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

Document Version	4.2.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Change Description
09.12.2011	4.2.0	AUTOSAR Administration	• Introduced multiple formats per DTC • Reworked Dem_ResetEventStatus behavior • Reworked Dlt interaction • Reworked Dem/Dcm interface • Corrected include-structure and RTE interfaces • Refined several aspects on features
18.10.2010	4.1.0	AUTOSAR Administration	• Reworked Dem/Dcm interface • Extended definition of “Diagnostic Monitor” • Introduced "Event significance" and "DTC suppression" • Reworked OBD (esp. interface for service $02, readiness, and permanent memory) • Reworked file-structure • Finalization of issues on Revision 1

04.12.2009	4.0.0	AUTOSAR Administration	• Improved and extended of DEM functional description (especially: status bit handling, event displacement, debouncing, fault confirmation and indicator handling, event combination, enable- & storage conditions, event related data, operation cycle management) • Introduced the approach for functional diagnostics of SW-Cs • Added Dlt interaction and debugging interface • Document structure reworked and extended • Legal disclaimer revised
02.08.2008	3.0.0	AUTOSAR Administration	• Document structure reworked and extended • Add APIs and configuration parameters for OBD support • Improve interaction between DCM and software components • Legal disclaimer revised
28.11.2007	2.2.0	AUTOSAR Administration	• Improvement of RTE compliance • Improvement of configuration part • Improvement of document structure • Rework and tightening of data type useage • New API added • Document meta information extended • Small layout adaptations made

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

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 service component Diagnostic Event Manager (Dem) is responsible for processing and storing diagnostic events (errors) and associated 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 and to other BSW modules.

The basic target of the Dem specification document is to define the ability for a common approach of a “diagnostic fault memory” for automotive manufacturers and component suppliers

This specification defines the functionality, API and the configuration of the AUTOSAR basic software module Diagnostic Event Manager (Dem). Parts of the internal behavior are manufacturer specific and described in the Limitations chapter.

2 Acronyms and abbreviations

Acronym:	*Description:*
N_OK	Not OK
P-Code	Power train code
Freeze frame	Freeze frame is defined as a record of data (DIDs/PIDs). Freeze frames are the same as SnapShotRecords in ISO 14229-1 [15].
Extended data record	An extended data record is a record to store specific information assigned to a fault.
Monitor	A diagnostic monitor is a routine entity determining the proper functionality of a component. Alternatively the term “diagnostic function” can be used.
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, etc.)
Aging	Unlearning/deleting of a no longer failed event/DTC after a defined number of operation cycles from event memory
Healing	Switching of the warning indicator including the handling of reported passed results over a period of time / several operation cycles
PossibleErrors	PossibleErrors means the ApplicationErrors as defined in meta model
Event debouncing	Debouncing is a specific mechanism (e.g. counter-based) to evaluate, if the diagnostic event gets qualified. This works on top of potential signal debouncing and can be done within the SW-C or inside the Dem.
Event qualification	A diagnostic event is qualified in case of a passed or a failed result is set (Dem-internal or reported from another BSW module or SW-C).
Event confirmation	A diagnostic event is confirmed in case of repeated detection of qualified events over cycles or time evaluated by means of fault confirmation counters. Therefore, also the UDS DTC Status bit 3 (ConfirmedDTC) is set.
Event memory overflow indication	The event memory overflow indication indicates, if this specific event memory is full and the next event occurs to be stored in this event memory.
Readiness	The readiness refers to the tested bits TestNotCompletedSinceLastClear (bit 4) and TestNotCompleteThisOperationCycle (bit 6) of the UDS DTC Status Byte.

Abbreviation:	*Description:*
API	Application Programming Interface
BSW	Basic Software
CRC	Cyclic Redundancy Check
Dcm	Diagnostic Communication Manager
Dem	Diagnostic Event Manager
Det	Development Error Tracer
DID	Data Identifier
Dlt	Diagnostic Log and Trace
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
EcuM	Electronic Control Unit Manager
FDC	Fault Detection Counter
FiM	Function Inhibition Manager
HW	Hardware
ID	Identification/Identifier
ISO	International Standardization Organization
IUMPR	In Use Monitoring Performance Ratio
MIL	Malfunction Indication Light
NVRAM	Non volatile RAM
OBD	Onboard Diagnostics
OEM	Original Equipment Manufacturer (Automotive Manufacturer)
OS	Operating System

