem_GetFaultDetectionCounter()
Dem_GetIndicatorStatus()
Xxx_DemInitMonitor{EventID}()
Xxx_DemInit{Function}()
Xxx_DemTriggerOnDTCStatus()
Xxx DemTriggerOnEventStatus()
Xxx DemGetDataValueByDataIdentifier()
Xxx DemGetExtendedDataRecord()
Xxx_DemGetFaultDetectionCounter()
BSW
Dem_ReportErrorStatus()
NVRAM
NvM_WriteBlock()
NvM_ReadBlock()
ECU -SM
Dem_ShutDown()
Dem_Init()
Dem_PreInit()
DEM
Dem_SetDTCFilter()
Dem_SetDTCFilterForRecords()
Dem_SetViewFilter()
Dem_GetStatusOfDTC()
Dem_GetDTCStatusAvailabilityMask()
Dem_GetNumberOfFilteredDTC()
Dem_GetNextFilteredDTC()
Dem_GetDTCByOccurenceTime()
Dem_GetViewIDOfDTC()
Dem_GetNextFilteredRecord()
Dem_GetNextFilteredDTCAndFDC()
Dem_GetTranslationType()
Dem GetSeverityOfDTC()
Dem DisableDTCRecordUpdate()
Dem_EnableDTCRecordUpdate()
Dem_GetDTCOfFreezeFrameRecord()
Dem_GetFreezeFrameDataByDTC()
Dem_GetFreezeFrameDataIdentifierByDTC()
Dem_GetSizeOfFreezeFrame()
Dem_GetExtendedDataRecordByDTC()
Dem_GetSizeOfExtendedDataRecordByDTC()
Dem_ClearDTC()
Dem_DisableDTCStorage()
Dem_EnableDTCStorage()
Dem_DisableEventStatusUpdate()
Dem_EnableEventStatusUpdate()
DCM
Dem_GetVersionInfo()
Other

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
Std_Types.h	Std_ReturnType
	Std_VersionInfoType

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.	

8.2.2 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_ReturnSetViewFilterType

Name:	Dem_ReturnSetViewFilterType		
Type:	uint8		
Range:	DEM_WRONG_ID	0x01	Wrong View ID selected
	DEM_VIEW_ID_ACCEPTED	0x00	View ID was accepted
Description:	Used to return the status of updating the View filter for a functional addressing.		

8.2.2.3 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.		

8.2.2.4 Dem_ReturnGetNextFilteredDTCType

Name:	Dem_ReturnGetNextFilteredDTCType		
Type:	uint8		
Range:	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.		

8.2.2.5 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.		

8.2.2.6 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.		

8.2.2.7 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.		

8.2.2.8 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.		

8.2.2.9 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.		

8.2.2.10 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.		

8.2.2.11 Dem_ReturnGetExtendedDataRecordByDTCType

Name:	Dem_ReturnGetExtendedDataRecordByDTCType		
Type:	uint8		
Range:	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.		

8.2.2.12 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.		

8.2.2.13 Dem_ReturnGetFreezeFrameDataByDTCType

Name:	Dem_ReturnGetFreezeFrameDataByDTCType		
Type:	uint8		
Range:	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.		

8.2.2.14 Dem_ReturnGetSizeOfExtendedDataRecordByDTCType

Name:	Dem_ReturnGetSizeOfExtendedDataRecordByDTCType		
Type:	uint8		
Range:	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.		

8.2.2.15 Dem_ReturnGetSizeOfFreezeFrameType

Name:	Dem_ReturnGetSizeOfFreezeFrameType		
Type:	uint8		
Range:	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.		

8.2.2.16 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.		

8.2.2.17 Dem_ReturnGetViewIDOfDTCType

Name:	Dem_ReturnGetViewIDOfDTCType		
Type:	uint8		
Range:	DEM_VIEWID_WRONG_DTCKIND	0x02	DTC kind wrong
	DEM_VIEWID_WRONG_DTC	0x01	Wrong DTC
	DEM_VIEWID_OK	0x00	Status of ViewID is OK
Description:	Used to return the status of Dem_GetViewIDOfDTC.		

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 BSWs.

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

The function DEM_PreInit shall be called by the ECU State Manager during the startup phase of the ECU before the NVRAM Manager has finished the restore of NVRAM data.

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.

Dem170: The function Dem_Init shall set the static status variable to a value not equal to 0.

The function Dem_Init shall be 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.

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.

Dem102: The function Dem_Shutdown shall finalize all pending operations in the DEM module to prepare the internal states and event data for transfer to the NVRAM.

After a call of Dem_Shotdown the Dem calls NvM_WriteBlock. The relevant block IDs are configured in the subcontainer DemNvramBlockId and the references DemNvramBlockIdRef.
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.

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
Description:	Queue the status of an event.	

Dem184: The function Dem_SetEventStatus shall store the relevant Event 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.

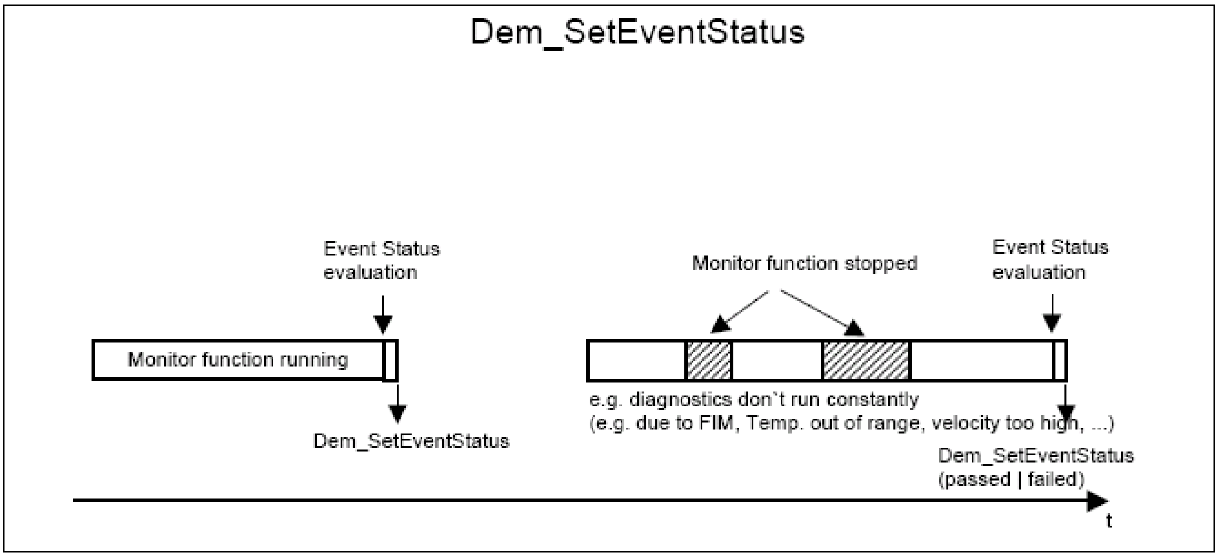
Dem192: If the the function Dem_SetEventStatus is used with a pre-debounced status (DEM_EVENT_STATUS_PASSED, DEM_EVENT_STATUS_FAILED), then a pre-stored FreezeFrame of the corresponding event shall be discarded (same behaviour like Dem_ClearPrestoredFreezeFrame).

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 Dem158, Dem159]

No API parameter checks are required for the function 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	None	

(inout):		
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: reset of event status was successful E_NOT_OK: reset of event status failed
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 (ISO14229-1 [10]) to 0.

The function Dem_ResetEventStatus will be called by a Monitor Function.

The function Dem_ResetEventStatus does not influence the status bit 6 (“TestNotCompletedThisMonitoringCycle”). [ref. to Dem158, Dem159] and the pre-stored FreezeFrame. [ref. to Dem158, Dem159].

Refer to ISO14229: DTC Status Bit Definition, Table D.14, Bit0 Test failed and Bit6 TestNotCompletedThisMonitoringCycle.

No API parameter checks are required for the function Dem_ResetEventStatus.

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 environmental data during running failure monitoring phase).

Dem191: The caputre 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.

No API parameter checks are required for the function Dem_PrestoreFreezeFrame.

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 Freeze Frame	

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, uint8* 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 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.

Configuration of Dem_GetEventStatus: EventId

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	EventFailed	TRUE - Last Failed

(out):		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, uint8 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 DEM_DTC_KIND_ALL_DTCS Select all DTCs DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant 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 DEM_GET_DTCOFEVENT_WRONG_DTCKIND: 0x04 DTC kind wrong, DEM_GET_DTCOFEVENT_WRONG_EVENTID: 0x05 Wrong EventId
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.

Configuration of Dem_GetDTCOfEvent: Mapping of events to DTCs is configured in the DEM module. Mapping is "n to 1" xor "1 to n". Cross dependencies between "n to 1" and "1 to n" relationships are not allowed.

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.	

The function Dem_ReportErrorStatus is an interface for BSW Components to report Errors during start up (even before DEM initialization) and normal operation. At a first step, it is assumed, that all incoming results are considered as debounced. If a central pre-debouncing is provided, this API shall be used to support them for the BSW.

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

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, uint8 DTCKind, uint8 DTCOrigin, uint8 FilterWithSeverity, uint8 DTCSeverity, uint8 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) DEM_DTC_KIND_ALL_DTCS selects all DTCs, DEM_DTC_KIND_EMISSION_REL_DTCS selects 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. DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information ocated in the secondary memory, DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory The definition and use of the different memory types is OEM specific.
	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. DEM_FILTER_WITH_SEVERITY_YES 0x00 Severity information used, DEM_FILTER_WITH_SEVERITY_NO 0x01 Severity information not used
	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
	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 Xxx_DemGetFaultDetectionCounter. DEM_FILTER_FOR_FDC_YES 0x00 Fault Detection Counter information used, DEM_FILTER_FOR_FDC_NO 0x01 Fault Detection Counter information not used
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.	

Dem057: The function Dem_SetDTCFilter shall set the filter mask over all DTCs for the function Dem_GetNextFilteredDTC, Dem_GetNextFilteredDTCAndFDC as well as Dem_GetNextFilteredDTCAndSeverity and reset the internal counter to the first event that matches the filter settings.

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_SetViewFilter

Dem211:

Service name:	Dem_SetViewFilter	
Syntax:	Dem_ReturnSetViewFilterType Dem_SetViewFilter(uint8 ViewId)	
Service ID[hex]:	0x14	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ViewId	The ViewId is a parameter used to select a specific view.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_ReturnSetViewFilterType	Status of the operation of type Dem_ReturnSetViewFilterType.
Description:	Sets a view filter.	

Dem058: The function Dem_SetViewFilter shall set a mask to process only the events of a functional addressable function with the following DCM <-> DEM API calls (Chapter 8.3.5).

The DEM module’s environment shall call the function Dem_SetViewFilter again with another ViewId to to change the view .

The chosen ViewId is reset to the default value (0xFF → all functional groups are visible) inside the DEM module when the DEM module’s environment has called the function Dem_SetDTCFilter or the function Dem_Shutdown.

The function Dem_SetViewFilter may be used in case that only a special functional group shall be addressed (e.g. wiper system, window lifter). After setting a ViewID only events from the selected group are accessible in the event memory.

The DEM module’s environment shall use the function Dem_SetViewFilter for function oriented diagnostics on ECUs with multiple functions.

Configuration of Dem_SetViewFilter: The assignment of an EventId to a specific view has to be configured / calibrated.

8.3.5.1.4 Dem_GetStatusOfDTC

Dem212:

Service name:	Dem_GetStatusOfDTC	
Syntax:	Dem_ReturnGetStatusOfDTCType Dem_GetStatusOfDTC(uint32 DTC, uint8 DTCKind, uint8 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 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. DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, DEM_DTC_ORIGIN_SECONDARY_MEMORY

		Event information ocated in the secondary memory, DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory The definition and use of the different memory types is OEM specific.
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 ISO14229-1 [10].

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).

8.3.5.1.5 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.
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 ISO14229-1 [10]) supported by the DEM module.

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

8.3.5.1.6 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).

8.3.5.1.7 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. 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).

8.3.5.1.8 Dem_GetDTCByOccurrenceTime

Dem218:

Service name:	Dem_GetDTCByOccurrenceTime	
Syntax:	Dem_ReturnGetDTCByOccurrenceTimeType Dem_GetDTCByOccurrenceTime(uint8 DTCRequest, uint8 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. DEM_FIRST_DET_CONFIRMED_DTC 0x03 first detected confirmed DTC requested, DEM_MOST_RECENT_FAILED_DTC 0x02 most recent failed DTC requested, DEM_MOST_REC_DET_CONFIRMED_DTC 0x04 most recently detected confirmed DTC requested, DEM_FIRST_FAILED_DTC 0x01 first failed DTC requested
	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
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.	

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

Dem221: The function Dem_GetDTCByOccurrenceTime shall return the appropriate operation status (ref. to 8.2.2.12) and set the DTC value to zero (0) if no DTC is matching the requested point in time.

8.3.5.1.9 Dem_GetViewIDOfDTC

Dem222:

Service name:	Dem_GetViewIDOfDTC	
Syntax:	Dem_ReturnGetViewIDOfDTCType Dem_GetViewIDOfDTC(uint32 DTC, uint8 DTCKind, uint8* ViewId)	
Service ID[hex]:	0x36	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	This parameter defines the requested DTC.
	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
Parameters (inout):	None	
Parameters (out):	ViewId	The ViewId is a parameter used to select a specific view.
Return value:	Dem_ReturnGetViewIDOfDTCType	Status of the operation of type Dem_ReturnGetViewIDOfDTCType.
Description:	Gets the ViewID of a DTC.	

Dem223: The function Dem_GetViewIDOfDTC shall provide the capability to get the according ViewID (e.g. wiper system, window lifter, …) of a specific DTC.

The parameter ViewID of the function Dem_GetViewIDOfDTC is equivalent to the parameter FunctionalUnit in ISO14229-1 [10].

8.3.5.1.10 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.

8.3.5.1.11 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

8.3.5.1.12 Dem_GetNextFilterdDTCAndSeverity

Dem281:

Service name:	Dem_GetNextFilteredDTCAndSeverity	
Syntax:	Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredDTCAndSeverity(uint32* DTC, uint8* DTCStatus, uint8* 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.

