Document Title	Specification of Diagnostics Event Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	019
Document Classification	Standard
Document Version	2.2.0
Document Status	Final
Part of Release	2.1
Revision	20

Document Change History

Date	Version	Changed by	Change Description
07.06.2010	2.2.0	AUTOSAR Administration	• Correction of Dem_DTCGroupType according to UDS specification (ISO 14229-1) • Legal disclaimer revised
24.01.2007	2.1.1	AUTOSAR Administration	• “Advice for users” revised • “Revision Information” added
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

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction and functional overview

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.

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.
Extended Data Record	An Extended Data Record is a record to store application-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	represent 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
Optional	In this specification, "optional" means the requirement is not mandatory for implementation. Optional does not mean that the feature can be selected by configuration.

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
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
EMI	Electro Magnetic Interference
EOL	End Of Line
ESD	Electro Static Discharge
ESP	Electronic Stability Program
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 Memory
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

1 Related documentation

1.1 Input documents

[1] List of Basic Software Modules,
AUTOSAR_BasicSoftwareModules.pdf

[2] Specification of ECU Configuration,
AUTOSAR_ECU_Configuration.pdf

[3] Layered Software Architecture,
AUTOSAR_LayeredSoftwareArchitecture.pdf

[4] General Requirements on Basic Software Modules,
AUTOSAR_SRS_General.pdf

[5] Specification of Diagnostic Communication Manager
AUTOSAR_SWS_DCM.pdf

1.2 Related standards and norms

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

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

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

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

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

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

2 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.

2.1 Limitations

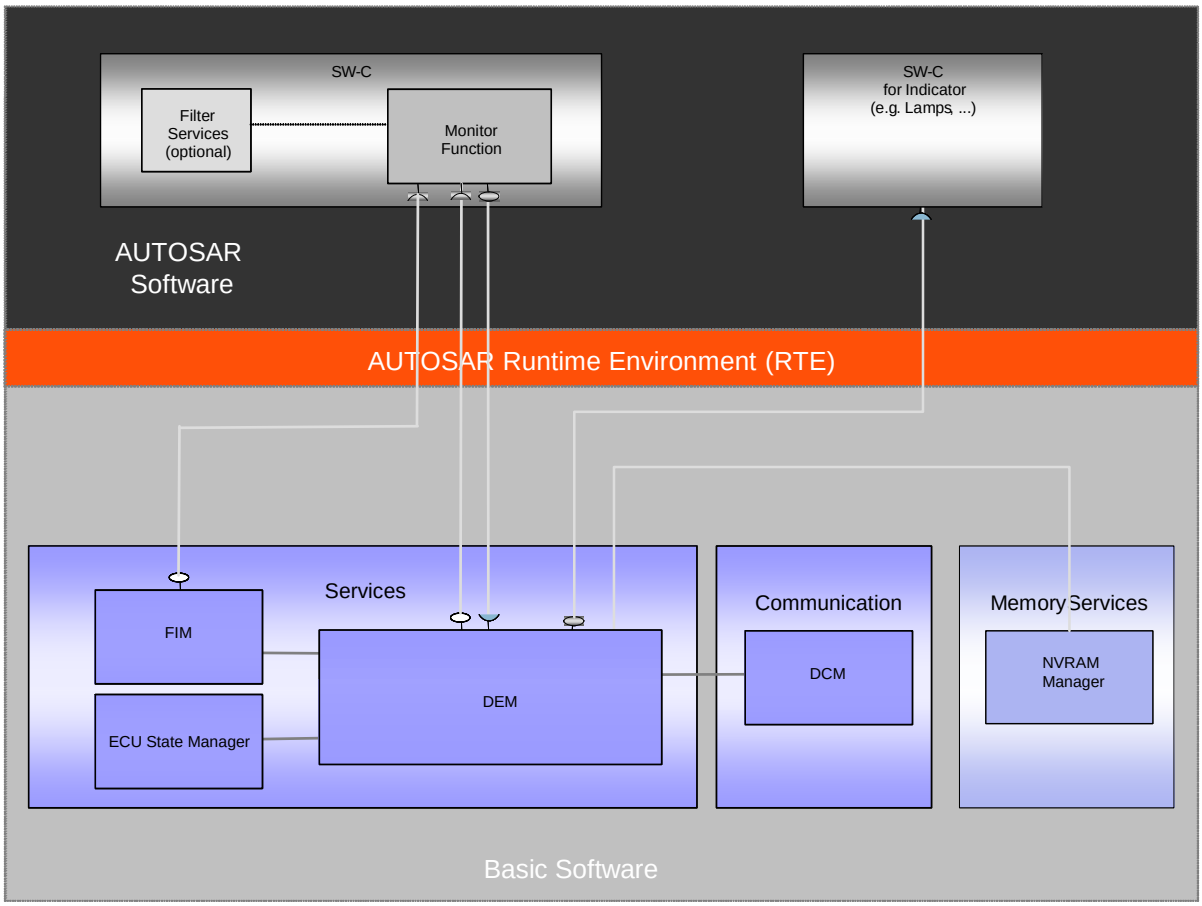
Timing constrains have to be considered for the whole ECU. If there are explicit needs for faster responses from the DEM than the DEM basic cycle time, special measures have to be implemented that are not specified in this AUTOSAR document. This is especially the case in ECUs with many events.

2.2 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.

3 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.

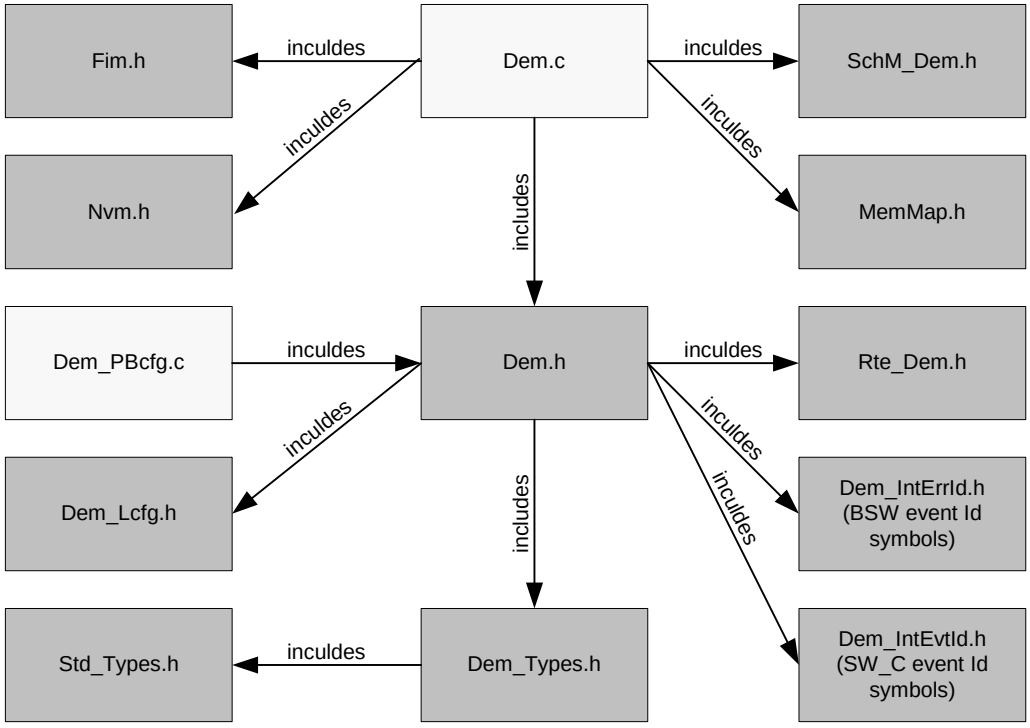
3.1 File structure

3.1.1 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.c – the main functionality
- Dem_Types.h – for all DEM data types
- Dem.h – provides the DEM API functions to other modules
- Dem_Lcfg.h – for link time configurable parameters
- Dem_PBcfg.h – for post build time configurable parameters
- Dem_IntErrId.h – for BSW EventId Symbols
- Dem_IntEvtId.h – for SW-C EventId Symbols
- SchM_Dem.h – for Basic Software Module Scheduler symbols

3.1.2 Header file structure



4 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, Dem065
[BSW158] Separation of configuration from implementation	Dem108
[BSW159] Tool-based configuration	Dem120, Ref. to chapter 10. configuration definitions
[BSW160] Human-readable configuration data	Dem120, 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	Dem120, 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	Dem118
[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	Dem118
[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

Requirement	*Satisfied by*
[BSW335] Status values naming convention	Implemented
[BSW336] Shutdown interface	Not applicable
[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	Dem119
[BSW345] Pre-Build configuration	Dem119
[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	Dem118
[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	Dem120
[BSW00389] Containers shall have names	Dem120
[BSW00390] Parameter content shall be unique within the module	Dem118
[BSW00391] Parameter shall have unique names	Dem118
[BSW00392] Parameters shall have a type	Dem118
[BSW00393] Parameters shall have a range	Dem118
[BSW00394] Specify the scope of the parameters	Dem118
[BSW00395] List the required parameters (per parameter)	Dem118
[BSW00396] Configuration classes	Dem119
[BSW00397] Pre-compile-time parameters	Dem119

Requirement	Satisfied by
[BSW00398] Link-time parameters	Dem119
[BSW00399] Loadable Post-build time parameters	Dem119
[BSW00400] Selectable Post-build time parameters	Dem119
[BSW00401] Documentation of multiple instances of configuration parameters	Dem119
[BSW00402] Published information	Dem112
[BSW00404] Reference to post build time configuration	Dem119
[BSW00405] Reference to multiple configuration sets	Dem119
[BSW00406] Check module initialization	Dem123, Dem124
[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 6.3.5) Dem042, Dem020, Dem041
[BSW04002] Basic SW Module for Diagnostic	Defined by AUTOSAR architecture

event (error) management [approved]	
[BSW04030] Interface between DEM and Monitoring SW Component	refer to [BSW103]
[BSW04057] Classification of event [approved]	Dem057, Dem058, Dem047, Dem104
[BSW04061] Multiple or parallel usage from different applications of the DEM functionality [approved]	Dem038
[BSW04063] Single EventId for each monitoring path	Dem005, Dem006, Dem023
[BSW04064] Event buffer must be configurable concerning size [approved]	See chapter Configuration specification (Chapters 5.1.2, 8.2)
[BSW04065] Clearing of events and event groups [approved]	See [BSW111], [BSW113]
[BSW04066] Provision of a `Secondary Event Memory [approved]	Dem010, Dem063
[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]	Dem017, Dem043
[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 8.2) See Type definition Dem_DTCTranslationFormatType
[BSW04076] System Cycle definition [approved]	Dem019, Dem047
[BSW04077] Interface between DEM and NVRAM function	Dem106

5 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).

5.1 DEM core variables

5.1.1 ´Diagnostic Event´ definition

A ´Diagnostic Event´ defines the atomic unit that can be handled by the DEM. 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.

Dem005: Each Diagnostic Event is represented by an EventId. The EventId shall be unique per DEM. Each event is assigned to only one Monitor Path. Two Monitor Paths can never manipulate the same EventId.

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

Dem023: The DEM receives via the RTE events from Monitor Function(s) (SW-Component). The events can be pre-debounced by the Monitor Function or using an optional filter, implemented and provided by the DEM for event qualification (ref. to 5.3.13.1). The Monitor function can use a generic filter library to pre-debounce an event.

Dem104: Within the DEM different attributes are assigned to each EventId. The DEM shall support the calibration/configuration of Event specific attributes to achieve a specific behavior. The assignment can be simplified by grouping sets of attributes to event classes.
Possible attributes depending on supported DEM functionality are:

- DTC(s)
- event priority (priority of an event, in view of full event buffer)
- Healing cycles (cycles necessary to heal/erase event)
- Healing allowed (general switch to allow healing or not)
- Identification of the destination of an event (ref. to Dem_DTCOriginType)
- Emission relevant (OBD fault definition)
- Indicator request (Indicator to be requested by EventId)
- Indicator counters (counters for activation/deactivation of indicators)
- ViewId

5.1.2 ´Event Memory´ description

The ´Event Memory´ is the storage area/array of the events and associated data (example: chapter 5.3.1) 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.

5.2 Event counting and status management

Dem001: The following API functions Dem_SetEventStatus, Dem_ResetEventStatus, Dem_GetEventStatus shall be provided by the DEM for OBD relevant and non OBD relevant ECUs. The functions Dem_PrestoreFreezeFrame/ Dem_ClearPrestoreFreezeFrame, Xxx_DemInitMonitor{EventId}, Dem_GetEventFailed, Dem_GetEventTested shall be provided by the DEM for OBD relevant systems and are optional for non OBD relevant systems. They are the interface functions of the DEM to control the behavior of the events inside the DEM. The functions are described in detail in chapter 6.3.

Dem009: The deletion of DTCs must be provided related to each single event, event groups and all events.
ClearDiagnosticInformation (14 hex) Service of ISO14229 defines and covers the required actions and the deletion of related memory areas like FreezeFrames. The groups are defined in ISO14229 Definition of GroupOfDTC and range of DTC numbers, Annex D1.
The API function Dem_ClearDTC shall provide the erase functionality related to a request by the DCM. Accoriding to the ISO service, only a DTC is transmitted which can either represent a single DTC or a group of DTC. This distinction has to be made within the DEM when triggered by the DCM.
For this API functions the initialization of the corresponding Monitor Function (DTC -> EventId -> Xxx_DemInitMonitor{EventId}) is managed by Xxx_DemInitMonitor{EventId}.

Dem003: An Interface shall be provided to initialize the monitor function. This function shall be event specific (Chapter 5.3).
With the API 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.
The association between an EventId and a corresponding Xxx_DemInitMonitor{EventId} function is done by configuration of DEM.

Dem010: The DEM shall be capable to support several event memories (the event memories can be physically located in the same address range).
For the DCM-DEM Interface the Dem_DTCOriginType is defined to distinguish between the memory areas. See Dem063
The intention is to allow OEM specific operations on the different memory areas (Primary, Secondary memory and mirror memory).

The usage of several event memories is not mandatory.

Dem011: The DEM shall control the counting of the number of event/DTC occurrences according to DTCStatusMask and DTC Type.
Dem013: The ISO14229-1 format of DTCs shall be supported (3 bytes). OBD ECUs shall support 2 byte DTCs according to ISO 15031-6 as well. (DEM_TYPE_OF_DTC_SUPPORTED)

Dem034: The DEM shall be capable of enabling (Dem_EnableEventStatusUpdate) and disabling (Dem_DisableEventStatusUpdate) the update of all event states. Meaning: when disabled, calls to Dem_SetEventStatus or Dem_ResetEventStatus will not lead to changes in internal states of DEM (ref. to section 6.3.3.1).

Dem035: The DEM shall be capable of enabling (Dem_EnableDTCStorage) and disabling (Dem_DisableDTCStorage) the storage of all event records. Meaning: when disabled, the update of an event status does not result in changes in the event memory (no DTC storage) (ref. to section 6.3.3.1).

5.3 DEM core functionalities

5.3.1 Overview DEM Structure

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

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	…

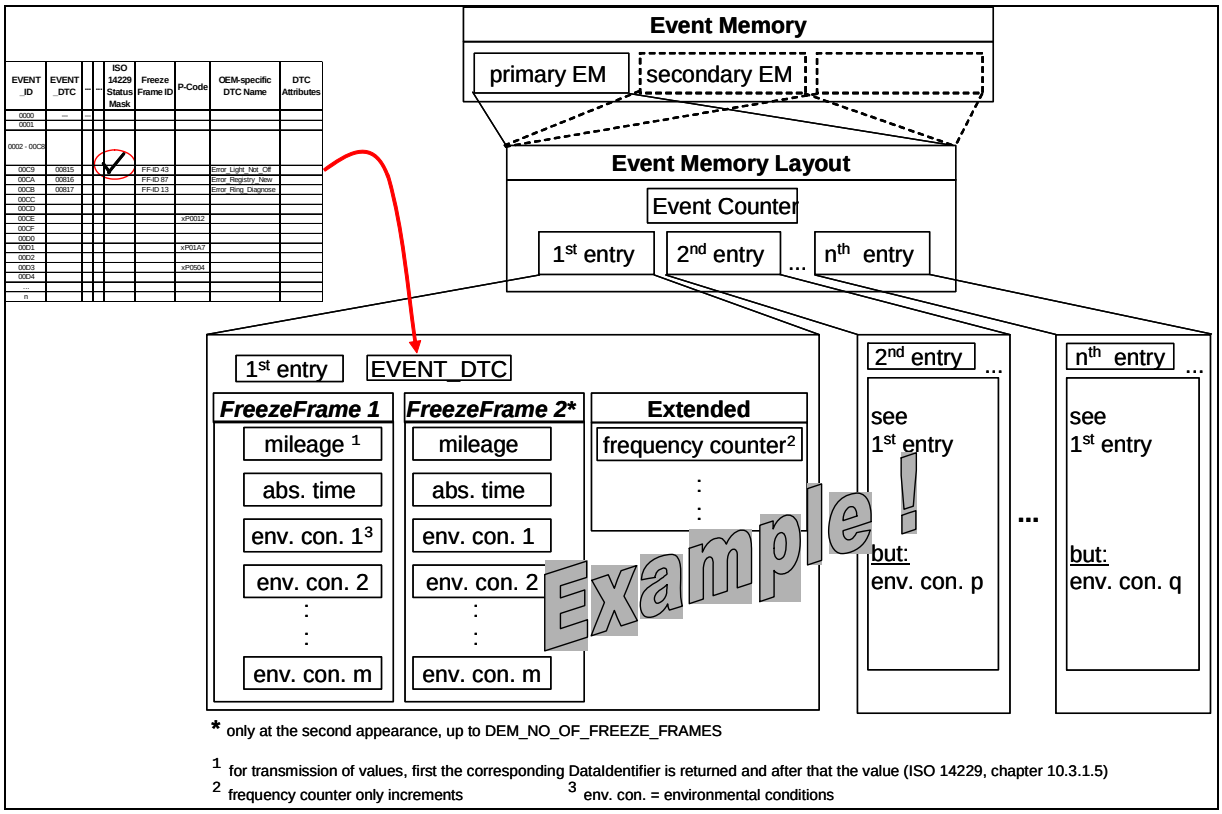
Minimum DEM configuration independant of DEM functionality

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



5.3.2 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: Event unlearn counters (e.g. for failure healing) shall be supported event specific. In case of OBD-relevant events they shall be based on the OBD/ISO defined cycles.

Dem014: The DTC Status mask shall be supported according to ISO14229. The availability of specific status mask information shall be checked by calling Dem_GetDTCStatusAvailabilityMask (Dem060) before calling the Status mask filter function. The DCM shall set the Status mask as required (Dem_SetDTCFilter, Dem057).

Dem042: The DEM shall use the internal event status information to meet the DCM-requirement of a DTC-status request. If the basic information (ref. to 6.2.1.5) of the DEM is not sufficient, the DEM has to provide further status information (event pending). 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 event the DEM shall perform the event status transition immediately for each status supported for that event when requested via API call.

Dem038: (optional) To select only events of a specific function group (wiper, seat, climate control) the function Dem_SetViewFilter (Dem058) shall be used. All consecutive API calls from the DCM shall work only with the events of these functions groups.

Dem015: (optional) The DEM shall 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. An IndicatorId is assigned to an EventId by configuration or calibration.

Dem016: The DEM shall support the execution of an event specific function upon status change of each event (e.g. FIM or ISO14229 service 86hex "ResponseOnEvent Request"). For this purpose the function prototype Xxx_DemTriggerOnEventStatus (ref. to Dem044) shall be used.

Dem017: The occurrence order of events shall be recognizable by e.g. mileage, time stamps and/or age. Reoccurrence of healed events shall be handled like new events since they were previously erased. The DEM shall provide enough memory space to store all high priority faults.

Dem033: (optional) Severity shall be assigned to events regarding the importance of the specific events according to ISO14229, Annex D, DTCSeverityMask and DTCSeverity bit definitions. Severity levels defined there are: no severity available, Maintenance, Check at next Halt, Check immediately.
The API call “Dem_SetDTCFilter” (Dem057) allows filtering for DTCs with severity information.

5.3.3 DEM Statistics

Dem020: (required for OBD-relevant diagnostics)
The DEM shall 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).

5.3.4 Event combination

Large SW applications or SW components normally consist of a high number of Monitor Functions especially if the SW components have to process several sensors or a high number of actuators.
SW Components in themselves are getting more and more sophisticated and try to take over or replace functions which were formerly handled by hardware components or at least to compensate for reduced quality of sensors or actuators.
Subsequently the number, complexity and quality of Monitor Functions is increasing continuously, trying to cover as detailed as possible a malfunction of e.g. hardware components via similar or slightly different supervision criteria to get maximal fault coverage.

Based on the reasons mentioned above it could be possible to have several events which point to the same hardware defect or malfunction.
Related to the “customer of the fault information” (i.e. the mechanic in the plant or service station) the meaning of the DTC/event could lead to identical repair instructions but might have different basic root causes i.e. events. Event combination is an optional feature which is implementation-specific. A possible handling of event combination is included in Dem024, Dem025 and Dem026.

Dem024: (optional) The DEM shall 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 update of additional event specific information (e.g. self-healing, frequency counters, environmental data)

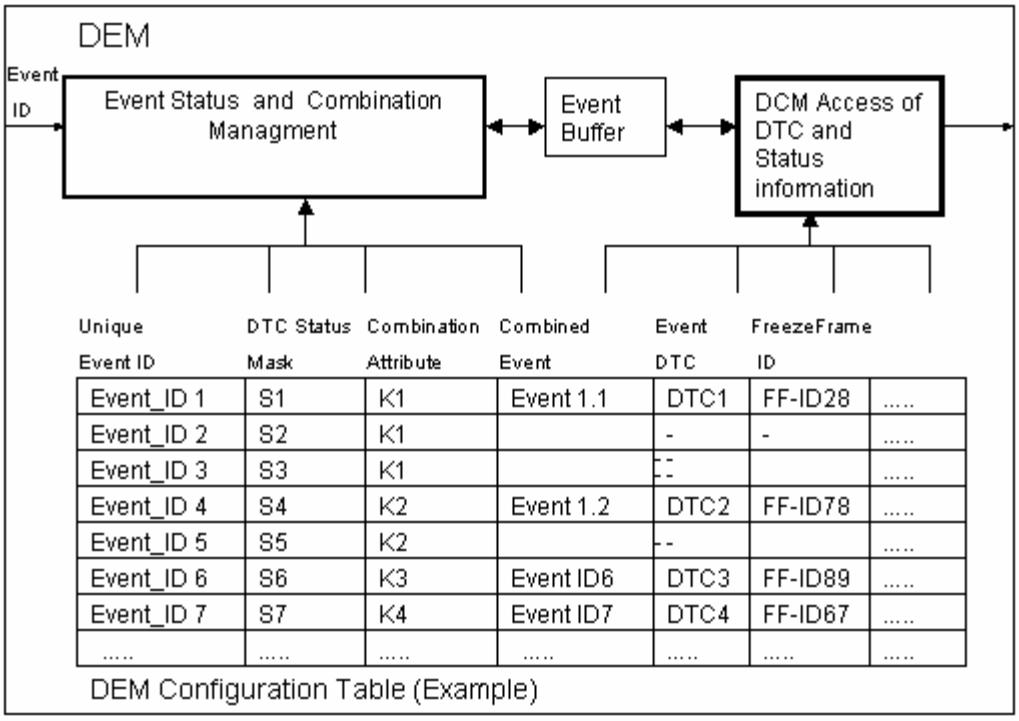
When combined events are supported the consistency between the combined event status and associated Monitor Functions must be ensured when updating the status or clearing individual events of the combined event or the combined event itself.

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

Related to the erasing of the ´combined event´ resulting from a failure healing mechanism (unlearn counter) corresponding Monitor Functions are usually not 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: (optional) The configuration of the DEM must cover the enabling and disabling of “event combinations”, hence this functionality could be used as an option.

Dem026: (optional) If “combined diagnostic events” are supported the configuration of the DEM shall allow for assigning each “diagnostic event” the attribute “combined diagnostic EventId”.



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

DEM Configuration Table (Example)

5.3.5 Environmental Data

The ´Environmental Data´ is 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 shall support one or several FreezeFrames with different sets of environmental data. The DEM is not in charge of validity of environmental data. Time related data consistency of environmental data is depending on data source and storage time.

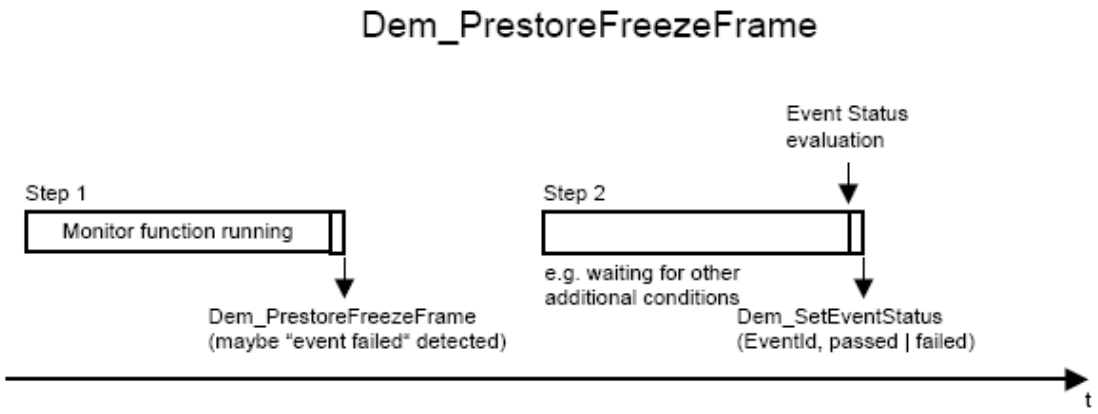
Dem021: The DEM shall support the EventId specific storage of one or several of the FreezeFrames defined by Dem039:

Dem040 : The number and the size of each FreezeFrame that can be stored by the DEM 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: (optional) Dem_PrestoreFreezeFrame

The DEM shall support the storage of a FreezeFrame regardless of the status change of an event. 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 shall support the storage of additional information associated to a specific event that is not contained in a freeze-frame (extended data, e.g. frequency counters, self-healing counters, etc.).

Note:
At moment the UDS service (0x19, 0x05, 0xXX) is not supported by DEM, because snapshot records are DTC related and therefore not unique in the ECU. For OBD ECUs the UDS service (0x19, 0x05,0x00) shall be supported.

5.3.6 Main use cases for DEM interaction with Monitor Functions

To support fault tracing of different types of error scenarios using information from Monitor Functions with varying capabilities in applications that are dedicated to support several characteristics, there is a need to distinguish between different types of event characteristics including associated FreezeFrames. This often results in conflicting or impossible requirements on the capability of Monitor Functions. The paragraph below names four such conflicting requirements, followed by a discussion:
1. Wait until high degree of confidence is achieved by the monitor function, i.e. the error is established, then save FreezeFrame data and set Test Failed in order to store DTC that points towards the root cause of the problem.
2. When error causes drivability disturbance in the vehicle, immediately save FreezeFrame data and set Test Failed to store DTC even though the error may not yet be established (error may disappear but driver has noticed and may complain, i.e. workshop needs all the help it can get). This “unreliable” information is classified in ISO as DTC with suffix 68h, i.e. xxyy68h. Furthermore, in some cases certain

drivability disturbances, like engine stall or too long cranking time, are not possible to classify as errors at all (root cause could be driver mistake or no fuel) but are needed if an error, not possible to monitor at the time it caused the disturbance, is the root cause to an unexpected stall or long cranking time.
3. If error is OBD relevant, wait until error is established before storing a corresponding OBD DTC with OBD FreezeFrame.
4. If Monitor Function of error needs to use local conditions to change DTC status.