PID	Parameter Identification
PTO	Power Take Off
RAM	Random Access Memory
ROM	Read-only Memory
RTE	Runtime Environment
SSCP	synchronous server call point
SW	Software
SW-C	Software Component
UDS	Unified Diagnostic Services

3 Related documentation

3.1 Input documents

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

[2] Specification of ECU Configuration,
AUTOSAR_TPS_ECUConfiguration.pdf

[3] Layered Software Architecture,
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

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

[5] Requirements on Diagnostic,
AUTOSAR_SRS_Diagnostic.pdf

[6] Specification of NVRAM Manager,
AUTOSAR_SWS_NVRAMManager.pdf

[7] Specification of Diagnostic Communication Manager
AUTOSAR_SWS_DiagnosticCommunicationManager.pdf

[8] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[9] Software Component Template,
AUTOSAR_TPS_SoftwareComponentTemplate.pdf

[10] Specification of Function Inhibition Manager,
AUTOSAR_SWS_FunctionInhibitionManager.pdf

[11] Specification of Diagnostic Log and Trace,
AUTOSAR_SWS_DiagnosticLogAndTrace.pdf

3.2 Related standards and norms

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

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

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

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

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

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

[18] SAE J1979 Rev May 2007

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

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

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

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

4 Constraints and assumptions

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

[Dem126] 「There shall only be one Dem module available per ECU.」()

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

4.1 Limitations

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

The Dem is able to support additional event memories (Permanent memory, Mirror memory and Secondary memory), but the specific event memory processing is not defined in detail.

Some details on the interaction between Dem and specific emission-related SW-C are not specified in this specification, since they are dependent on the SW-C implementation. The following functionality is not defined:

- Malfunction Indicator Lamp (MIL)-activation (MIL handler interaction to Dem, MIL-bulb check, readiness blinking, blinking in case of catalyst damaging misfire, etc.)
- misfire fault handling (debouncing over all cylinders, filtering single / multiple misfire faults)
- support of similar conditions for the specific healing of misfire and fuel system faults

Note: For OBD2, it is required that misfire and fuel system fault shall only be healed (yielding leaving service $07) under the similar conditions as they have been detected. The "similarity" is derived from a "window" spanned by ranges on engine speed, engine load and temperature conditions being present at the time of fault detection.

Absolute freeze frame record addressing functionality (refer to chapter 7.8.2.2) and configuration is not specified.

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

The Dcm may lock the event memory update (refer to Dem270) while processing the “read diagnostic data” service, due to architectural design.

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

Post build time loadable configuration is not supported by the Dem configuration model and respective parameter set.