8.3.5.1.13 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 configurted translation formats Bit 0: 2 byte ISO15031-6 DTC Bit 1: 3 byte ISO14229-1 DTC Bit 2: Customer specific DTC Bit 3: SAEJ1939 Bit 4: WWH-OBD-format Set the according Bit to '1' means DTC format is supported. Combination of different DTC formats is possible (e.g. ISO 15031-6 and ISO14229-1 is coded by 0x0011b).
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.

8.3.5.1.14 Dem_GetSeverityOfDTC

Dem232:

Service name:	Dem_GetSeverityOfDTC	

Syntax:	`Dem_ReturnGetSeverityOfDTCType Dem_GetSeverityOfDTC(`
	`    uint32 DTC,`
	`    uint8* 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 ISO14229-1 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 environmental 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 ISO14229-1 and ISO15031-5. 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, uint8 DTCOrigin, uint8 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.
	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. DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information ocated in the secondary memory,

		DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory The definition and use of the different memory types is OEM specific.
	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
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, uint8 DTCKind, uint8 DTCOrigin, uint8 RecordNumber, uint16 DataId, uint8* DestBuffer,

	uint8* BufSize)	
Service ID[hex]:	0x1d	
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 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. DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information ocated in the secondary memory, DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary 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.
	DataId	This parameter specifies the PID (ISO15031-5) (Mode2 individual parameter or the whole FreezeFrame data set) or data identifier (ISO14229-1) that shall be copied to the destination buffer.
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 Freeze Frame Data by DTC	

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, ...

The DCM does not know the DEM module internal structure so it requests per Identifier to get special PIDs for instance, not intended to get all FreezeFrame data value by value. In case of DataId=All FreezeFrame Data will be transferred at once.

8.3.5.2.5 Dem_GetFreezeFrameDataIdentifierByDTC

Dem237:

Service name:	Dem_GetFreezeFrameDataIdentifierByDTC	
Syntax:	Dem_ReturnGetFreezeFrameDataIdentifierByDTCType Dem_GetFreezeFrameDataIdentifierByDTC(uint32 DTC, uint8 DTCKind, uint8 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 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. DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information ocated in the secondary memory, DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary 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 Freeze Frame 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,
	   uint8 DTCKind,
	   uint8 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 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. DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information ocated in the secondary memory, DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary 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):	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, uint8 DTCKind, uint8 DTCOrigin, uint8 ExtendedDataNumber, uint8* DestBuffer, uint8* BufSize)	
Service ID[hex]:	0x20	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	This is the DTC the 'Extended Data Record' is assigned to.
	DTCKind	s the requested DTC, either only OBD- TCs TCS Select all DTCs ION_REL_DTCS Select OBD-relevant DTCs e requested DTC, either only OBD-relevant TCS Select all DTCs ION_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. DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information ocated in the secondary

		memory, DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary 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):	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:	Gets extended data record by DTC	

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, uint8 DTCKind, uint8 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 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. DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information ocated in the secondary memory, DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary 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, uint8 DTCKind, uint8 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 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. DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information ocated in the secondary memory, DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory The definition and use of the different memory types is OEM specific.
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 (environmental and/or FreezeFrame data, …).

Configuration of Dem_ClearDTC: The initialization of the corresponding Monitor Function (DTC → EventId → <Xxx>_DemInitMonitor{EventName}) is managed by <Xxx>_Dem_InitMonitor{EventId}.

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(uint32 DTCGroup, uint8 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. DEM_DTC_GROUP_BODY_DTCS selects group of body DTCs, DEM_DTC_GROUP_EMISSION_REL_DTCS selects group of OBD-relevant DTCs, DEM_DTC_GROUP_ALL_DTCS selects all DTCs, DEM_DTC_GROUP_CHASSIS_DTCS selects group of chassis DTCs, DEM_DTC_GROUP_NETWORK_COM_DTCS selects group of network communication DTCs, DEM_DTC_GROUP_POWERTRAIN_DTCS selects group of powertrain DTCs
	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
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_ReturnControlDTCStorageType	Status of the operation of type Dem_ReturnControlDTCStorageType.
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(uint32 DTCGroup, uint8 DTCKind)	
Service ID[hex]:	0x25	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTCGroup	Defines the group of DTC that shall be disabled to store in event memory. DEM_DTC_GROUP_BODY_DTCS selects group of body DTCs, DEM_DTC_GROUP_EMISSION_REL_DTCS selects group of OBD-relevant DTCs, DEM_DTC_GROUP_ALL_DTCS selects all DTCs, DEM_DTC_GROUP_CHASSIS_DTCS selects group of chassis DTCs, DEM_DTC_GROUP_NETWORK_COM_DTCS selects group of network communication DTCs, DEM_DTC_GROUP_POWERTRAIN_DTCS selects group of powertrain DTCs
	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
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_ReturnControlDTCStorageType	Status of the operation of type Dem_ReturnControlDTCStorageType.
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(uint32 DTCGroup, uint8 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. DEM_DTC_GROUP_BODY_DTCS selects group of body DTCs, DEM_DTC_GROUP_EMISSION_REL_DTCS selects group of OBD-relevant DTCs, DEM_DTC_GROUP_ALL_DTCS selects all DTCs, DEM_DTC_GROUP_CHASSIS_DTCS selects group of chassis DTCs, DEM_DTC_GROUP_NETWORK_COM_DTCS selects group of network communication DTCs, DEM_DTC_GROUP_POWERTRAIN_DTCS selects group of powertrain DTCs
	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
Parameters	None	

(inout):		
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 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(uint32 DTCGroup, uint8 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. DEM_DTC_GROUP_BODY_DTCS selects group of body DTCs, DEM_DTC_GROUP_EMISSION_REL_DTCS

		selects group of OBD-relevant DTCs, DEM_DTC_GROUP_ALL_DTCS selects all DTCs, DEM_DTC_GROUP_CHASSIS_DTCS selects group of chassis DTCs, DEM_DTC_GROUP_NETWORK_COM_DTCS selects group of network communication DTCs, DEM_DTC_GROUP_POWERTRAIN_DTCS selects group of powertrain DTCs
	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
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.

8.4 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 module.

Currently Mandatory Interfaces are non-available.

8.4.2 Optional Interfaces

This chapter defines all interfaces, which are required to fulfill an optional functionality of the 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_WriteAll	Initiates a multi block write request.
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.
NvM_ReadAll	Initiates a multi block read request.

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.

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 <Xxx>_DemInitMonitor{*EventName}*

Dem256:

Service name:	<Xxx>_DemInitMonitor{EventName}	

Syntax:	`Std_ReturnType <Xxx>_DemInitMonitor{EventName}(`
	`    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}.	

The function <Xxx>_DemInitMonitor{EventName} shall init the Monitor Function of an specific Event {*EventName*}. By the parameter Dem_InitMonitorKind the type of initialisation is chosen.

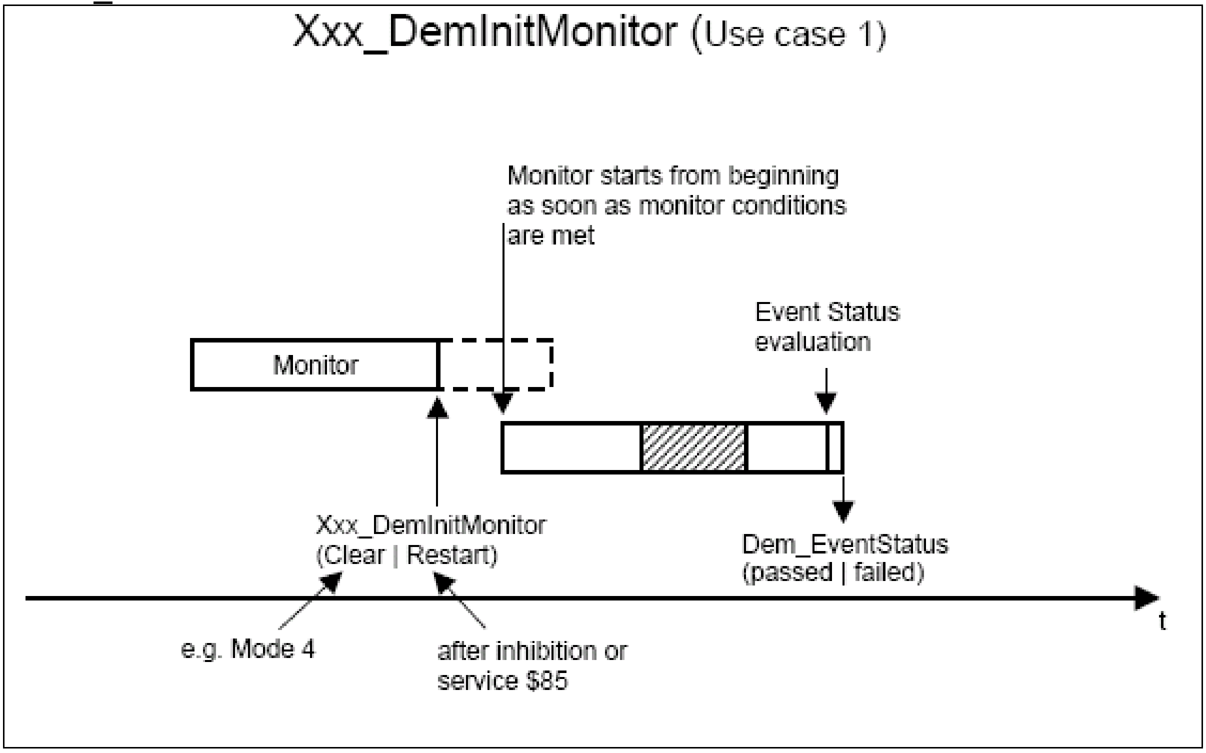
The function <Xxx>_DemInitMonitor{EventName} is called from DEM module and has to be provided by SW_C. (Ref to Dem003)

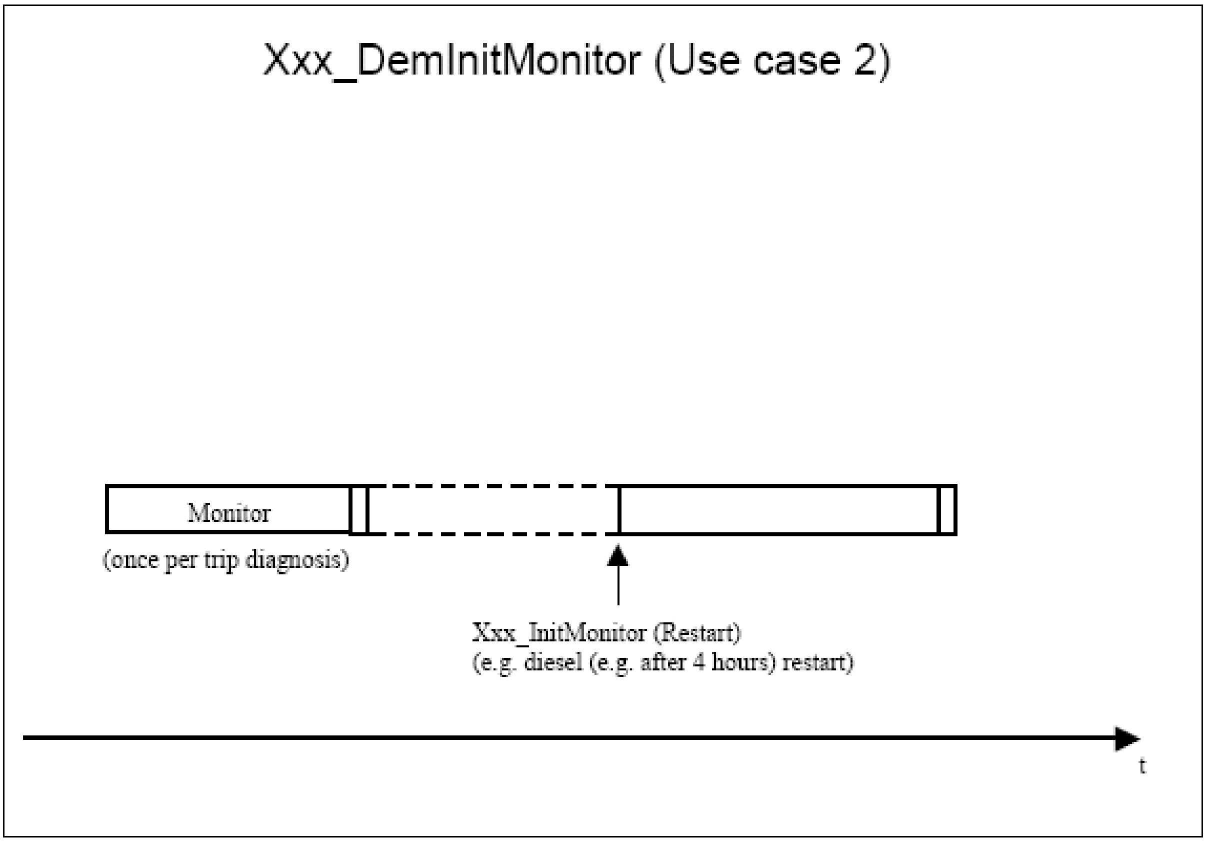
No API parameter checks are required for the function <Xxx>_DemInitMonitor{EventName}.

Caveats of <Xxx>_DemInitMonitor{EventName}: DEM configuration during integration of Monitor Functions is system specific.

Configuration of <Xxx>_DemInitMonitor{EventName}: The link between the EventId and the corresponding function <Xxx>_DemInitMonitor{EventName} is configured within the DEM module.

Example for the function name after configuration:
<Xxx>_DemInitMonitor0x2709


Xxx_DemInitMonitor (Use case 1)
Monitor starts from beginning as soon as monitor conditions are met
Event Status evaluation
Monitor
Xxx_DemInitMonitor (Clear | Restart)
Dem_EventStatus (passed | failed)
t
e.g. Mode 4
after inhibition or service $85


Xxx_DemInitMonitor (Use case 2)
Monitor
(once per trip diagnosis)
Xxx_InitMonitor (Restart)
(e.g. diesel (e.g. after 4 hours) restart)
t

8.4.3.1.2 <Xxx>_DemInit*{Function}*

Dem258:

Service name:	<Xxx>_DemInit{Function}	

Syntax:	`Std_ReturnType <Xxx>_DemInit{Function}(`
	
	`)`	

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 <Xxx>_DemInit*{Function}* to reset the {Function} of the Module <Xxx>.