Discussion:
Requirements 1 and 3 are possible to combine, i.e. wait until error is established error before event is set to Failed. The difference is only the way the information is presented, i.e. one event with its FreezeFrame data may be used to store ISO DTC, OBD DTC, ISO FreezeFrame and OBD FreezeFrame.
However, requirement 2 has totally different timing and is therefore not possible to fulfill with same event and FreezeFrame. Here the FreezeFrame may have the same parameters but shall be saved immediately upon first indication of error which could be supported by pre-store FreezeFrame.
Finally requirement 4 is basically a different type of requirement.

The outcome is the following use cases with some new requirements for the DEM:

Use case A.
Drivability not affected FreezeFrame relevant when error is established:
Monitor functions for errors which do not cause drivability disturbance before a reliable error determination is possible during all driving conditions.
These monitors may use a single event with FreezeFrame data saved at Test Failed. DTC and FreezeFrame is calibrated according to ISO and may also be calibrated to set OBD DTC with OBD FreezeFrame.

Use case B.
Drivability not affected FreezeFrame relevant when error is established:
In cases where FreezeFrame data needs to be saved before Test Failed is set, the DEM shall support pre-store FreezeFrame. Note that no service information is available before Test Failed is set (OK since drivability is not disturbed). If used also for OBD, the OBD FreezeFrame may be saved at pre-store.
An example is an offset error within the engine coolant temperature sensor which can only be detected at start-up but the error is not established before the engine pre-heater is determined as not used (by then FreezeFrame parameters are useless).

Use case C.
Drivability affected, error not yet established (FreezeFrame may not yet be relevant):
Monitor functions for errors that cause drivability disturbance before a reliable error determination is possible during driving conditions likely to occur.
DEM will support these two different events associated to the same detection mechanism. Each event will be treated separately from the each other. Both events will save FreezeFrame data as soon as the event is set to Test Failed but one event will have a very short de-bounce time and different settings, ref. to Dem090 below.

Use case D.
System events.

The DEM will support separate monitor functions that generate so called system events not associated to errors (e.g. engine stall) as they are treated the same way like any other event. A difference may be DTC storage settings, ref. to Dem090 below.

Dem090: (optional) To support events set before error is established as well as system-events, it shall be possible to select DTC storage that does not use pending status at all.
DTCs shall be possible to calibrate with 68h at the end (i.e. xxyy68h according to ISO_DIS_15031-6.4_(E) dated Jan 14th 2004).
Currently they shall be accessible through the same service as other ISO DTCs (i.e. service 19 sub mode 02).
They shall be treated as separate entities by the DEM, i.e. supporting ISO 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.

Dem091: (optional) The DEM shall support direct status change of events on request from function. Ref.: “Dem_SetEventStatus”.

5.3.7 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 ISO standard) to the internal EventId’s.
It is possible to have more than one translation for the internal EventId (ref. to Dem006:).

5.3.8 DEM Cycle Management

Different Operation cycles are used by the DEM (ref. to ISO14229-1). Those cycles could either be provided by other BSW modules and SW-C or generated by the DEM 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.

5.3.9 NVRAM Manager Access

Dem106: The non-volatile memory blocks (configurable in size) are used by the DEM 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 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 copy to NVRAM, NVRAM data will be incomplete 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 redundancy data blocks.
If there is the necessity for the DEM to store and restore data between Dem_Init and Dem_Shutdown the DEM shall call NvM_WriteBlock and NvM_ReadBlock.
If the DEM retrieves an error message from NVRAM Manager while trying to call the services NvM_WriteBlock or NvM_ReadBlock the DEM shall retry to access the NVRAM Manager only for a configurable number of times. A configurable time limit between the retries has to be defined.
Was the call of NvM_ReadBlock after the defined recurrences not successful, the DEM could generate a DTC in the RAM area of event memory (behavior depends on OEM). Note: All additional informations, like occurrence counter and freeze frames, are not meaningful, because the information could be volatile.
Was the call of NvM_WriteBlock after the defined recurrences not successful, the DEM could generate a DTC in the RAM area of event memory (behavior depends on OEM). Note: The Information will be lost after ECU power down.

5.3.10 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: Upon changes of reported event status, the DEM shall inform BSW (e.g. FIM) or SW-C about the new status. For this treatment the function prototype Xxx_DemTriggerOnEventStatus (ref. to Dem044) shall be used. The linkage between

a diagnostic event and a Xxx_DemTriggerOnEventStatus function is part of configuration (ref. to 8.2.6).
The information is also passed to the FIM if Dem_DisableDTCStorage is called.

Dem031: The inhibition relations between events and application software depend on the event status. However, the DEM 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 (ref. to Dem_EventStatusExtendedType).
By this extension, function entities for complex application, e.g. long expression, can be stopped upon pending status.

Dem032: For possible plausibility checks of the FIM, re-building, start-up etc. of inhibition relations by the FIM, the DEM shall provide API to read out event status (ref. to Dem_GetEventStatus).

5.3.11 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 direct to event memory
- Errors 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: Unlearning/healing of BSW errors shall be possible. 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).

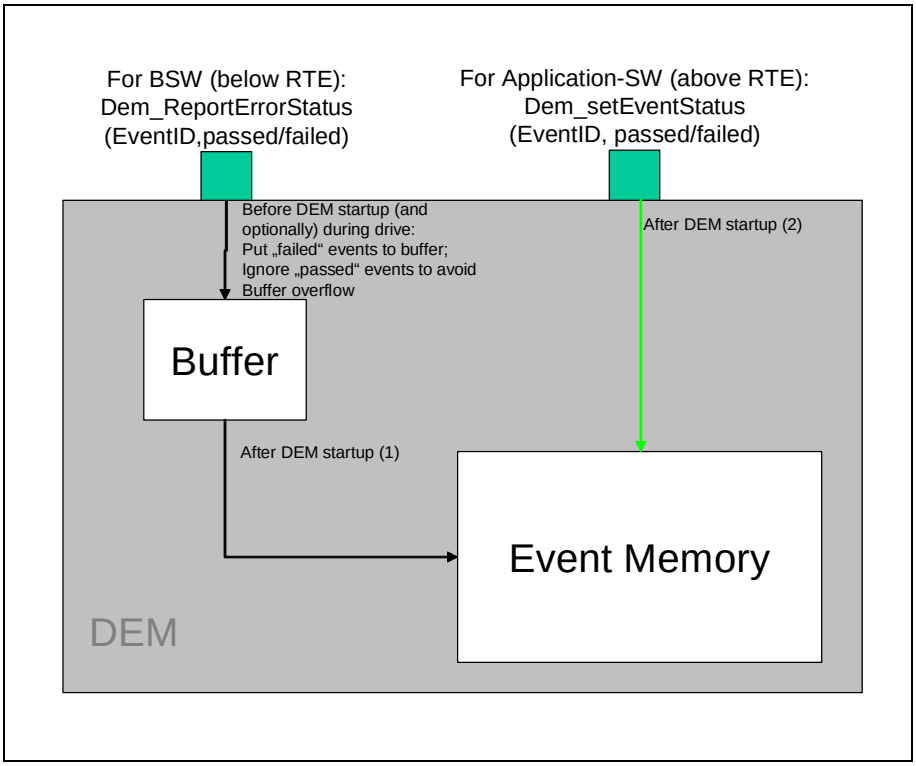
Dem107: Due to the fact that the DEM is not available during startup a separate interface shall be provided by the DEM to BSW to report errors. Therefore, the API call Dem_ReportErrorStatus shall be used.

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 finished) the buffer is processed by a cyclic task of the DEM and all contained events are reported to the event memory.

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

If the interface Dem_ReportErrorStatus is used during normal operation, the reported events are directly passed through the Buffer to the Event Memory. In case that the Event memory can not be accessed, reported events are buffered (FIFO).

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.



5.3.12 DEM startup behavior

Dem123: A static status variable denoting if the DEM module is initialized shall be initialized with value 0 before any APIs of the DEM module is called.
The initialization function of the DEM module shall set the static status variable to a value not equal to 0.

Dem124: If an API call occurs before the DEM is initialized the Development Error Tracer (DET) shall be called to set the error code DEM_E_UNINIT in case that the DET is activated. The function Dem_ReportErrorStatus is excluded from this requirement.

5.3.13 Debouncing of events

5.3.13.1 Distinction between ´signal de-bouncing´, ´event debouncing´ and ´event qualification´

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

1. **Signal debouncing** - (i.e. conditioning) is done in HW, incl. EMI, ESD, LPF. Signal de-bouncing in hardware is the responsibility of the ECU-HW designer and is not part of specification work in WP4.2.2.1.4.
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 DEM (if configured), the monitor function have 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 [9].
   Event qualification is processed inside DEM if the event de-bouncing is done inside DEM. Otherwise the Event qualification is done by monitor function.
   For OBD-units, event qualification according to legal requirements is mandatory.

5.3.13.2 Event De-bouncing algorithms

The DEM offers some standard algorithm for debouncing in order to avoid multiple implementations of the same mechanism in several monitoring modules. Below the defined algorithms are described. Important to note, that a debounce mechanism can be configured per EventID (for details see Chap. 0). This is 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{*EventId}* to reset the DTCFaultDetectionCounter after the DTC was cleared.

5.3.13.2.1 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.

5.3.13.2.1.1 Representation the DTCFaultDetectionCounter:

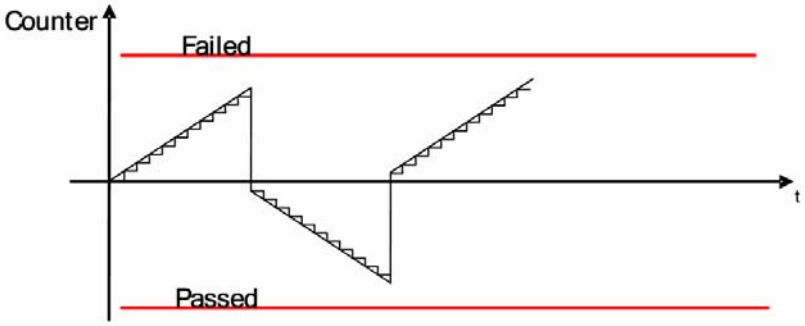
The counter base 1:1 relation with maximum value of 127 and the minimum value of -128. If the pre-debouncing has been finished then the DTCFaultDetectionCounter is either 127 (this means “TestFailed”) or -128 (this means “Passed”). When the debouncing is in progress the counter value can be derived from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

Different mechanisms are possible. According to the DTCFaultDetection definition, the counter increment upon a PREFAILED report and decrement 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.

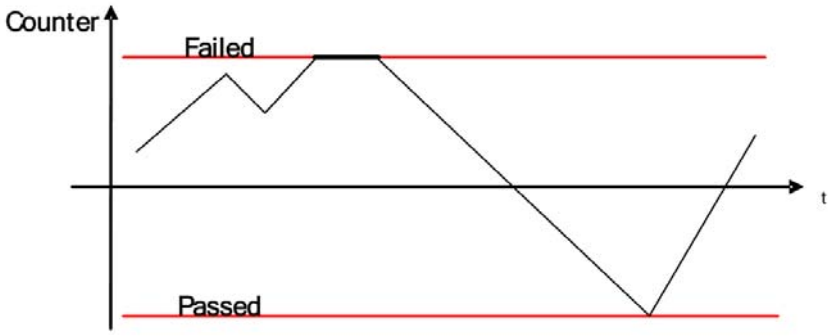
Since range of -128 to +127 is fixed, different limits for FAILED and PASSED detection in the internal debouncing can be converted into the 1-byte range via different step sizes. Therefore, it shall be possible to either defined a parameter set for step size (assuming equal levels for FAILED and PASSED 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 a “event-in-a-row”-behavior. According to ISO 14229-1 (version of Nov. 2005) the behavior of the DTCFaultDetectionCounter indicates an asymmetric behavior where the jump is only active upon PREFAILED reports in order to start FAILED detection always from the “0”-level.

- Events in a row counter



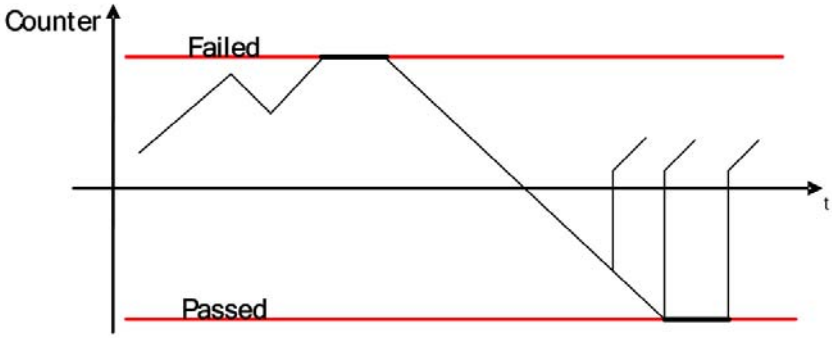
Note, that the steps should indicate individual incrementing and decrementing per report and also to show the combination of a jump and a step upon a change of the reported result.

- Events up-down-counter



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

- Count-in – Count-out/Jump-in



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

5.3.13.2.1.2 Use Cases

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

5.3.13.2.1.3 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)

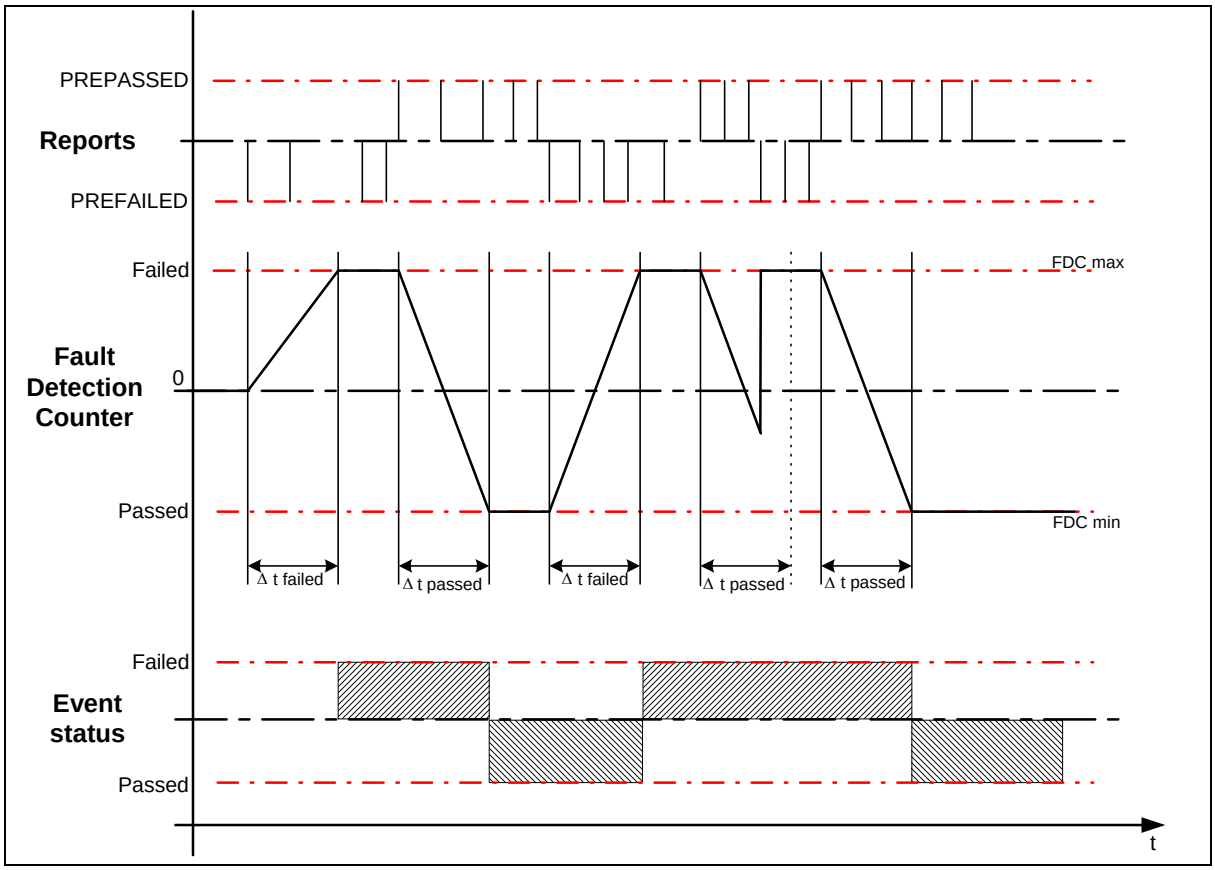
- Switch for the activation of Jump-UP (boolean) (upon PREFAILED report within PREPASSED / PASSED range)
- Switch for the activation of Jump-DOWN (boolean) (upon PREFAILED report within PREPASSED / PASSED range) – only in combination with Jump-UP activation.

5.3.13.2.2 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.

5.3.13.2.2.1 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.



5.3.13.2.2.2 Use Cases

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

5.3.13.2.2.3 Parameter:

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

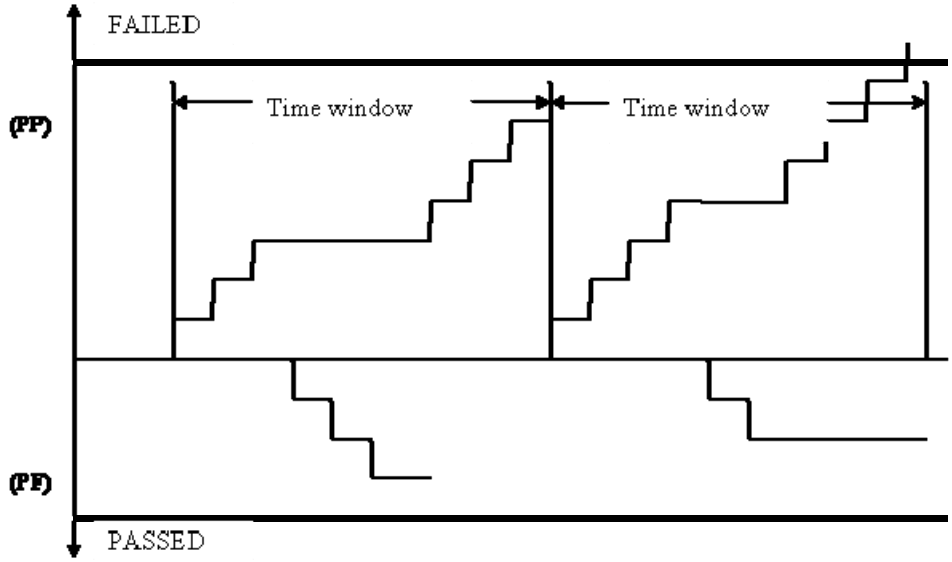
5.3.13.2.3 Error occurrence frequency based

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.

5.3.13.2.3.1 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.



5.3.13.2.3.2 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'

5.3.13.2.3.3 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)

5.4 Auxiliary explanations and definitions

5.4.1 Requirements on variables

5.4.1.1 Variables provided to DEM

The DEM requires several input values for computation. We distinguish between two different kinds of values:

- Environmental data to be stored in FreezeFrames

The values are requested by a data ID mechanism (by calling Xxx_GetDataValueByIdentifier) via the data ID configuration table according to ISO14229-1.

This list shall also contain parameters required as PIDs according to ISO15031-5 in order to fulfill OBD mode02.

- Data for the calculation of operation cycles or statistical data (mileage for the computation of the return value of API Dem_GetDistanceMIL)

This data is requested via normal RTE interface (for example engine speed or ambient temperature).

5.4.1.2 Variables returned from DEM

The negative return value WRONG_DTCORIGIN is only returned if an unavailable event memory /origin is requested. In case that a DTC is requested which is available but has a different origin than the requested one the return value WRONG_DTC is returned.

5.5 Version check

Dem067: A pre-processor check in the *.c file shall ensure that the right version of the *.h file is included.

5.6 Error classification

Dem115: Values for production code EventId's 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 report 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.

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

5.7 Error detection

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

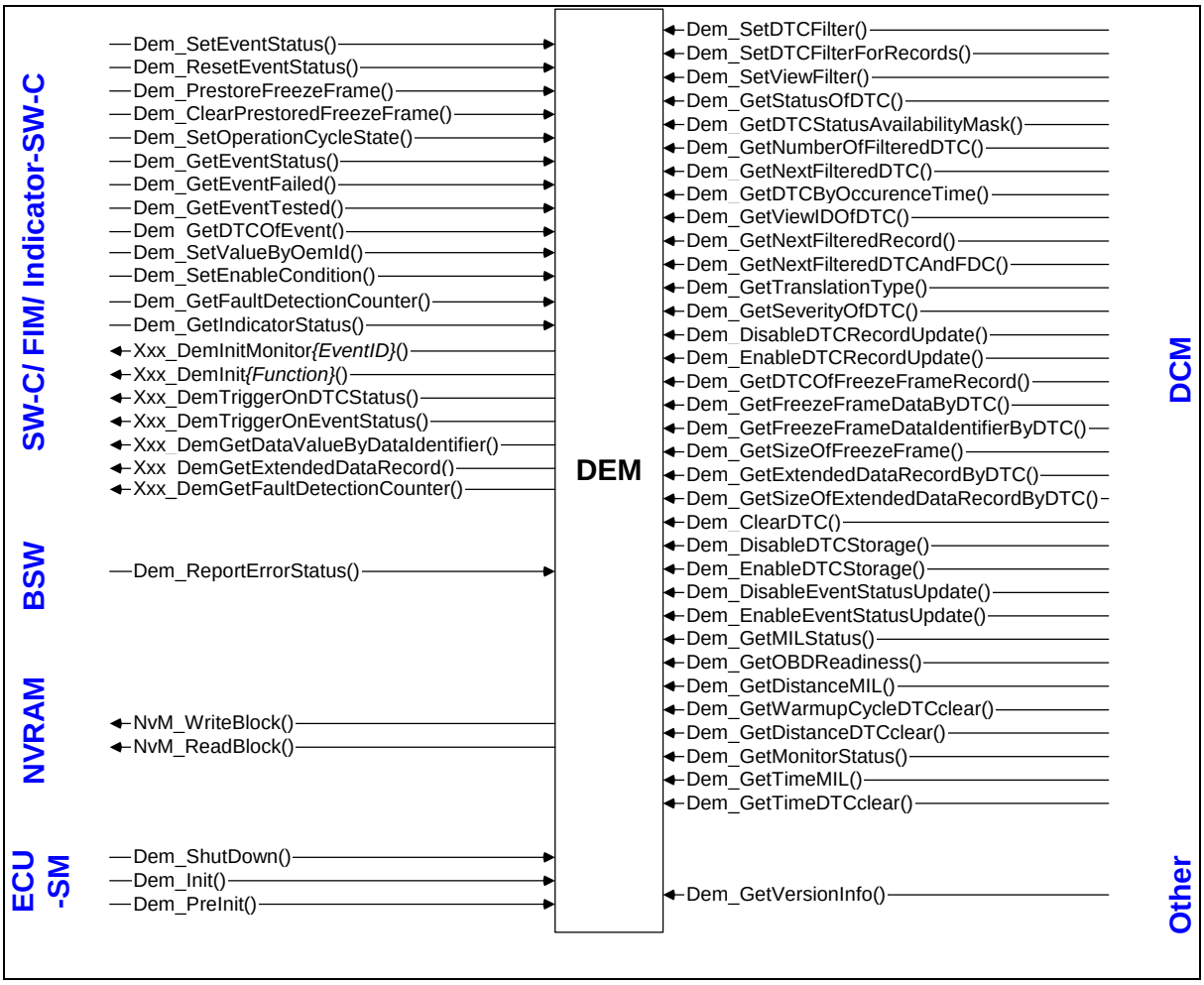
Dem114: If the *DEM_DEV_ERROR_DETECT* switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 5.6 and chapter 6.

5.8 Error notification

Dem117: Detected development errors will be reported to the error hook of the Development Error Tracer (DET) if the pre-processor switch *DEM_DEV_ERROR_DETECT* is set (ref. to chapter 10).

6 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.



6.1 Imported types

6.1.1 Standard types

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

- Std_Types.h

- uint8
- uint16
- uint32

- boolean
- Std_ReturnType
- Std_VersionInfoType

6.1.2 NVM types

In this chapter all types included from module NVM are listed.

- NvM_ReturnType

6.2 Type definitions

Dem118: The following Data Types shall be used for the functions defined in this specification.

6.2.1 DEM data types

6.2.1.1 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.	

6.2.1.2 Dem_DTCType

Type:	uint32	
Range:	0…0xFFFFFF	Identifier of DTC
Description:	Diagnostic Trouble Code (DTC) assigned to each event stored in the event memory. The DTC is 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).The Dem services shall report these 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.

6.2.1.3 Dem_ViewIdType

Type:	uint8	
Range:	1…255	Total number of view IDs
Description:	Identification of a view by assigned identification number. 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 limit number of views according to configuration of DEM_NUMBER_OF_VIEWS	

6.2.1.4 Dem_EventStatusType

Type:	uint8	
Range:	0x00	DEM_EVENT_STATUS_PASSED Monitor Function reports the debounced test result passed
	0x01	DEM_EVENT_STATUS_FAILED Monitor Function reports the debounced test result failed
	0x02	DEM_EVENT_STATUS_PREPASSED Monitor Function reports the non debounced test result passed
	0x03	DEM_EVENT_STATUS_ PREFAILED Monitor Function reports the non debounced test result failed
	0x04..0x1F	reserved
	0x<*xx*> (0x20..0xFF)	DEM_EVENT_STATUS_<Custom> Monitor Function reports custom status, optional, for complex OBD applications and / or transitions of, e.g. misfire entries.
Description:	See API Dem_SetEventStatus	

6.2.1.5 Dem_EventStatusExtendedType

Type:	uint8	
Range:	Bit0	testFailed (=1; Passed = 0)
	Bit1	testFailedThisOperationCycle (= 1; not yet failed this cycle = 0)
	Bit2	pendingDTC (= 1; Not Pending DTC status = 0)
	Bit3	confirmedDTC(= 1; Not Confirmed DTC status = 0)
	Bit4	testNotCompletedSinceLastClear (=1; Test CompletedSince Last Clear = 0)

	Bit5	testFailedSinceLastClear (=1; test not Failed Since Last Clear = 0)
	Bit6	testNotCompletedThisOperationCycle (= 1; Completed this cycle = 0)
	Bit7	warningIndicatorRequested (=1)
Description:	See API Dem_GetEventStatus	

6.2.1.6 Dem_DTCKindType

Type:	uint8	
Range:	0x01	DEM_DTC_KIND_ALL_DTCS Select all DTCs
	0x02	DEM_DTC_KIND_EMISSION_ REL_DTCS Select OBD-relevant DTCs
Description:	Used to define the DTC kind (type).	

6.2.1.7 Dem_DTCGroupType

Type:	Uint32	
Range:	0x000000	DEM_DTC_GROUP_EMISSION_REL_DTCS Select group of OBD-relevant DTCs
	Depends on configuration	DEM_DTC_GROUP_POWERTRAIN_DTCS Select group of powertrain DTCs
	Depends on configuration	DEM_DTC_GROUP_CHASSIS_DTCS Select group of chassis DTCs
	Depends on configuration	DEM_DTC_GROUP_BODY_DTCS Select group of body DTCs
	Depends on configuration	DEM_DTC_GROUP_NETWORK_COM_DTCS Select group of network communication DTCs
	0xFFFFFF	DEM_DTC_GROUP_ALL_DTCS Select all DTCs
Description:	Used to define the group of DTCs. The user can add further groups. Unused bytes shall be filled with 00.	

6.2.1.8 Dem_DTCOriginType

Type:	uint8	
Range:	0x01	DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory

	0x02	DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information located in the secondary memory
	0x03	DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory
Description:	**Dem063:** The Dem_DTCOriginType is used to differ between the different event memories. Used to define the origin of the DTC. The definition and use of the different memory types is OEM specific.	