4.2 Applicability to car domains

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

5 Dependencies to other modules

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

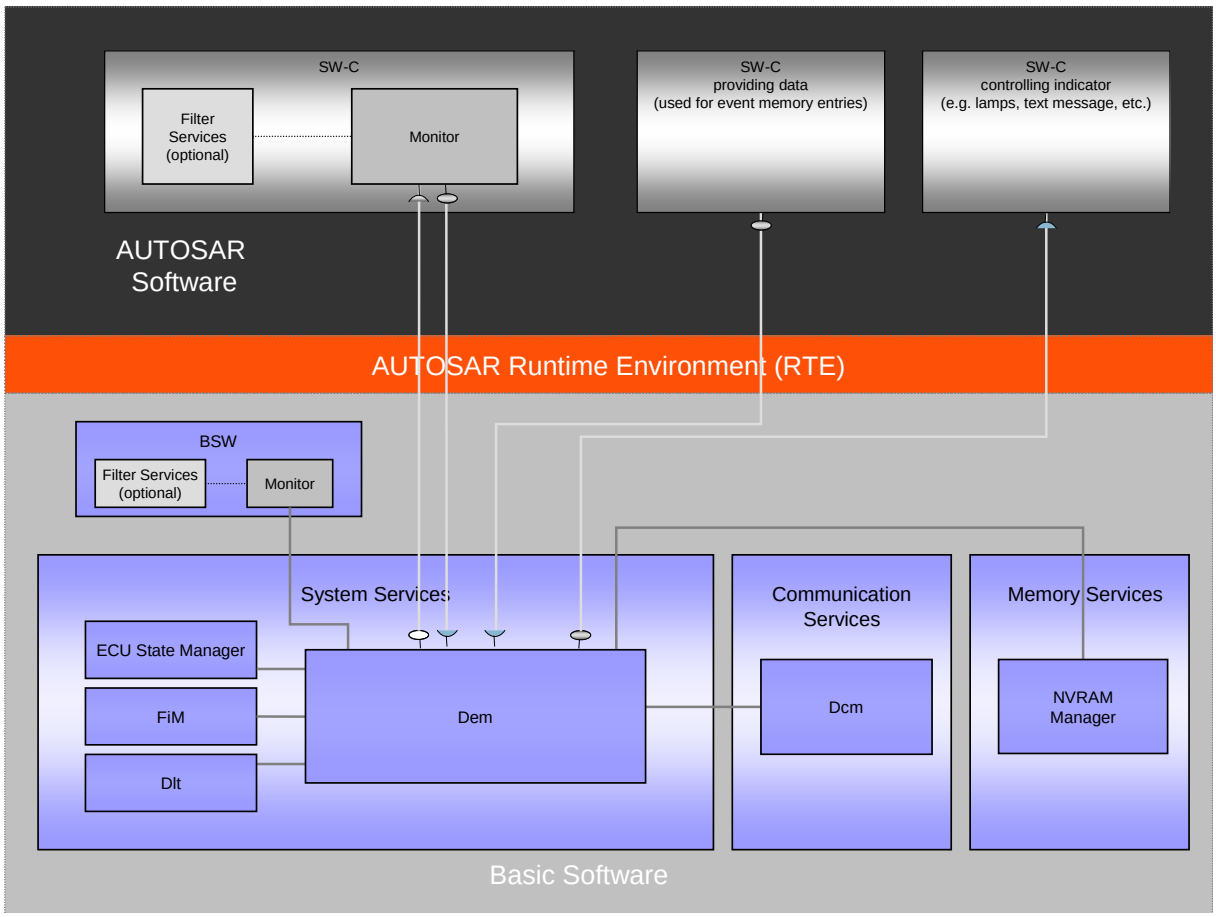

Figure 1 Dependencies of the Diagnostic Event Manager (Dem) to other software modules

- The **Function Inhibition Manager (FiM)** (refer to [10]) stands for the evaluation and assignment of events to the required actions for Software Components (e.g. inhibition of specific “Monitors”). 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.

- The **Diagnostic Log and Trace (Dlt)** (refer to [11]) provides a generic Logging and Tracing functionality for the Dem. The Dem informs and updates the Diagnostic Log and Trace (Dlt) upon changes of the event status and provides access on the current event related data in order to log and trace this information.

- The **Diagnostic Communication Manager (Dcm)** (refer to [7]) 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, etc.) which will be transferred to the external diagnostic scan tool afterwards.

- **Software-Components (SW-C)** and **Basic Software (BSW) modules** can access the Dem to update and/or retrieve current event status information. SW-Cs and BSW modules can retrieve data from the Dem e.g. to turn the indicator lamps on or off. The **monitor** is a sub-component of a SW-C / BSW module.

- **Data Provider** SW-Cs and/or BSW modules will provide data (e.g. event related data) required by the Dem, for example, to be able to create event memory entries.

- The **NVRAM Manager (NvM)** (refer to [5]) provides mechanisms to store data blocks in NVRAM. 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 **ECU State Manager (EcuM)** is responsible for the basic initialization and de-initialization of basic software components including Dem.

- The **RTE** implements scheduling mechanisms for BSW, e.g. assigns priority and memory protection to each BSW module used in an ECU.

5.1 File structure

5.1.1 Code file structure

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

- Dem_Lcfg.c – for link time configurable parameters (but **not** used by Dem in this version).
- Dem_PBcfg.c – for post build time configurable parameters

These files shall contain all link time and post-build time configurable parameters.⌋ (BSW158, BSW346, BSW00380, BSW00381, BSW00383, BSW00384, BSW00412, BSW00415, BSW00419, BSW00435, BSW00436)

5.1.2 Header file structure

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

- Dem.h – header file of Dem module
- Dem_Types.h – for Dem data types (not defined in Rte_Dem_Type.h)