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

Caveats of <Xxx>_DemInit{Function}: DEM module configuration during integration of Monitor Functions is system specific.

Configuration of <Xxx>_DemInit{Function}: During system configuration 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.
{Function} is a placeholder for a real unique function name, provided by the SW-C.

8.4.3.1.3 <Xxx>_DemTriggerOnEventStatus{EventName}

Dem259:

Service name:	<Xxx>_DemTriggerOnEventStatus{EventName}	

Syntax:	`Std_ReturnType <Xxx>_DemTriggerOnEventStatus{EventName}(`
	`    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 <Xxx>_DemTriggerOnEventStatus{EventName} that will be triggered by the DEM module on changes of the extended Event status.

Caveats of <Xxx>_DemTriggerOnEventStatus{EventName}: In case of disabling the event status update the function <Xxx>_DemTriggerOnEventStatus{EventName} does not need to be called because no status change can be reported.

Configuration of <Xxx>_DemTriggerOnEventStatus{Eventname}: 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 <Xxx>_DemTriggerOnDTCStatus

Dem260:

Service name:	<Xxx>_DemTriggerOnDTCStatus	

Syntax:	`Std_ReturnType <Xxx>_DemTriggerOnDTCStatus(`
	`    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 <Xxx>_DemTriggerOnDTCStatus that will be triggered by the DEM module on changes of the DTC status.

Configuration of <Xxx>_DemTriggerOnDTCStatus: 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 <Xxx>_DemGetDataValueByDataIdentifier

Dem261:

Service name:	<Xxx>_DemGetDataValueByDataIdentifier{DataId}	

Syntax:	`Std_ReturnType <Xxx>_DemGetDataValueByDataIdentifier{DataId}(`
	`    uint8* DataValueBuffer`
	`)`	

Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	DataValueBuffer	Pointer to the buffer in the DEM which should be filled with the requested value
Return value:	Std_ReturnType	In Case of Data value of the requested data identifier is available and successful supplied as out parameter the API returns E_OK. If there is no data value available for a certain data identifier the API returns E_NOT_OK. 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 data value by a data identifier.	

The function <Xxx>_DemGetDataValueByDataIdentifier{DataId} (provided by a SW-C) returns the associated data value for a requested data identifier.

Dem283: The function <Xxx>_DemGetDataValueByDataIdentifier{DataId} shall be called by the DEM module as soon as it requires the data value to use it in a specific FreezeFrame. The DataID (resp. PID) can be used directly.

The software component which is responsible for providing and updating the data values (e.g. vehicle speed, RPM…) has to provide the function <Xxx>_DemGetDataValueByDataIdentifier{DataId}, too.

8.4.3.1.6 <Xxx>_DemGetExtendedDataRecord{recordNumber}

Dem262:

Service name:	<Xxx>_DemGetExtendedDataRecord{RecordNumber}

Syntax:	`Std_ReturnType <Xxx>_DemGetExtendedDataRecord{RecordNumber}(`
	`    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 <Xxx>_DemGetExtendedDataRecord{RecordNumber} (provided by a SW-C) may be used in case the Extended Data Record is provided by the application.

The function <Xxx>_DemGetExtendedDataRecord{RecordNumber} (provided by a SW-C) returns the associated Extended Data Record for a requested ExtendedDataRecordNumber.

The DEM module will use the function <Xxx>_DemGetExtendedDataRecord{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 <Xxx>_DemGetExtendedDataRecord{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 <Xxx>_DemGetFaultDetectionCounter{EventName}

Dem263:

Service name:	<Xxx>_DemGetFaultDetectionCounter{EventName}	

Syntax:	`Std_ReturnType <Xxx>_DemGetFaultDetectionCounter{EventName}(`
	`    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 <Xxx>_DemGetFaultDetectionCounter{EventName} to request the current Fault Detection Counter for a given EventID.

Dem265: The DEM module shall use the function <Xxx>_DemGetFaultDetectionCounter{EventName} only if debouncing is not done by the DEM module itself.

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.

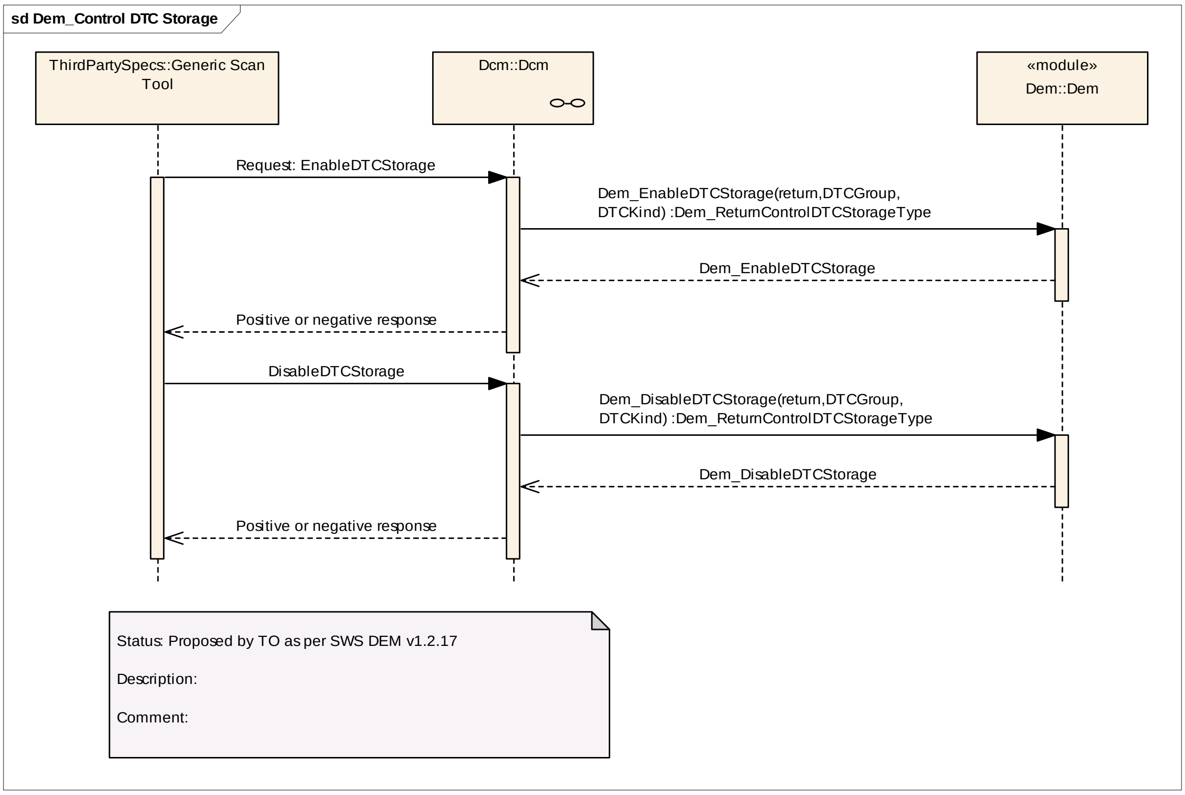
Configuration of Dem_MainFunction: The cyclic time for the main function has to be defined as an operating system task or run able entity.

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.

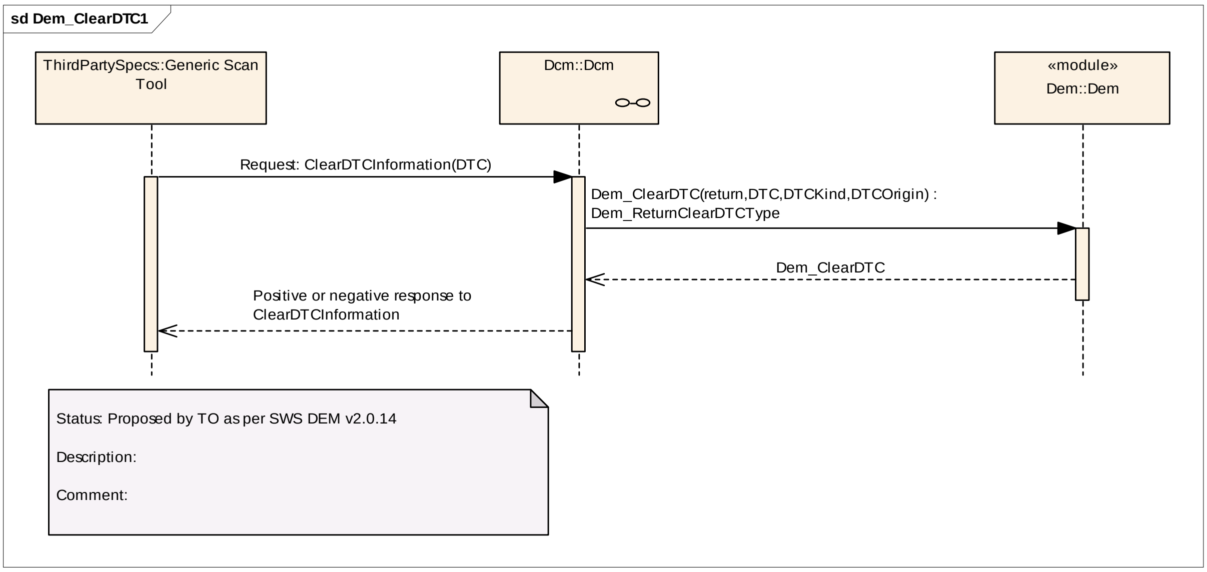
On pre condition: On pre-condition means that no cycle time can be defined. The function will be called when conditions are fulfilled. Alternatively, the function may be called cyclically however the cycle time will be assigned dynamically during runtime by other modules.