6.2.1.9 Dem_DTCTranslationFormatType

Type:	uint8	
Range:	0x01	DEM_DTC_TRANSLATION_ISO15031_6 DTCs of ISO15031-6
	0x02	DEM_DTC_TRANSLATION_ISO14229_1 DTCs of ISO14229-1
	0x03	DEM_DTC_TRANSLATION_CUSTOMER Customer specific DTCs
	0x04	DEM_DTC_TRANSLATION_INTERNAL Internal by EventId
Description:	Used to define the DTCTranslationFormat.	

6.2.1.10 Dem_DTCSeverityType

Type:	uint8	
Range:	0x00	DEM_SEVERITY_NO_SEVERITY No severity information available
	0x20	DEM_SEVERITY_MAINTENANCE_ONLY Maintenance required
	0x40	DEM_SEVERITY_CHECK_AT_NEXT_HALT Check at next halt
	0x80	DEM_SEVERITY_CHECK_IMMEDIATELY Check immediately
Description:	Used to define the DTC severity.	

6.2.1.11 Dem_DTCRequestType

Type:	uint8	
Range:	0x01	DEM_FIRST_FAILED_DTC First failed DTC requested

	0x02	DEM_MOST_RECENT_FAILED_DTC Most recent failed DTC requested
	0x03	DEM_FIRST_DET_CONFIRMED_DTC First detected confirmed DTC requested
	0x04	DEM_MOST_REC_DET_CONFIRMED_DTC Most recently detected confirmed DTC requested
Description:	Request type for Dem_GetDTCByOccurrenceTime.	

6.2.1.12 Dem_EventPriorityType

Type:	uint8	
Range:	0…255	Define the priority of each event. 0 is the lowest priority.
Description:	Used to define the priority of an event (ref. to configuration chapter)	

6.2.1.13 Dem_DataByteType

Type:	uint8	
Range:	0…255	DataByte (DataA, DataB, DataC, DataD) according ISO15031-5 definition
Description:	Used to return the value of DataA/DataB/DataC/DataD, depending on request	

6.2.1.14 Dem_IndicatorIdType

Type:	uint8	
Range:	0…255	indicator lamp ID
Description:	Used to request an indicator type	

6.2.1.15 Dem_IndicatorStatusType

Type:	uint8	
Range:	0x00	DEM_INDICATOR_OFF Indicator off
	0x01	DEM_INDICATOR_CONTINUOUS Continuous on
	0x02	DEM_INDICATOR_BLINKING Blinking mode
	0x03	DEM_INDICATOR_BLINK_CONT Continuous and blinking mode The SW-C for indicator is responsible to decide if the indicator is blinking or continuously on.
Description:	Used to return the status of Dem_GetIndicatorStatus and Dem_GetMILStatus	

6.2.1.16 Dem_OperationCycleIdType

Type:	uint8	
Range:	0…255	The OperationCycleId, e.g. the ignition cycle is assigned to an ID.
Description:	Used to select the individual operation cycle in the API Dem_SetOperationCycleState.	

6.2.1.17 Dem_OperationCycleStateType

Type:	uint8	
Range:	0x01	DEM_CYCLE_STATE_START Start of operation cycle
	0x02	DEM_CYCLE_STATE_END End of operation cycle
Description:	Used to indicate start and end of operation cycle by API Dem_SetOperationCycleState.	

6.2.1.18 Dem_FilterWithSeverityType

Type:	uint8	
Range:	0x00	DEM_FILTER_WITH_SEVERITY_YES Severity information used
	0x01	DEM_FILTER_WITH_SEVERITY_NO Severity information not used
Description:	Used to specify the usage of severity information	

6.2.1.19 Dem_InitMonitorKindType

Type:	uint8	
Range:	0x01	DEM_INIT_MONITOR_CLEAR Monitor Function of the EventId is cleared and all internal values and states are reseted.
	0x02	DEM_INIT_MONITOR_RESTART Monitor Function of the EventId is requested to restart
Description:	See API Xxx_DemInitMonitor{*EventId}*	

6.2.1.20 Dem_DTCStatusMaskType

Type:	uint8	
Range:	0x00…0xFF	Match DTCStatusMask as defined in ISO14229-1
Description:	Used to set the current status	

6.2.1.21 Dem_FilterForFaultDetectionCounterType

Type:	uint8	
Range:	0x00	DEM_FILTER_FOR_ FAULTDETECTIONCOUNTER_YES Fault Detection Counter information used
	0x01	DEM_FILTER_FOR_ FAULTDETECTIONCOUNTER_NO Fault Detection Counter information not used
Description:	Used to specify the usage of Fault Detection Counter information	

6.2.1.22 Dem_FaultDetectionCounterType

Type:	sint8	
Range:	-128dec…127dec	PASSED … FAILED according to ISO 14229-1
Description:	Used to report Fault Detection Counter via API Dem_GetFaultDetectionCounter	

6.2.2 DEM return types

6.2.2.1 Dem_ReturnSetDTCFilterType

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

6.2.2.2 Dem_ReturnSetViewFilterType

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

6.2.2.3 Dem_ReturnGetStatusOfDTCType

Type:	uint8

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

6.2.2.4 Dem_ReturnGetNextFilteredDTCType

Type:	uint8	
Range:	0x00	DEM_FILTERED_OK Returned next filtered DTC
	0x01	DEM_FILTERED_NO_MATCHING_DTC No DTC matched
	0x02	DEM_FILTERED_WRONG_DTCKIND DTC kind wrong
Description:	Used to return the status of Dem_GetNextFilteredDTC.	

6.2.2.5 Dem_ReturnClearDTCType

Type:	uint8	
Range:	0x00	DEM_CLEAR_OK DTC successfully cleared
	0x01	DEM_CLEAR_WRONG_DTC Wrong DTC
	0x02	DEM_CLEAR_WRONG_DTCORIGIN Wrong DTC origin
	0x03	DEM_CLEAR_WRONG_DTCKIND DTC kind wrong
	0x04	DEM_CLEAR_FAILED DTC not cleared
Description:	Used to return the status of Dem_ClearDTC**.**	

6.2.2.6 Dem_ReturnControlDTCStorageType

Type:	unit8	
Range:	0x00	DEM_CONTROL_DTC_STORAGE_OK DTC storage control successful
	0x01	DEM_CONTROL_DTC_STORAGE_N_OK DTC storage control not successful
	0x02	DEM_CONTROL_DTC_WRONG_DTCGROUP DTC storage control not successful because group of DTC was wrong
Description:	Used to return the status of Dem_DisableDTCStorage and Dem_EnableDTCStorage.	

6.2.2.7 Dem_ReturnControlEventUpdateType

Type:	unit8	
Range:	0x00	DEM_CONTROL_EVENT_UPDATE_OK Event storage control successful
	0x01	DEM_CONTROL_EVENT_UPDATE_N_OK Event storage control not successful
	0x02	DEM_CONTROL_EVENT_WRONG_DTCGROUP Event storage control not successful because group of DTC was wrong
Description:	Used to return the status of Dem_DisableEventStatusUpdate and Dem_EnableEventStatusUpdate.	

6.2.2.8 Dem_ReturnGetDTCOfFreezeFrameRecordType

Type:	unit8	
Range:	0x00	DEM_GET_DTCOFFF_OK DTC successfully returned
	0x01	DEM_GET_DTCOFFF_WRONG_RECORD Wrong record
	0x02	DEM_GET_DTCOFFF_NO_DTC_FOR_RECORD No DTC for record available
	0x03	DEM_GET_DTCOFFF_WRONG_DTCKIND DTC kind wrong
Description:	Used to return the status of Dem_GetDTCOfFreezeFrameRecord.	

6.2.2.9 Dem_ReturnGetFreezeFrameDataIdentifierByDTCType

Type:	unit8

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

6.2.2.10 Dem_ReturnGetExtendedDataRecordByDTCType

Type:	uint8	
Range:	0x00	DEM_RECORD_OK Extended data record successfully returned
	0x01	DEM_RECORD_WRONG_DTC Wrong DTC
	0x02	DEM_RECORD_WRONG_DTCORIGIN Origin wrong
	0x03	DEM_RECORD_WRONG_DTCKIND DTC kind wrong
	0x04	DEM_RECORD_WRONG_NUMBER Record number wrong
	0x05	DEM_RECORD_WRONG_BUFFERSIZE Provided buffer to small
Description:	Used to return the status of Dem_GetExtendedDataRecordByDTC.	

6.2.2.11 Dem_ReturnGetDTCByOccurrenceTimeType

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

6.2.2.12 Dem_ReturnGetDTCOfEventType

Type:	uint8	
Range:	0x00	DEM_GET_DTCOFEVENT_OK DTC successfully returned
	0x01	DEM_GET_DTCOFEVENT_WRONG_EVENTID Wrong EventId
	0x02	DEM_GET_DTCOFEVENT_WRONG_DTCKIND DTC kind wrong
Description:	Used to return the status of Dem_GetDTCOfEvent.	

6.2.2.13 Dem_ReturnGetFreezeFrameDataByDTCType

Type:	uint8	
Range:	0x00	DEM_GET_FFDATABYDTC_OK FreezeFrame data successfully returned
	0x01	DEM_GET_FFDATABYDTC_WRONG_DTC Wrong DTC
	0x02	DEM_GET_FFDATABYDTC_WRONG_ DTCORIGIN Wrong DTC origin
	0x03	DEM_GET_FFDATABYDTC_WRONG_DTCKIND DTC kind wrong
	0x04	DEM_GET_FFDATABYDTC_WRONG_ RECORDNUMBER Wrong Record Number
	0x05	DEM_GET_FFDATABYDTC_WRONG_DATAID Wrong DataID
	0x06	DEM_GET_FFDATABYDTC_WRONG_ BUFFERSIZE provided buffer size to small
Description:	Used to return the status of Dem_GetFreezeFrameDataByDTC.	

6.2.2.14 Dem_ReturnGetSizeOfExtendedDataRecordByDTCType

Type:	uint8	
Range:	0x00	DEM_GET_SIZEOFEDRBYDTCTYPE_OK Size successfully returned.
	0x01	DEM_GET_SIZEOFEDRBYDTCTYPE_WRONG_ DTC Wrong DTC

	0x02	DEM_GET_SIZEOFEDRBYDTCTYPE_WRONG_DTCORIGIN Wrong DTC origin
	0x03	DEM_GET_SIZEOFEDRBYDTCTYPE_WRONG_DTCKIND DTC kind wrong
	0x04	DEM_GET_SIZEOFEDRBYDTCTYPE_WRONG_RECORDNUMBER Wrong Record Number
Description:	Used to return the status of Dem_GetSizeOfExtendedDataRecordByDTC.	

6.2.2.15 Dem_ReturnGetSizeOfFreezeFrameType

Type:	uint8	
Range:	0x00	DEM_GET_SIZEOFFREEZEFRAMETYPE_OK Size successfully returned.
	0x01	DEM_GET_SIZEOFFREEZEFRAMETYPE_WRONG_DTC Wrong DTC
	0x02	DEM_GET_SIZEOFFREEZEFRAMETYPE_WRONG_DTCORIGIN Wrong DTC origin
	0x03	DEM_GET_SIZEOFFREEZEFRAMETYPE_WRONG_DTCKIND DTC kind wrong
	0x04	DEM_GET_SIZEOFFREEZEFRAMETYPE_WRONG_RECORDNUMBER Wrong Record Number
Description:	Used to return the status of Dem_GetSizeOfFreezeFrame.	

6.2.2.16 Dem_ReturnGetSeverityOfDTCType

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

6.2.2.17 Dem_ReturnGetViewIDOfDTCType

Type:	uint8	
Range:	`0x00`	DEM_VIEWID_OK Status of ViewID is OK
	`0x02`	DEM_VIEWID_WRONG_DTC Wrong DTC
	`0x03`	DEM_VIEWID_WRONG_DTCKIND DTC kind wrong
Description:	Used to return the status of Dem_GetViewIDOfDTC.	

6.3 Function definitions

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

6.3.1 Dem_GetVersionInfo

Service name:	Dem_GetVersionInfo	

Syntax:	`void Dem_GetVersionInfo (`
	`  Std_VersionInfoType *versioninfo`
	`)`	

Service ID [hex]:	0	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	yes	
Parameters (in):	none	
Parameters (out):	`versioninfo`	Pointer to where to store the version information of this module.
Return value:	none	
Description:	**Dem110:** This service returns the version information of this module. The version information includes: - Module Id - Vendor Id - Vendor specific version numbers (BSW00407). **Dem111:** This function shall be pre compile time configurable `On/Off` by the configuration parameter: `DEM_VERSION_INFO_API` Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.	
Caveats:	`--`	
Configuration:	`--`	

6.3.2 Interface ECU State Manager ⇔ DEM

6.3.2.1 Dem_PreInit

Service name:	Dem_PreInit
Syntax:	`void Dem_PreInit(void)`
Service ID:	1
Sync/Async:	Synchronous
canBeInvoked-Concurrently:	no
Description:	This function shall be used to initialize the internal states necessary to process events reported by BSWs by using Dem_ReportError. 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.
Caveats:	--
Configuration:	--

6.3.2.2 Dem_Init

Service name:	Dem_Init
Syntax:	`void Dem_Init(void)`
Service ID:	2
Sync/Async:	Synchronous
canBeInvoked-Concurrently:	no
Description:	**Dem065:** This function 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:	The DEM is not functional until this function has been called.
Configuration:	--

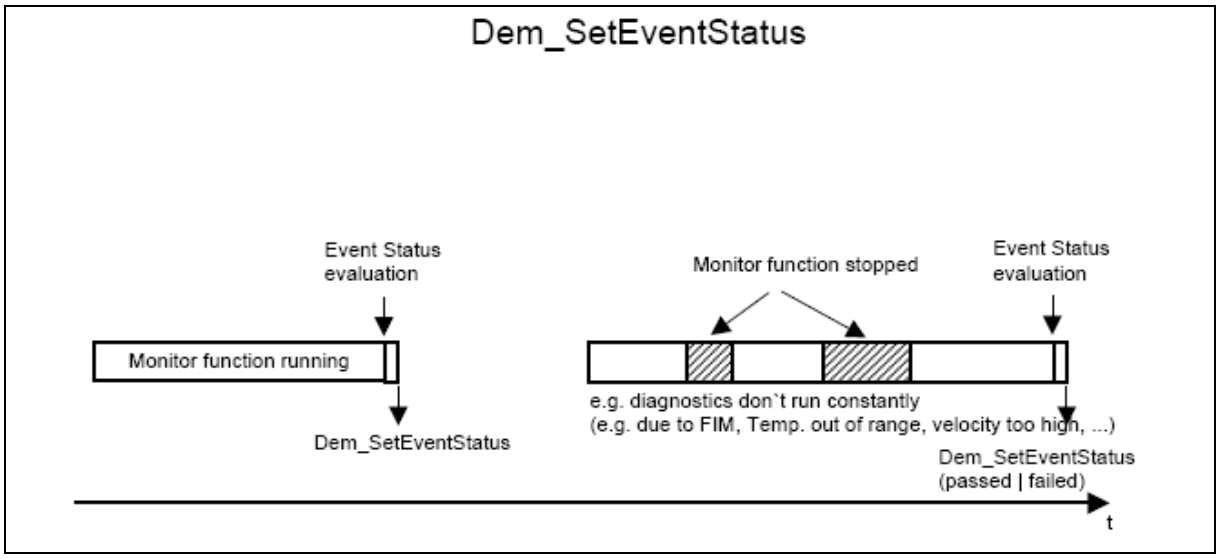
6.3.2.3 Dem_Shutdown

Service name:	Dem_Shutdown
Syntax:	`void Dem_Shutdown(void)`
Service ID:	3
Sync/Async:	Synchronous
canBeInvoked-Concurrently:	no
Description:	**Dem102:** This function is used to finalize all pending operations in the DEM to prepare the internal states and event data for transfer to the NVRAM.
Caveats:	Once this function has been executed no further updates are applied to the DEM internal event data.
Configuration:	--

6.3.3 Interface SW-Components via RTE ⇔ DEM

6.3.3.1 Dem_SetEventStatus

Service name:	Dem_SetEventStatus	
Syntax:	`Std_ReturnType Dem_SetEventStatus ( Dem_EventIdType EventId, Dem_EventStatusType EventStatus )`	
Service ID:	4	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	yes	
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 (out):	None	
Return value:	`Std_ReturnType`	E_OK set of event status was successful E_NOT_OK set of event status failed
Description:	Service for reporting the Event Status to the DEM. This service shall be used to report an Event Status as soon a new test result is available. This Service stores the event to the Event Memory. API is called from the Monitor Function. [ref. to Dem001] No API parameter checks required.	
Caveats:	DEM configuration during integration of Monitor Functions is system specific.	
Configuration:	--	



6.3.3.2 Dem_ResetEventStatus

Service name:	Dem_ResetEventStatus	
Syntax:	`Std_ReturnType Dem_ResetEventStatus ( Dem_EventIdType EventId )`	
Service ID:	5	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	yes	
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 (out):	None	
Return value:	`Std_ReturnType`	E_OK reset of event status was successful E_NOT_OK reset of event status failed
Description:	This service shall be used to reset the Event Status stored in the Event Memory in the DEM, without the usage of API Dem_SetEventStatus(EventId, Passed), because no new test result is available at this time. With this API the status bit "Failed" defined by Dem_EventStatusExtendedType is set to 0. API is called by Monitor Function. Refer to ISO14229: DTC Status Bit Definition, Table D.14, Bit0 Test failed. Dem_ResetEventStatus does not influence the status bit 6 ("testNotCompletedThisMonitoringCycle"). [ref. to Dem001] No API parameter checks required.	
Caveats:	DEM configuration during integration of Monitor Functions is system specific.	
Configuration:	--	

6.3.3.3 Dem_PrestoreFreezeFrame

Service name:	Dem_PrestoreFreezeFrame	
Syntax:	`Std_ReturnType Dem_PrestoreFreezeFrame ( Dem_EventIdType EventId )`	
Service ID:	6	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	yes	
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 (out):	None	
Return value:	`Std_ReturnType`	E_OK PreStoreFreezeFrame was successful E_NOT_OK PreStoreFreezeFrame failed
Description:	This API call is used to capture the FreezeFrame data for a specific EventId before the Monitor Function reports the event status DEM_EVENT_STATUS_FAILED to the DEM by calling Dem_SetEventStatus (e.g. rapid changing of environmental data during running failure monitoring phase). If the DEM does not receive any request to pre-store a FreezeFrame, FreezeFrame capture is linked to the API call Dem_SetEventStatus. This API call triggers the FreezeFrame storage. If Dem_SetEventStatus(EventId, Passed) is called the corresponding pre-stored FreezeFrame is discarded (same behaviour like Dem_ClearPrestoredFreezeFrame). The API call Dem_ResetEventStatus does not influence the pre-stored FreezeFrame. [ref. to Dem001] API is called from Monitor Function. No API parameter checks required.	
Caveats:	DEM configuration during integration of Monitor Functions is system specific.	
Configuration:	During configuration the DEM the capability of pre-store functionality for the required event has to be defined.	

6.3.3.4 Dem_ClearPrestoredFreezeFrame

Service name:	Dem_ClearPrestoredFreezeFrame	
Syntax:	`Std_ReturnType Dem_ClearPrestoredFreezeFrame ( Dem_EventIdType EventId )`	
Service ID:	7	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	yes	
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 (out):	None	
Return value:	`Std_ReturnType`	E_OK ClearPreStoreFreezeFrame was successful E_NOT_OK ClearPreStoreFreezeFrame failed
Description:	**Dem050:** The API shall be called to delete or release the prestored FreezeFrame for specific EventId. If the API Dem_SetEventStatus (passed|failed) is called it has the same effect – that means it's not necessary to call the API Dem_ClearPrestoredFreezeFrame directly after Dem_SetEventStatus.	
Caveats:	DEM configuration during integration of Monitor Functions is system specific.	
Configuration:	During integration in the DEM the capability of pre-store functionality for the required event has to be defined.	

6.3.3.5 Dem_SetOperationCycleState

Service name:	Dem_SetOperationCycleState	
Syntax:	`Std_ReturnType Dem_SetOperationCycleState ( Dem_OperationCycleIdType OperationCycleId, Dem_OperationCycleStateType CycleState )`	
Service ID:	8	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	no	
Parameters (in):	`OperationCycleId`	Identification of operation cycle, like power cycle, driving cycle, …
	`CycleState`	`uint8 (Start, End)`

Parameters (in/out):	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:	**Dem047:** DEM shall be called by the SW- Component as soon as it detects the status change of the CycleState for the Operation Cycle. This function can be DEM internal for DEM self calculated operation cycles.	
Caveats:	None	
Configuration:	The OperationCycleId shall be configured in view of sender receiver communication.	

6.3.3.6 Dem_GetEventStatus

Service name:	Dem_GetEventStatus	
Syntax:	`Std_ReturnType Dem_GetEventStatus ( Dem_EventIdType EventId, Dem_EventStatusExtendedType *EventStatusExtended )`	
Service ID:	10	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	yes	
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 (in/out):	None	
Parameters (out):	`EventStatus Extended`	For explanation see Dem_EventStatusExtendedType definition chapter
Return value:	`Std_ReturnType`	E_OK: get of event status was successful E_NOT_OK: get of event status failed
Description:	**Dem051:** This API shall be used to read the event status from the DEM. This API is provided to be used by SW-Components or other basic software modules e.g. FIM. Bit 0 is set to 1 if the last event status update by API 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 is set if at least one time the API Dem_SetEventStatus (passed | failed) is called with failed this cycle. Intended to be used for defaults reset only at next key on. Bit 2 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 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 and 5 is currently not supported but can be extended if needed for interface Xxx_Dem_TriggerOnEventStatus. Bit 6 is set if at least one time the API Dem_SetEventStatus (passed | failed) is called within this cycle (the usage of different cycles is application-specific, if only one cycle is the differentiation is obsolete). Bit 7 reports the status of any warning indicators associated with a particular DTC. For the DCM the API Dem_GetStatusOfDTC is used.	
Caveats:	None	

Configuration:	EventId

6.3.3.7 Dem_GetEventFailed

Service name:	Dem_GetEventFailed	
Syntax:	`Std_ReturnType Dem_GetEventFailed ( Dem_EventIdType EventId, boolean *EventFailed )`	
Service ID:	11	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	yes	
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 (in/out):	None	
Parameters (out):	`EventFailed`	TRUE – Last Failed FALSE – not Last Failed
Return value:	`Std_ReturnType`	E_OK: get of "EventFailed" was successful E_NOT_OK: get of "EventFailed" was not successful
Description:	**Dem052:** This API shall be used to read Bit 0 of Dem_EventStatusExtendedType from the DEM. For the DCM the API Dem_GetStatusOfDTC is used.	
Caveats:	None	
Configuration:	EventId	

6.3.3.8 Dem_GetEventTested

Service name:	Dem_GetEventTested	
Syntax:	`Std_ReturnType Dem_GetEventTested ( Dem_EventIdType EventId, Boolean *EventTested )`	
Service ID:	12	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	yes	
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 (in/out):	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:	**Dem053:** This API shall be used to read negated Bit 6 of Dem_EventStatusExtendedType from the DEM. For the DCM the API Dem_GetStatusOfDTC is used.	
Caveats:	None	
Configuration:	EventId.	

6.3.3.9 Dem_GetDTCOfEvent

Service name:	Dem_GetDTCOfEvent	
Syntax:	`Std_ReturnType Dem_GetDTCOfEvent ( Dem_EventIdType EventId, Dem_DTCKindType DTCKind, Dem_DTCType *DTCOfEvent, Dem_ReturnGetDTCOfEventType *StatusDTCOfEvent )`	
Service ID:	13	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	yes	
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 OBD-relevant or non OBD-relevant
Parameters (out):	`DTCOfEvent`	Receives the DTC value returned by the function. If the return value of the function is other than `OK` this parameter does not contain valid data.
	`StatusDTCOfEvent`	Status of the operation of type Dem_ReturnGetDTCOfEventType
Return value:	`Std_ReturnType`	E_OK: get of DTC was successful E_NOT_OK: get of DTC failed
Description:	Gets the DTC which is mapped to EventId by DEM Configuration.	
Caveats:	-	
Configuration:	Mapping of Events to DTCs is configured in DEM. Mapping is “n to 1” or “1 to n”.	

6.3.3.10 Dem_SetValueByOemId

Service name:	Dem_SetValueByOemId	
Syntax:	`Std_ReturnType Dem_SetValueByOemId ( uint16 OemID, uint8 *DemDataValueByDataIDBuffer, uint8 DemDataValueByDataIDBufferLength )`	
Service ID:	56	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	no	
Parameters (in):	`OemID`	This OEM specific parameter identifies a data value the DEM requires for internal processing, e.g. vehicle speed or mileage.
	`DemDataValueByDataID BufferLength`	Data length of the value to be set
Parameters (out):	`DemDataValueByDataID Buffer`	Pointer to the buffer with the value to be set
Return value:	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:	This API shall be used to 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.	
Caveats:	-	
Configuration:	`Required configuration parameters:` • `OemID` • `real Name of the assigned value`	

6.3.3.11 Dem_SetEnableCondition

Service name:	Dem_SetEnableCondition	
Syntax:	`Std_ReturnType Dem_SetEnableCondition ( uint8 EnableConditionID, boolean ConditionFulfilled )`	
Service ID:	57	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	no	
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).

Return value:	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:	This API shall be used to 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. This API is **optional** and depends on the automotive manufacturer.
Caveats:	optional
Configuration:	`Required configuration parameters per event:` • `EnableConditionID` • `EnableConditionStatus`

6.3.3.12 Dem_GetFaultDetectionCounter

Service name:	Dem_GetFaultDetectionCounter	
Syntax:	`Std_ReturnType Dem_GetFaultDetectionCounter ( Dem_EventIdType EventId, Dem_FaultDetectionCounterType *EventIdFaultDetectionCounter )`	
Service ID:	62	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	no	
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 (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 `OK` this parameter does not contain valid data.
Return value:	`Std_ReturnType`	E_OK: request of severity was successful E_NOT_OK: request of severity failed
Description:	The API shall be used by SW-C to request the current Fault Detection Counter for a given EventID.	
Caveats:	--	
Configuration:	--	

6.3.3.13 Dem_GetIndicatorStatus

Service name:	Dem_GetIndicatorStatus	
Syntax:	`Std_ReturnType Dem_GetIndicatorStatus ( Dem_IndicatorIdType IndicatorId, Dem_IndicatorStatusType *IndicatorStatus )`	
Service ID:	41	
Sync/Async:	Synchronous	
canBeInvoked-Concurrently:	no	
Parameters (in):	`IndicatorId`	Number of indicator
Parameters (out):	`IndicatorStatus`	Status of the indicator, like on, off, blinking.
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Description:	**Dem046:** This function shall be used to read the indicator-status derived from the event status as a summary of all assigned events.	
Caveats:	--	
Configuration:	The assignment for the Dem_IndicatorId to indicator has to be done. Examples for indicators: lamps, different text messages, icons, …	

6.3.4 Interface BSW-Components ⇔ DEM

6.3.4.1 Dem_ReportErrorStatus

Service name:	Dem_ReportErrorStatus	
Syntax:	`void Dem_ReportErrorStatus ( Dem_EventIdType EventId, Dem_EventStatusType EventStatus )`	
Service ID:	15	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	yes	
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 (out):	-	-
Return value:	-	-
Description:	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.	
Caveats:	--	
Configuration:	The size of the buffer queue needs to be configured (ref. to `DEM_BSW_ERROR_BUFFER_SIZE`)	

6.3.5 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.

6.3.5.1 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.