- Rte_Dem_Type.h – Dem-specific Application Types Header file
- Std_Types.h – includes all definitions of standard types
- Dcm_Types.h – for all imported Dcm types (refer to chapter 8.1)
- Dem_Lcfg.h – for link time configurable parameters (but **not** used by Dem in this version)
- Dem_PBcfg.h – for post build time configurable parameters
- SchM_Dem.h – for Basic Software Module Scheduler symbols
- Fim.h – for Function Inhibition Manager symbols
- Dlt.h – for Diagnostic Log & Trace symbols
- NvM.h – for NVRAM Manager symbols
- MemMap.h – for memory mapping
- Rte_Dem.h – for Dem-internal RTE symbols
- <…>.h – contains all C-callback declarations (refer to DemHeaderFileInclusion in DemGeneral) configured for the Dem」(BSW00447)

[Dem152] 「The Dem module shall provide all types, APIs (unless limited in chapter 8.3) and all required symbolic names via Dem.h. The Dem configuration tool shall assign ECU dependent values to the Id symbols.」(BSW00409)

Note: The symbolic names are generated for configuration containers containing an identifier parameter, like event Id symbols (refer to Dem115), operation cycles, indicators, enable/storage conditions, etc.

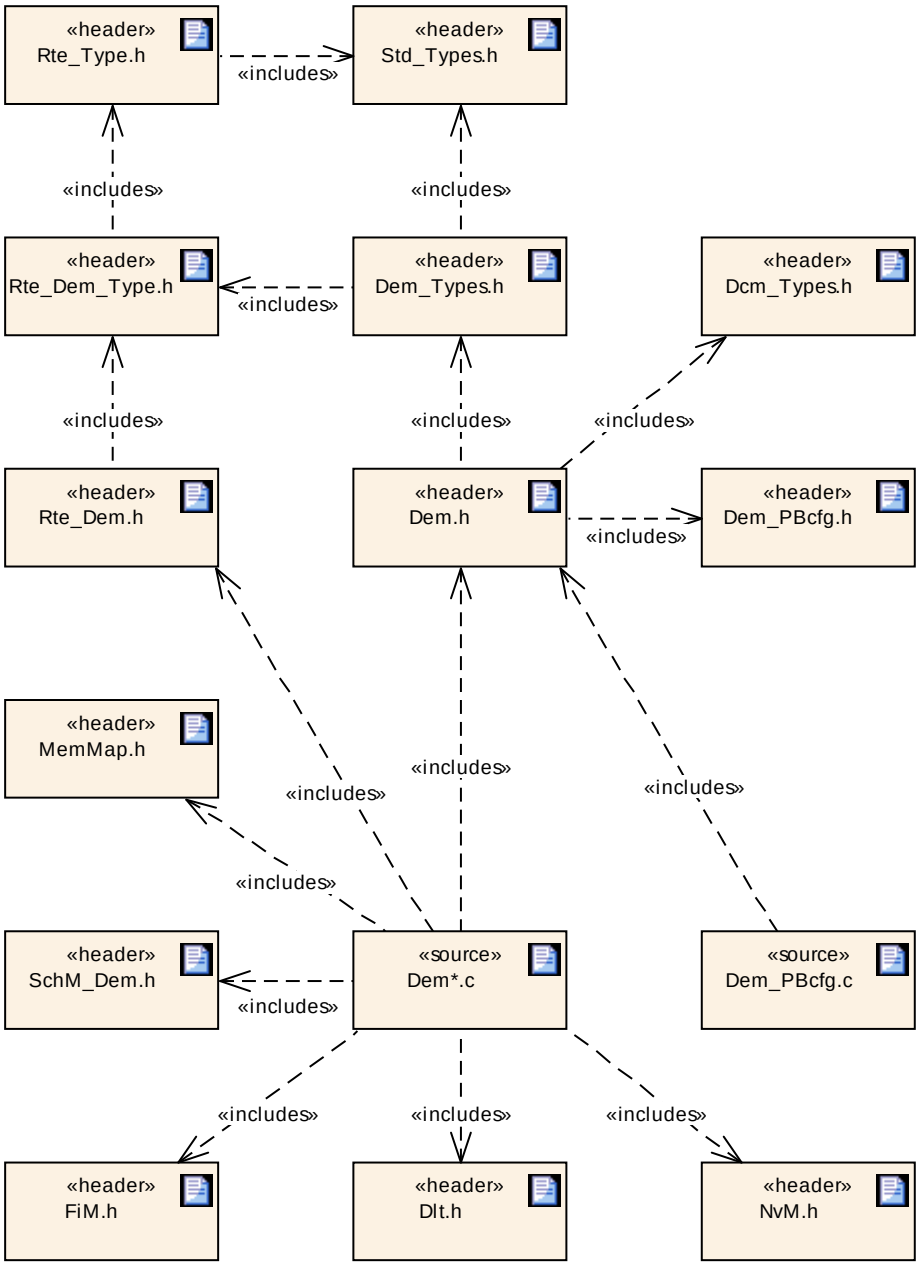

Figure 2 Header file structure

6 Requirements traceability

6.1 Document: General requirements on Basic Software Modules

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

Requirement	*Satisfied by*
[BSW331] Separation of error and status values	Not applicable
[BSW333] Documentation of callback function context	Documentation requirement
[BSW334] Provision of XML file	Implementation requirement
[BSW335] Status values naming convention	Implemented
[BSW336] Shutdown interface	Dem182
[BSW337] Classification of errors	Not applicable
[BSW338] Detection and Reporting of development errors	Not applicable
[BSW339] Reporting of production relevant errors and exceptions	Not applicable
[BSW341] Microcontroller compatibility documentation	Not applicable
[BSW342] Usage of source code and object code	Implementation requirement
[BSW343] Specification and configuration of time	Dem715_Conf, Dem716_Conf, Dem717_Conf
[BSW344] Post-Build configuration	Dem267, Dem268
[BSW345] Pre-Build configuration	Dem267, Dem268
[BSW346] Basic set of module files	Dem108