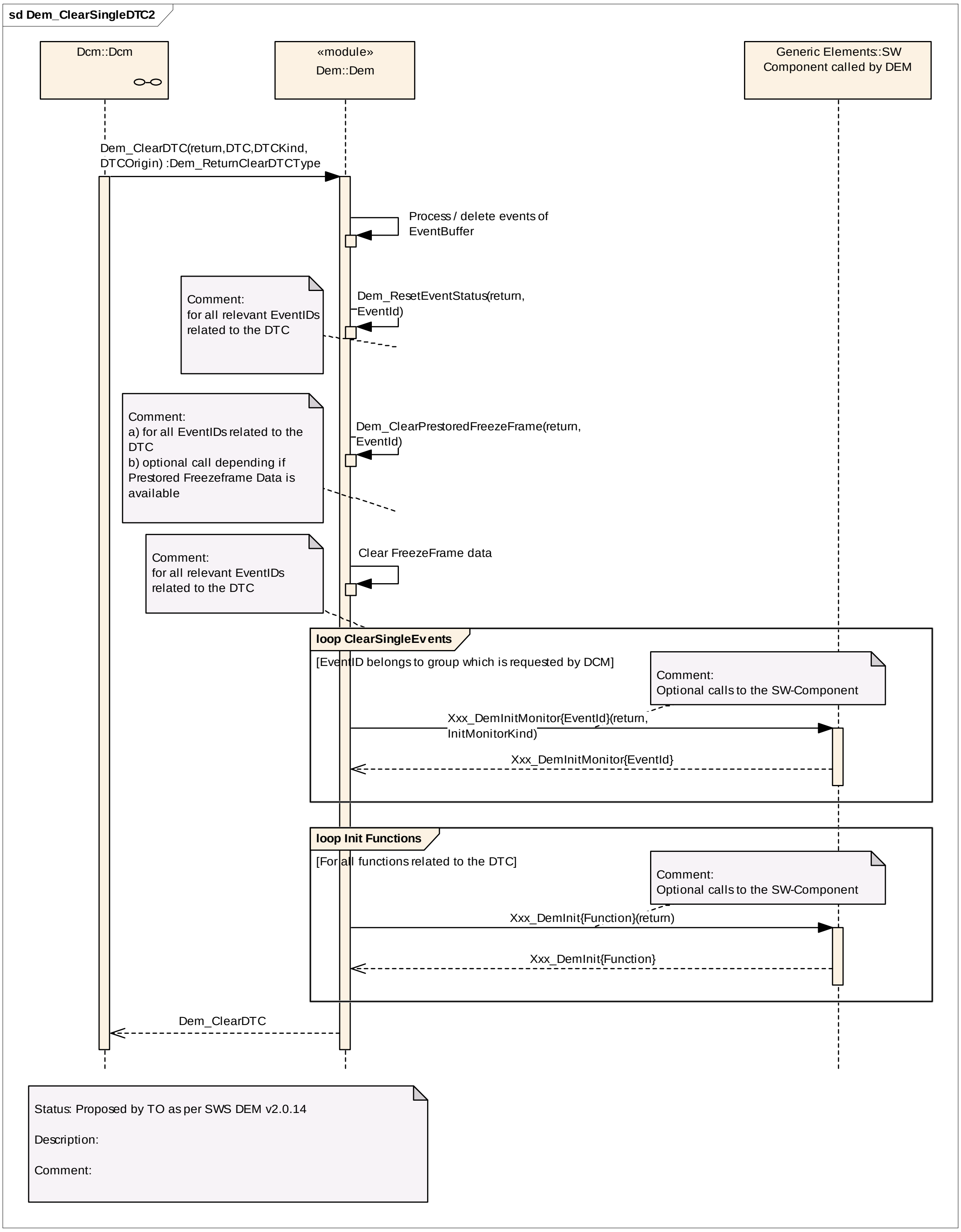
9 Sequence diagrams

9.1 ControlDTCStorage

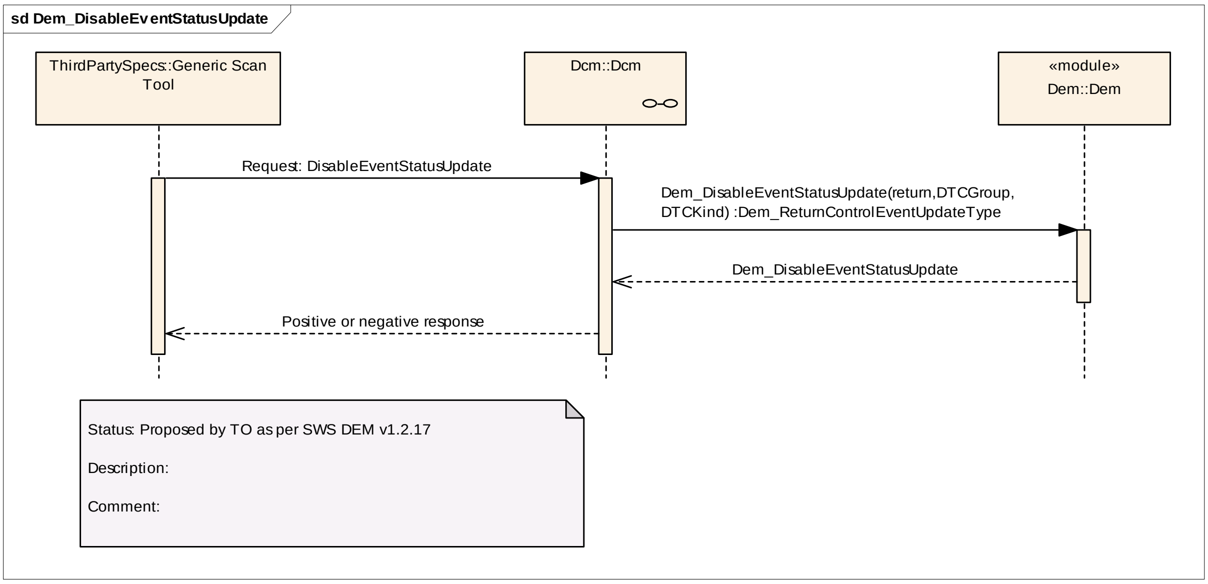

sd Dem_Control DTC Storage
ThirdPartySpecs::Generic Scan Tool
Dcm::Dcm
«module»
Dem::Dem
Request: EnableDTCStorage
Dem_EnableDTCStorage(return,DTCGroup, DTCKind) :Dem_ReturnControlDTCStorageType
Dem_EnableDTCStorage
Positive or negative response
DisableDTCStorage
Dem_DisableDTCStorage(return,DTCGroup, DTCKind) :Dem_ReturnControlDTCStorageType
Dem_DisableDTCStorage
Positive or negative response
Status: Proposed by TO as per SWS DEM v1.2.17
Description:
Comment:

9.2 Dem_ClearDTC

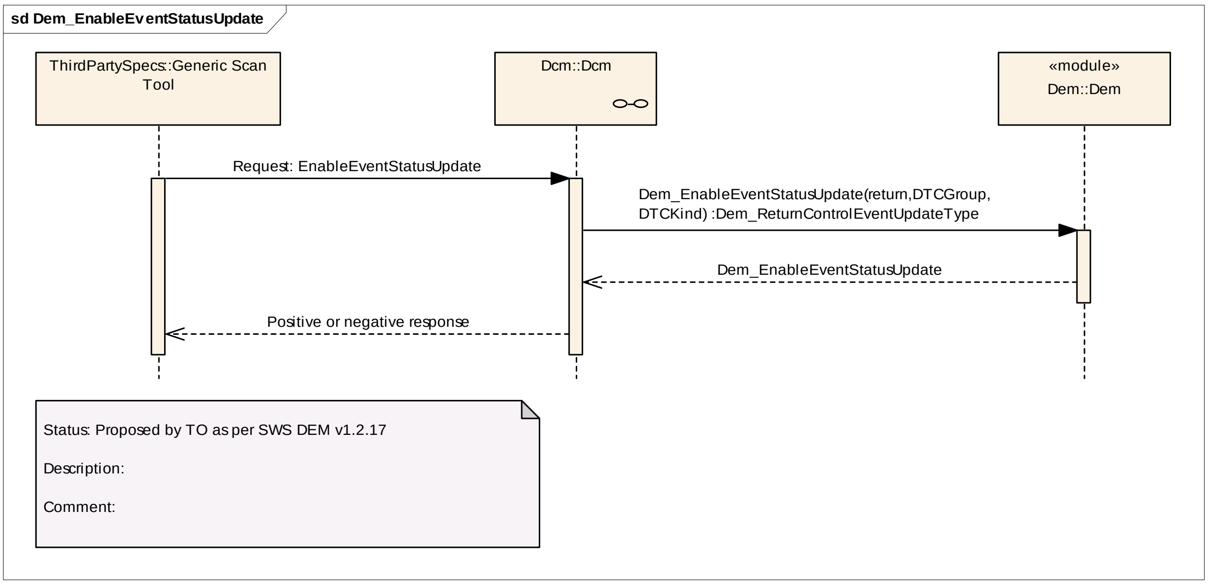

sd Dem_ClearDTC1
ThirdPartySpecs::Generic Scan Tool
Dcm::Dcm
«module»
Dem::Dem
Request: ClearDTCInformation(DTC)
Dem_ClearDTC(return,DTC,DTCKind,DTCOrigin) :
Dem_ReturnClearDTCType
Dem_ClearDTC
Positive or negative response to
ClearDTCInformation
Status: Proposed by TO as per SWS DEM v2.0.14
Description:
Comment:


sd Dem_ClearSingleDTC2
Dcm::Dcm
«module»
Dem::Dem
Generic Elements::SW Component called by DEM
Dem_ClearDTC(return,DTC,DTCKind, DTCOrigin) :Dem_ReturnClearDTCType
Process / delete events of EventBuffer
Comment: for all relevant EventIDs related to the DTC
Dem_ResetEventStatus(return, EventId)
Comment: a) for all EventIDs related to the DTC b) optional call depending if Prestored Freezeframe Data is available
Dem_ClearPrestoredFreezeFrame(return, EventId)
Comment: for all relevant EventIDs related to the DTC
Clear FreezeFrame data
loop ClearSingleEvents
[EventID belongs to group which is requested by DCM]
Comment: Optional calls to the SW-Component
Xxx_DemInitMonitor{EventId}(return, InitMonitorKind)
Xxx_DemInitMonitor{EventId}
loop Init Functions
[For all functions related to the DTC]
Comment: Optional calls to the SW-Component
Xxx_DemInit{Function}(return)
Xxx_DemInit{Function}
Dem_ClearDTC
Status: Proposed by TO as per SWS DEM v2.0.14
Description:
Comment:

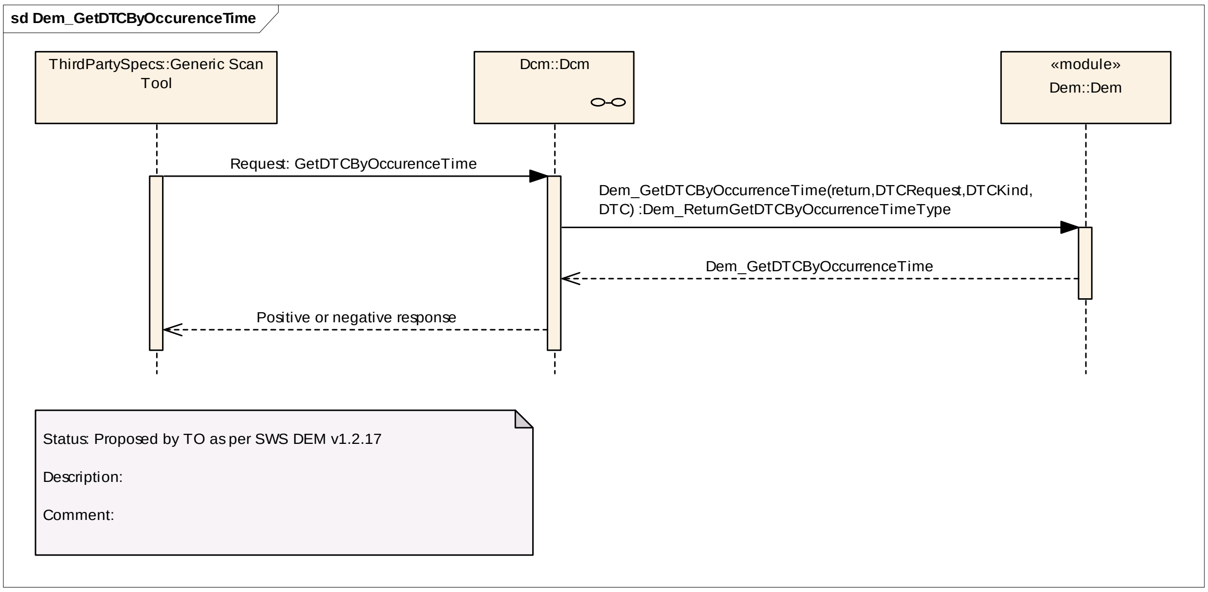
9.3 Dem_DisableEventStatusUpdate


sd Dem_DisableEventStatusUpdate
ThirdPartySpecs::Generic Scan Tool
Dcm::Dcm
«module»
Dem::Dem
Request: DisableEventStatusUpdate
Dem_DisableEventStatusUpdate(return,DTCGroup, DTCKind) :Dem_ReturnControlEventUpdateType
Dem_DisableEventStatusUpdate
Positive or negative response
Status: Proposed by TO as per SWS DEM v1.2.17
Description:
Comment:

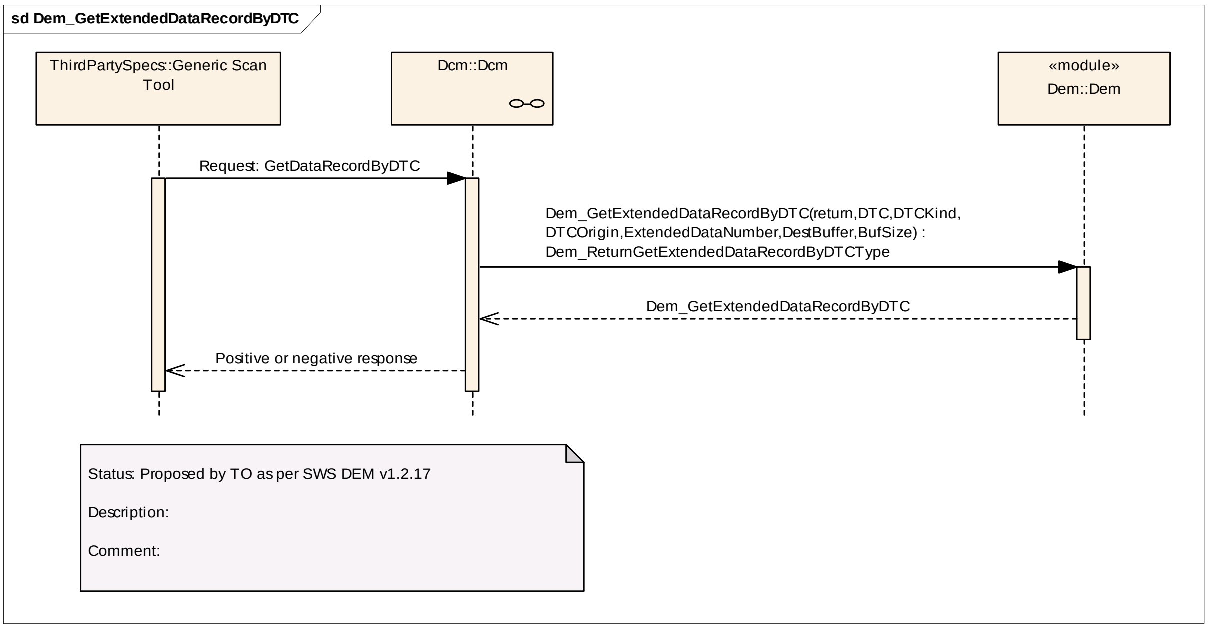
9.4 Dem_EnableEventStatusUpdate


sd Dem_EnableEventStatusUpdate
ThirdPartySpecs::Generic Scan Tool
Dcm::Dcm
«module»
Dem::Dem
Request: EnableEventStatusUpdate
Dem_EnableEventStatusUpdate(return,DTCGroup, DTCKind) :Dem_ReturnControlEventUpdateType
Dem_EnableEventStatusUpdate
Positive or negative response
Status: Proposed by TO as per SWS DEM v1.2.17
Description:
Comment:

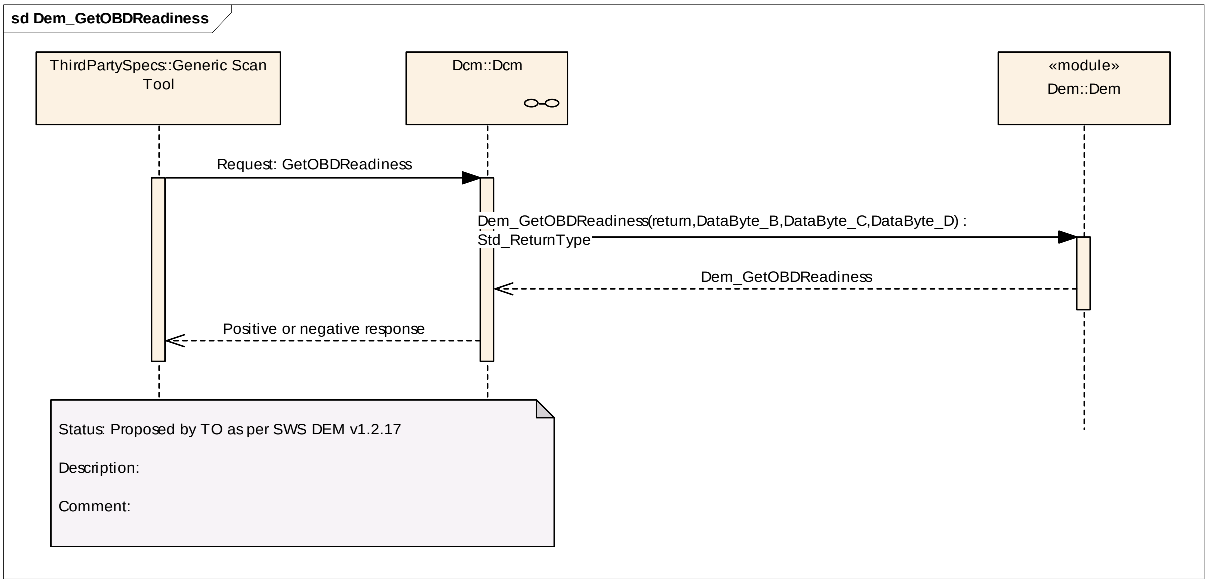
9.5 Dem_GetDTCByOccurrenceTime


sd Dem_GetDTCByOccurenceTime
ThirdPartySpecs::Generic Scan Tool
Dcm::Dcm
«module»
Dem::Dem
Request: GetDTCByOccurenceTime
Dem_GetDTCByOccurrenceTime(return,DTCRequest,DTCKind, DTC) :Dem_ReturnGetDTCByOccurrenceTimeType
Dem_GetDTCByOccurrenceTime
Positive or negative response
Status: Proposed by TO as per SWS DEM v1.2.17
Description:
Comment:

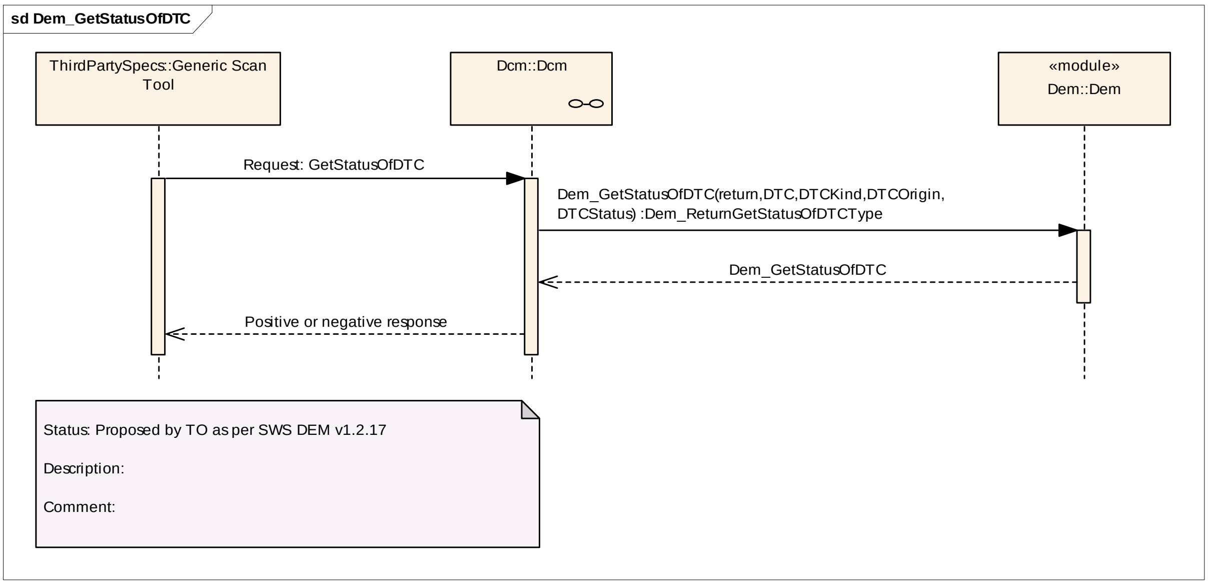
9.6 Dem_GetExtendedDataRecordByDTC


sd Dem_GetExtendedDataRecordByDTC
ThirdPartySpecs::Generic Scan Tool
Dcm::Dcm
«module»
Dem::Dem
Request: GetDataRecordByDTC
Dem_GetExtendedDataRecordByDTC(return,DTC,DTCKind, DTCOrigin,ExtendedDataNumber,DestBuffer,BufSize) : Dem_ReturnGetExtendedDataRecordByDTCType
Dem_GetExtendedDataRecordByDTC
Positive or negative response
Status: Proposed by TO as per SWS DEM v1.2.17
Description:
Comment:

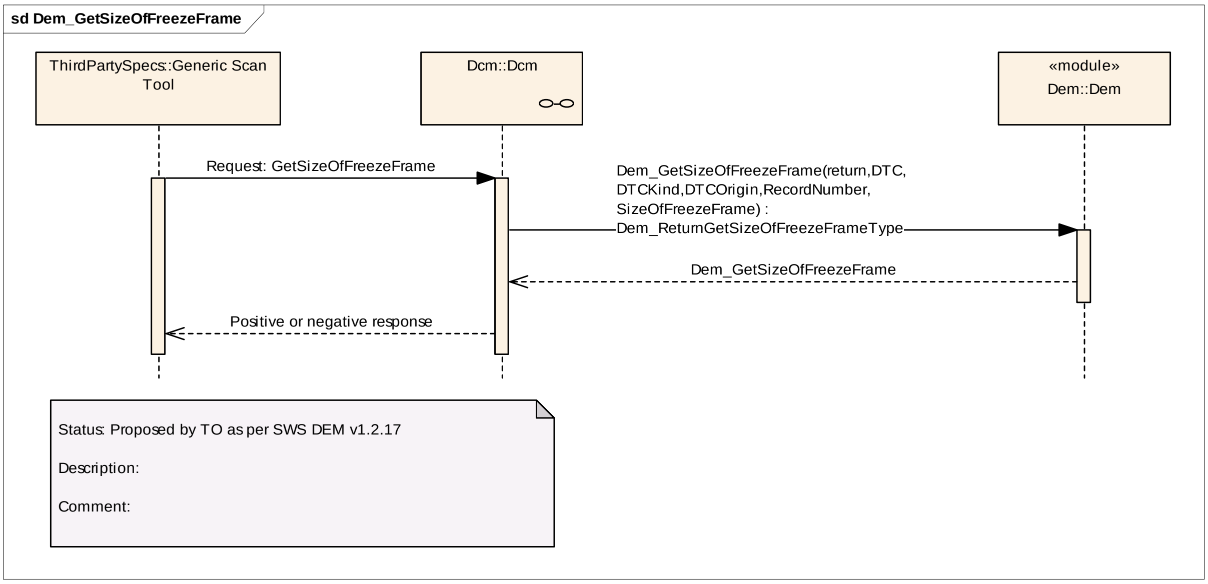
9.7 Dem_GetOBDReadiness


sd Dem_GetOBDReadiness
ThirdPartySpecs::Generic Scan Tool
Dcm::Dcm
«module»
Dem::Dem
Request: GetOBDReadiness
Dem_GetOBDReadiness(return,DataByte_B,DataByte_C,DataByte_D) :
Std_ReturnType
Dem_GetOBDReadiness
Positive or negative response
Status: Proposed by TO as per SWS DEM v1.2.17
Description:
Comment:

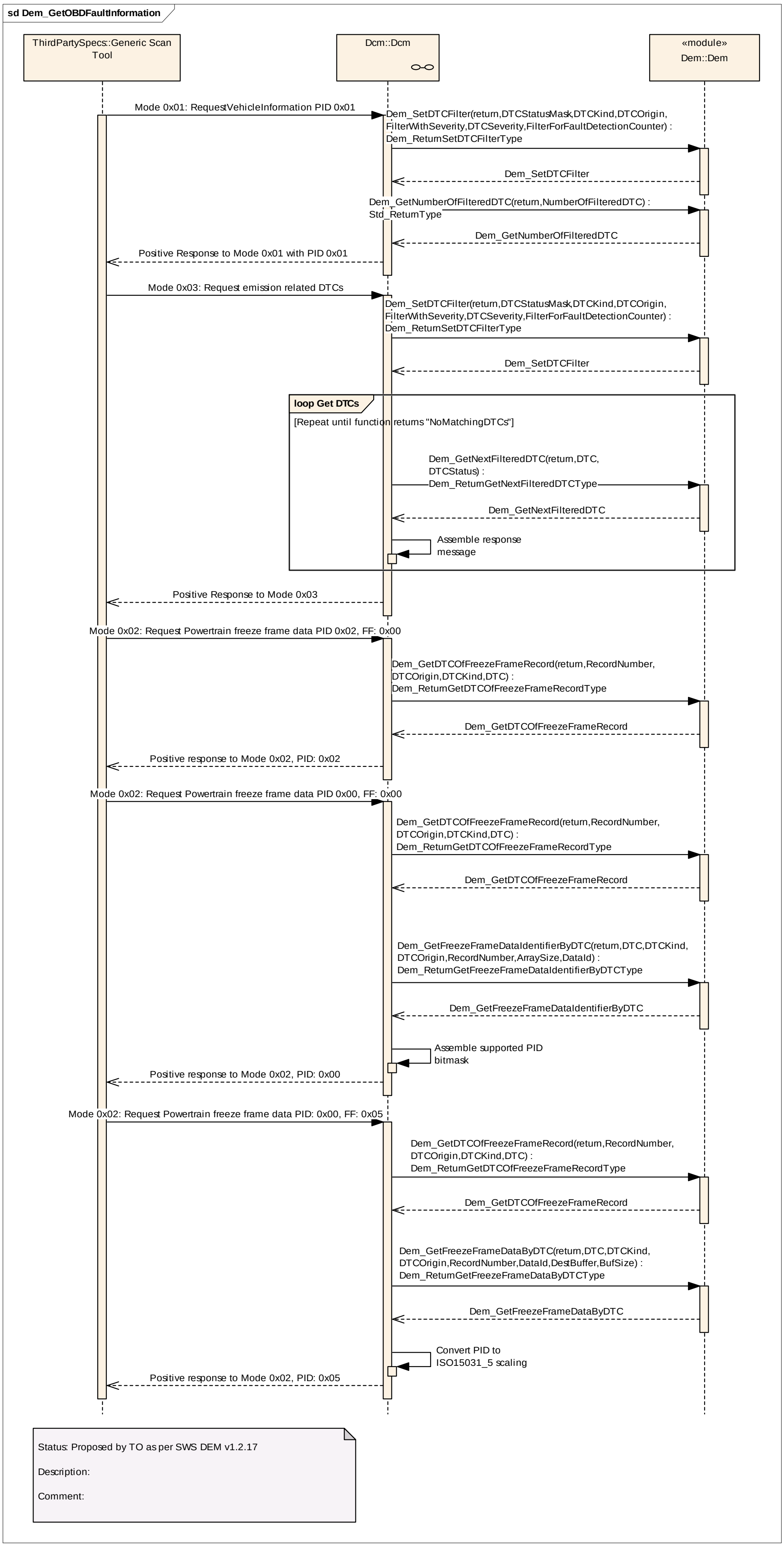
9.8 Dem_GetStatusOfDTC


sd Dem_GetStatusOfDTC
ThirdPartySpecs::Generic Scan Tool
Dcm::Dcm
«module»
Dem::Dem
Request: GetStatusOfDTC
Dem_GetStatusOfDTC(return,DTC,DTCKind,DTCOrigin,
DTCStatus) :Dem_ReturnGetStatusOfDTCType
Dem_GetStatusOfDTC
Positive or negative response
Status: Proposed by TO as per SWS DEM v1.2.17
Description:
Comment:

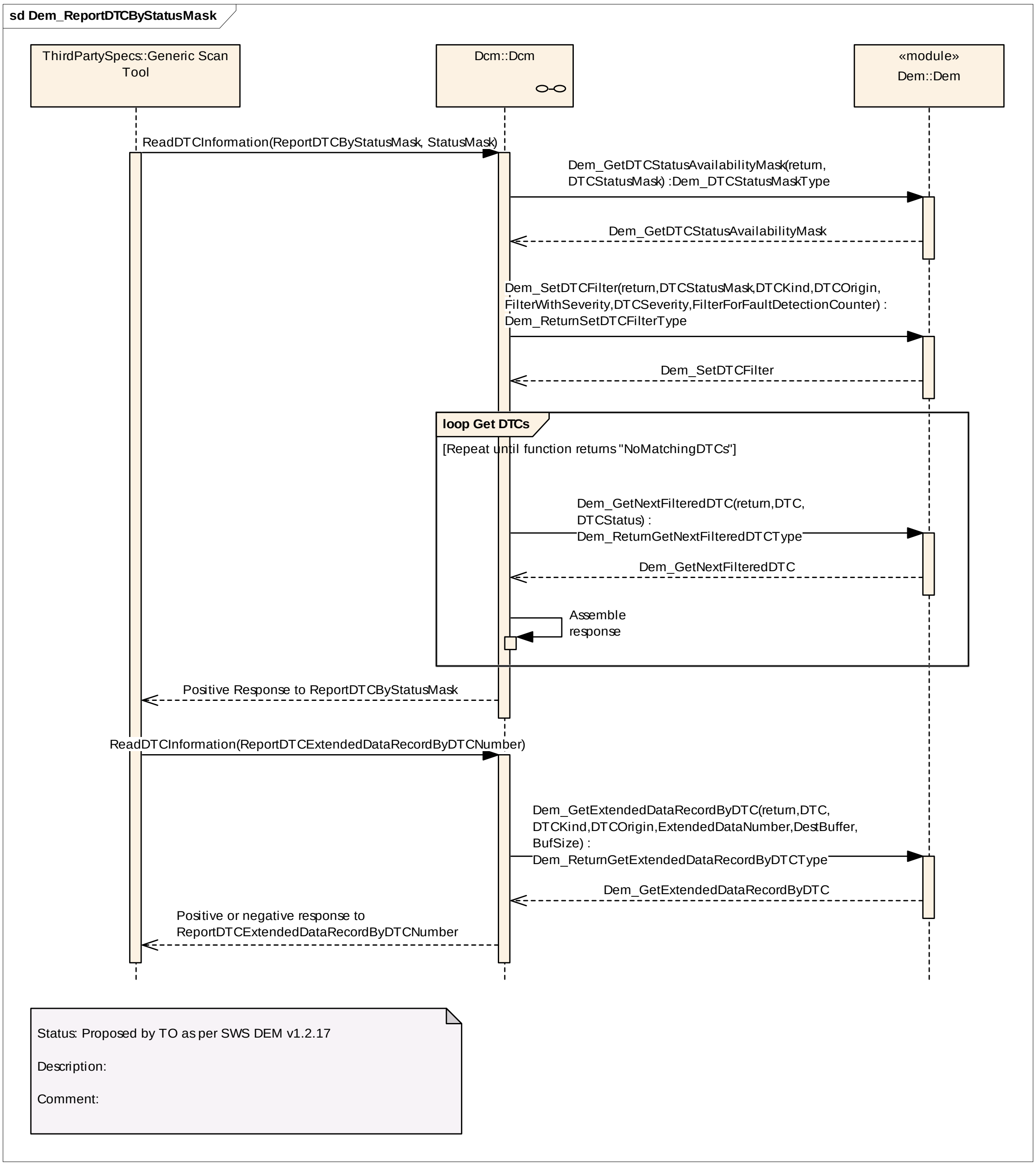
9.9 Dem_GetSizeOfFreezeFrame


sd Dem_GetSizeOfFreezeFrame
ThirdPartySpecs::Generic Scan Tool
Dcm::Dcm
«module»
Dem::Dem
Request: GetSizeOfFreezeFrame
Dem_GetSizeOfFreezeFrame(return,DTC,
DTCKind,DTCOrigin,RecordNumber,
SizeOfFreezeFrame) :
Dem_ReturnGetSizeOfFreezeFrameType
Dem_GetSizeOfFreezeFrame
Positive or negative response
Status: Proposed by TO as per SWS DEM v1.2.17
Description:
Comment:

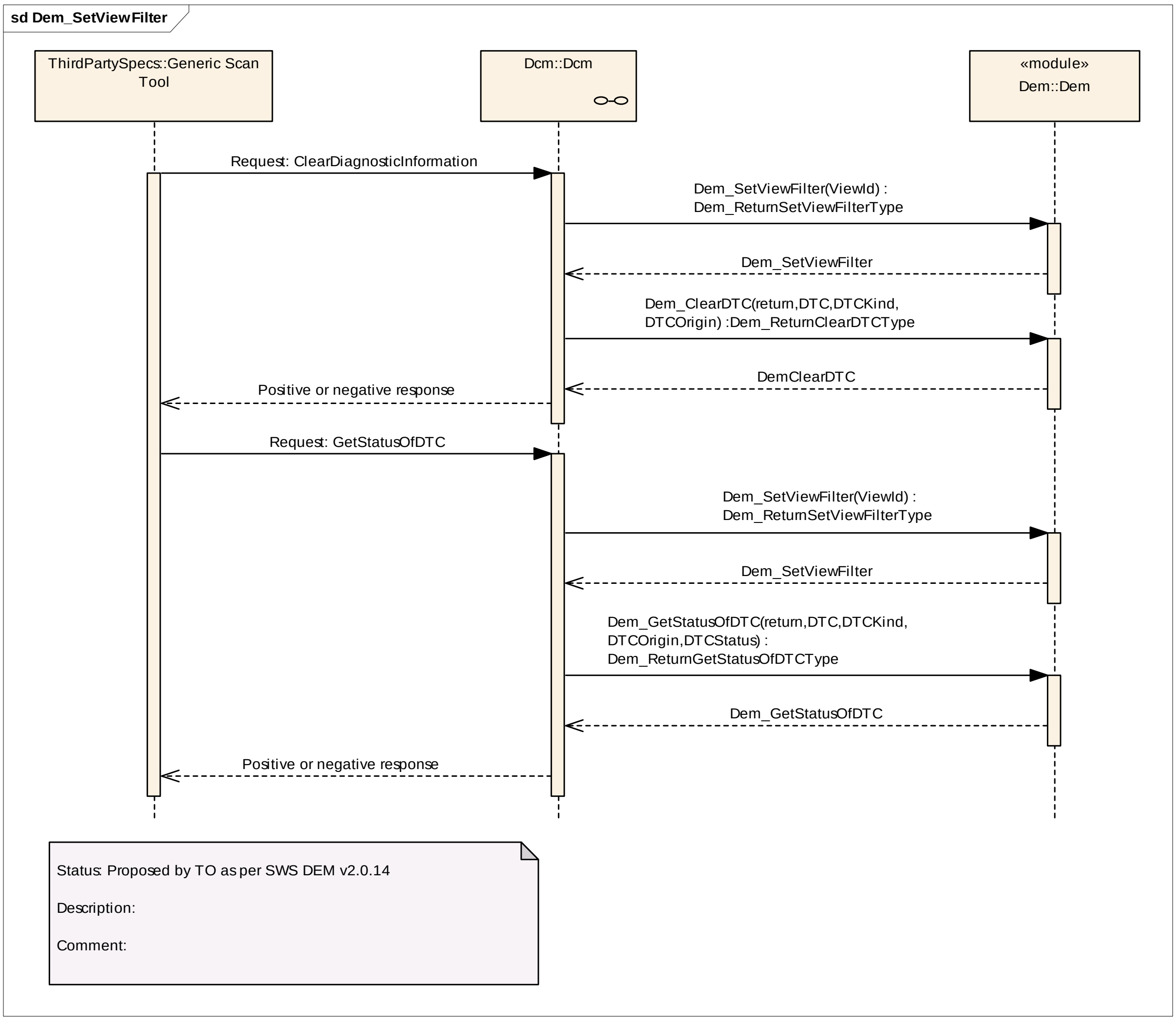
9.10 GetOBDFaultInformation


sd Dem_GetOBDFaultInformation
ThirdPartySpecs::Generic Scan Tool
Dcm::Dcm
«module»
Dem::Dem
Mode 0x01: RequestVehicleInformation PID 0x01
Dem_SetDTCFilter(return,DTCStatusMask,DTCKind,DTCOrigin, FilterWithSeverity,DTCSeverity,FilterForFaultDetectionCounter) : Dem_ReturnSetDTCFilterType
Dem_SetDTCFilter
Dem_GetNumberOfFilteredDTC(return,NumberOfFilteredDTC) : Std_ReturnType
Dem_GetNumberOfFilteredDTC
Positive Response to Mode 0x01 with PID 0x01
Mode 0x03: Request emission related DTCs
Dem_SetDTCFilter(return,DTCStatusMask,DTCKind,DTCOrigin, FilterWithSeverity,DTCSeverity,FilterForFaultDetectionCounter) : Dem_ReturnSetDTCFilterType
Dem_SetDTCFilter
loop Get DTCs
[Repeat until function returns "NoMatchingDTCs"]
Dem_GetNextFilteredDTC(return,DTC, DTCStatus) : Dem_ReturnGetNextFilteredDTCType
Dem_GetNextFilteredDTC
Assemble response message
Positive Response to Mode 0x03
Mode 0x02: Request Powertrain freeze frame data PID 0x02, FF: 0x00
Dem_GetDTCOfFreezeFrameRecord(return,RecordNumber, DTCOrigin,DTCKind,DTC) : Dem_ReturnGetDTCOfFreezeFrameRecordType
Dem_GetDTCOfFreezeFrameRecord
Positive response to Mode 0x02, PID: 0x02
Mode 0x02: Request Powertrain freeze frame data PID 0x00, FF: 0x00
Dem_GetDTCOfFreezeFrameRecord(return,RecordNumber, DTCOrigin,DTCKind,DTC) : Dem_ReturnGetDTCOfFreezeFrameRecordType
Dem_GetDTCOfFreezeFrameRecord
Dem_GetFreezeFrameDataIdentifierByDTC(return,DTC,DTCKind, DTCOrigin,RecordNumber,ArraySize,DataId) : Dem_ReturnGetFreezeFrameDataIdentifierByDTCType
Dem_GetFreezeFrameDataIdentifierByDTC
Assemble supported PID bitmask
Positive response to Mode 0x02, PID: 0x00
Mode 0x02: Request Powertrain freeze frame data PID: 0x00, FF: 0x05
Dem_GetDTCOfFreezeFrameRecord(return,RecordNumber, DTCOrigin,DTCKind,DTC) : Dem_ReturnGetDTCOfFreezeFrameRecordType
Dem_GetDTCOfFreezeFrameRecord
Dem_GetFreezeFrameDataByDTC(return,DTC,DTCKind, DTCOrigin,RecordNumber,DataId,DestBuffer,BufSize) : Dem_ReturnGetFreezeFrameDataByDTCType
Dem_GetFreezeFrameDataByDTC
Convert PID to ISO15031_5 scaling
Positive response to Mode 0x02, PID: 0x05
Status: Proposed by TO as per SWS DEM v1.2.17
Description:
Comment:

9.11 ReportDTCByStatusMask

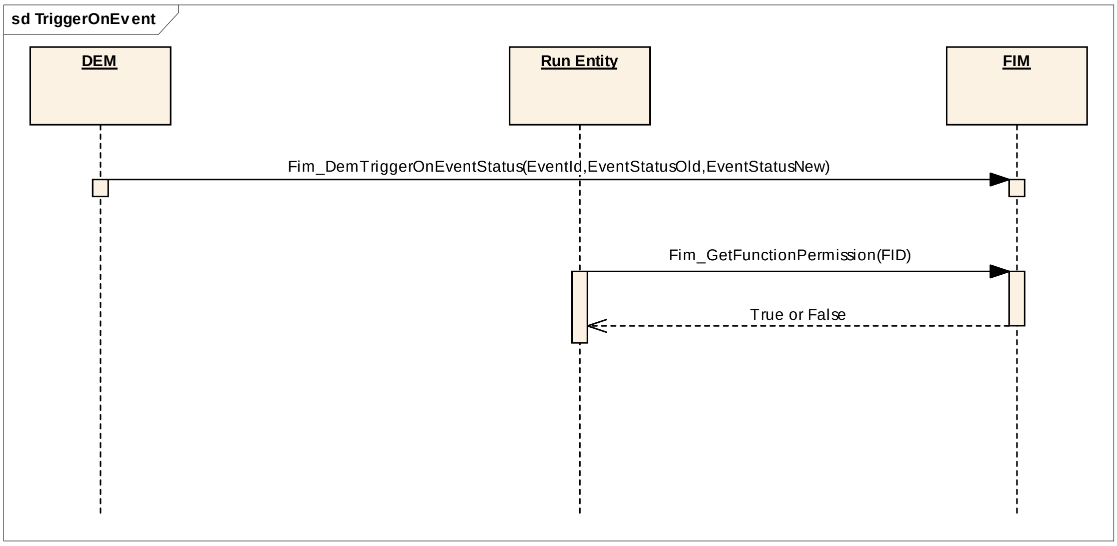

sd Dem_ReportDTCByStatusMask
ThirdPartySpecs::Generic Scan Tool
Dcm::Dcm
«module»
Dem::Dem
ReadDTCInformation(ReportDTCByStatusMask, StatusMask)
Dem_GetDTCStatusAvailabilityMask(return, DTCStatusMask) :Dem_DTCStatusMaskType
Dem_GetDTCStatusAvailabilityMask
Dem_SetDTCFilter(return,DTCStatusMask,DTCKind,DTCOrigin, FilterWithSeverity,DTCSeverity,FilterForFaultDetectionCounter) : Dem_ReturnSetDTCFilterType
Dem_SetDTCFilter
loop Get DTCs
[Repeat until function returns "NoMatchingDTCs"]
Dem_GetNextFilteredDTC(return,DTC, DTCStatus) : Dem_ReturnGetNextFilteredDTCType
Dem_GetNextFilteredDTC
Assemble response
Positive Response to ReportDTCByStatusMask
ReadDTCInformation(ReportDTCExtendedDataRecordByDTCNumber)
Dem_GetExtendedDataRecordByDTC(return,DTC, DTCKind,DTCOrigin,ExtendedDataNumber,DestBuffer, BufSize) : Dem_ReturnGetExtendedDataRecordByDTCType
Dem_GetExtendedDataRecordByDTC
Positive or negative response to ReportDTCExtendedDataRecordByDTCNumber
Status: Proposed by TO as per SWS DEM v1.2.17
Description:
Comment:

9.12 Dem_SetViewFilter

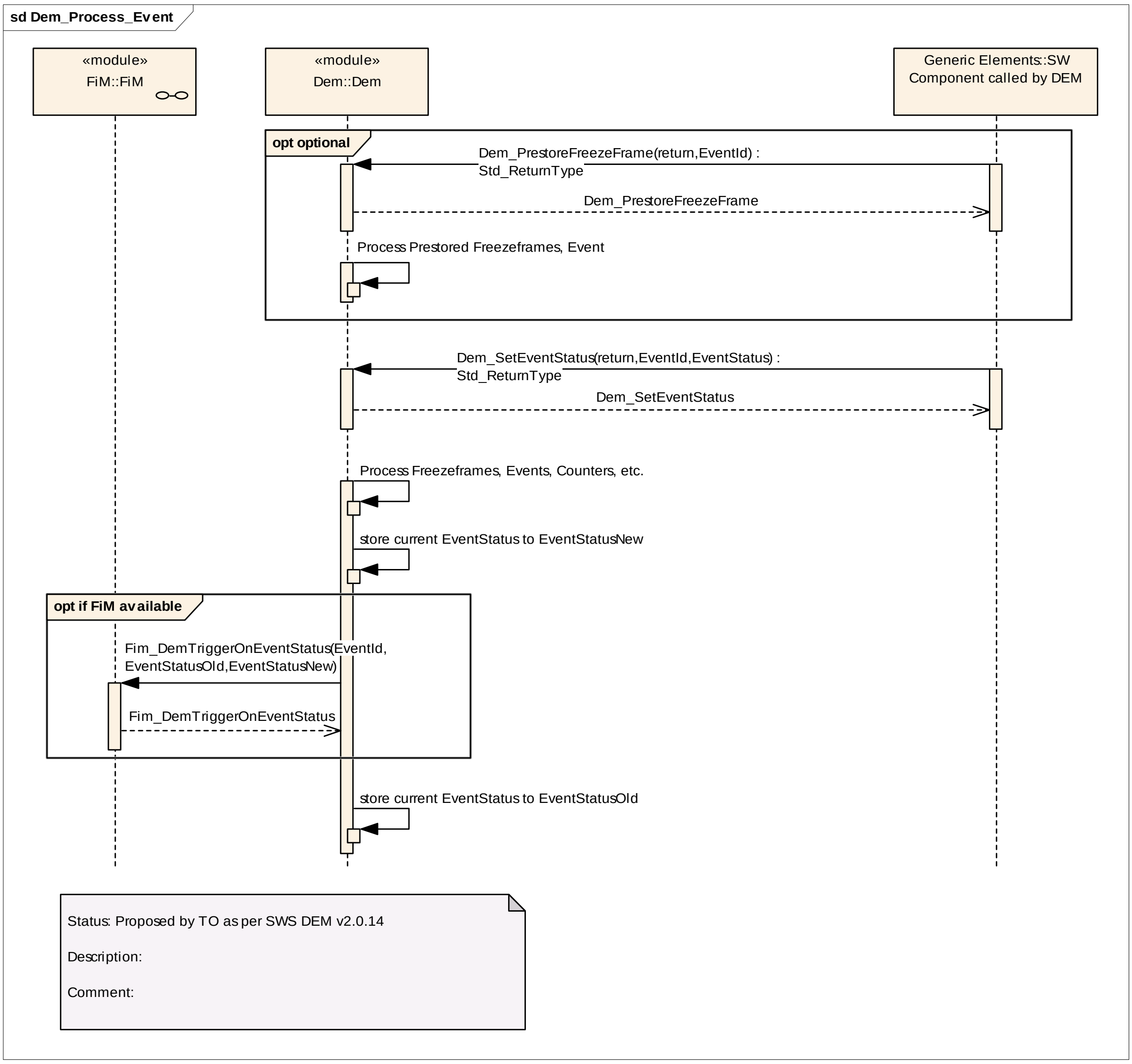

sd Dem_SetViewFilter
ThirdPartySpecs::Generic Scan Tool
Dcm::Dcm
«module»
Dem::Dem
Request: ClearDiagnosticInformation
Dem_SetViewFilter(ViewId) : Dem_ReturnSetViewFilterType
Dem_SetViewFilter
Dem_ClearDTC(return,DTC,DTCKind, DTCOrigin) :Dem_ReturnClearDTCType
DemClearDTC
Positive or negative response
Request: GetStatusOfDTC
Dem_SetViewFilter(ViewId) : Dem_ReturnSetViewFilterType
Dem_SetViewFilter
Dem_GetStatusOfDTC(return,DTC,DTCKind, DTCOrigin,DTCStatus) : Dem_ReturnGetStatusOfDTCType
Dem_GetStatusOfDTC
Positive or negative response
Status: Proposed by TO as per SWS DEM v2.0.14
Description:
Comment:

Note: this is only an example for the use of Dem_SetViewFilter

9.13 Fim_DemTriggerOnEventStatus


sd TriggerOnEvent
DEM
Run Entity
FIM
Fim_DemTriggerOnEventStatus(EventId,EventStatusOld,EventStatusNew)
Fim_GetFunctionPermission(FID)
True or False

9.14 ProcessEvent (Example)


sd Dem_Process_Event
«module»
FiM::FiM
«module»
Dem::Dem
Generic Elements::SW Component called by DEM
opt optional
Dem_PrestoreFreezeFrame(return,EventId) : Std_ReturnType
Dem_PrestoreFreezeFrame
Process Prestored Freezeframes, Event
Dem_SetEventStatus(return,EventId,EventStatus) : Std_ReturnType
Dem_SetEventStatus
Process Freezeframes, Events, Counters, etc.
store current EventStatus to EventStatusNew
opt if FiM available
Fim_DemTriggerOnEventStatus(EventId, EventStatusOld,EventStatusNew)
Fim_DemTriggerOnEventStatus
store current EventStatus to EventStatusOld
Status: Proposed by TO as per SWS DEM v2.0.14
Description:
Comment:

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	DemMaxNumberEventEntryMir {DEM_MAX_NUMBER_EVENT_ENTRY_MIR}	
Description	Maximum number of events which can be stored in the mirror memory (typically up to 30) (ref. to example ch. 7)	
Multiplicity	1	
Type	IntegerParamDef	
Range	0 .. 255	

Default value	0		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DemTypeOfOriginSupported, if mirror memory is not available, then set to 0		

SWS Item	**Dem128 :**		
Name	DemMaxNumberEventEntryPrm {DEM_MAX_NUMBER_EVENT_ENTRY_PRM}		
Description	Maximum number of events which can be stored in the primary memory (typically up to 20) (ref. to example ch. 7)		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	30		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem128 :**		
Name	DemMaxNumberEventEntrySec {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	0		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DemTypeOfOriginSupported, if secondary memory is not available, then set to 0		

SWS Item	**Dem128 :**		
Name	DemMaxNumberPrestoredFF {DEM_MAX_NUMBER_PRESTORED_FF}		
Description	Defines the maximum number for prestored freeze frames. 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	**Dem128 :**

Name	DemNvmAccessRetry {DEM_NVM_ACCESS_RETRY}		
Description	Maximum number of retries to access NVRAM Manager.		
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	DemNvmAccessRetryTimeDelay {DEM_NVM_ACCESS_RETRY_TIMEDELAY}		
Description	Time in milliseconds between two retries to NVRAM Manager in case that first access failed.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
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	DEM_TYPE_OF_DTC_SUPPORTED is defined in the DEM SWS as uint8. However, it is used in a way similar to an enum. Bit 0: 2 byte ISO15031-6 DTC Bit 1: 3 byte ISO14229-1 DTC Bit 2: Customer specific DTC Bit 3: SAEJ1939 Bit 4: WWH-OBD-format Set the according Bit to '1' means DTC format is supported. Combination of different DTC formats is possible (e.g. ISO 15031-6 and ISO14229-1 is coded by 0x0011b).
Multiplicity	1
Type	IntegerParamDef

Range	0 .. 31		
Default value	--		
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..*	This container contains the configuaration (parameters) for Enable Conditions.
DemExtendedDataClass	0..*	This class contains the combinations of extended data records (ExtendedDataClassRec).
DemExtendedDataRecClass	0..253	This container contains the configuration (parameters) for ExtendedDataClassRecords
DemFreezeFrameClass	0..255	This container contains the configuration (parameters) for FreezeFrameClass.
DemFreezeFrameIdClass	0..255	This container contains the configuration (parameters) for FreezeFrameClass.
DemGetDataValByDataId	0..*	This container contains the configuaration (parameters) for GetDataValueByDataIdentifier functions.
DemGetExtDataRecord	0..*	This container contains the configuration (parameters) for GetExtendedDataRecord functions.
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.
DemInitFunction	0..*	This container contains the configuaration (parameters) for Xxx_DemInit{Function}
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.
DemTriggerOnDTCStatusTgt	0..*	This container contains the configuaration (parameters) for DemTriggerOnDTCStatus functions.
DemTriggerOnEventStatusTgt	0..*	This container contains the configuaration (parameters) for DemTriggerOnEventStatus functions.

DemView	0..255	This container contains the configuaration (parameters) for Views. 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 VIEW_NAME.

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 name (readability of code) and IndicatorID 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 DemView

SWS Item	**Dem138 :**
Container Name	DemView{VIEW}
Description	This container contains the configuaration (parameters) for Views. 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 VIEW_NAME.
Configuration Parameters	

SWS Item	**Dem138 :**	
Name	DemViewID {ViewID}	
Description	Unique Identifier of a View. Implementation Type: uint8	
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

10.2.6 DemTriggerOnEventStatus

SWS Item	**Dem140 :**
Container Name	DemTriggerOnEventStatus{TriggerOnEventStatus}
Description	This container contains the configuration for TriggerOnEvent functions.
Configuration Parameters	

No Included Containers

10.2.7 DemGetExtDataRecord

SWS Item	**Dem139 :**
Container Name	DemGetExtDataRecord{Xxx_DemGetExtendedDataRecord}
Description	This container contains the configuration (parameters) for GetExtendedDataRecord functions.
Configuration Parameters	

SWS Item	**Dem139 :**		
Name	DemGetExtDataRecordFnc {PREFIX_GetExtendedDataRecord}		
Description	DemGetExtDataRecord function name.		
Multiplicity	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 DemGetDataValByDataId

SWS Item	**Dem139 :**
Container Name	DemGetDataValByDataId{Xxx_DemGetDataValueByDataIdentifier}
Description	This container contains the configuaration (parameters) for GetDataValueByDataIdentifier functions.
Configuration Parameters	

SWS Item	**Dem139 :**
Name	DemGetDataValByDataIdListFnc
Description	DemGetDataValByDataIdList function name.

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

No Included Containers

10.2.9 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	**Dem130 :**		
Name	DemInitMonitorName {InitMonitorName}		
Description	Monitor function which has to be initialized for the event (ref. to Xxx_DemInitMonitor(EventId)		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem132 :**
Name	DemDTCClassRef {DTCClass}

Description	This container contains the configuration (parameters) for DTCClass.		
Multiplicity	1..2		
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	This container contains the configuration (parameters) for FreezeFrameClass.		
Multiplicity	0..255		
Type	Reference to DemFreezeFrameClass		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem140 :**		
Name	DemTriggerOnEventStatusTargetRef		
Description	Reference to Xxx_DemTriggerOnEventStatus function. The possible selection depends on Xxx_DemTriggerOnEventStatusList.		
Multiplicity	1..*		
Type	Reference to DemTriggerOnEventStatusTgt		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemEventClass	1	This container contains the configuaration (parameters) for EventClass
DemTriggerOnEventStatus	0..1	This container contains the configuration for TriggerOnEvent functions.

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 DemExtendedDataRecClass		
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	**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..3		
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.	
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 freeze frames 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..1		
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	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		

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		

SWS Item	**Dem130 :**		
Name	DemSupportedViewName {SUPPORTED_VIEW_NAME}		
Description	view name of supported view A view describes a functional group like a wiper system or a window lifter for the access of corresponding DTCs and related data. SUPPORTED_VIEW_NAME selects a view in which the event is visible. Example: WIPERSYSTEM refers to functionality wiper system, WINDOWLIFTER refers to functionality window lifter, ... This parameter is optional.		
Multiplicity	1		
Type	Reference to DemView		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
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	1..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 Bit 0: Indicator is active if event in status FAILED Bit 1: Indicator is blinking if event in status FAILED		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 7		
Default value	--		
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	0 .. 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	Kind of DTC (OBD relevant or not) ImplementationType: Dem_DTCKindType		
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		

SWS Item	**Dem140 :**		
Name	DemTriggerOnDTCStatusTargetRef		
Description	Reference to Xxx_DemTriggerOnEventStatus function. The possible selection depends on Xxx_DemTriggerOnEventStatusList.		
Multiplicity	1..*		
Type	Reference to DemTriggerOnDTCStatusTgt		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemInit	0..1	This container contains the configuration (parameters) for "Xxx_DemInit{Function} calls at Dem_ClearSingleDTC2Generic

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	For OBD relevant data Multiple PIDs can be relevant per Freezeframe. This parameter is optional!		
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 DemFreezeFrameIdClass		
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	**Dem146 :**		
Name	DemGetFaultDetectionCntFnc {Prefix_DemGetFaultDetectionCounter}		
Description	Defines a real name of an API assigned to the monitoring path. This name shall be used by a code generator as function name.		
Multiplicity	1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

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		

No Included Containers

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 configuaration (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		
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		

No Included Containers

10.2.26 DemInit

SWS Item	**--**
Container Name	DemInit{DEM_INIT_FUNC}
Description	This container contains the configuration (parameters) for "Xxx_DemInit{Function} calls at Dem_ClearSingleDTC2Generic
Configuration Parameters	

SWS Item	**--**		
Name	DemInitListTargetRef		
Description	Refernce to "Xxx_DemInit{Function} called at Dem_ClearSingleDTC2Generic		
Multiplicity	1..*		
Type	Reference to DemInitFunction		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.27 DemInitFunction

SWS Item	**Dem139 :**
Container Name	DemInitFunction{Xxx_DemInitFunction}
Description	This container contains the configuaration (parameters) for Xxx_DemInit{Function}
Configuration Parameters	

SWS Item	**Dem130 :**		
Name	DemInitFunctionName {InitMonitorName}		
Description	"Xxx_DemInit{Function} which has to be called by DEM or other API to reset the {Function} of the Module Xxx		
Multiplicity	1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

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

No Included Containers

10.2.28 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 DemFreezeFrameIdClass

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

SWS Item	**Dem136 :**		
Name	DemDataID {DataId}		
Description	For enhanced diagnostics Multiple DataIDs can be relevant per FreezeFrame. This parameter is optional!		
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	DemFreezeFrameIdDataSize {DID}		
Description	For OBD relevant data Multiple PIDs can be relevant per Freezeframe. This parameter is optional!		
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	**Dem136 :**		
Name	DemPID {PID}		
Description	For OBD relevant data Multiple PIDs can be relevant per Freezeframe. This parameter is optional!		
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		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemGetDataValByDataIdTgt	1	This container contains the configuration for GetDataValueByDataIdentifierList.

10.2.30 DemExtendedDataRecClass

SWS Item	**Dem135 :**
Container Name	DemExtendedDataRecClass{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		

SWS Item	**Dem140 :**		
Name	DemGetExtendedDataTargetRef		
Description	Reference to Xxx_GetExtendedData function. The possible selection depends on XxxDemGetExtDataRecordList.		
Multiplicity	1		
Type	Reference to DemGetExtDataRecord		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.31 DemTriggerOnEventStatusTgt

SWS Item	**Dem139 :**
Container Name	DemTriggerOnEventStatusTgt{Xxx_DemTriggerOnEventStatus}
Description	This container contains the configuaration (parameters) for DemTriggerOnEventStatus functions.
Configuration Parameters	

SWS Item	**Dem139 :**		
Name	DemTrigOnEventStatusFnc {PREFIX_DemTriggerOnEventStatusList}		
Description	DemTriggerOnEventStatus function name.		
Multiplicity	1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.32 DemTriggerOnDTCStatusTgt

SWS Item	**Dem139 :**
Container Name	DemTriggerOnDTCStatusTgt{Xxx_DemTriggerOnEventStatus}
Description	This container contains the configuaration (parameters) for DemTriggerOnDTCStatus functions.
Configuration Parameters	

SWS Item	**Dem139 :**		
Name	DemTrigOnDTCStatusFnc {PREFIX_DemTriggerOnEventStatusList}		
Description	DemTriggerOnDTCStatus function name.		
Multiplicity	1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.3 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.