6.3.5.1.1 Dem_SetDTCFilter

Service name:	Dem_SetDTCFilter	
Syntax:	`Dem_ReturnSetDTCFilterType Dem_SetDTCFilter ( Dem_DTCStatusMaskType DTCStatusMask, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, Dem_FilterWithSeverityType FilterWithSeverity, Dem_DTCSeverityType DTCSeverity, Dem_FilterForFaultDetectionCounterType FilterForFaultDetectionCounter )`	
Service ID:	19	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTCStatusMask`	According ISO14229-1 StatusOfDTC Values: 0x00: Report all supported DTCs 0x01…0xFF: Match DTCStatusMask as defined in ISO14229-1
	`DTCKind`	Defines the functional group of DTCs to be reported (e.g. all DTC, OBD-relevant DTC)
	`DTCOrigin`	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.
	`FilterWithSeverity`	This flag defines whether severity information (ref. to parameter below) shall be used for filtering. This is to allow for coexistence of DTCs with and without severity information.
	`DTCSeverity`	This parameter contains the DTCSeverityMask according to ISO14229-1.
	`FilterForFault DetectionCounter`	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.

Return value:	Status of the operation of type Dem_ReturnSetDTCFilterType
Description:	**Dem057:** This API shall be used to set the filter mask over all DTCs for the API's Dem_GetNextFilteredDTC. The function resets the internal counter to the first event that matches the filter settings.
Caveats:	--
Configuration:	--

6.3.5.1.2 Dem_SetDTCFilterForRecords

Service name:	Dem_SetDTCFilterForRecords	
Syntax:	`Dem_ReturnSetDTCFilterType Dem_SetDTCFilterForRecords ( uint16 *NumberOfFilteredRecords )`	
Service ID:	63	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`none`	
Parameters (out):	`NumberOfFiltered Records`	Number of snapshot records currently stored in the event memory.
Return value:	Status of the operation of type Dem_ReturnSetDTCFilterType	
Description:	The filtered snapshot records can be retrieved by the function Dem_GetNextFilteredRecord. This filter always belongs to primary memory.	
Caveats:	--	
Configuration:	--	

6.3.5.1.3 Dem_SetViewFilter

Service name:	Dem_SetViewFilter	
Syntax:	`Dem_ReturnSetViewFilterType Dem_SetViewFilter ( Dem_ViewIdType ViewId )`	
Service ID:	20	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`ViewId`	The ViewId is a parameter used to select a specific view (ref. to Dem_ViewIdType).
Return value:	Status of the operation of type Dem_ReturnSetViewFilterType.	
Description:	**Dem058:** Dem_SetViewFilter sets a mask to process only the events of a functional addressable function with the following DCM <-> DEM API calls (Chapter 6.3.5). To change the view the function Dem_SetViewFilter must be called again with another ViewId. The chosen ViewId is reset to the default value (0xFF → all functional groups are visible) inside the DEM when Dem_SetDTCFilter and Dem_Shutdown is called. This function can be used optional 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. This function shall be used for function oriented diagnostics on ECUs with multiple functions.
Caveats:	--
Configuration:	The assignment of an EventId to a specific view has to be configured / calibrated.

6.3.5.1.4 Dem_GetStatusOfDTC

Service name:	Dem_GetStatusOfDTC	
Syntax:	`Dem_ReturnGetStatusOfDTCType Dem_GetStatusOfDTC ( Dem_DTCType DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, Dem_DTCStatusMaskType *DTCStatus )`	
Service ID:	21	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTC`	For this DTC its status is requested
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant
	`DTCOrigin`	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.
Parameters (out):	`DTCStatus`	This parameter receives the status information of the requested DTC. If the return value of the function call is other than `OK` this parameter does not contain valid data.
Return value:	Status of the operation of type Dem_ReturnGetStatusOfDTCType.	
Description:	**Dem059:** This API shall be used to read the status of a DTC to the parameter DTCStatus according to ISO14229.	
Caveats:	--	
Configuration:	--	

If the DTC is not stored in one of the available event memories the parameter DTCOrigin is neglected. E.g. when DTC status is pending.

6.3.5.1.5 Dem_GetDTCStatusAvailabilityMask

Service name:	Dem_GetDTCStatusAvailabilityMask	
Syntax:	`Std_ReturnType Dem_GetDTCStatusAvailabilityMask ( Dem_DTCStatusMaskType *DTCStatusMask )`	
Service ID:	22	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (out):	`DTCStatusMask`	The value from type Dem_DTCStatusMaskType 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:	**Dem060:** The API shall be used to get the DTC Status availability mask. That means the DTC status information (according to ISO14229) supported by the DEM. Only supported bits can be used as filter parameters in the API Dem_SetDTCFilter	
Caveats:	--	
Configuration:	--	

6.3.5.1.6 Dem_GetNumberOfFilteredDTC

Service name:	Dem_GetNumberOfFilteredDTC	
Syntax:	`Std_ReturnType Dem_GetNumberOfFilteredDTC ( uint16 *NumberOfFilteredDTC )`	
Service ID:	23	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (out):	`NumberOfFilteredDTC`	The number of DTCs matching the defined status mask.
Return value:	`Std_ReturnType`	E_OK: get of number of DTC was successful E_NOT_OK: get of number of DTC failed
Description:	**Dem061:** The API shall be used to get the number of DTC matching the defined status mask. The DTC Status mask filter is set by the API Dem_SetDTCFilter.	
Caveats:	DTC filter has been set up properly before function call (Dem_SetDTCFilter).	
Configuration:	--	

6.3.5.1.7 Dem_GetNextFilteredDTC

Service name:	Dem_GetNextFilteredDTC	
Syntax:	`Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredDTC ( Dem_DTCType *DTC, Dem_DTCStatusMaskType *DTCStatus )`	
Service ID:	24	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`none`	
Parameters (out):	`DTC`	Receives the DTC value returned by the function. If the return value of the function is other than `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 `OK` this parameter does not contain valid data.
Return value:	Status of the operation to retrieve a DTC from the DEM.	
Description:	**Dem062:** The API shall be used to return the current DTC and its associated status from the DEM matching the filter criteria defined by the API call Dem_SetDTCFilter. After having returned the data the function skips to the next DTC matching the filter criteria. To receive all DTCs matching the filter criteria this function shall be called continuously until the return value of the function is “NoMatchingDTC”. 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).	
Caveats:	--	
Configuration:	--	

6.3.5.1.8 Dem_GetDTCByOccurrenceTime

Service name:	Dem_GetDTCByOccurrenceTime	
Syntax:	`Dem_ReturnGetDTCByOccurrenceTimeType Dem_GetDTCByOccurrenceTime ( Dem_DTCRequestType DTCRequest, Dem_DTCKindType DTCKind, Dem_DTCType *DTC )`	
Service ID:	25	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTCRequest`	This parameter defines the request type of the DTC.
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant
Parameters (out):	`DTC`	Receives the DTC value returned by the function. If the return value of the function is other than `OK` this parameter does not contain valid data.
Return value:	Status of the operation of type Dem_ReturnGetDTCByOccurrenceTimeType.	
Description:	**Dem043:** The API provides the capability to access specific events stored at certain important points in time, The API call will return one DTC according to the Dem_DTCRequestType. If no DTC is matching the requested point in time the function will return the appropriate operation status (ref. to 6.2.2.11) and the DTC value will be zero (0).	
Caveats:	--	
Configuration:	--	

6.3.5.1.9 Dem_GetViewIDOfDTC

Service name:	Dem_GetViewIDOfDTC	
Syntax:	`Dem_ReturnGetViewIDOfDTCType Dem_GetViewIDOfDTC ( Dem_DTCType DTC, Dem_DTCKindType DTCKind, Dem_ViewIdType *ViewId )`	
Service ID:	54	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTC`	This parameter defines the requested DTC.
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant

Parameters (out):	`ViewId`	The ViewId is a parameter used to select a specific view (ref. to Dem_ViewIdType).
Return value:	Status of the operation of type Dem_ReturnGetViewIDOfDTCType.	
Description:	The API provides the capability to get the according ViewID (e.g. wiper system, window lifter, …) of a specific DTC. The parameter ViewID is equivalent to the parameter FunctionalUnit in ISO14229-1 [9].	
Caveats:	--	
Configuration:	--	

6.3.5.1.10 Dem_GetNextFilteredRecord

Service name:	Dem_GetNextFilteredRecord	
Syntax:	`Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredRecord ( Dem_DTCType *DTC, uint8 *SnapshotRecord )`	
Service ID:	58	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`none`	
Parameters (out):	`DTC`	Receives the DTC value returned by the function. If the return value of the function is other than 0K this parameter does not contain valid data.
	`SnapshotRecord`	Snapshot Record Number for the reported DTC.

Return value:	Status of the operation to retrieve a DTC from the DEM.
Description:	The API shall be used to return the current DTC and its associated Snapshot Record numbers from the DEM matching the filter criteria defined by the API call Dem_SetDTCFilterForRecords. After having returned the data the function skips to the next Record matching the filter criteria. To receive all Records matching the filter criteria this function shall be called continuously until the return value of the function is “NoMatchingDTC”.
Caveats:	--
Configuration:	--

6.3.5.1.11 Dem_GetNextFilteredDTCAndFDC

Service name:	Dem_GetNextFilteredDTCAndFDC	
Syntax:	`Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredDTCAndFDC ( Dem_DTCType *DTC, Dem_FaultDetectionCounterType *DTCFaultDetectionCounter, )`	
Service ID:	59	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`none`	
Parameters (out):	`DTC`	Receives the DTC value returned by the function. If the return value of the function is other than `OK` this parameter does not contain valid data.
	`DTCFaultDetection Counter`	This parameter receives the Fault Detection Counter information of the requested DTC. If the return value of the function call is other than `OK` this parameter does not contain valid data.

Return value:	Status of the operation to retrieve a DTC from the DEM.
Description:	The API shall be used to return the current DTC and its associated Fault Detection Counter (FDC) from the DEM matching the filter criteria defined by the API call Dem_SetDTCFilter. After having returned the data the function skips to the next DTC matching the filter criteria. To receive all DTCs matching the filter criteria this function shall be called continuously until the return value of the function is “NoMatchingDTC”.
Caveats:	--
Configuration:	--

6.3.5.1.12 Dem_GetTranslationType

Service name:	Dem_GetTranslationType	
Syntax:	`Std_ReturnType Dem_GetTranslationType ( Dem_DTCTranslationFormatType *TranslationFormat )`	
Service ID:	60	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`none`	
Parameters (out):	`TranslationFormat`	The translation provides the configurted translation format
Return value:	Status of the operation of type Dem_ReturnGetViewIDOfDTCType.	
Description:	The API provides the capability to get the configured translation format of the ECU.	
Caveats:	--	
Configuration:	--	

6.3.5.1.13 Dem_GetSeverityOfDTC

Service name:	Dem_GetSeverityOfDTC	
Syntax:	`Std_ReturnType Dem_GetSeverityOfDTC ( Dem_DTCType DTC, Dem_DTCOriginType DTCOrigin, Dem_DTCSeverityType *DTCSeverity, Dem_ReturnGetSeverityOfDTCType *StatusGetSeverity )`	
Service ID:	14	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	yes	
Parameters (in):	`DTC`	The Severity assigned to this DTC should be returned
	`DTCOrigin`	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.

Parameters (out):	`DTCSeverity`	This parameter contains the DTCSeverityMask according to ISO14229-1.
	`StatusGetSeverity`	Status of the operation of type Dem_ReturnGetSeverityOfDTCType.
Return value:	`Std_ReturnType`	E_OK: request of severity was successful E_NOT_OK: request of severity failed
Description:	Gets Severity of the requested DTC.	
Caveats:	Dem_DTCKindType not needed, because Severity is only available for ISO14229-1 DTCs	
Configuration:	--	

6.3.5.2 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].

6.3.5.2.1 Dem_DisableDTCRecordUpdate

Service name:	Dem_DisableDTCRecordUpdate	
Syntax:	`Std_ReturnType Dem_DisableDTCRecordUpdate(void)`	
Service ID:	26	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	None	
Parameters (out)	None	
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	**Dem068**: (optional) This function shall be used 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. This function is used to prevent the DEM 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. DTC status information update is not affected by this function.	
Caveats:	--	
Configuration:	--	

6.3.5.2.2 Dem_EnableDTCRecordUpdate

Service name:	Dem_EnableDTCRecordUpdate	
Syntax:	`Std_ReturnType Dem_EnableDTCRecordUpdate(void)`	
Service ID:	27	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	None	
Parameters (out)	None	
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	**Dem069**:(optional)) The function Dem_EnableDTCRecordUpdate is the counterpart to the function Dem_DisableDTCRecordUpdate. It shall be called after the FreezeFrame and extended data record were protected by the function Dem_DisableDTCRecordUpdate and after the access by subsequent API-calls is finished. It is called to release the data contained in this record so that the data can be accessed or manipulated by the external application, e.g. DCM, again.	
Caveats:	--	
Configuration:	--	

6.3.5.2.3 Dem_GetDTCOfFreezeFrameRecord

Service name:	Dem_GetDTCOfFreezeFrameRecord	
Syntax:	`Dem_ReturnGetDTCOfFreezeFrameRecordType Dem_GetDTCOfFreezeFrameRecord ( uint8 RecordNumber, Dem_DTCOriginType DTCOrigin, Dem_DTCKindType DTCKind, Dem_DTCType *DTC )`	
Service ID:	28	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
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.
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant
Parameters (out):	`DTC`	Receives the DTC value returned by the function. If the return value of the function is other than `OK` this parameter does not contain valid data.
Return value:	Status of the operation of type Dem_ReturnGetDTCOfFreezeFrameRecordType.	
Description:	**Dem070:** This API shall be used to return the DTC associated with the FreezeFrame selected via its absolute record number.	
Caveats:	The record number has to be unique throughout the whole ECU. This function is only required for OBD-relevant ECUs.	
Configuration:	--	

6.3.5.2.4 Dem_GetFreezeFrameDataByDTC

Service name:	Dem_GetFreezeFrameDataByDTC	
Syntax:	`Dem_ReturnGetFreezeFrameDataByDTCType Dem_GetFreezeFrameDataByDTC ( Dem_DTCType DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, uint8 RecordNumber, uint16 DataId, uint8 *DestBuffer, uint8 *BufSize )`	
Service ID:	29	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTC`	This is the DTC the FreezeFrame is assigned to.
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant
	`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.
	`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.
Parameter (in|out):	`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.
Parameter (out):	`DestBuffer`	This parameter contains a byte pointer that points to the buffer to which the FreezeFrame data shall be written.
Return value:	Status of the operation of type Dem_ReturnGetFreezeFrameDataByDTCType.	
Description:	**Dem071:** This function shall be used to copy a specific PID/DataId of a FreezeFrame selected via the associated DTC number and an optional FreezeFrame RecordNumber to the destination buffer. Shall be transmitted as complete record with format PID followed by data, PID – data, ... The DCM does not know the DEM internal structure so it asks 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.	
Caveats:	Assumptions: The software engineer implementing the DEM shall be an expert in OBD-relevant onboard diagnostics and familiar with the services as defined in ISO15031-5.	
Configuration:	--	

6.3.5.2.5 Dem_GetFreezeFrameDataIdentifierByDTC

Service name:	Dem_GetFreezeFrameDataIdentifierByDTC	
Syntax:	`Dem_ReturnGetFreezeFrameDataIdentifierByDTCType Dem_GetFreezeFrameDataIdentifierByDTC ( Dem_DTCType DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, uint8 RecordNumber, uint8 *ArraySize, const uint16 **DataId )`	
Service ID:	30	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTC`	This is the DTC the FreezeFrame is assigned to.
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant
	`DTCOrigin`	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.
	`RecordNumber`	This parameter is a unique identifier for a FreezeFrame record as defined in ISO15031-5 and ISO14229-1.
Parameters (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:	Status of the operation of type Dem_ReturnGetFreezeFrameDataIdentifierByDTCType.	
Description:	**Dem073:** This function shall be used to return the data identifiers and the number of data identifiers of a FreezeFrame which belong to a specific DTC.	
Caveats:	--	
Configuration:	--	

6.3.5.2.6 Dem_GetSizeOfFreezeFrame

Service name:	Dem_GetSizeOfFreezeFrame	
Syntax:	`Dem_ReturnGetSizeOfFreezeFrameType Dem_GetSizeOfFreezeFrame ( Dem_DTCType DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, uint8 RecordNumber uint16 *SizeOfFreezeFrame )`	
Service ID:	31	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTC`	This is the DTC the FreezeFrame is assigned to.
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant
	`DTCOrigin`	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.
	`RecordNumber`	This parameter is a unique identifier for a FreezeFrame record as defined in ISO15031-5 and ISO14229-1.

Parameter (out):	`SizeOfFreezeFrame`	Number of bytes in the requested FreezeFrame.
Return value:	Status of the operation of type Dem_ReturnGetSizeOfFreezeFrameType	
Description:	**Dem074:** The API shall be used to return the size of the requested FreezeFrame. This size only represents the number of user data bytes (pure FreezeFrame data) and does not contain any FreezeFrame structure information.	
Caveats:	--	
Configuration:	--	

6.3.5.2.7 Dem_GetExtendedDataRecordByDTC

Service name:	Dem_GetExtendedDataRecordByDTC	
Syntax:	`Dem_ReturnGetExtendedDataRecordByDTCType Dem_GetExtendedDataRecordByDTC ( Dem_DTCType DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, uint8 ExtendedDataNumber, uint8 *DestBuffer, uint8 *BufSize )`	
Service ID:	32	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTC`	This is the DTC the 'Extended Data Record' is assigned to.
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant
	`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.
	`ExtendedDataNumber`	Identification of requested Extended data record. The requested record is copied to the destination buffer.
Parameter (in|out):	`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.
Parameter (out):	`DestBuffer`	This parameter contains a byte pointer that points to the buffer to which the Extended Data shall be written.
Return value:	Status of the operation of type Dem_ReturnGetExtendedDataRecordByDTCType.	
Description:	**Dem075:** This function shall be used to return the complete Extended Data Record for the requested DTC. The format of the data is raw hexadecimal values and is not standardized to comply with predefined scaling methods.	
Caveats:	--	
Configuration:	Values of 'Extended Data Record' have to be defined.	

6.3.5.2.8 Dem_GetSizeOfExtendedDataRecordByDTC

Service name:	Dem_GetSizeOfExtendedDataRecordByDTC	
Syntax:	`Dem_ReturnGetSizeOfExtendedDataRecordByDTCType Dem_GetSizeOfExtendedDataRecordByDTC ( Dem_DTCType DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin, uint8 ExtendedDataNumber, uint16 *SizeOfExtendedDataRecord )`	
Service ID:	33	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTC`	This is the DTC the 'Extended Data Record' is assigned to.
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant
	`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.
	`ExtendedDataNumber`	Identification of requested Extended data record. The requested record is copied to the destination buffer.
Parameter (out):	`SizeOfExtendedDataRecord`	Pointer to Size of the requested data record
Return value:	Status of the operation of type Dem_ReturnGetSizeOfExtendedDataRecordByDTCType	
Description:	**Dem076:** The API shall be used to return the size of the requested 'Extended Data Record' frame. This size only represents the number of user data bytes stored in the 'Extended Data Record'.	
Caveats:	--	
Configuration:	Values of 'Extended Data Record' have to be defined.	

6.3.5.3 Clear DTC information

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

6.3.5.3.1 Dem_ClearDTC

Service name:	Dem_ClearDTC	
Syntax:	`Dem_ReturnClearDTCType Dem_ClearDTC ( Dem_DTCType DTC, Dem_DTCKindType DTCKind, Dem_DTCOriginType DTCOrigin )`	
Service ID:	34	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
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 OBD-relevant or non OBD-relevant
	`DTCOrigin`	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be cleared from.
Return value:	Status of the operation of type Dem_ReturnClearDTCType.	
Description:	**Dem077:** This API shall be used to clear ♦ all the event status related the specified DTC ♦ all associated event memory entries for these events (environmental and/or FreezeFrame data, …)	
Caveats:	--	
Configuration:	The initialization of the corresponding Monitor Function (DTC → EventId → Xxx_DemInitMonitor{*EventId*}) is managed by Xxx_Dem_InitMonitor{EventId}.	

6.3.5.4 Control DTC storage

This section defines the API-calls to enable and disable DTC storage in the DEM.

6.3.5.4.1 Dem_DisableDTCStorage

Service name:	Dem_DisableDTCStorage	
Syntax:	`Dem_ReturnControlDTCStorageType Dem_DisableDTCStorage ( Dem_DTCGroupType DTCGroup, Dem_DTCKindType DTCKind )`	
Service ID:	36	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTCGroup`	Defines the group of DTC that shall be disabled to store in event memory.
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant

Return value:	Status of the operation of type Dem_ReturnControlDTCStorageType.
Description:	**Dem079:** This function shall be called to disable the storage of DTCs in the event memory (Dem035). DTC status information update is not affected by this function. This 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 ECU's 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.
Caveats:	--
Configuration:	--

6.3.5.4.2 Dem_EnableDTCStorage

Service name:	Dem_EnableDTCStorage	
Syntax:	`Dem_ReturnControlDTCStorageType Dem_EnableDTCStorage ( Dem_DTCGroupType DTCGroup, Dem_DTCKindType DTCKind )`	
Service ID:	37	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTCGroup`	Defines the group of DTC that shall be enabled to store in event memory.
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant

Return value:	Status of the operation of type Dem_ReturnControlDTCStorageType.
Description:	**Dem080:** This function shall be called to enable the storage of DTCs in the event memory (Dem035). See also Dem_DisableDTCStorage.
Caveats:	--
Configuration:	--

6.3.5.4.3 Dem_DisableEventStatusUpdate

Service name:	Dem_DisableEventStatusUpdate	
Syntax:	`Dem_ReturnControlEventUpdateType Dem_DisableEventStatusUpdate ( Dem_DTCGroupType DTCGroup, Dem_DTCKindType DTCKind )`	
Service ID:	38	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTCGroup`	Defines the group of DTC that shall be disabled to update event status.
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant

Return value:	Status of the operation of type Status of the operation of type Dem_ReturnControlDTCStorageType
Description:	**Dem081:** This function shall be called to disable the update of the event status. This API has influence only on the execution of the API's Dem_SetEventStatus and Dem_ResetEventStatus. This is defined by configuration (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. This feature / API-call 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.).
Caveats:	--
Configuration:	Depending on configuration within the API Dem_DisableEventStatusUpdate the reaction on the event status is defined. For example: the execution of Dem_ResetEventStatus is possible also during this phase.

6.3.5.4.4 Dem_EnableEventStatusUpdate

Service name:	Dem_EnableEventStatusUpdate	
Syntax:	`Dem_ReturnControlEventUpdateType Dem_EnableEventStatusUpdate ( Dem_DTCGroupType DTCGroup, Dem_DTCKindType DTCKind )`	
Service ID:	39	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	`DTCGroup`	Defines the group of DTC that shall be enabled to update event status.
	`DTCKind`	This parameter defines the requested DTC, either OBD-relevant or non OBD-relevant

Return value:	Status of the operation of type Status of the operation of type Dem_ReturnControlDTCStorageType
Description:	**Dem082:** This function shall be called to enable the update of the event status (Dem034). See also Dem_DisableEventStatusUpdate.
Caveats:	--
Configuration:	--

6.3.5.5 Get DEM statistical data / legislated data / indicator status

6.3.5.5.1 Dem_GetMILStatus

Service name:	Dem_GetMILStatus	
Syntax:	`Std_ReturnType Dem_GetMILStatus ( Dem_IndicatorStatusType *MILStatus )`	
Service ID:	40	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	None	
Parameters (in/out):	None	
Parameters (out):	`MILStatus`	Returns the status of the MIL
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Description:	**Dem045:** This function shall be used to read the MIL-Status required for ISO15031-5 [10]	
Caveats:	--	
Configuration:	This function shall only be supported and implemented by OBD-relevant ECUs.	

6.3.5.5.2 Dem_GetOBDReadiness

Service name:	Dem_GetOBDReadiness	
Syntax:	`Std_ReturnType Dem_GetOBDReadiness ( Dem_DataByteType *DataByte_B, Dem_DataByteType *DataByte_C, Dem_DataByteType *DataByte_D )`	
Service ID:	42	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	None	
Parameters (in/out):	None	

Parameters (out):	`DataByte_B`	Data B of PID01h of ISO15031-5
	`DataByte_C`	Data C of PID01h of ISO15031-5
	`DataByte_D`	Data D of PID01h of ISO15031-5
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Description:	**Dem083:** This function shall be used to read the PID01h of ISO15031-5 Data B/C/D information	
Caveats:	--	
Configuration:	This function shall only be supported and implemented by OBD-relevant ECUs. The readiness group shall be configured in the DEM.	

6.3.5.5.3 Dem_GetDistanceMIL

Service name:	Dem_GetDistanceMIL	
Syntax:	`Std_ReturnType Dem_GetDistanceMIL ( Dem_DataByteType *DataByte_A, Dem_DataByteType *DataByte_B )`	
Service ID:	43	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	None	
Parameters (in/out):	None	
Parameters (out):	`DataByte_A`	Data A of PID21h of ISO15031-5
	`DataByte_B`	Data B of PID21h of ISO15031-5
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Description:	**Dem084:** This function shall be used to read the PID21h of ISO15031-5 Data A/B [10] information	
Caveats:	--	
Configuration:	This function shall only be supported and implemented by OBD-relevant ECUs.	

6.3.5.5.4 Dem_GetWarmupCycleDTCclear

Service name:	Dem_GetWarmupCycleDTCclear	
Syntax:	`Std_ReturnType Dem_GetWarmupCycleDTCclear ( Dem_DataByteType *DataByte_A )`	
Service ID:	44	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	None	
Parameters (in/out):	None	
Parameters (out):	`DataByte_A`	Data A of PID30h of ISO15031-5
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Description:	**Dem085:** This function shall be used to read the PID30h of ISO15031-5 Data A [10] information	
Caveats:	--	
Configuration:	This function shall only be supported and implemented by OBD-relevantECUs.	

6.3.5.5.5 Dem_GetDistanceDTCclear

Service name:	Dem_GetDistanceDTCclear	
Syntax:	`Std_ReturnType Dem_GetDistanceDTCclear ( Dem_DataByteType *DataByte_A, Dem_DataByteType *DataByte_B )`	
Service ID:	45	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	None	
Parameters (in/out):	None	
Parameters (out):	`DataByte_A`	Data A of PID31h of ISO15031-5
	`DataByte_B`	Data B of PID31h of ISO15031-5
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Description:	**Dem086:** This function shall be used to read the PID31h of ISO15031-5 Data A/B [10] information	
Caveats:	--	
Configuration:	This function shall only be supported and implemented by OBD-relevantECUs.	

6.3.5.5.6 Dem_GetMonitorStatus

Service name:	Dem_GetMonitorStatus	
Syntax:	`Std_ReturnType Dem_GetMonitorStatus ( Dem_DataByteType *DataByte_B, Dem_DataByteType *DataByte_C, Dem_DataByteType *DataByte_D )`	
Service ID:	46	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	None	
Parameters (in/out):	None	
Parameters (out):	`DataByte_B`	Data B of PID41h of ISO15031-5
	`DataByte_C`	Data C of PID41h of ISO15031-5
	`DataByte_D`	Data D of PID41h of ISO15031-5
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Description:	**Dem087:** This function shall be used to read the PID41h of ISO15031-5 Data B/C/D [10] information	
Caveats:	--	
Configuration:	This function shall only be supported and implemented by OBD-relevantECUs. The readiness group shall be configured in the DEM.	

6.3.5.5.7 Dem_GetTimeMIL

Service name:	Dem_GetTimeMIL	
Syntax:	`Std_ReturnType Dem_GetTimeMIL ( Dem_DataByteType *DataByte_A, Dem_DataByteType *DataByte_B )`	
Service ID:	47	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	None	
Parameters (in/out):	None	
Parameters (out):	`DataByte_A`	Data A of PID4Dh of ISO15031-5
	`DataByte_B`	Data B of PID4Dh of ISO15031-5
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Description:	**Dem088:** This function shall be used to read the PID4Dh of ISO15031-5 Data A/B [10] information	
Caveats:	--	
Configuration:	This function shall only be supported and implemented by OBD-relevantECUs.	