[BSW347] Naming separation of drivers	Not applicable
[BSW348] Standard type header	Not applicable
[BSW350] Development error detection keyword	Not applicable
[BSW353] Platform specific type header	Not applicable
[BSW355] Do not redefine AUTOSAR integer data types	Implementation requirement
[BSW357] Standard API return type	Not applicable
[BSW358] Return type of init() functions	Implemented
[BSW359] Return type of callback functions	Not applicable
[BSW360] Parameters of callback functions	Not applicable
[BSW361] Compiler specific language extension header	Not applicable
[BSW369] Do not return development error codes via API	Not applicable
[BSW370] Separation of callback interface from API	Implementation requirement
[BSW371] Do not pass function pointers via API	Implemented
[BSW373] Main processing function naming convention	No main processing function used
[BSW374] Module vendor identification	Not applicable
[BSW375] Notification of wake-up reason	Not applicable
[BSW376] Return type and parameters of main processing functions	No main processing function used
[BSW377] Module specific API return types	Chapter 8.2
[BSW378] AUTOSAR boolean type	Implementation requirement
[BSW379] Module identification	Not applicable
[BSW00380] Separate C-Files for configuration parameters	Dem108
[BSW00381] Separate configuration header file for pre-compile time parameters	Dem108
[BSW00382] Not-used configuration elements need to be listed	Not applicable
[BSW00383] List dependencies of configuration files	Dem108
[BSW00384] List dependencies to other modules	Dem108
[BSW00385] List possible error notifications	Dem113, Dem114
[BSW00386] Configuration for detecting an error	Dem116
[BSW00387] Specify the configuration class of callback function	Not applicable
[BSW00388] Introduce containers	Ref. to chapter 10. configuration definitions
[BSW00389] Containers shall have names	Ref. to chapter 10. configuration definitions
[BSW00390] Parameter content shall be unique within the module	Chapter 8.2
[BSW00391] Parameter shall have unique names	Chapter 8.2
[BSW00392] Parameters shall have a type	Chapter 8.2
[BSW00393] Parameters shall have a range	Chapter 8.2
[BSW00394] Specify the scope of the parameters	Chapter 8.2
[BSW00395] List the required parameters (per parameter)	Chapter 8.2