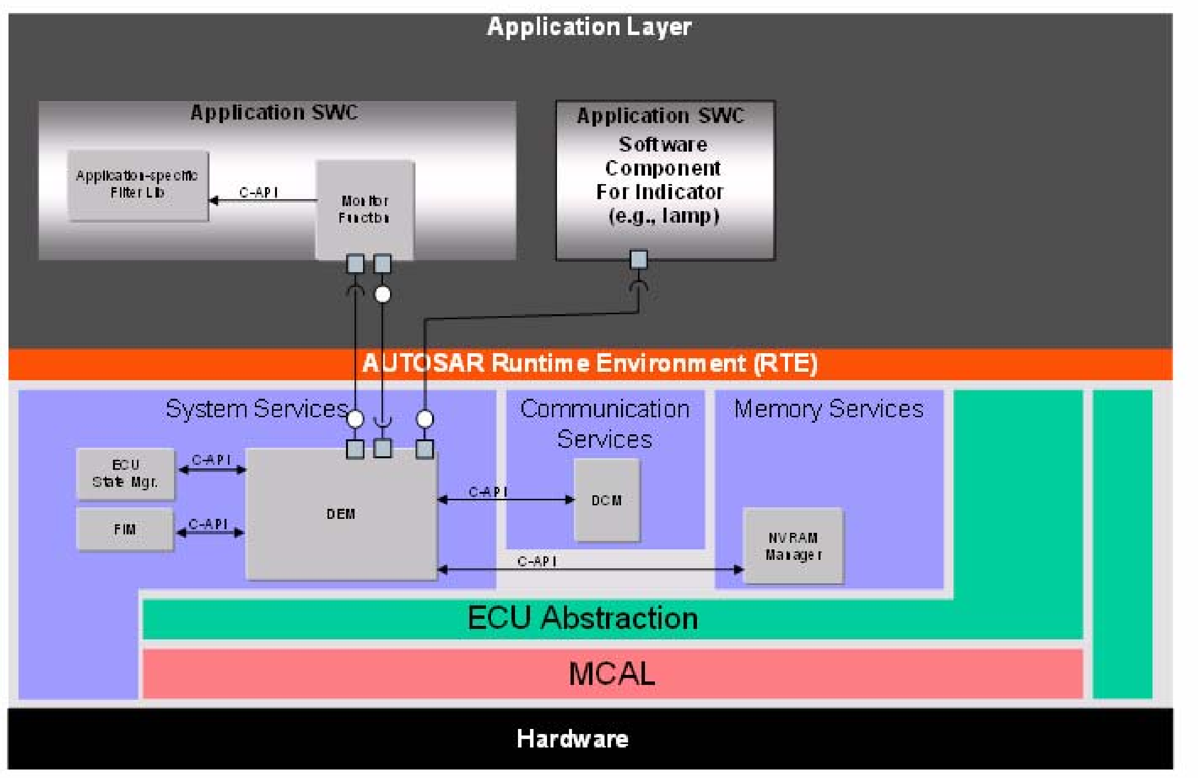


Figure 1: DEM in the ECU software architecture.

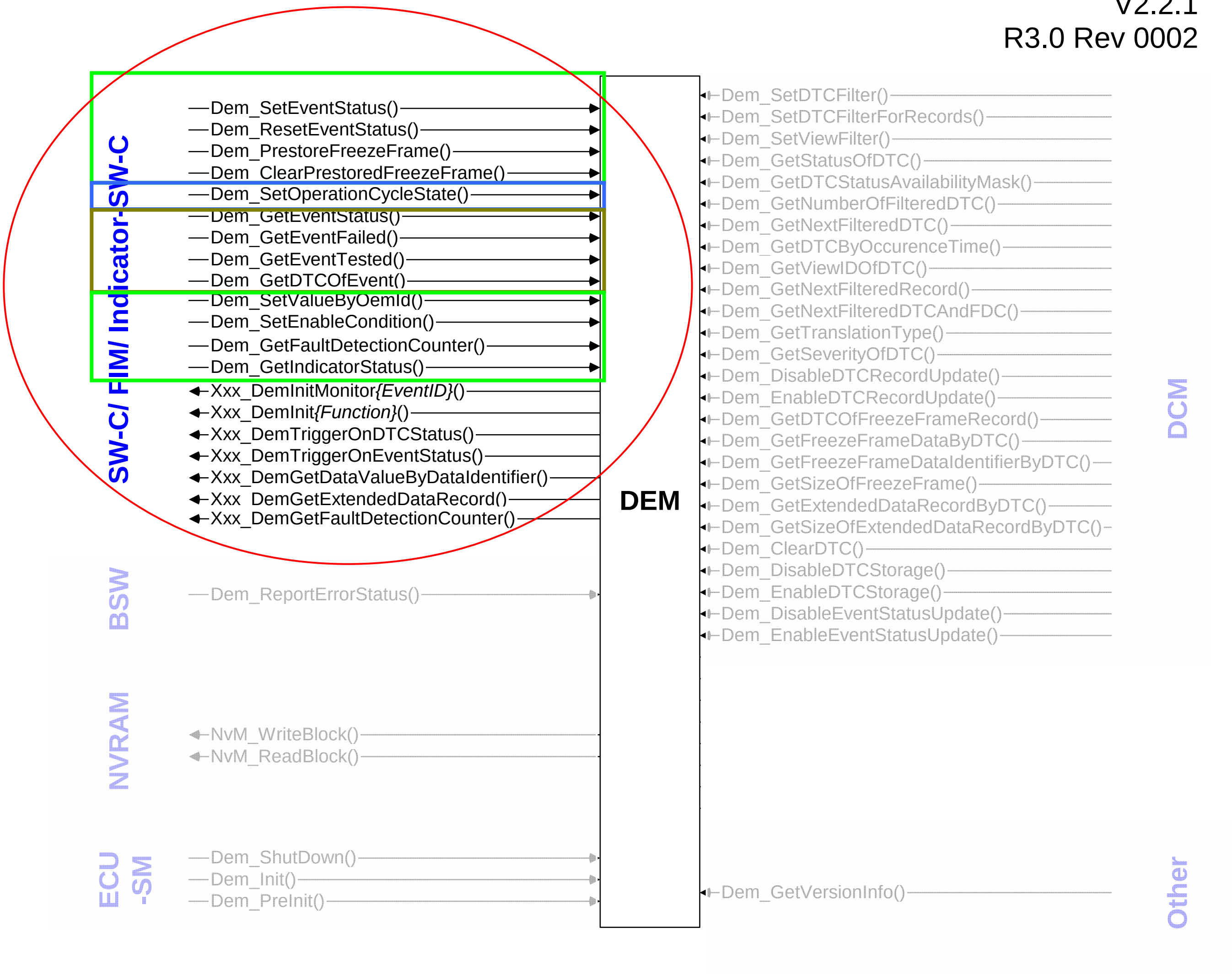

SW-C/ FIM/ Indicator-SW-C
Dem_SetEventStatus()
Dem_ResetEventStatus()
Dem_PrestoreFreezeFrame()
Dem_ClearPrestoredFreezeFrame()
Dem_SetOperationCycleState()
Dem_GetEventStatus()
Dem_GetEventFailed()
Dem_GetEventTested()
Dem_GetDTCOfEvent()
Dem_SetValueByOemId()
Dem_SetEnableCondition()
Dem_GetFaultDetectionCounter()
Dem_GetIndicatorStatus()
Xxx_DemInitMonitor{EventID}()
Xxx_DemInit{Function}()
Xxx_DemTriggerOnDTCStatus()
Xxx_DemTriggerOnEventStatus()
Xxx_DemGetDataValueByDataIdentifier()
Xxx_DemGetExtendedDataRecord()
Xxx_DemGetFaultDetectionCounter()
BSW
Dem_ReportErrorStatus()
NVRAM
NvM_WriteBlock()
NvM_ReadBlock()
ECU -SM
Dem_ShutDown()
Dem_Init()
Dem_PreInit()
DEM
Dem_SetDTCFilter()
Dem_SetDTCFilterForRecords()
Dem_SetViewFilter()
Dem_GetStatusOfDTC()
Dem_GetDTCStatusAvailabilityMask()
Dem_GetNumberOfFilteredDTC()
Dem_GetNextFilteredDTC()
Dem_GetDTCByOccurenceTime()
Dem_GetViewIDOfDTC()
Dem_GetNextFilteredRecord()
Dem_GetNextFilteredDTCAndFDC()
Dem_GetTranslationType()
Dem_GetSeverityOfDTC()
Dem_DisableDTCRecordUpdate()
Dem_EnableDTCRecordUpdate()
Dem_GetDTCOfFreezeFrameRecord()
Dem_GetFreezeFrameDataByDTC()
Dem_GetFreezeFrameDataIdentifierByDTC()
Dem_GetSizeOfFreezeFrame()
Dem_GetExtendedDataRecordByDTC()
Dem_GetSizeOfExtendedDataRecordByDTC()
Dem_ClearDTC()
Dem_DisableDTCStorage()
Dem_EnableDTCStorage()
Dem_DisableEventStatusUpdate()
Dem_EnableEventStatusUpdate()
DCM
Dem_GetVersionInfo()
Other

Figure 2: Communication relationships of the DEM. The red circle indicates RTE-relevant communication

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 this document above.

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

A DEM 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

Type Name	Definition	Used in
Dem_EventStatusType	UInt8 (DEM_EVENT_STATUS_PASSED, DEM_EVENT_STATUS_FAILED, DEM_EVENT_STATUS_PREPASSED, DEM_EVENT_STATUS_PREFAILED, DEM_EVENT_STATUS_<Custom>)	Dem_SetEventStatus
Dem_EventStatusExtended-	Uint8	Dem_GetEventStatus,

Type		<Xxx>_DemTriggerOn-EventStatus <Xxx>_DemTriggerOnDTC-Status
Dem_DTCKindType	UInt8 (DEM_DTC_KIND_ALL_DTCS, DEM_DTC_KIND_EMISSION_REL_DTCS)	Dem_GetDTCOfEvent
Dem_DTCType	UInt32	Dem_GetDTCOfEvent <Xxx>_DemTriggerOnDTC-Status
Dem_ReturnGetDTCOfEvent-Type	UInt8 (DEM_GET_DTCOFEVENT_OK, DEM_GET_DTCOFEVENT_WRONG_EVENTID, DEM_GET_DTCOFEVENT_WRONG_TRANSLATION)	Dem_GetDTCOfEvent
Dem_InitMonitorKindType	UInt8 (DEM_INIT_MONITOR_CLEAR, DEM_INIT_MONITOR_RESTART)	Dem_InitMonitorForEvent
Dem_DTCStatusMaskType	Uint8	Dem_DTCStatus-Changed
Dem_OperationCycleIdType	Uint8	Dem_SetOperationCycle-State
Dem_OperationCycleState-Type	UInt8 (DEM_CYCLE_STATE_START, DEM_CYCLE_STATE_END)	Dem_SetOperationCycle-State
Dem_FaultDetectionCounter-Type	SInt8	Dem_GetFaultDetection-Counter <Xxx>_DemGetFault-DetectionCounter
Dem_IndicatorStatusType	…	Dem_GetIndicatorStatus

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_ReturnGetDTCOfEventType {
      LOWER-LIMIT=0;
      UPPER-LIMIT=2;
      0 -> DEM_GET_DTCOFEVENT_OK
      1 -> DEM_GET_DTCOFEVENT_WRONG_EVENTID
      2 -> DEM_GET_DTCOFEVENT_WRONG_DTCKIND
}

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
}

ArrayType Dem_MaxExtendedDataRecordType {
      ELEMENT-TYPE-REF=UInt8;
      MAX-NUMBER-OF-ELEMENTS=[size of largest Record];
}

ArrayType Dem_MaxDataValueType {
      ELEMENT-TYPE-REF=UInt8;
      MAX-NUMBER-OF-ELEMENTS=[size of largest DID];
}
```

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-C and the DEM. 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.

```
ClientServerInterface DiagnosticMonitor {
    PossibleErrors {
        E_NOT_OK = 1
    }

    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 Dem_DTCType DTC,
        OUT Dem_ReturnGetDTCOfEvent StatusDTCOfEvent, ERR{E_NOT_OK});
    PrestoreFreezeFrame(ERR{E_NOT_OK}); // OPTIONAL, only if DEM uses OBD
    ClearPrestoredFreezeFrame(ERR{E_NOT_OK}); // OPTIONAL, only if DEM uses
                                              OBD
    GetFaultDetectionCounter(OUT Dem_FaultDetectionCounterType
        EventIdFaultDetectionCounter, ERR{E_NOT_OK});
}

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

// optional interface
ClientServerInterface ValueByOemId {
    PossibleErrors {
        E_NOT_OK = 1
    }
    SetValueByOemId(IN UInt16 OemID, OUT UInt8 DataValue,
        IN UInt8 DataLength, ERR{E_NOT_OK});
}

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

// optional interface
ClientServerInterface IndicatorStatus {
    PossibleErrors {
        E_NOT_OK = 1
    }
    GetIndicatorStatus (OUT IndicatorStatusType IndicatorStatus,
    ERR{E_NOT_OK});
}
```

11.4.2 Callback functions

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.

```
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 Dem_InitMonitorKindType
InitMonitorKind,
        ERR{E_NOT_OK});
}

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});
}

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
}

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
}

ClientServerInterface CallbackGetDataValueByDataID{DataId} {
    PossibleErrors {
        E_NOT_OK = 1
    }
    // used to get freeze frame data from SW-C
    GetDataValueByDataIdentifier(
      INOUT UInt8 DataValueBuffer[size of largest DID],
      INOUT Dem_MaxDataValueType DataId,
      ERR{E_NOT_OK});
}

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

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});
}
```

11.4.3 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.

11.4.4 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>;
      };


ServiceComponent Dem {

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

RequirePort CallbackInitMonitor CBInitEvt_<EventName>;
RequirePort CallbackInitMonitor CBInitEvt_<EventName>;
…
RequirePort CallbackInitMonitor CBInitEvt_<EventName>;

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

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

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

ProvidePort ValueByOemId ValByOemId;

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

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

// the <DataId> has to be in the format '0xNNNN' (e.g. '0x0200')
RequirePort CallbackGetDataValueByDataID CBValByDID_<DataID>;
RequirePort CallbackGetDataValueByDataID CBValByDID_<DataID>;
…
RequirePort CallbackGetDataValueByDataID CBValByDID_<DataID>;

// the <RecordNumber > has to be in the format '0xNN' (e.g. '0x05')
RequirePort CallbackGetExtendedDataRecord CBExtDataRec_<RecordNumber>;
RequirePort CallbackGetExtendedDataRecord CBExtDataRec_<RecordNumber>;
…
RequirePort CallbackGetExtendedDataRecord CBExtDataRec_<RecordNumber>;

RequirePort CallbackGetFaultDetectCounter CBFaultDetectCtr_<EventName>;
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 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


// 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>}
};
```

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

12.2 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

12.3 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

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.
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