6.3.5.5.8 Dem_GetTimeDTCclear

Service name:	Dem_GetTimeDTCclear	
Syntax:	`Std_ReturnType Dem_GetTimeDTCclear ( Dem_DataByteType *DataByte_A, Dem_DataByteType *DataByte_B )`	
Service ID:	48	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameters (in):	None	
Parameters (in/out):	None	
Parameters (out):	`DataByte_A`	Data A of PID4Eh of ISO15031-5
	`DataByte_B`	Data B of PID4Eh of ISO15031-5
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Description:	**Dem089:** This function shall be used to read the PID4Eh of ISO15031-5 Data A/B [10] information	
Caveats:	--	
Configuration:	This function shall only be supported and implemented by OBD-relevant ECUs.	

6.4 Expected Interfaces

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

6.4.1 Mandatory Interfaces

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

API function	***Module***	***Description***
NvM_WriteBlock	NvM	Service to write a block to non volatile memory.
NvM_ReadBlock	NvM	Service to read a block from non volatile memory
SchM_ActMainFunction_Dem	SchM	The DEM module implementation invokes the SchM_ActMainFunction API to trigger the activation of a corresponding main processing function.
SchM_CancelMainFunction_Dem	SchM	The DEM module implementation invokes the SchM_CancelMainFunction API to trigger the cancellation of a corresponding main processing function.

6.4.2 Optional Interfaces

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

API function	*Module*	*Description*	*Configuration parameter (description ref. to chapter 10)*
Det_ReportError	DET	Development error notification	DEM_DEV_ERROR_DETECT
Fim_DemTriggerOnEvent-Status	FIM	ref. to Xxx_DemTriggerOnEventStatus	Ref. to chapter TriggerOnEventStatusList

6.4.3 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.

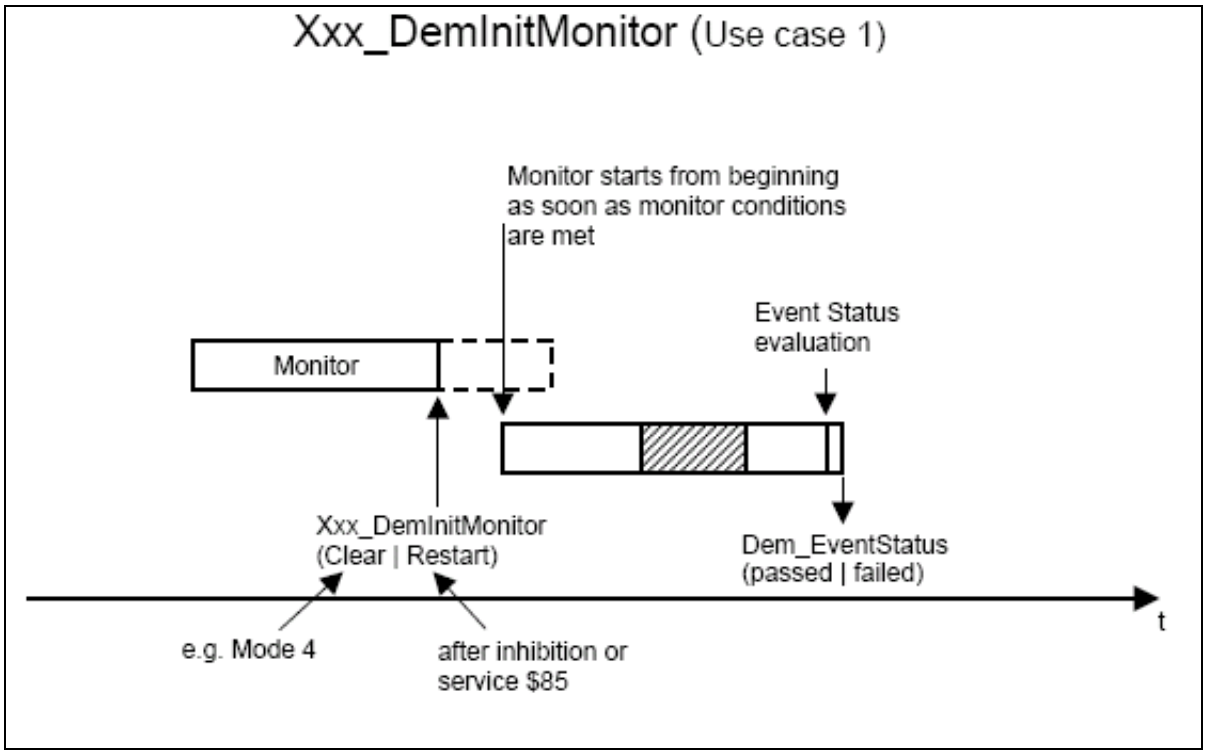
6.4.3.1 Interface SW-Components/FIM ⇔ DEM

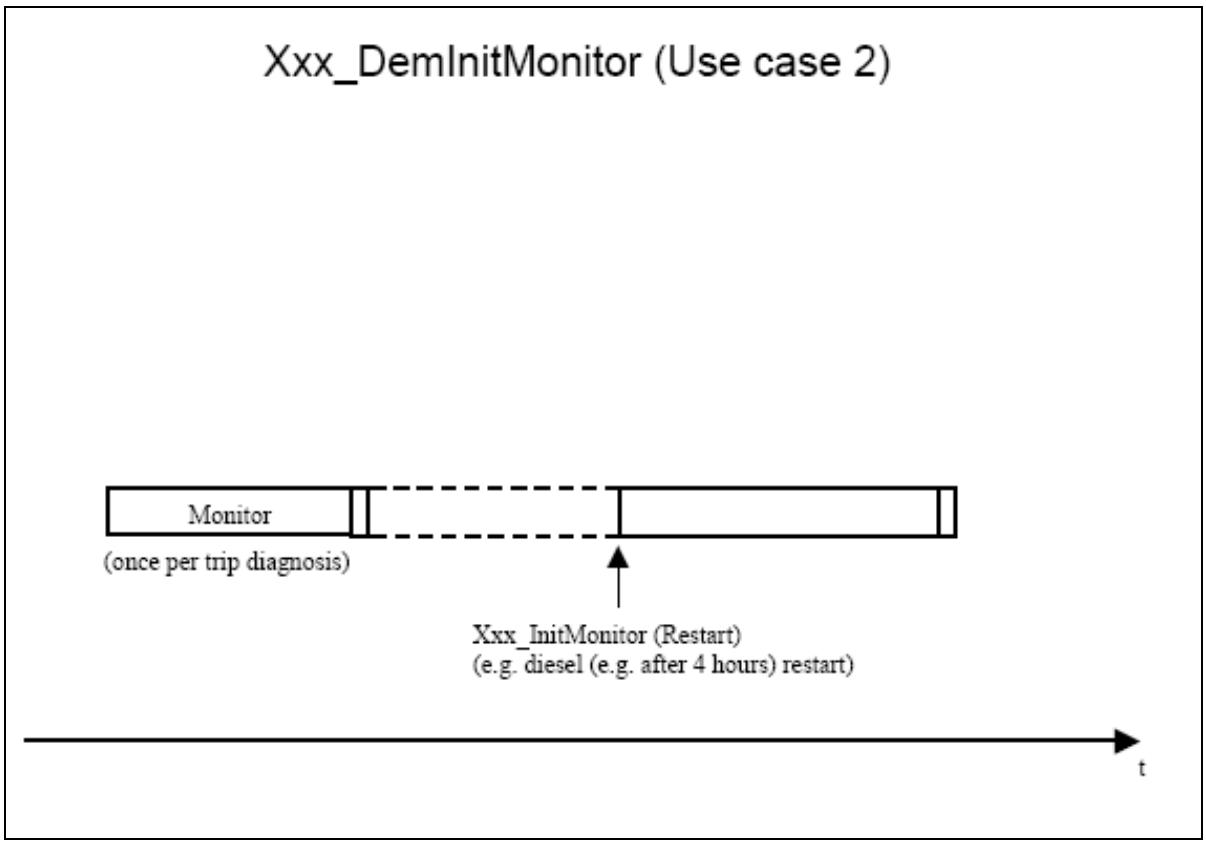
6.4.3.1.1 Xxx_DemInitMonitor{*EventId*}

Service name:	Xxx_DemInitMonitor{*EventId*}	

Syntax:	`Std_ReturnType Xxx_DemInitMonitor{EventId}`
	`(`
	`  Dem_InitMonitorKindType  InitMonitorKind`
	`)`	

Service ID:	49	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	yes	
Parameters (in):	`InitMonitorKind`	uint8 {Clear, Restart} Identification of the type of init function to be called from Monitor Function.
Parameters (out):	None	
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	This service shall be used to init the Monitor Function of an specific Event {*EventId*}. By the parameter Dem_InitMonitorKind the type of initialisation is chosen. API is called from DEM and has to be provided by SW_C and BSW. (Ref to Dem003) No API parameter checks required.	
Caveats:	DEM configuration during integration of Monitor Functions is system specific.	
Configuration:	The link between the EventId and the corresponding API (Xxx_DemInitMonitor) is configured within the DEM.	


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

6.4.3.1.2 Xxx_DemInit*{Function}*

Service name:	Xxx_DemInit{*Function*}	
Syntax:	`Std_ReturnType Xxx_DemInit{Function}(void)`	
Service ID:	50	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	yes	
Parameters (in):	None	
Parameters (out):	None	
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	**Dem049:** Xxx_DemInit*{Function}* shall be called by DEM or other API to reset the {Function} of the Module Xxx. For example: Adaptations may be reset in case of clear DEM (on service 04/ISO15031-5 request).	
Caveats:	DEM configuration during integration of Monitor Functions is system specific.	
Configuration:	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.	

6.4.3.1.3 Xxx_DemTriggerOnEventStatus

Service name:	Xxx_DemTriggerOnEventStatus	
Syntax:	`Std_ReturnType Xxx_DemTriggerOnEventStatus ( Dem_EventIdType EventId, Dem_EventStatusExtendedType EventStatusOld, Dem_EventStatusExtendedType EventStatusNew )`	
Service ID:	51	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	Not-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)
	`EventStatusOld`	Extended event status before change
	`EventStatusNew`	Extended event status after change
Parameters (in/out):	None	
Parameters (out):	None	
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	**Dem044:** Function prototype that shall be provided by SW-Components or FIM that want to be triggered by the DEM on changes of the extended Event status. For configuration ref. to 8.2.6.	
Caveats:	In case of disabling the event status update the function need not be called because no status change can be reported.	
Configuration:	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.	

6.4.3.1.4 Xxx_DemTriggerOnDTCStatus

Service name:	Xxx_DemTriggerOnDTCStatus	
Syntax:	`Std_ReturnType Xxx_DemTriggerOnDTCStatus ( Dem_DTCType DTC, Dem_EventIdType EventId, Dem_DTCStatusMaskType DTCStatusOld, Dem_DTCStatusMaskType DTCStatusNew )`	
Service ID:	52	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	Non reentrant	
Parameters (in):	`DTC`	This is the DTC the change trigger is assigned to.
	`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)
	`DTCStatusOld`	DTC status before change
	`DTCStatusNew`	DTC status after change
Parameters (in/out):	None	
Parameters (out):	None	
Return value:	`Std_ReturnType`	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	Function prototype that shall be provided by SW-Components that want to be triggered by the DEM on changes of the DTC status.	
Caveats:	None	
Configuration:	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.	

6.4.3.2 Interface DEM ⇔ SW-Components

6.4.3.2.1 Xxx_DemGetDataValueByDataIdentifier

Service name:	Xxx_DemGetDataValueByDataIdentifier	
Syntax:	`Std_ReturnType Xxx_DemGetDataValueByDataIdentifier ( uint16 DataID, uint8 *DemDataValueByDataIDBuffer )`	
Service ID:	53	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameter (in):	`DataID`	Data identifier for a specific data value. Here, the ISO14229 data identifier can be used or the PID (15031-5).
Parameter (out)	`DemDataValueByDataID Buffer`	Pointer to the buffer in the DEM which should be filled with the requested value
Return value:	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:	The API call (provided by a SW-C) returns the associated data value for a requested data identifier. It is called by the DEM as soon as it requires this 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 this API call, too.	
Caveats:	--	
Configuration:	--	

6.4.3.2.2 Xxx_DemGetExtendedDataRecord

Service name:	Xxx_DemGetExtendedDataRecord	
Syntax:	`Std_ReturnType Xxx_DemGetExtendedDataRecord ( uint8 ExtendedDataRecordNumber, uint8 *ExtendedDataRecord )`	
Service ID:	61	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Parameter (in):	`ExtendedDataRecordNumber`	This parameter specifies the identifier for a particular record the data values shall be stored.
Parameter (out)	`ExtendedDataRecord`	Pointer to the buffer in the DEM which should be filled with the requested value
Return value:	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:	This interface can be optionally used in case the Extended Data Record is provided by the application. The API call (provided by a SW-C) returns the associated Extended Data Record for a requested ExtendedDataRecordNumber. It is called by the DEM 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, …).	
Caveats:	--	
Configuration:	The configuration has to provide one or several ExtendedDataRecordNumber(s) assigned to one DTC because the DEM 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).	

6.4.3.2.3 Xxx_DemGetFaultDetectionCounter

Service name:	Xxx_DemGetFaultDetectionCounter	
Syntax:	`Std_ReturnType Xxx_DemGetFaultDetectionCounter ( Dem_EventIdType EventId, Dem_FaultDetectionCounterType *EventIdFaultDetectionCounter )`	
Service ID:	62	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
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 (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 OK this parameter does not contain valid data.

Return value:	`Std_ReturnType`	E_OK: request of severity was successful E_NOT_OK: request of severity failed
Description:	The API shall be used by DEM to request the current Fault Detection Counter for a given EventID. The Dem shall use this API only if debouncing is not done by DEM itself.	
Caveats:	--	
Configuration:	--	

6.5 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.

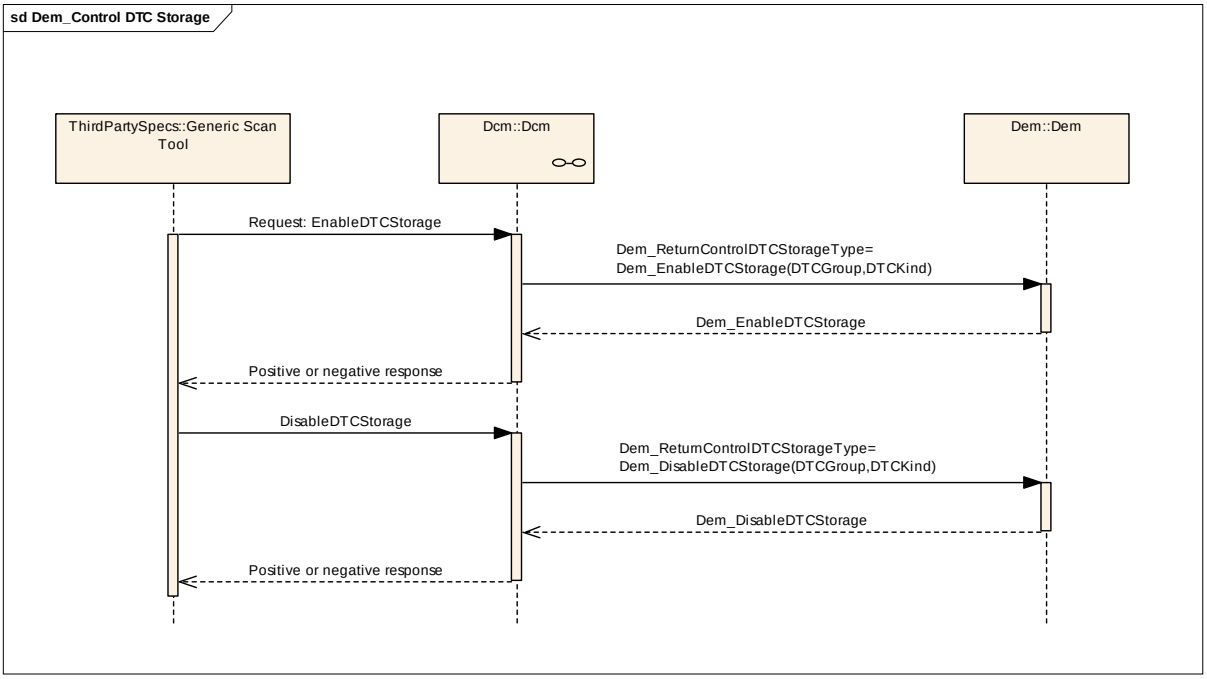
6.5.1 Dem_MainFunction

Service name:	Dem_MainFunction	
Syntax:	`Std_ReturnType Dem_MainFunction(void)`	
Service ID [hex]:	55	
Sync/Async:	Synchronous	
CanBeInvoked-Concurrently:	no	
Return value:	`Std_ReturnType`	The return value specified whether the function call has been successful or not according to Std_ReturnType.
Description:	**Dem125:** This function is used to process all not event based DEM internal functions. It shall be called periodically as cyclic task by the software system (e.g. by operating system).	
Timing:	fixed cyclic	
Pre condition:	--	
Configuration:	The cyclic time for the main function has to be defined as an operating system task or run able entity.	

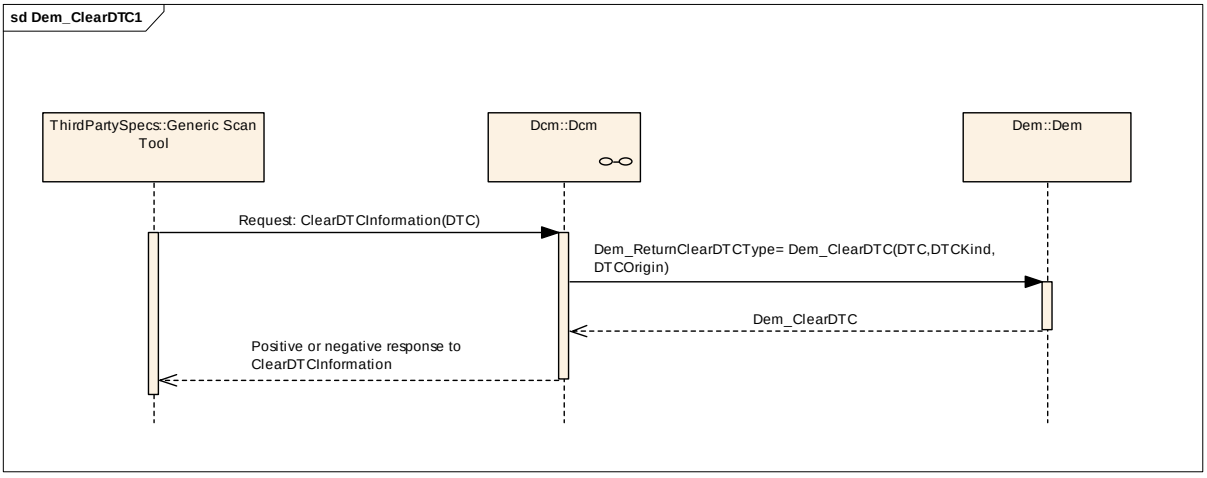
Terms and definitions:
Fixed cyclic: Fixed cyclic means that one cycle time is defined at configuration and shall not be changed because functionality is requiring that fixed timing (e.g. filters).
Variable cyclic: Variable cyclic means that the cycle times are defined at configuration, but might be 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.

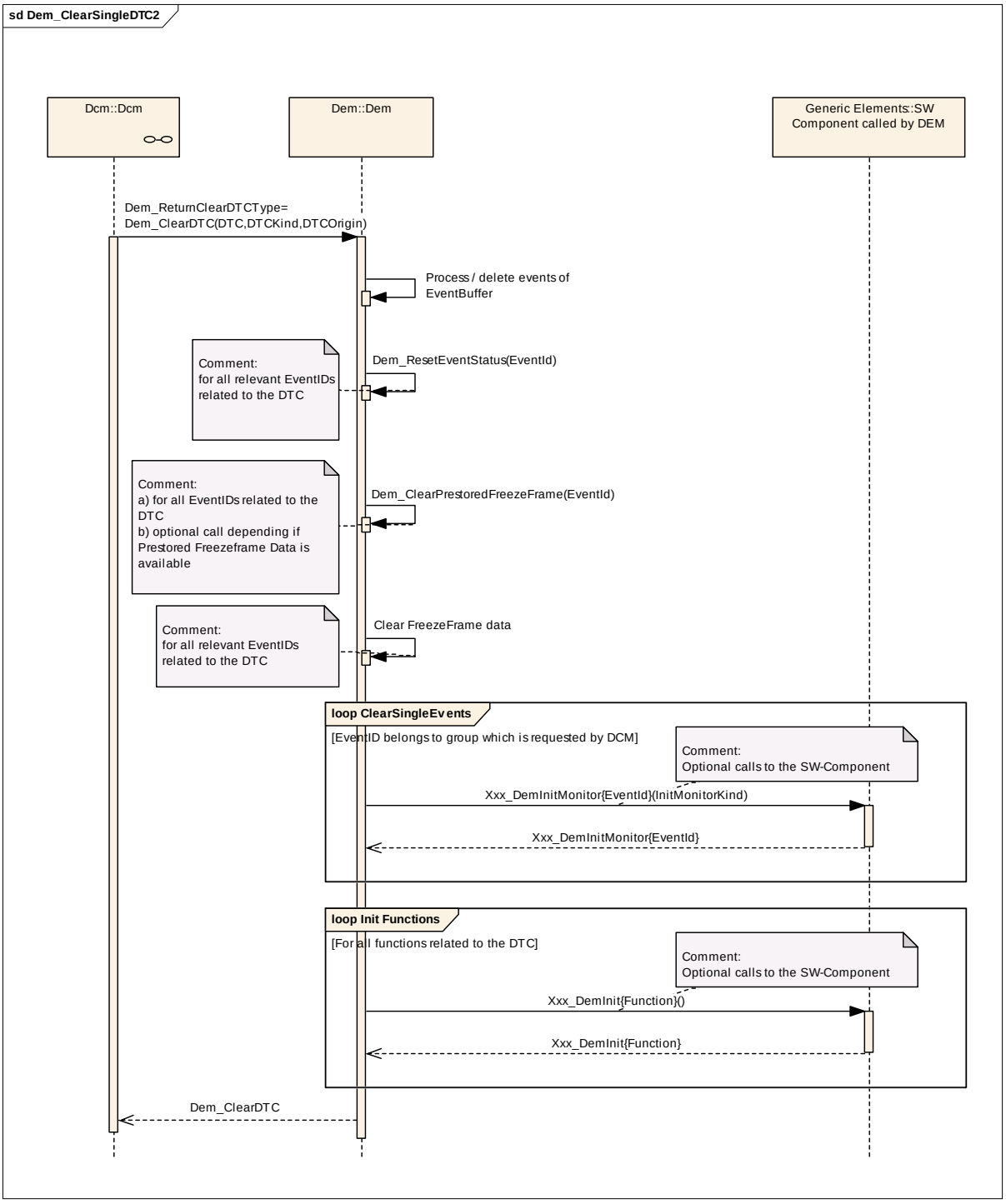
7 Sequence diagrams

7.1 ControlDTCStorage

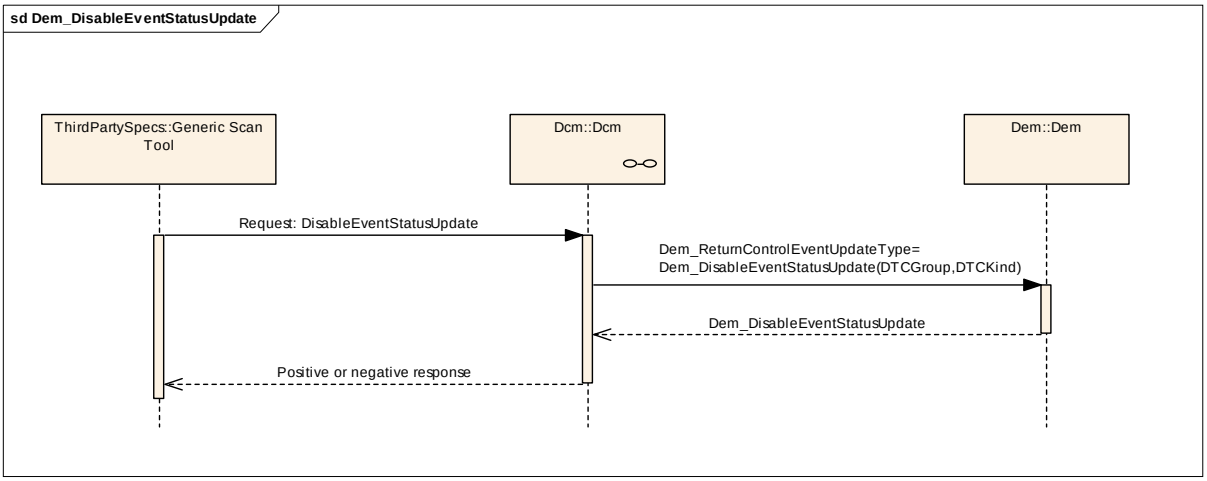


7.2 Dem_ClearDTC

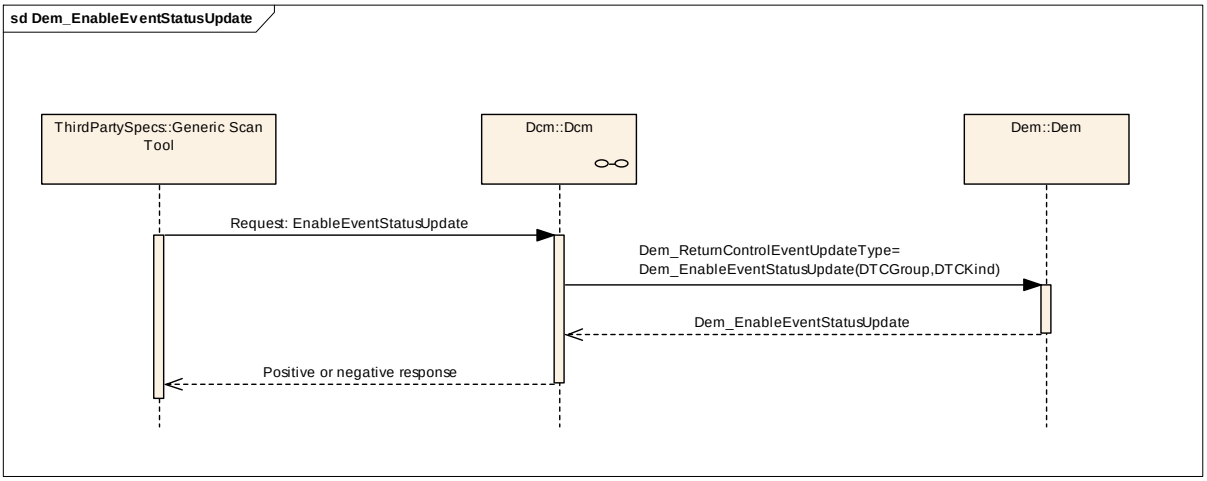



sd Dem_ClearSingleDTC2
Dcm::Dcm
Dem::Dem
Generic Elements::SW Component called by DEM
Dem_ReturnClearDTCType= Dem_ClearDTC(DTC,DTCKind,DTCOrigin)
Process / delete events of EventBuffer
Comment: for all relevant EventIDs related to the DTC
Dem_ResetEventStatus(EventId)
Comment: a) for all EventIDs related to the DTC b) optional call depending if Prestored Freezeframe Data is available
Dem_ClearPrestoredFreezeFrame(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}(InitMonitorKind)
Xxx_DemInitMonitor{EventId}
loop Init Functions
[For all functions related to the DTC]
Comment: Optional calls to the SW-Component
Xxx_DemInit{Function}()
Xxx_DemInit{Function}
Dem_ClearDTC