Requirement	Satisfied by
[BSW00396] Configuration classes	Dem267, Dem268
[BSW00397] Pre-compile-time parameters	Dem267, Dem268
[BSW00398] Link-time parameters	Dem267, Dem268
[BSW00399] Loadable Post-build time parameters	Dem267, Dem268
[BSW00400] Selectable Post-build time parameters	Dem267, Dem268
[BSW00401] Documentation of multiple instances of configuration parameters	Dem267, Dem268
[BSW00402] Published information	Dem112
[BSW00404] Reference to post build time configuration	Dem267, Dem268
[BSW00405] Reference to multiple configuration sets	Dem267, Dem268
[BSW00406] Check module initialization	Dem124, Dem169, Dem170
[BSW00407] Function to read out published parameters	Dem110, Dem111
[BSW00408] Configuration parameter naming convention	Implemented
[BSW00409] Header files for production code error IDs	Dem152 and note below
[BSW00410] Compiler switches shall have defined values	Implementation requirement
[BSW00411] Get version info keyword	Dem110, Dem111,Dem112
[BSW00412] Separate H-File for configuration parameters	Dem108
[BSW00413] Accessing instances of BSW modules	Implemented
[BSW00414] Parameter of init function	Implemented
[BSW00415] User dependent include files	Dem108
[BSW00416] Sequence of Initialization	Implemented
[BSW00417] Reporting of Error Events by Non-Basic Software	Dem107
[BSW00418] Allocation of error detection	Dem117
[BSW00419] Separate C-Files for pre-compile time configuration parameters	Dem108
[BSW00420] Production relevant error event rate detection	Dem107
[BSW00421] Reporting of production relevant error events	Dem107
[BSW00422] Debouncing of production relevant error status	Dem004
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Implemented
[BSW00424] BSW main processing function task allocation	Implementation Requirement
[BSW00425] Trigger conditions for schedulable objects	Implementation Requirement
[BSW00426] Exclusive areas in BSW modules	Implementation Requirement
[BSW00427] ISR description for BSW modules	Implementation Requirement
[BSW00428] Execution order dependencies of main processing functions	Implementation Requirement
[BSW00429] Restricted BSW OS functionality access	Implementation Requirement
[BSW00431] The BSW Scheduler module implements task bodies	Implementation Requirement
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Implementation Requirement
[BSW00433] Calling of main processing functions	Not applicable
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable
[BSW00435] Header File Structure for the Basic Software Scheduler	Dem108
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	Dem108

6.2 Document: Requirements on Diagnostic

Requirement	Satisfied by
General	
[BSW04010] Interface between Diagnostic service	Interface Dcm ⇔ Dem (Chapter 8.3.3.31),

handling and Diagnostic Event (error) management	Dem042, Dem020, Dem041, Dem208, Dem241
[BSW04082] Support of ISO15031-5 and SAE J1979	Chapter 7.7.1.2 and 7.7.1.4
[BSW04065] Clearing of events and event groups	Chapter 7.8.2.3
[BSW04067] Counting and evaluation of events according to ISO 14229-1 DTCStatusMask	Dem011, Chapter 7.3.1 and 7.8.2.1 (esp. Dem059)
[BSW04097] Decentralized modular diagnostic configuration of SW-Cs	Chapter 8.3.3.22 and 8.4.3.8
Diagnostic Event Mananger (Dem)	
[BSW04002] Basic SW Module for Diagnostic event (error) management	Defined by AUTOSAR architecture
[BSW04057] Classification of event	Dem057, Dem058, Dem156, Dem157
[BSW04061] Distinction between different function groups	Dem153
[BSW04063] Single EventId for each monitoring path	Dem153, Dem154
[BSW04066] Support of different event (fault) memories	Dem010, Dem559, Dem208, Dem212, Dem235, Dem236, Dem238, Dem239, Dem240, Dem241
[BSW04068] Debouncing of diagnostic events	Chapter 7.3.3, Dem019
[BSW04106] Aging/Unlearning of diagnostic events	Chapter 7.3.3, Dem019
[BSW04069] Warning indicator handling	Chapter 7.3.10.1 and 7.3.10.3
[BSW04070] Event ‘occurrence order’ definition	Dem161, Dem162, Dem219, Dem221, Dem396, Dem410
[BSW04071] Event importance definition	Chapter 7.1.1
[BSW04072] Extended event information	Dem internal
[BSW04073] Event combination and compression	Dem024, Dem025, Dem026
[BSW04074] Event related data	Chapter 7.3.7 (esp. Dem039, Dem040, Dem070, Dem071, Dem073, Dem074, Dem075, Dem076)
[BSW04079] The size of a FreezeFrame shall be reported to the Dcm by the Dem	Dem074
[BSW04104] Configuration of event related data	Chapter 7.3.7.4
[BSW04075] Event and DTC assignment	Configuration parameter “DemDTCClassRef” (Chapter 10.2)
[BSW04076] System Cycle definition	Dem019, Dem480, Dem388
[BSW04091] Notifications about new freeze frame data	Chapter 7.3.7.5
[BSW04092] Control of event handling	Dem514, Dem515, Dem516
[BSW04093] Memory Overflow indication	Chapter 7.3.2.2
[BSW04095] Enable & Storage conditions	Chapter 7.3.6
[BSW04096] Status bit support & handling	Chapter 7.3.1.1, 7.3.1.2 and 7.3.1.3
[BSW04105] Event memory management	Chapter 7.3.2.1 and 7.3.2.3
[BSW04102] Reporting of stored events	Chapter 7.3.2.4
[BSW04099] Notification about data changes for log & trace (Dlt)	Dem517
[BSW04107] Defensive behavior of the Dem module	Dem339
Diagnostic Event Mananger (Dem) - Interfaces and APIs	
[BSW04077] Interaction with NVRAM manager	Chapter 7.8.4
[BSW04030] Interaction with Software Components (SW-Cs)	Chapter 7.8.1
[BSW04031] Interaction with Function Inhibition Manager (FIM)	Dem029
Configuration	
[BSW04059] Configuration of timing parameter	Dem426
[BSW04064] Event buffer must be configurable concerning size	Configuration parameters “DemMaxNumberEventEntry<…>” (Chapter 10.2)

7 Functional specification

The **Diagnostic Event Manager (Dem)** handles and stores the events detected by diagnostic monitors in both Software Components (SW-Cs) and Basic software (BSW) modules. The stored event information is available via an interface to other BSW modules or SW-Cs.

Figure 3 shows the Dem configuration. DemGeneral contains the global part of the configuration and DemConfigSet contains the multiple configuration part.

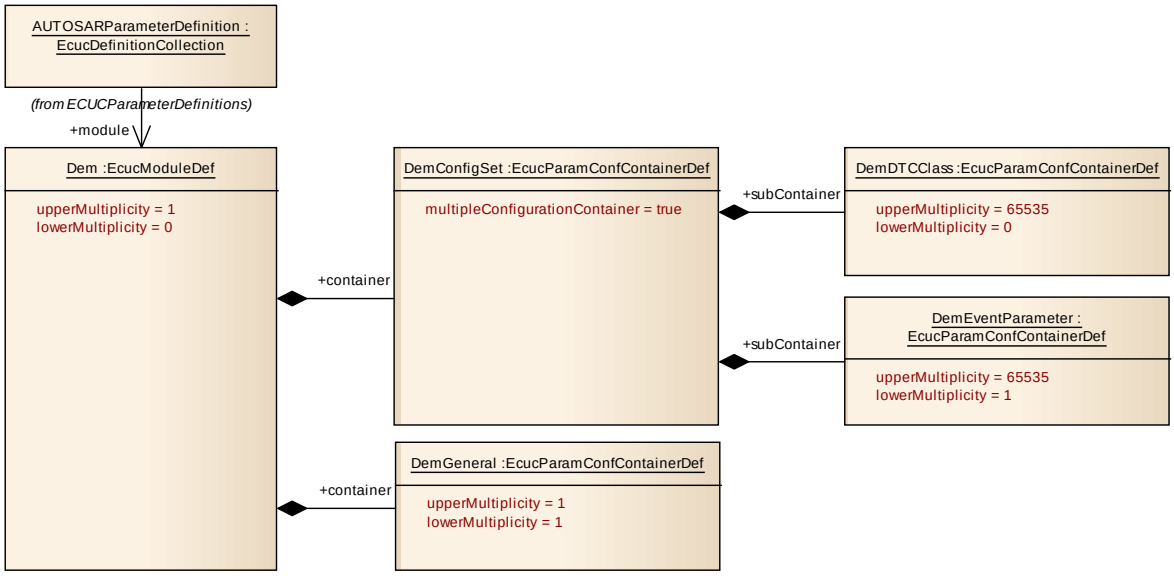

Figure 3 Top-level view of Dem configuration

7.1 Diagnostic event definition

A ´Diagnostic Event´ defines the atomic unit that can be handled by the Dem module. The status of a ´Diagnostic Event´ represents the result of a monitor (refer to chapter 7.1.6). The Dem receives the result of a monitor from SW-C via the RTE or other BSW modules.

The Dem module 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. stores the freeze frame.

[Dem153] ⌈The Dem module shall represent each Diagnostic Event by an EventId and the related EventName.⌋(BSW04061, BSW04063)

All monitors and BSW modules use the EventId as a symbolic EventName. The Dem configuration tool replaces the symbolic names by numbers.