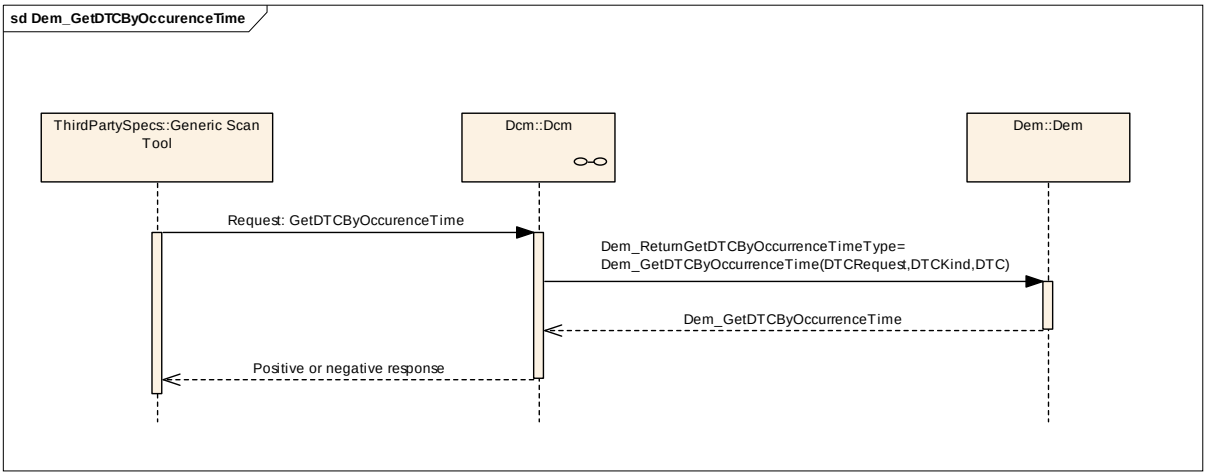
7.3 Dem_DisableEventStatusUpdate



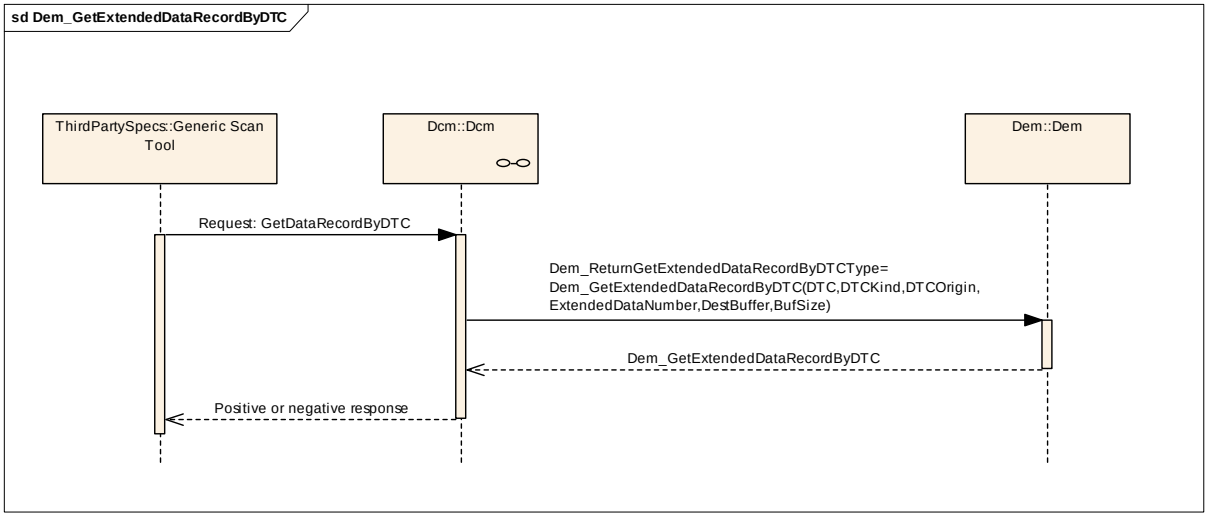
7.4 Dem_EnableEventStatusUpdate



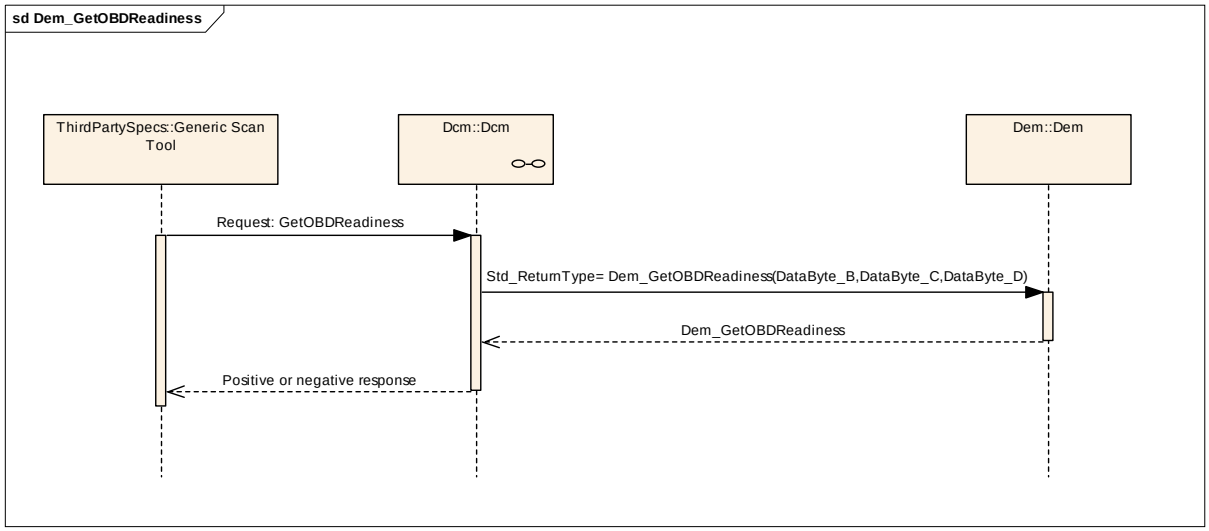
7.5 Dem_GetDTCByOccurrenceTime



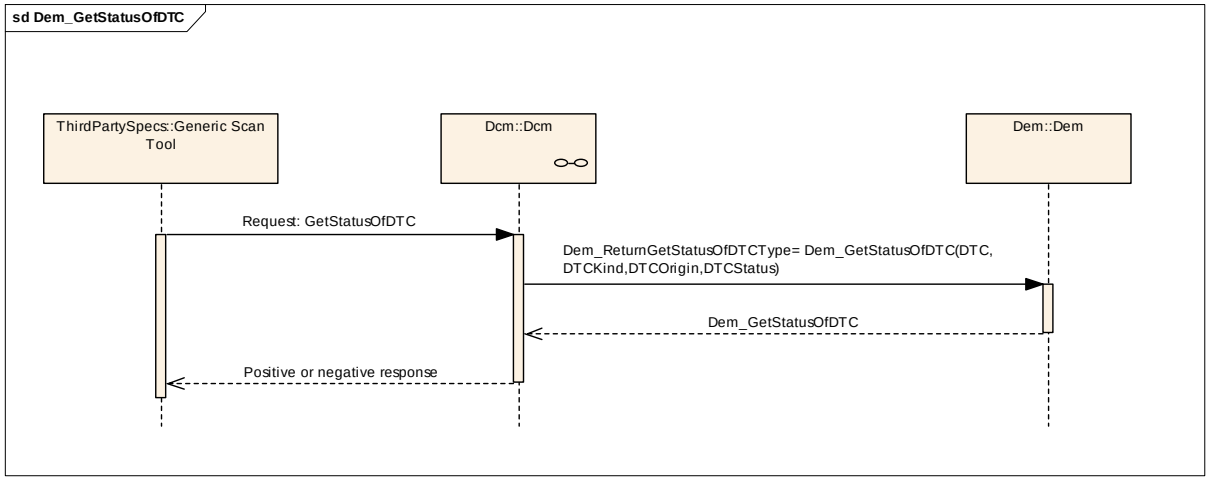
7.6 Dem_GetExtendedDataRecordByDTC



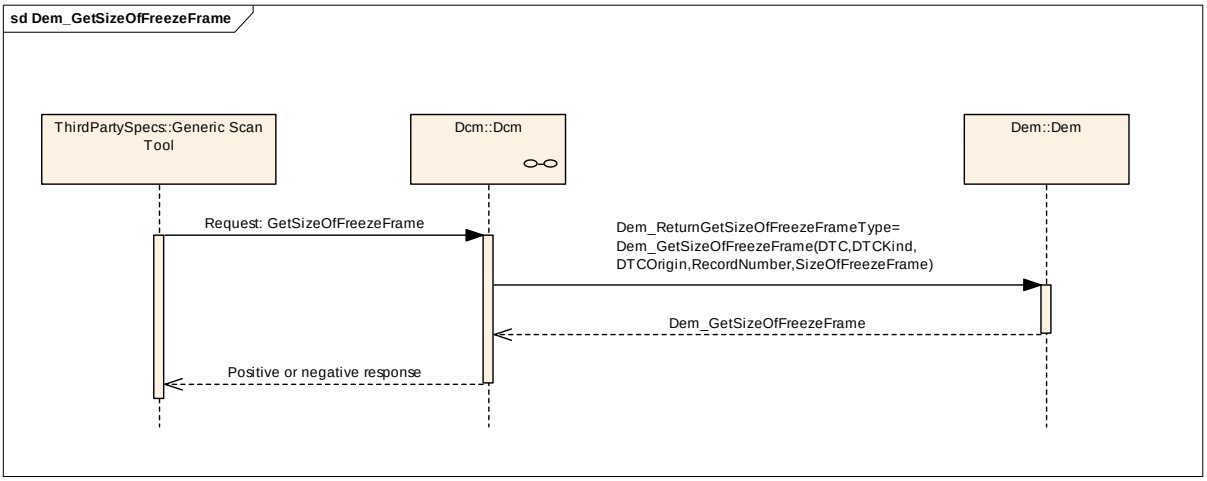
7.7 Dem_GetOBDReadiness



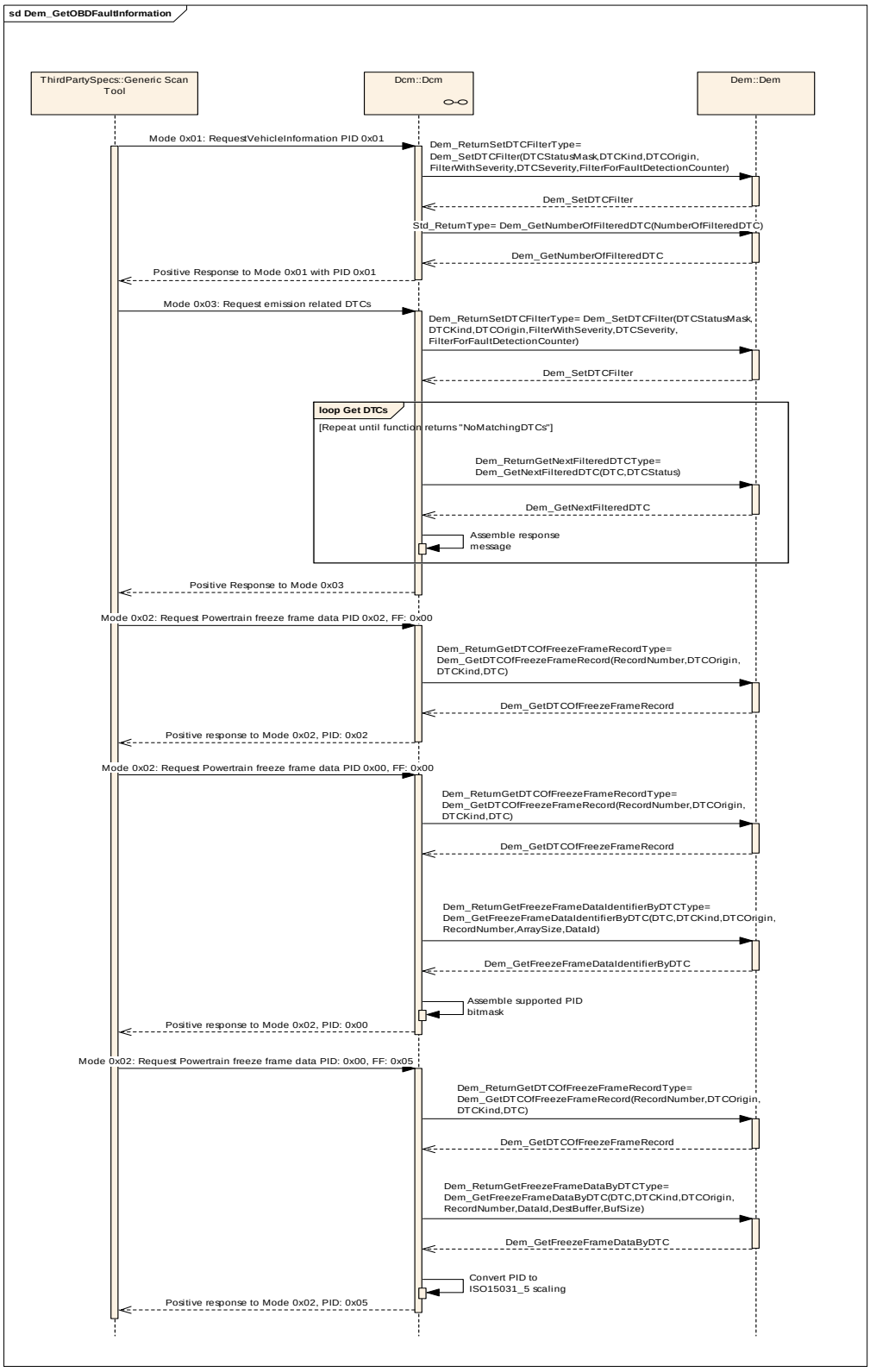
7.8 Dem_GetStatusOfDTC



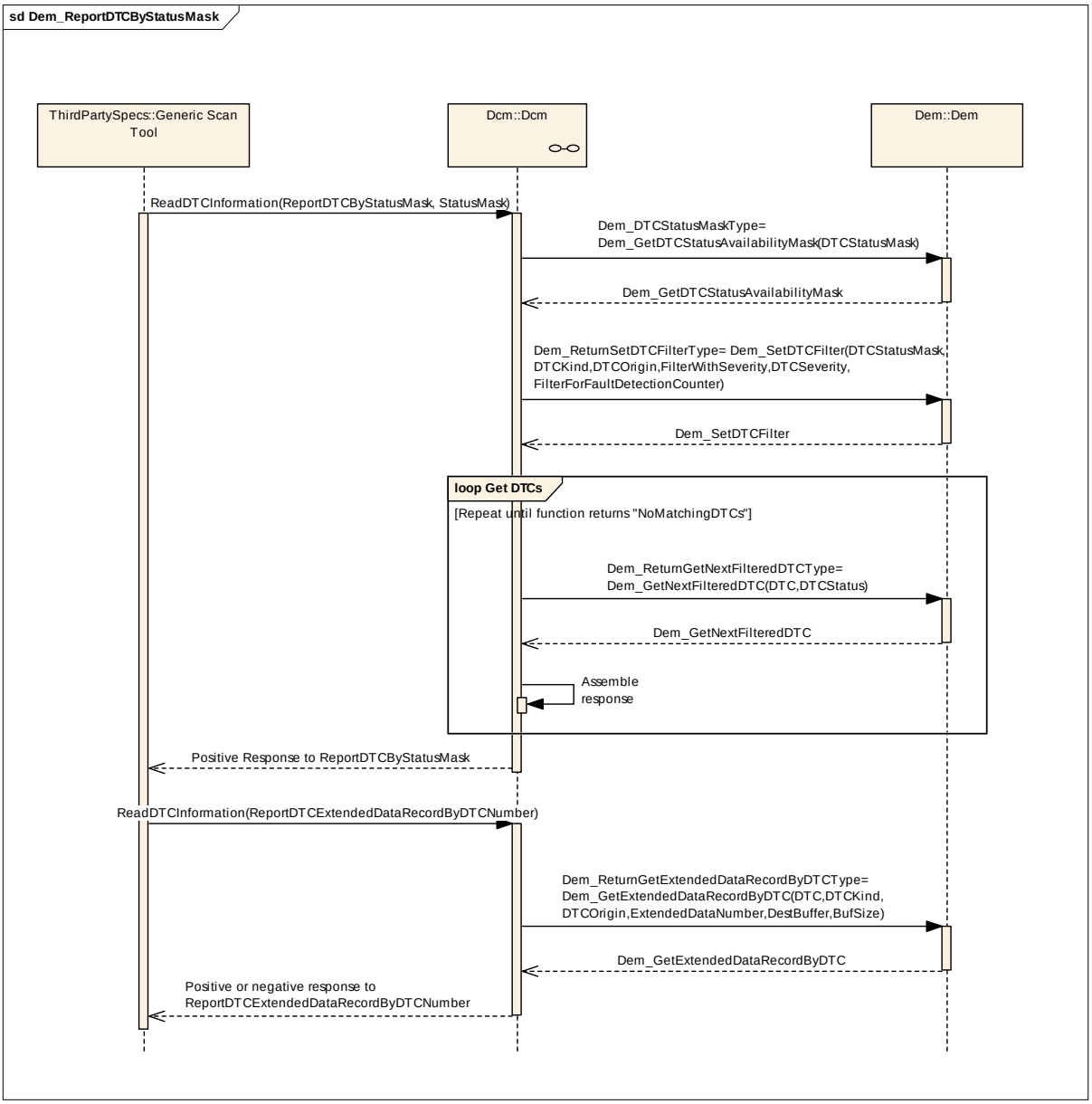
7.9 Dem_GetSizeOfFreezeFrame



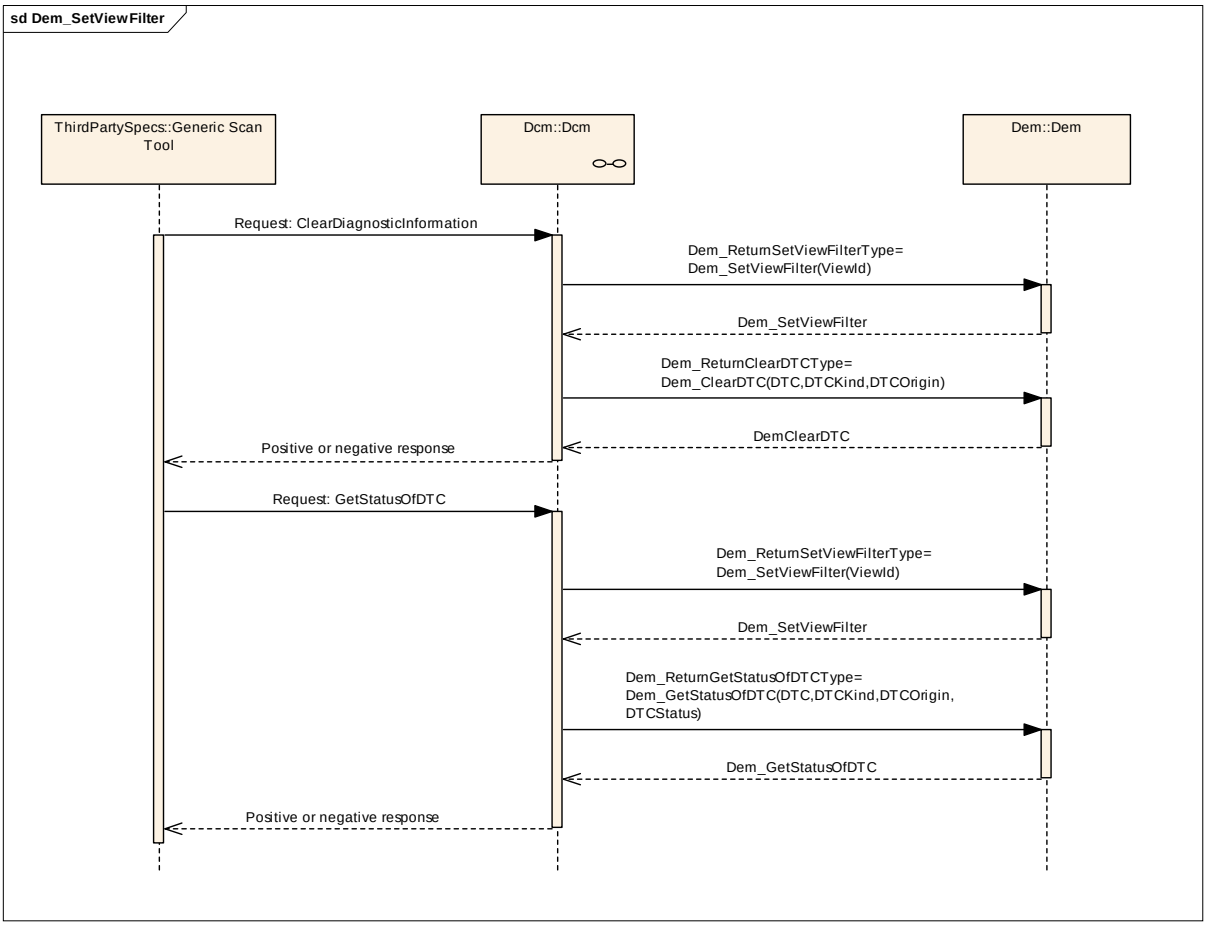
7.10 GetOBDFaultInformation



7.11 ReportDTCByStatusMask

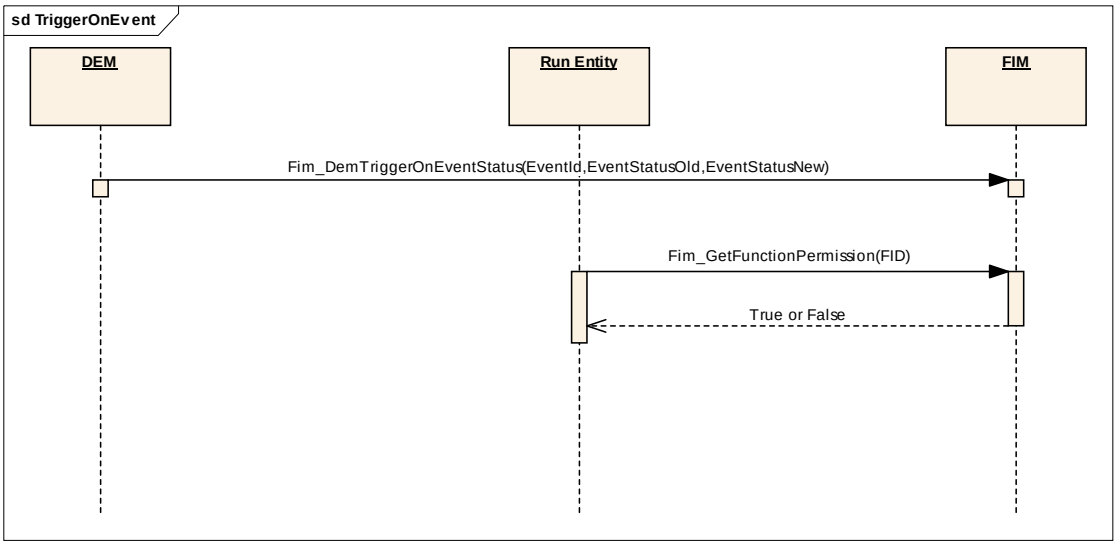


7.12 Dem_SetViewFilter

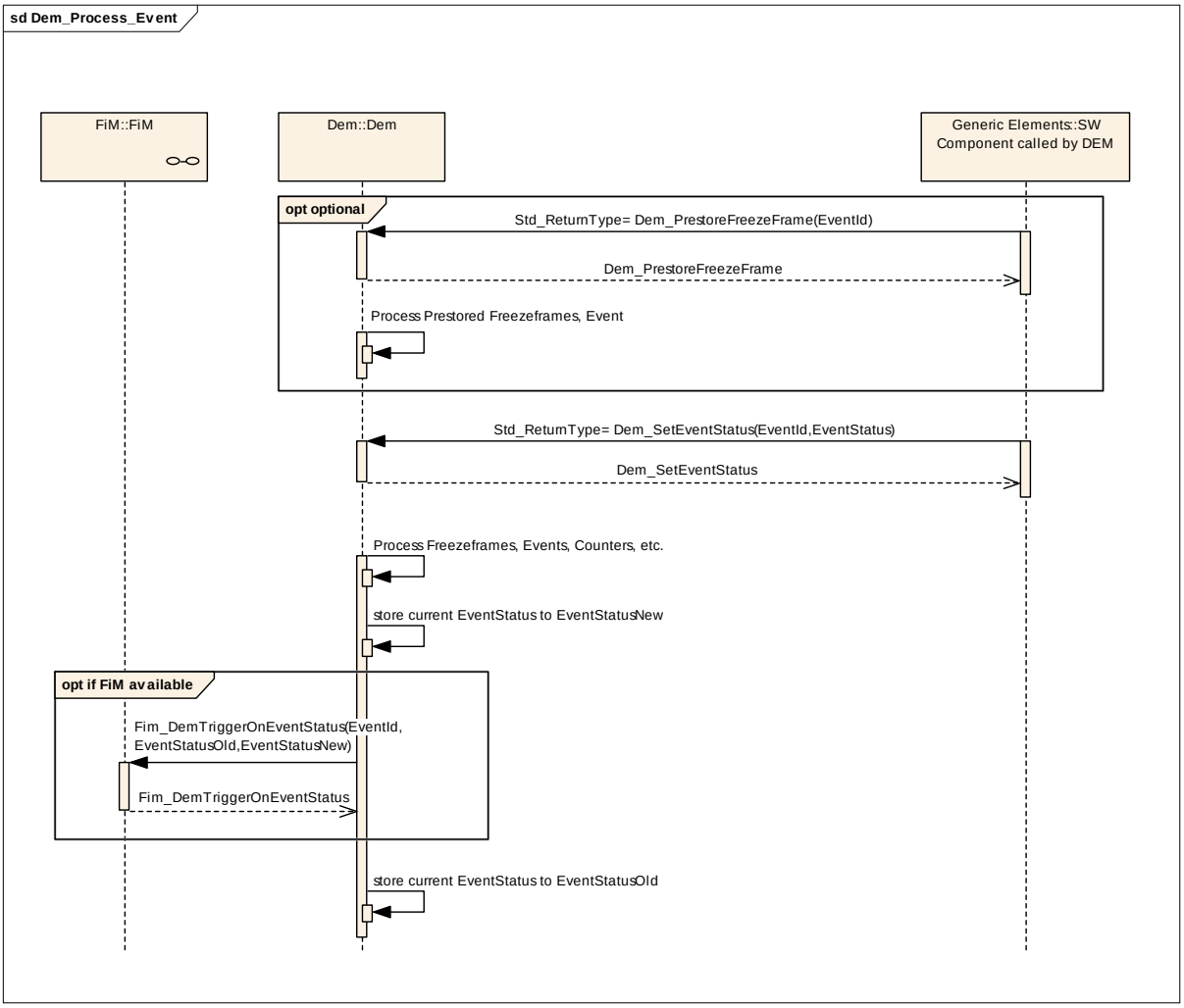


Note: this is only an example for the use of Dem_SetViewFilter

7.13 Fim_Dem_TriggerOnEventStatus



7.14 ProcessEvent (Example)



8 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 8.2 specifies the structure (containers) and the parameters of the module DEM.

Chapter 8.2.14 specifies published information of the module DEM.

8.1 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.

8.1.1 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.

8.1.2 Variants

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

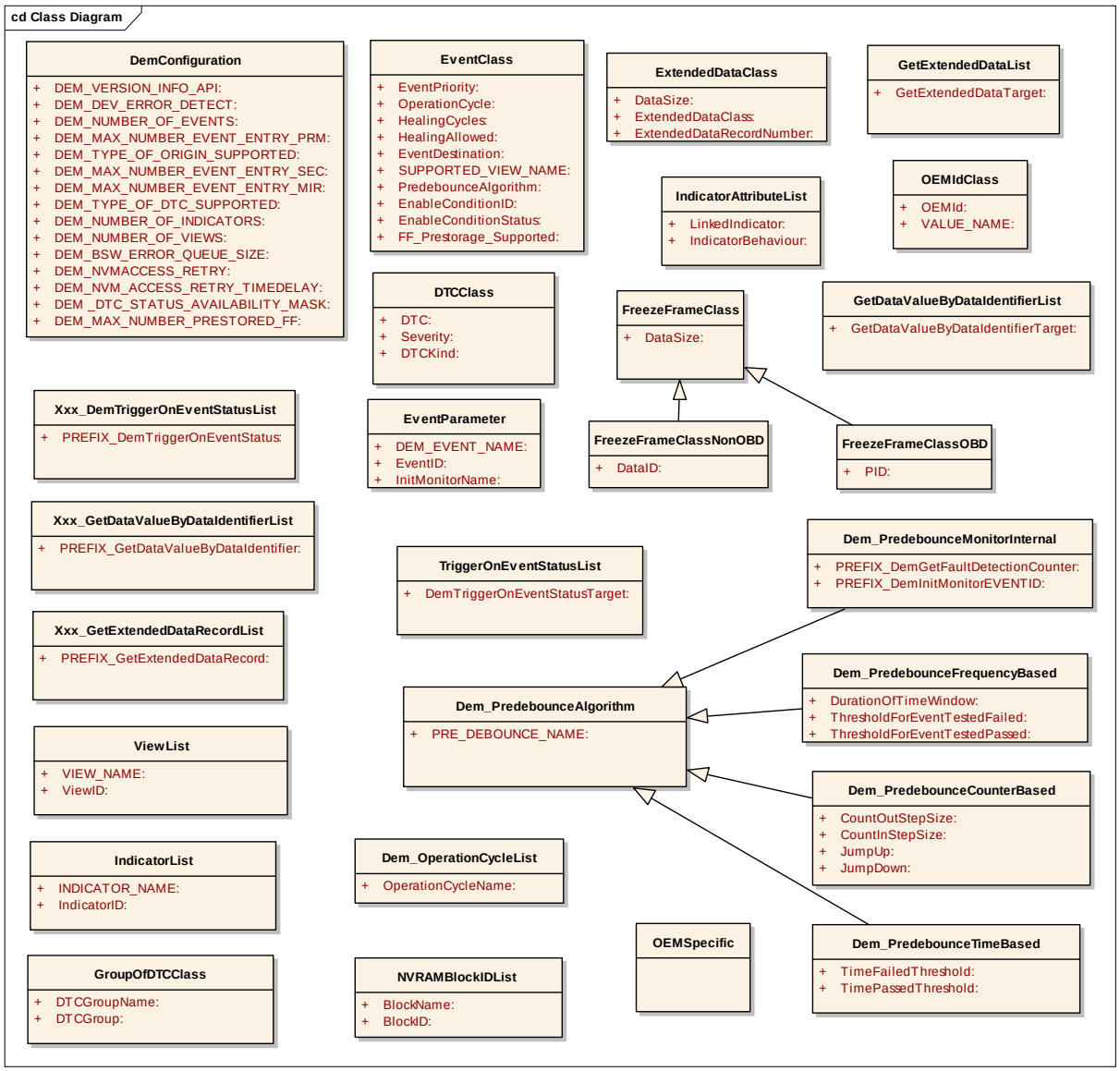
8.1.3 Containers

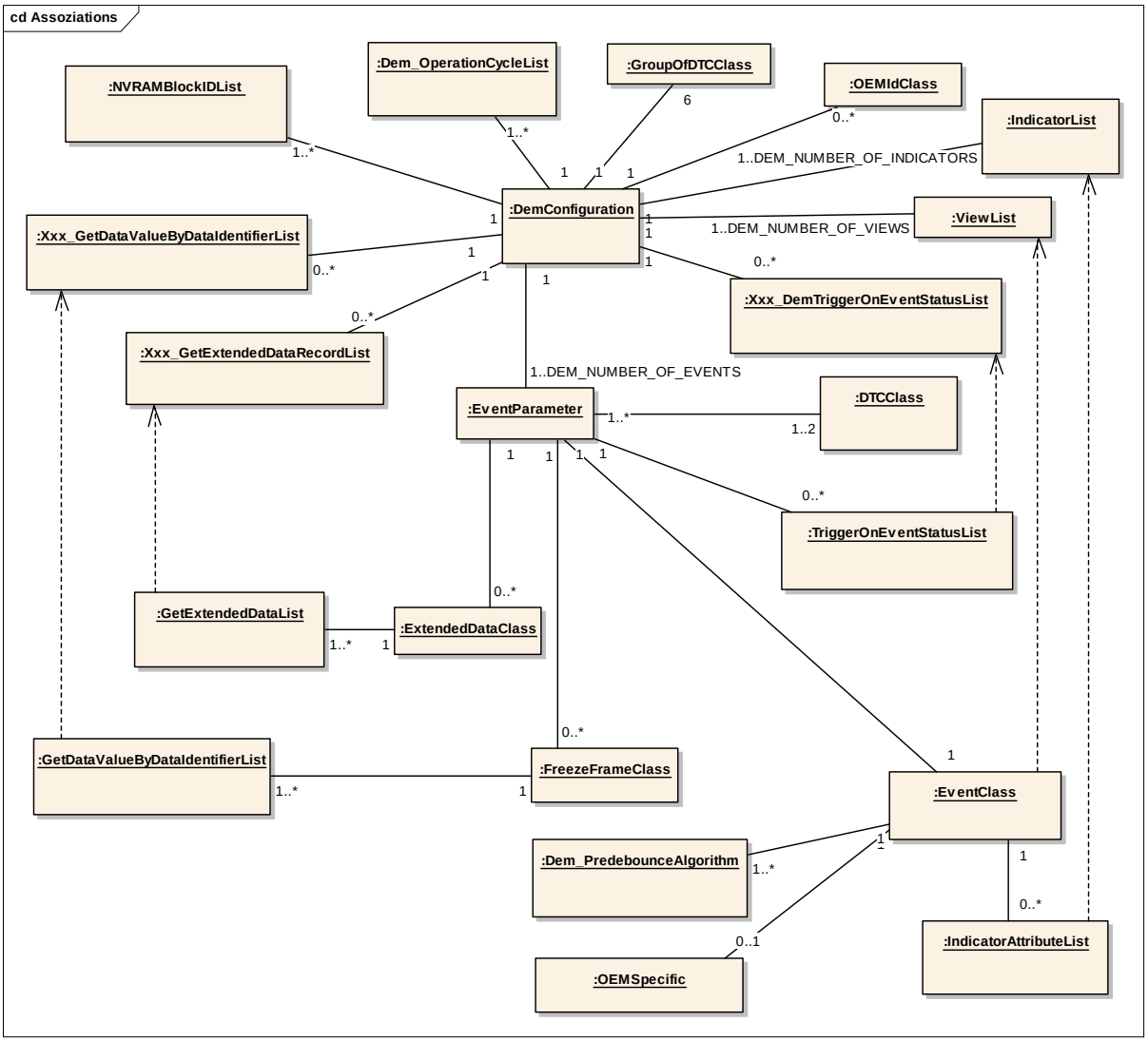
Dem120: Containers shall structure the set of configuration parameters. This means:
- *all* configuration parameters are kept in containers.

- (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.

8.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 5 and Chapter 6.





8.2.1 Variants

Dem119: The following configuration parameters shall be available:

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

Link time configurable parameters are not used in this specification.

8.2.2 DemConfiguration

SWS Item	**Dem128:**
Container Name	DemConfiguration
Description	This container contains the configuration (parameters) of the BSW DEM.
Configuration Parameters	

Name	DEM_VERSION_INFO_API		
Description	Switches the function Dem_GetVersionInfo ON or OFF.		
Type	#define		
Unit	--		
Range	STD_ON	supported	
	STD_OFF	Not supported	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	none		

Name	DEM_DEV_ERROR_DETECT		
Description	Switches the Development Error Detection and Notification ON or OFF.		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	none		

Name	DEM_NUMBER_OF_EVENTS		
Description	Maximum of defined events which are present in the system (typically up to 500)		
Type or Unit	Implementation specific		
Range	0..65535		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	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 chapter 7)		
Type or Unit	#define		
Range	0..255		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	DEM_TYPE_OF_ORIGIN_SUPPORTED

Description	Definition of origins available at ECU. Bit 0: DEM_DTC_ORIGIN_SECONDARY_MEMORY Bit 1: DEM_DTC_ORIGIN_MIRROR_MEMORY A primary memory is always available. Set the according Bit to '1' means origin is available. Combination of different origins is possible (e.g. DEM_DTC_ORIGIN_MIRROR_MEMORY and DEM_DTC_ORIGIN_SECONDARY_MEMORY supported is coded by 0x0011b) This parameter is optional, because multiple origins are not supported by all automotive manufacturer.		
Type or Unit	`#define`		
Range	`2 Bit`		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	`DEM_MAX_NUMBER_EVENT_ENTRY_SEC`		
Description	Maximum number of events which can be stored in the secondary memory. Minimum = default value = 0		
Type or Unit	`#define`		
Range	`0..255`		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	DEM_TYPE_OF_ORIGIN_SUPPORTED, if secondary memory is not available, then set to 0.		

Name	`DEM_MAX_NUMBER_EVENT_ENTRY_MIR`		
Description	Maximum number of events which can be stored in the mirror memory Minimum = default value = 0		
Type or Unit	`#define`		
Range	`0..255`		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	DEM_TYPE_OF_ORIGIN_SUPPORTED, if mirror memory is not available, then set to 0.		

Name	`DEM_TYPE_OF_DTC_SUPPORTED`		
Description	Defines the type of DTC which is supported by ECU 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 supported is coded by 0x0011b)		
Type or Unit	`#define`		

Range	5 Bit		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	DEM_NUMBER_OF_INDICATORS		
Description	Maximum Numbers of Indicators supported by DEM (this parameter is not mandatory for all ECUs)		
Type or Unit	#define		
Range	0..255	If DEM_NUMBER_OF_INDICATORS is 0, than Indicators are not supported.	
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	DEM_NUMBER_OF_VIEWS		
Description	Maximum number of views supported by the DEM (this parameter is not mandatory for all ECUs) This parameter is optional.		
Type or Unit	#define		
Range	0..256	If DEM_NUMBER_OF_VIEWS is 0, than views are not supported	
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	DEM_BSW_ERROR_BUFFER_SIZE		
Description	Maximum number of Elements in buffer for handling of BSW Errors (ref. to Dem107:		
Type or Unit	#define		
Range	0..255		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	DEM_NVM_ACCESS_RETRY		
Description	Maximum number of retries to access NVRAM Manager		
Type or Unit	#define		
Range	0..255		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	DEM_NVM_ACCESS_RETRY_TIMEDELAY		
Description	Time in milliseconds between two retries to NVRAM Manager in case that first access failed.		
Type or Unit	#define		
Range	0.. 65535		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	DEM_DTC_STATUS_AVAILABILITY_MASK		
Description	Mask for the supported DTC status bits by the DEM. This mask is used by UDS service 0x19.		
Type or Unit	#define		
Range	0.. 255		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	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.		
Type or Unit	#define		
Range	0..255	Number of possible freeze frames	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EventParameter	0..65535	Multiplicity depends on DEM_NUMBER_OF_EVENTS
IndicatorList	0..255	Multiplicity depends on DEM_ NUMBER_OF_INDICATORS
GroupOfDTCList	6	--
ViewList	0..255	Multiplicity depends on DEM_NUMBER_OF_VIEWS (optional container)
Xxx_DemTriggerOnEvent StatusList	0..n	--
OEMIdClass	0..n	--
Dem_OperationCycleList	1..n	--
NVRAMBlockIDList	1..n	--

SWS Item	**Dem129:**
Container Name	IndicatorList
Description	This container contains the configuration (parameters) for Indicators.
Configuration Parameters	

8.2.3 IndicatorList

Name	`INDICATOR_NAME`		
Description	Unique name of an indicator (readability of code) Not present after compilation. Only IndicatorID is used during runtime.		
Type or Unit	`#define`		
Range	`INDIATOR_<NAME>`		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	`IndicatorID`		
Description	Unique identifier of an INDICATOR		
Type or Unit	`Dem_IndicatorIdType`		
Range	`0..255`		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	DemConfiguration/DEM_NUMBER_OF_INDICATORS		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.4 GroupOfDTCList

SWS Item	**Dem137:**
Container Name	GroupOfDTCList
Description	This container contains the configuration (parameters) for DTC Groups.
Configuration Parameters	

Name	DTCGroupName		
Description	Name of a group of DTC according to Dem_DTCGroupType		
Type or Unit	Dem_DTCGroupType		
Range	--	--	
	--	--	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	Vehicle		
Dependency	None		

Name	DTCGroup		
Description	DTC of the selected group of DTC (according to ISO14229-1[9] Annex D1).		
Type or Unit	Dem_DTCType		
Range	$0..2^{24}$	--	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	Vehicle		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.5 ViewList

SWS Item	**Dem138:**
Container Name	ViewList
Description	This container contains the configuration (parameters) for Views.
Configuration Parameters	

Name	VIEW_NAME		
Description	Unique name of a view (readability of code) Not present after compilation. Only ViewID is used during runtime.		
Type or Unit	#define		
Range	VIEW_<NAME>		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	ViewID		
Description	Unique identifier of a VIEW		
Type or Unit	Dem_ViewIdType		
Range	0..255		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	

Scope	ECU
Dependency	DemConfiguration/DEM_NUMBER_OF_VIEWS

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

SWS Item	**Dem139:**
Container Name	Xxx_DemTriggerOnEventStatusList
Description	This container contains the configuration (parameters) for DemTriggerOnEventStatus functions.
Configuration Parameters	

8.2.6 Xxx_DemTriggerOnEventStatusList

Name	`PREFIX_DemTriggerOnEventStatus`		
Description	Unique prefix of a Xxx_DemTriggerOnEventStatus function. The prefix shall replace the Xxx in function name. The prefixes shall be matching to available SW_C and BSW.		
Type or Unit	`#define`		
Range	`<PREFIX>`		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.7 Xxx_DemGetDataValueByDataIdentifierList

SWS Item	**Dem139:**
Container Name	Xxx_DemGetDataValueByDataIdentifierList
Description	This container contains the configuration (parameters) for GetDataValueByDataIdentifier functions.
Configuration Parameters	

Name	`PREFIX_GetDataValueByDataIdentifierList`

Description	Unique prefix of a Xxx_DemGetDataValueByDataIdentifierList function. The prefix shall replace the Xxx in function name. The prefixes shall be matching to available SW_C and BSW.		
Type or Unit	`#define`		
Range	<PREFIX>		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.8 Xxx_DemGetExtendedDataRecordList

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

Name	`PREFIX_GetExtendedDataRecord`		
Description	Unique prefix of a Xxx_DemGetExtendedDataRecord function. The prefix shall replace the Xxx in function name. The prefixes shall be matching to available SW_C and BSW.		
Type or Unit	`#define`		
Range	<PREFIX>		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.9 EventParameter

SWS Item	**Dem130:**		
Container Name	EventParameter		
Description	This container contains the configuration (parameters) for events.		
Configuration Parameters			
Name	`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.		
Type or Unit	`#define`		
Range	`1..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.	
	`1..65535`	For ECUs with > 255 different events, the events should be sequentially ordered beginning with 1 and no gaps in between.	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	`InitMonitorName`		
Description	Monitor Function which has to be initialized for the event (ref. to Xxx_DemInitMonitor{EventId})		
Type or Unit	`#define`		
Range	`function name`	Function name	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Name	`DEM_EVENT_NAME`		
Description	Unique name of an EVENT (readability of code) Not present after compilation. Only EventId is used during runtime.		
Type or Unit	`#define`		
Range	--		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EventClass	1	--
FreezeFrameClass	0..n	--
ExtendedDataClass	0..n	--
DTCClass	1..2	--
TriggerOnEventStatusList	0..n	--

SWS Item	**Dem131:**
Container Name	EventClass
Description	This container contains the configuration (parameters) for EventClass.
Configuration Parameters	

8.2.10 EventClass

Name	`EventPriority`		
Description	priority of an event, in view of full event buffer (ref. to Dem104:)		
Type or Unit	`#define`		
Range	`0..255 (depends on OEM)`	Dem_EventPriorityType	
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	None		

Name	`OperationCycle`		
Description	Kind of operation cycle for the event storage(e.g. power cycle, driving cycle, ...)		
Type or Unit	`#define`		
Range	`0..255`	Dem_OperationCycleIdType	
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	None		

Name	`HealingCyclesCounter`		
Description	cycles necessary to heal/erase event. (ref. to Dem104:) This parameter is optional (depends on OEM).		
Type or Unit	`#define`		
Range	`1..256 (depends on OEM)`		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	None		

Name	`HealingCycle`
Description	Kind of operation cycle for heal/erase event. This parameter is optional (depends on OEM).
Type or Unit	`#define`

Range	`0..255 (depends on OEM)`	Dem_OperationCycleIdType	
	v		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	None		

Name	`HealingAllowed`		
Description	(DEM104) general switch to allow healing/unlearning or not		
Type or Unit	`#define`		
Range	`True`	healing/unlearning allowed	
	`False`	healing/unlearning not allowed	
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	None		

Name	`EventDestination`		
Description	Definition of event is assigned to origin. Bit 0: DEM_DTC_ORIGIN_PRIMARY_MEMORY Bit 1: DEM_DTC_ORIGIN_SECONDARY_MEMORY Bit 2: DEM_DTC_ORIGIN_MIRROR_MEMORY Set the according Bit to '1' means event is assigned to the origin. Combination of different origins is possible (e.g. DEM_DTC_ORIGIN_MIRROR_MEMORY and DEM_DTC_ORIGIN_PRIMARY_MEMORY supported is coded by 0x0101b) If an event is not assigned to an origin, then it is only handled internally and it is not visible externally. This parameter is optional, because multiple origins are not supported by all automotive manufacturer.		
Type or Unit	`#define`		
Range	`3 Bit`		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	DemConfiguration/DEM_TYPE_OF_ORIGIN_SUPPORTED		

Name	`SUPPORTED_VIEW_NAME`	
Description	view name of the 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 whih the event is visible. Example: WIPERSYSTEM refers to functionality wiper system, WINDOWLIFTER refers to functionality window lifter, … This parameter is optional	
Type or Unit	`#define`	
Range	`<VIEW_NAME>`	Name of the view, VIEW_NAME is defined in ViewList
	--	

Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	ViewList/VIEW_NAME		

Name	`PredebounceAlgorithm`		
Description	Selects a Predebounce Algorithm 0x00: No Predebouncing 0x01: Dem_PredebounceTimeBased 0x02: Dem_PredebounceCounterBased 0x03: DEM Dem_PredebounceFrequencyBased		
Type or Unit	`#define`		
Range	`0..3`		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	None		

Name	`EnableConditionID`		
Description	Defines a condition ID. This parameter is optional and depends on manufacturer.		
Type or Unit	`#define`		
Range	`0..2`^{16}		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	None		

Name	`EnableConditionStatus`		
Description	Defines a status for enable or disable of storage of a event. This parameter is optional and depends on manufacturer.		
Type or Unit	`uint8`		
Range	--		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	None		

Name	`FF_Prestorage_Supported`		
Description	If this parameter set to 1, then the “Prestorage” of freeze frames is supported by the assigned event. This parameter is useful for calculation of buffer size.		
Type or Unit	#define		
Range	0	Prestorage is not supported by the event	
	1	Prestorage is supported by the event	
Configuration Class	***Pre-compile***	x	All Variants
	Link time	--	
	Post Build	--	

Scope	ECU
Dependency	None

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
OEMSpecific and optional parameters	0..1	For special optional parameters, like qualification timings.
IndicatorAttributeList	0..255	Muliplicity depends on DEM_NUMBER_OF_INDICATORS.
Dem_Predebounce-Algorithm	1..n	Used algorithm class (Dem_PredebounceMonitorInternal, Dem_PredebounceFrequencyBased, Dem_PredebounceCounterBased, Dem_predebounceTimeBased) depends on parameter EventClass.PredebounceAlgorithm

8.2.11 DTCClass

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

Name	DTC		
Description	Diagnostic Trouble Code		
Type or Unit	#define		
Range	0..2^{24}	for 2byte and 3byte DTC	
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	None		

Name	Severity		
Description	DTC severity. This parameter depends on automotive manufacturer and is optional.		
Type or Unit	#define		
Range	DemDTCSeverityType		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	None		

Name	DTCKind	
Description	Kind of DTC (OBD relevant or not)	
Type or Unit	#define	
Range	Dem_DTCKindType	
	--	

Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.12 FreezeFrameClass

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

Name	`PID`		
Description	For OBD relevant data Multiple PIDs can be relevant per FreezeFrame. (This parameter is optional.)		
Type or Unit	`#define`		
Range	`0..65535`		
	--		
Configuration Class	***Pre-compile***	X	Variant 1
	Link time	--	
	Post Build	X	Variant 2
Scope	ECU		
Dependency	None		

Name	`DataID`		
Description	For enhanced diagnostics Multiple DataIDs can be relevant per FreezeFrame. (This parameter is optional.)		
Type or Unit	`#define`		
Range	`0..65535`		
	--		
Configuration Class	***Pre-compile***	X	Variant 1
	Link time	--	
	Post Build	X	Variant 2
Scope	ECU		
Dependency	None		

Name	`GlobalParameter`		
Description	Mandatory parameters for OBD and common parameters for all events		
Type or Unit	`#define`		
Range	`0..65535`		
	--		
Configuration Class	***Pre-compile***	X	Variant 1
	Link time	--	

	Post Build	X	Variant 2
Scope	ECU		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
GetDataValueByData-IdentifierList	1	--

8.2.13 ExtendedDataClass

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

Name	`ExtendedDataClass`		
Description	Unique identifier of an extended data class		
Type or Unit	`#define`		
Range	`0..65535`		
	--		
Configuration Class	***Pre-compile***	X	Variant 1
	Link time	-	
	Post Build	X	Variant 2
Scope	ECU		
Dependency	None		

Name	`ExtendedDataRecordNumber`		
Description	This configuration parameter specifies an unique identifier for an ExtendedDataRecord. One or more ExtendedDataRecords can be assigned to one DTC.		
Type or Unit	`#define`		
Range	`0..253`	Because 0xFF and 0xFE are resereved by ISO	
	--		
Configuration Class	***Pre-compile***	X	Variant 1
	Link time	-	
	Post Build	X	Variant 2
Scope	ECU		
Dependency	None		

Name	`DataSize`		
Description	Defines the size of the extended Data Record in Bytes.		
Type or Unit	`#define`		
Range	`0..253`		
	--		
Configuration Class	***Pre-compile***	X	Variant 1
	Link time	-	
	Post Build	X	Variant 2
Scope	ECU		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
GetExtendedDataList	1	--

8.2.14 OEMSpecific and optional parameters

SWS Item	**Dem134:**
Container Name	OEMSpecific
Description	This container contains the configuration for OEM specific additional parameters.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
Depends on OEM	--	--

8.2.15 IndicatorAttributeList

SWS Item	**Dem133:**
Container Name	IndicatorAttributeList
Description	This container contains the event specific configuration for Indicators
Configuration Parameters	

Name	`LinkedIndicator`		
Description	indicator name of the used indicator		
Type or Unit	`#define`		
Range	`<INDIATOR_NAME>`	INDICATOR_NAME linked with event	
	--		
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	IndicatorList/INDICATOR_NAME		

Name	`IndicatorBehaviour`		
Description	Behaviour of the linked indicator Bit 0: Indicator is active if event in status FAILED Bit 1: Indicator is passive if event in status FAILED Bit 2: Indicator is blinking if event in status FAILED		
Type or Unit	`#define`		
Range	`3 bit`		
	--		
	Pre-compile	X	all Variants

	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.16 TriggerOnEventStatusList

SWS Item	**Dem140:**
Container Name	TriggerOnEventStatusList
Description	This container contains the configuration (parameters) for TriggerOnEvent functions.
Configuration Parameters	

Name	`DemTriggerOnEventStatusTarget`		
Description	Real name of Xxx_DemTriggerOnEventStatus function. The possible selection depends on Xxx_DemTriggerOnEventStatusList.		
Type or Unit	`#define`		
Range	`<Xxx>_DemTriggerOnEventStatus function`		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	Xxx_DemTriggerOnEventStatusList/ PREFIX_DemTriggerOnEventStatus		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.17 GetDataValueByDataIdentifierList

SWS Item	**Dem140:**
Container Name	GetDataValueByDataIdentifierList
Description	This container contains the configuration (parameters) for GetDataValueByDataIdentifier functions.
Configuration Parameters	

Name	`GetDataValueByDataIdentifierListTarget`		
Description	Real name of Xxx_ GetDataValueByDataIdentifier function. The possible selection depends on Xxx_DemGetDataValueByDataIdentifierList.		
Type or Unit	`#define`		
Range	`<Xxx>_ GetDataValueByDataId entifier function`		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	Xxx_DemGetDataValueByDataIdentifierList/ PREFIX_ GetDataValueByDataIdentifier		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.18 GetExtendedDataList

SWS Item	**Dem140:**
Container Name	GetExtendedDataList
Description	This container contains the configuration (parameters) for GetExtendedData functions.
Configuration Parameters	

Name	`GetExtendedDataTarget`		
Description	Real name of Xxx_GetExtendedData function. The possible selection depends on Xxx_ GetExtendedDataList		
Type or Unit	`#define`		
Range	`<Xxx>_GetExtendedDat a function`		
	--		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	Xxx_GetExtendedDataList/ PREFIX_GetExtendedData		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.19 OEMIdClass

SWS Item	**Dem141:**
Container Name	OEMIdClass
Description	This container contains the configuration (parameters) for OEMIdClass.
Configuration Parameters	

Name	OEMId		
Description	Defines a unique ID of a data value		
Type or Unit	#define		
Range	0..255		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	--		

Name	VALUE_NAME		
Description	Defines a real name of a variable assigned to the data value of OEMId. This name shall be used by a code generator as the variable name.		
Type or Unit	#define		
Range	<NAME>	Unique name	
	--		
Configuration Class	***Pre-compile***	x	All Variant
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.20 Dem_OperationCycleList

SWS Item	**Dem142:**
Container Name	Dem_OperationCycleList
Description	This container contains the configuration (parameters) for Dem_OperationCycleList
Configuration Parameters	

Name	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.		
Type or Unit	OperationCycleName		
Range	DEM_IGNITION	Ignition ON / OFF Cycle	
	DEM_OBD_DCY	OBD Driving Cycle	
	DEM_WARMUP	OBD Warm up Cycle	
	DEM_POWER	Power ON / OFF Cycle	
	Pre-compile		
Configuration Class	***Link time***	x	all Variants
	Post Build	--	
	ECU	--	
Scope	None		
Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.21 DemPredebounceTimeBased

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

Name	PreDebounceName		
Description	Defines the selected debounce algorithm		
Type or Unit	#define		
Range	Dem_PredebounceTimeBased		
	--		
Configuration Class	***Pre-compile***	x	All Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	--		

Name	TimeFailedThreshold		
Description	Defines the time out duration in ms for “Event Failed” qualification.		
Type or Unit	#define		
Range	$0..2^{32}$		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2

Scope	ECU
Dependency	--

Name	`TimePassedThreshold`		
Description	Defines the time out duration in ms for “Event Passed” qualification.		
Type or Unit	`#define`		
Range	`0..2`^{32}		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.22 DemPredebounceCounterBased

SWS Item	**Dem144:**
Container Name	DemPredebounceCounterBased
Description	This container contains the configuration (parameters) for DemPredebounceCounterBased.
Configuration Parameters	

Name	`PreDebounceName`		
Description	Defines the selected debounce algorithm		
Type or Unit	`#define`		
Range	`Dem_PredebounceCounterBased`		
	--		
Configuration Class	***Pre-compile***	x	All Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	--		

Name	`CountOutStepSize`		
Description	Defines the Step size for incrementation (PREFAILED).		
Type or Unit	`#define`		
Range	`0..127`		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	--		

Name	`CountInStepSize`
Description	Defines the Step size for decrementation (PREPASSED)

Type or Unit	`#define`		
Range	`0..127`		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	--		

Name	`JumpUp`		
Description	Switch for the activation of Jump-UP		
Type or Unit	`#define`		
Range	`0`	JumpUp not activated	
	`1`	JumpUp activated	
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	--		

Name	`JumpDown`		
Description	Switch for the activation of Jump-Down – only in combination with Jump-UP activation.		
Type or Unit	`#define`		
Range	`0`	JumpDown not activated	
	`1`	JumpDown activated	
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	JumpUp		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

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

8.2.23 DemPredebounceFrequencyBased

Name	`PreDebounceName`	
Description	Defines the selected debounce algorithm	
Type or Unit	`#define`	
Range	`DemPredebounceFreque`	

	ncyBased		
	--		
Configuration Class	***Pre-compile***	x	All Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	--		

Name	DurationOfTimeWindow		
Description	Defines duration of the Time Window.		
Type or Unit	#define		
Range	2^{32}		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	--		

Name	ThresholdForEventTestedFailed		
Description	Defines the threshold for FAILED-detection		
Type or Unit	#define		
Range	$0..2^{16}$		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	--		

Name	ThresholdForEventTestedPassed		
Description	Defines the threshold for PASSED-detection		
Type or Unit	#define		
Range	$0..2^{16}$		
	--		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	
	Post Build	x	Variant 2
Scope	ECU		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.24 DemPredebounceMonitorInternal

SWS Item	**Dem146:**		
Container Name	DemPredebounceMonitorInternal		
Description	This container contains the configuration (parameters) for DemPredebounceMonitorInternal		
Configuration Parameters			
Name	`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. Only the Prefix part can be defined.		
Type or Unit	`#define`		
Range	`Xxx`		
	--		
Configuration Class	***Pre-compile***	x	All Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	--		

Name	`Prefix_DemInitMonitorEVENTID`		
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. Only the Prefix part can be defined. Event Id part of the name is given by the assigned Event Id		
Type or Unit	`#define`		
Range	`Xxx`		
	--		
Configuration Class	***Pre-compile***	x	All Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.2.25 NVRAMBlockIDList

SWS Item	**Dem147:**		
Container Name	NVRAMBlockIDList		
Description	This container contains the configuration (parameters) for NVRAMBlockIDList		
Configuration Parameters			

Name	`BlockName`	
Description	Defines a real name of NVRAM block, this name is used as a symbolic name.	
Type or Unit	`#define`	
Range	`<Name>`	
	--	

Configuration Class	***Pre-compile***	x	All Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	--		

Name	`BlockID`		
Description	Defines a corresponding BlockID to the BlockName (see parameter above)		
Type or Unit	`#define`		
Range	`1..2`^{16}	Depends on NVRAM Manager configuration	
	--		
Configuration Class	***Pre-compile***	x	All Variants
	Link time	--	
	Post Build	--	
Scope	ECU		
Dependency	Used BlockIDs are defined by NVRAM manager configuration		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

8.3 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.

SWS Item	**Dem112:**	
Information elements		
Information element name	***Type / Range***	***Information element description***
`DEM_VENDOR_ID`	`#define/ uint16`	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
`DEM_MODULE_ID`	`#define/ uint8`	Module ID of this module from Module List
`DEM_AR_MAJOR_VERSION`	`#define/ uint8`	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
`DEM_AR_MINOR_VERSION`	`#define/ uint8`	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
`DEM_AR_PATCH_VERSION`	`#define/ uint8`	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
`DEM_SW_MAJOR_VERSION`	`#define/ uint8`	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
`DEM_SW_MINOR_VERSION`	`#define/ uint8`	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
`DEM_SW_PATCH_VERSION`	`#define/ uint8`	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

9 Service Diagnostic Event Manager (DEM)

9.1 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.

9.2 Overview

9.2.1 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-Cs as indicated in **Figure 1**.

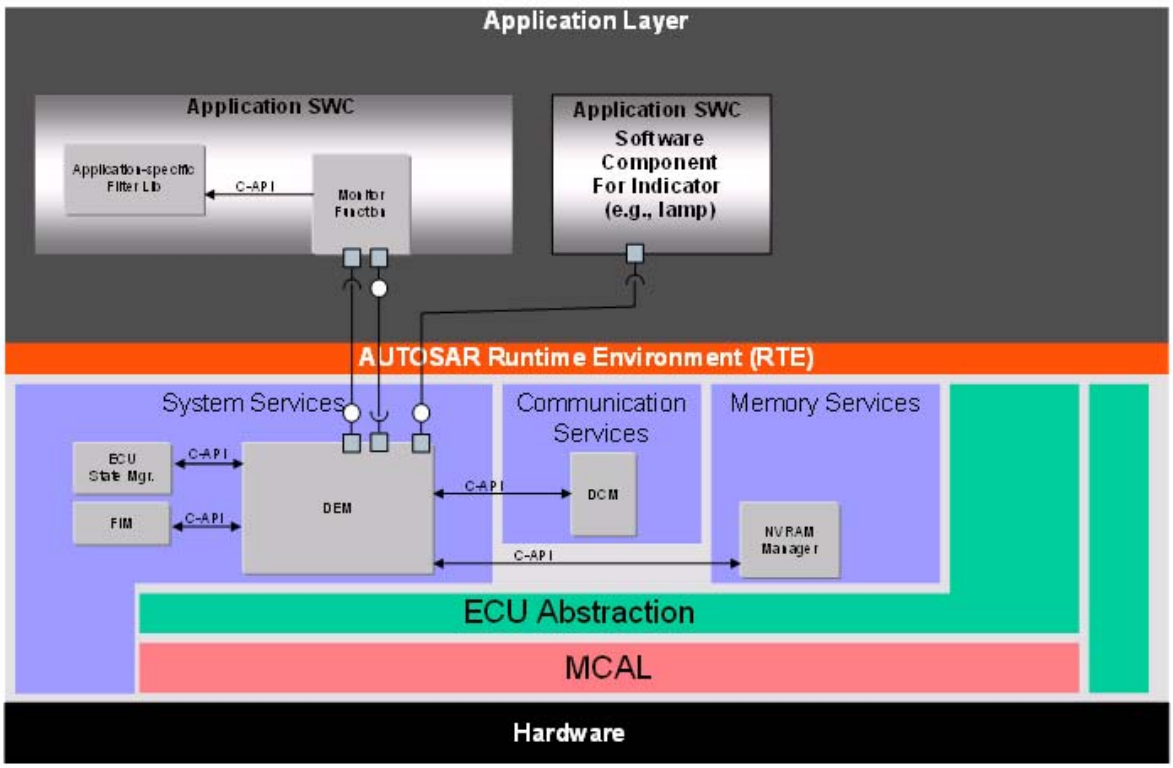


Figure 1: DEM in the ECU software architecture.

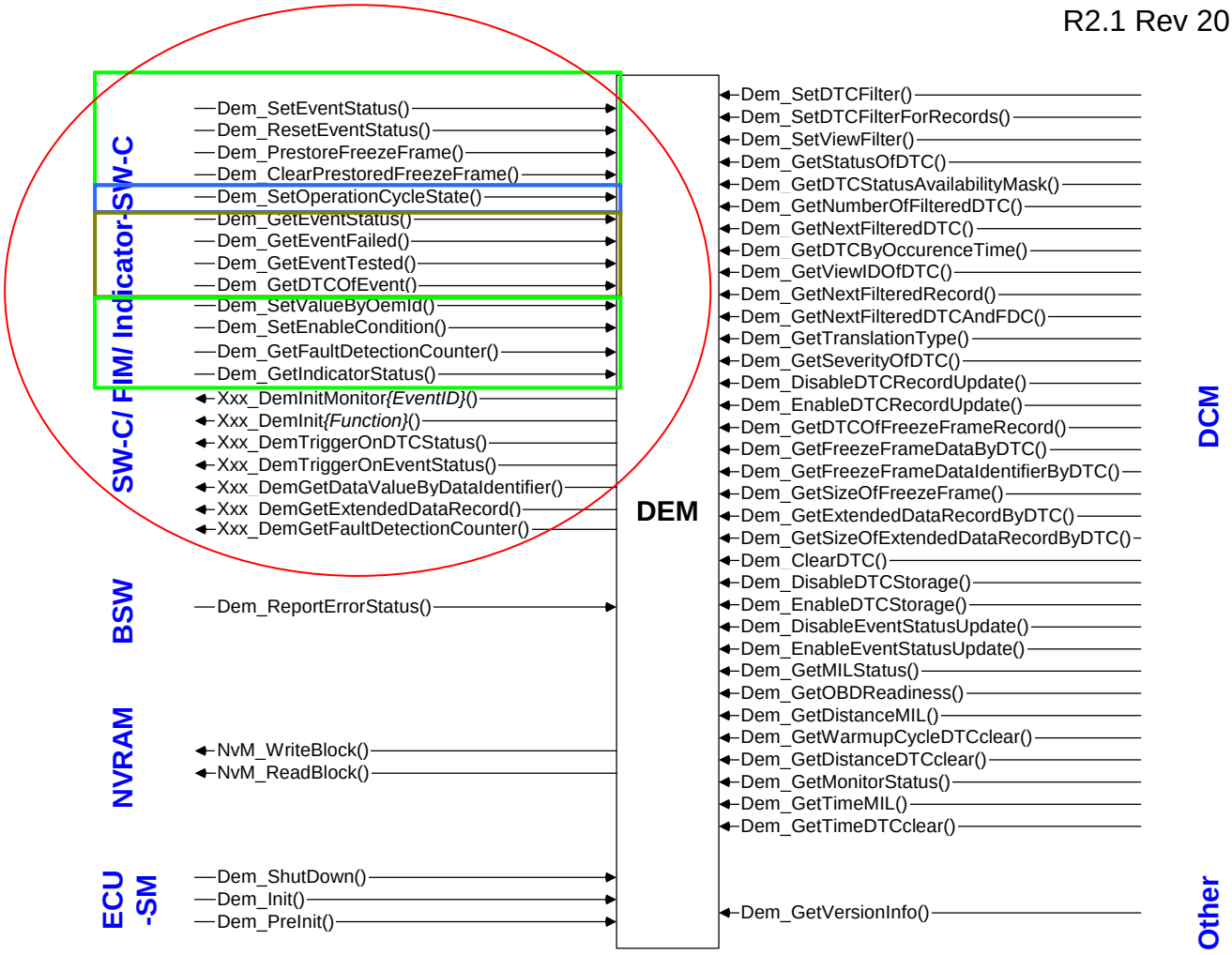


Figure 2: Communication relationships of the DEM. The red circle indicates RTE-relevant communication

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.

9.2.2 Requirements

The requirements for the functionality of the DEM service are specified in this document above.

9.2.3 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.

9.2.3.1 Initialization of event-specific part of the monitor

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

9.2.3.2 Initialization of function-specific part of the monitor

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

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

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

9.2.3.4 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.

9.2.3.5 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.

9.3 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-Type	Uint8	Dem_GetEventStatus, Dem_EventStatus-Changed
Dem_DTCTranslationFormat Type	UInt8 (DEM_DTC_TRANSLATION_ISO15031_6, DEM_DTC_TRANSLATION_ISO14229_1, DEM_DTC_TRANSLATION_CUSTOMER, DEM_DTC_TRANSLATION_INTERNAL)	Dem_GetDTCOfEvent
Dem_ReturnGetDTCOfEvent-Type	UInt8 (DEM_GET_DTCOFEVENT_OK, DEM_GET_DTCOFEVENT_WRONG_EVENTID, DEM_GET_DTCOFEVENT_WRONG_TRANSLATIO N)	Dem_GetDTCOfEvent
Dem_InitMonitorKindType	UInt8 (DEM_INIT_MONITOR_CLEAR, DEM_INIT_MONITOR_RESTART)	Dem_InitMonitorForEvent
Dem_DTCStatusMaskType	Uint8	Dem_DTCStatus-Changed
Dem_DTCOriginType	UInt8 (DEM_DTC_ORIGIN_PRIMARY_MEMORY, DEM_DTC_ORIGIN_SECONDARY_MEMORY, DEM_DTC_ORIGIN_MIRROR_MEMORY)	Dem_GetSeverityOfDTC
Dem_DTCSeverityType	UInt8 (DEM_SEVERITY_NO_SEVERITY, DEM_SEVERITY_MAINTENANCE_ONLY, DEM_SEVERITY_CHECK_AT_NEXT_HALT, DEM_SEVERITY_CHECK_IMMEDIATELY)	Dem_GetSeverityOfDTC
Dem_ReturnGetSeverityOf-DTC	UInt8 (DEM_GET_SEVERITYOFDTC_OK, DEM_GET_SEVERITYOFDTC_WRONG_DTC, DEM_GET_SEVERITYOFDTC_WRONG_DTCORIGI N, DEM_GET_SEVERITYOFDTC_NOSEVERITY)	Dem_GetSeverityOfDTC
Dem_OperationCycleIdType	Uint8	Dem_SetOperationCycle-State
Dem_CycleStateType	UInt8 (DEM_CYCLE_STATE_START, DEM_CYCLE_STATE_END)	Dem_SetOperationCycle-State

```
PrimitiveTypeWithSemantics Dem_EventStatusType {
      IntegerType {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
      // 4..255 -> custom status values
}

PrimitiveTypeWithSemantics Dem_DTCTranslationFormatType {
      IntegerType {LOWER-LIMIT=0, UPPER-LIMIT=3};
      0 -> DEM_DTC_TRANSLATION_ISO15031_6
      1 -> DEM_DTC_TRANSLATION_ISO14229_1
      2 -> DEM_DTC_TRANSLATION_CUSTOMER
      3 -> DEM_DTC_TRANSLATION_INTERNAL
}

PrimitiveTypeWithSemantics Dem_ReturnGetDTCOfEventType {
      IntegerType {LOWER-LIMIT=0, UPPER-LIMIT=2};
      0 -> DEM_GET_DTCOFEVENT_OK
      1 -> DEM_GET_DTCOFEVENT_WRONG_EVENTID
      2 -> DEM_GET_DTCOFEVENT_WRONG_TRANSLATION
}

PrimitiveTypeWithSemantics Dem_InitMonitorKindType {
      IntegerType {LOWER-LIMIT=0, UPPER-LIMIT=1};
      0 -> DEM_INIT_MONITOR_CLEAR
      1 -> DEM_INIT_MONITOR_RESTART
}
```

```
PrimitiveTypeWithSemantics Dem_DTCOriginType {
      IntegerType {LOWER-LIMIT=0, UPPER-LIMIT=2};
      0 -> DEM_DTC_ORIGIN_PRIMARY_MEMORY
      1 -> DEM_DTC_ORIGIN_SECONDARY_MEMORY
      2 -> DEM_DTC_ORIGIN_MIRROR_MEMORY
}

PrimitiveTypeWithSemantics Dem_DTCSeverityType {
      IntegerType {LOWER-LIMIT=0, UPPER-LIMIT=2};
      0 -> DEM_SEVERITY_NO_SEVERITY
      1 -> DEM_SEVERITY_MAINTENANCE_ONLY
      2 -> DEM_SEVERITY_CHECK_AT_NEXT_HALT
      3 -> DEM_SEVERITY_CHECK_IMMEDIATELY
}


PrimitiveTypeWithSemantics Dem_ReturnGetSeverityOfDTC {
      IntegerType {LOWER-LIMIT=0, UPPER-LIMIT=3};
      0 -> DEM_GET_SEVERITYOFDTC_OK
      1 -> DEM_GET_SEVERITYOFDTC_WRONG_DTC
      2 -> DEM_GET_SEVERITYOFDTC_WRONG_DTCORIGIN
      3 -> DEM_GET_SEVERITYOFDTC_NOSEVERITY
}


PrimitiveTypeWithSemantics Dem_CycleStateType {
      IntegerType {LOWER-LIMIT=0, UPPER-LIMIT=1};
      0 -> DEM_CYCLE_STATE_START
      1 -> DEM_CYCLE_STATE_END
}
```

9.4 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.
Figure 3 shows how AUTOSAR Software components (single or multiple instances) are connected via service ports to the DEM.

9.4.1 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 is provided per EventId by the *DEMService*. It has EventId as a port-defined argument.

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

      SetEventStatus(IN EventStatusType EventStatus);
```

```
      ResetEventStatus();
      GetEventStatus(OUT EventStatusExtendedType EventStatus);
      GetEventFailed (OUT Boolean Failed);
      GetEventTested (OUT Boolean Failed);
      GetDTCOfEvent (IN DTCTranslationFormatType Format, OUT UInt32 DTC,
                     OUT ReturnGetDTCOfEvent);
      PrestoreFreezeFrame(); //OPTIONAL, only for OBD-relevant events
      ClearPrestoredFreezeFrame(); //OPTIONAL, only for OBD-relevant events
      SetOperationCycleState(IN OperationCycleIdType OperationCycleId, IN
                             OperationCycleStateType CycleState);
      SetValueByOemId(IN uint16 OemID,OUT uint8 DemDataValueByDataIDBuffer,
                     IN uint8 DemDataValueByDataIDBufferLength);
      SetEnableCondition(IN uint8 EnableConditionID,IN boolean
                         ConditionFulfilled);
}
```

9.4.2 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. The EventId must **not** be passed as the first argument of the operation, because the monitor does not cope with EventIds explicitly.

The configuration of the callback functions must be done during ECU Configuration of the DEM. The *InitMonitorFunction* parameter of the DEM configuration must be configured with the name of the RTE_Call_XXX of the client/server operation.

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  {
  // Init functions are used to notify the monitor from the DEM (from DEM
SWS 8.4.3.1)
  InitMonitorForEvent(IN InitMonitorKindType InitMonitorKind);
}

ClientServerInterface CallbackInitMonitorForFunction  {
  // Init functions are used to notify the monitor from the DEM (from DEM
SWS 8.4.3.2)
  InitMonitorForFunction();
}

ClientServerInterface CallbackEventStatusChange {
  // used to notify the monitor from the DEM (from DEM SWS 8.4.3.1.3)
   EventStatusChanged(IN EventStatusExtendedType New,
EventStatusExtendedType Old);
    // this operation was called TriggerOnEventStatus
```

```
}

ClientServerInterface CallbackDTCStatusChange  {
        // used to notify the monitor from the DEM (from DEM SWS 8.4.3.1.4)
        DTCStatusChanged(     IN UInt32 DTC,
                                                IN DTCStatusMaskType New,
                        DTCStatusMaskType Old);
   // was called TriggerOnDTCStatus
ClientServerInterface CallbackGetDataValueByDataIdentifier  {
        // used to get data from SW-C
        GetDataValueByDataIdentifier (IN UInt16 DataID,
                                      OUT DemDataValueByDataIDBuffer New,
                                 DTCStatusMaskType Old);
   }
```

The DEM uses the RTE “indirect API” to implement callbacks.

```
      /* Return an array of all ports that provide the interface
CallbackEventStatusChange. Because of the specific naming conventions
chosen, the element n in this array of ports will reference to the port
CE<nnn>.  For example monitorCallbackPorts[1] will be a handle on port
CE001 */
      monitorCallbackPorts = Rte_Ports_EventCallbacks_P();
```

To signal that the status of an event has changed, the DEM must call:

```
      monitorCallbackPorts[n].StatusChanged(newStatus)
```

9.4.3 DTC APIs

```
ClientServerInterface DTCInfo  {
GetSeverityOfDTC(IN UInt32 DTC,
          IN DTCOriginType Origin,
          OUT DTCSeverityType Severity,
          OUT ReturnGetSeverityOfDTC ReturnValue);
}
ClientServerInterface OperationCycle {
    SetOperationCycleState(OperationCycleIdType      OperationCycleId,
CycleStateType CycleState)
}
```

9.4.4 Unused APIs

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

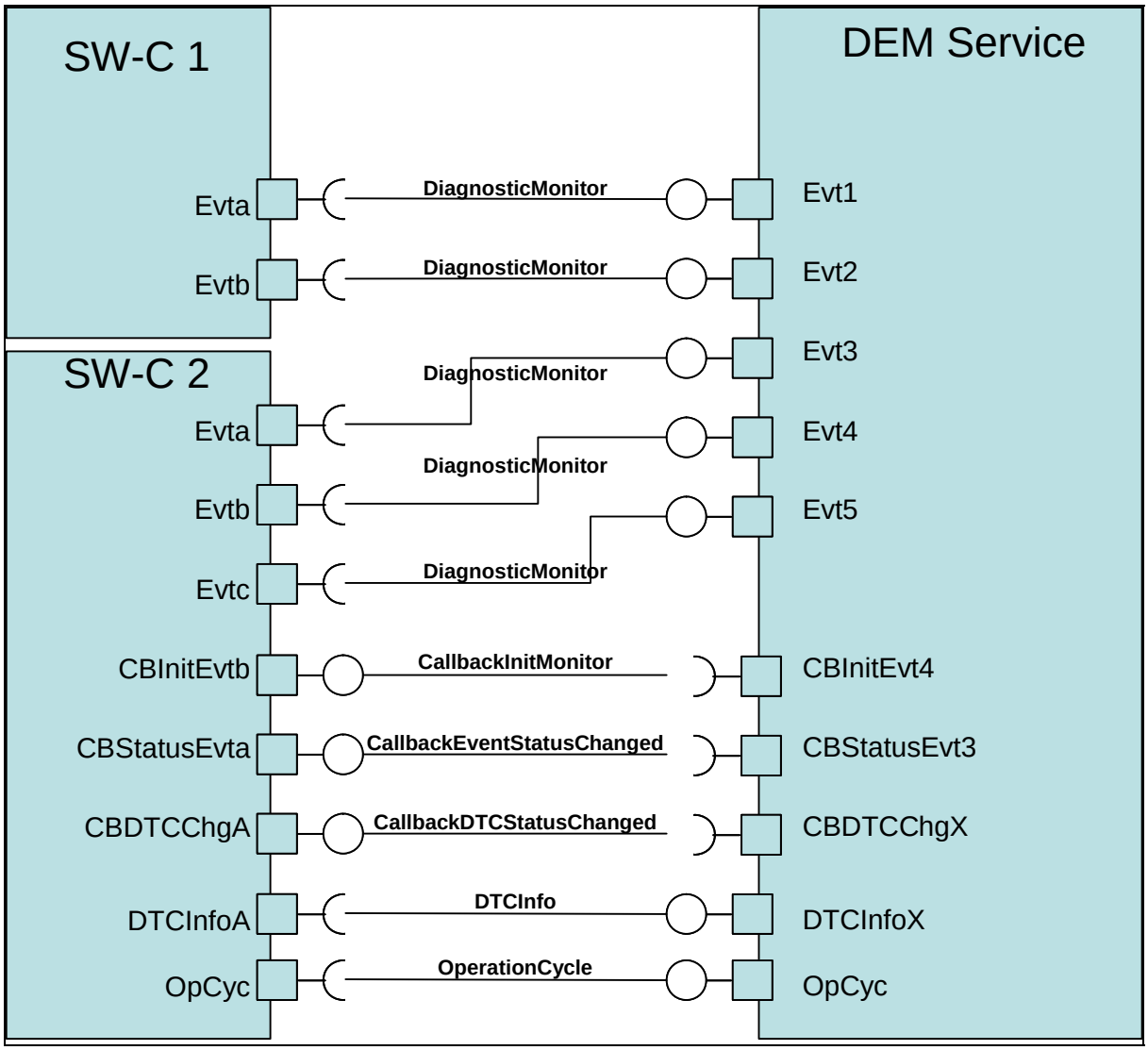


Figure 3: Example of SW-Cs connected to the DEM via service ports.

9.4.5 Definition of the Service DEM

```
IntegerType EventIdType {
          LOWER-LIMIT = 0;
          UPPER-LIMIT = <xx>;
     };
```

This type does not show up in the service ports of the client components, because the event identifier is implemented as port defined argument value, which is part of the InternalBehavior of the DEM Service. So the ECU dependency of EventIdType is not visible for the clients.

```
Service DEM {
ProvidePort DiagnosticMonitor Evt000
```

```
ProvidePort DiagnosticMonitor Evt001
…
ProvidePort DiagnosticMonitor EvtNNN

RequirePort CallbackInitMonitor CBInitEvt000
RequirePort CallbackInitMonitor CBInitEvt001
…
RequirePort CallbackInitMonitor CBInitEvtNNN

RequirePort CallbackEventStatusChanged CBStatusEvt000
RequirePort CallbackEventStatusChanged CBStatusEvt001
…
RequirePort CallbackEventStatusChanged CBStatusEvtNNN

RequirePort CallbackDTCStatusChanged CBStatusDTC000
RequirePort CallbackDTCStatusChanged CBStatusDTC001
…
RequirePort CallbackDTCStatusChanged CBStatusDTCNNN

ProvidePort DTCInfo DTCInfo000;
ProvidePort DTCInfo DTCInfo001;
…
ProvidePort DTCInfo DTCInfoNNN;

ProvidePort DiagnosticMonitor OpCyc;

};


/* 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 = FALSE
RunnableEntity ResetEventStatus
            symbol “DEM_ResetEventStatus”
            canbeInvokedConcurrently = FALSE
RunnableEntity GetEventStatus
            symbol “DEM_GetEventStatus”
            canbeInvokedConcurrently = FALSE
RunnableEntity GetEventFailed
            symbol “DEM_GetEventFailed”
            canbeInvokedConcurrently = FALSE
RunnableEntity GetEventTested
            symbol “DEM_GetEventTested”
            canbeInvokedConcurrently = FALSE
RunnableEntity GetDTCOfEvent
            symbol “DEM_GetDTCOfEvent”
            canbeInvokedConcurrently = FALSE
RunnableEntity PrestoreFreezeFrame
            symbol “DEM_PrestoreFreezeFrame”
            canbeInvokedConcurrently = FALSE
RunnableEntity ClearPrestoredFreezeFrame
            symbol “DEM_ClearPrestoredFreezeFrame”
            canbeInvokedConcurrently = FALSE
```

```
// for each port providing the DiagnosticMonitor, DTCInfo or OperationCycle
Interface:
PortArgument {port= PS0, value.type=EventIdType, value.value=0}
      …
PortArgument {port= PS<nn>, value.type=EventIdType, value.value=<nn>}
};
```

10 Changes to Release 1

10.1 Deleted SWS Items

SWS Item	***Rationale***
Dem101	Obsolete requirement
Dem008	Obsolete requirement
Dem012	Obsolete requirement
Dem030	Obsolete requirement

10.2 Changed SWS Items

SWS Item	***Rationale***
Dem005	Refinement of requirement
Dem006:	Refinement of requirement
Dem023:	Extension and 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
Dem106:	Extension 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

10.3 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
Dem128:	New requirement for configuration container
Dem129:	New requirement for configuration container
Dem137:	New requirement for configuration container

Dem138:	New requirement for configuration container
Dem139:	New requirement for configuration container
Dem130:	New requirement for configuration container
Dem131:	New requirement for configuration container
Dem132:	New requirement for configuration container
Dem136:	New requirement for configuration container
Dem135:	New requirement for configuration container
Dem134:	New requirement for configuration container
Dem140:	New requirement for configuration container