[Dem154]「The EventId and the related EventName shall be unique per Dem module represented by the ECU configuration (refer to Dem126).」(BSW04063)

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

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

The Dem module supports several event-specific configuration parameters as shown in the following figures. For a detailed description, refer to chapter 10 Configuration specification.

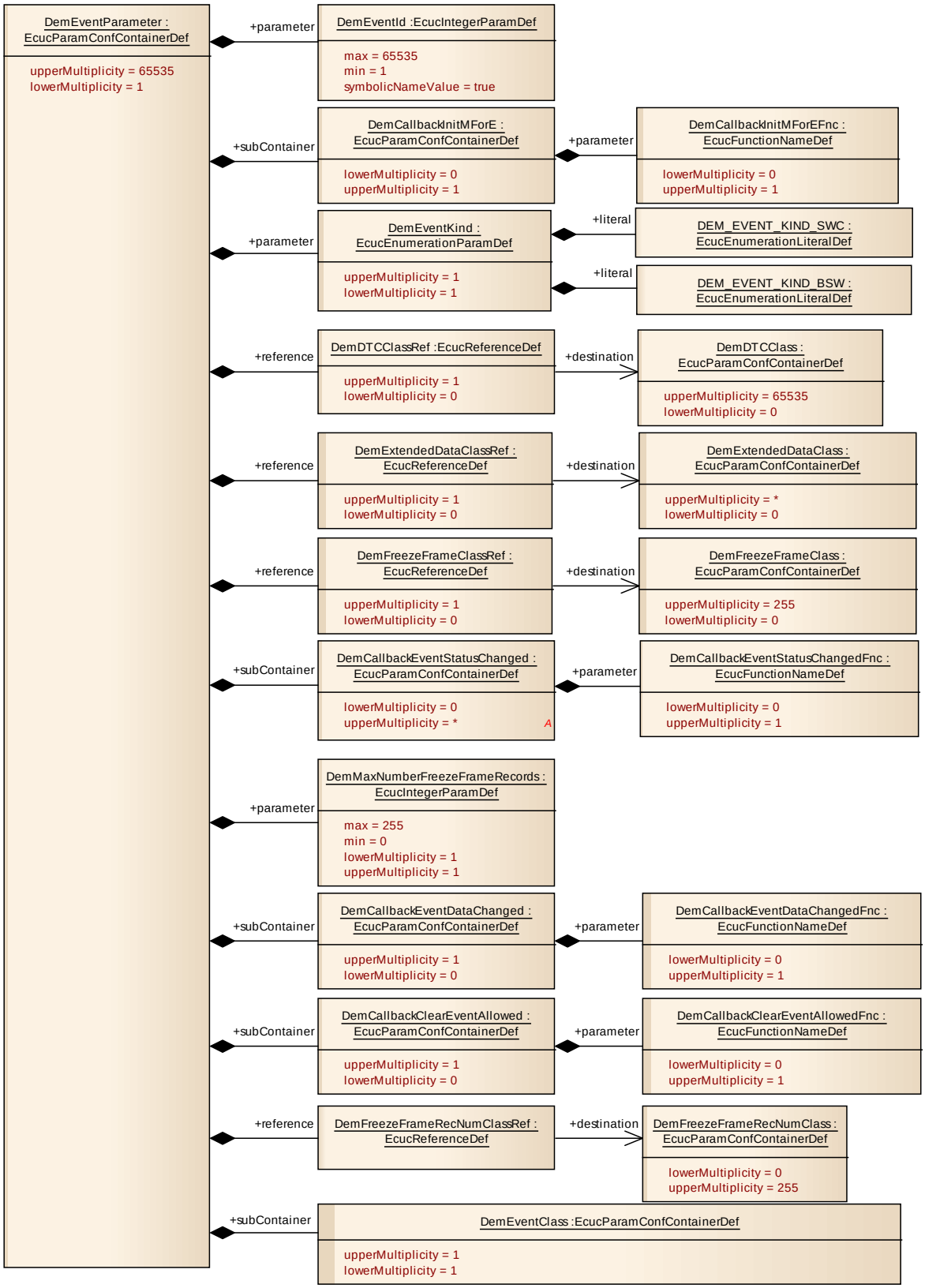

Figure 4 Event parameter configuration

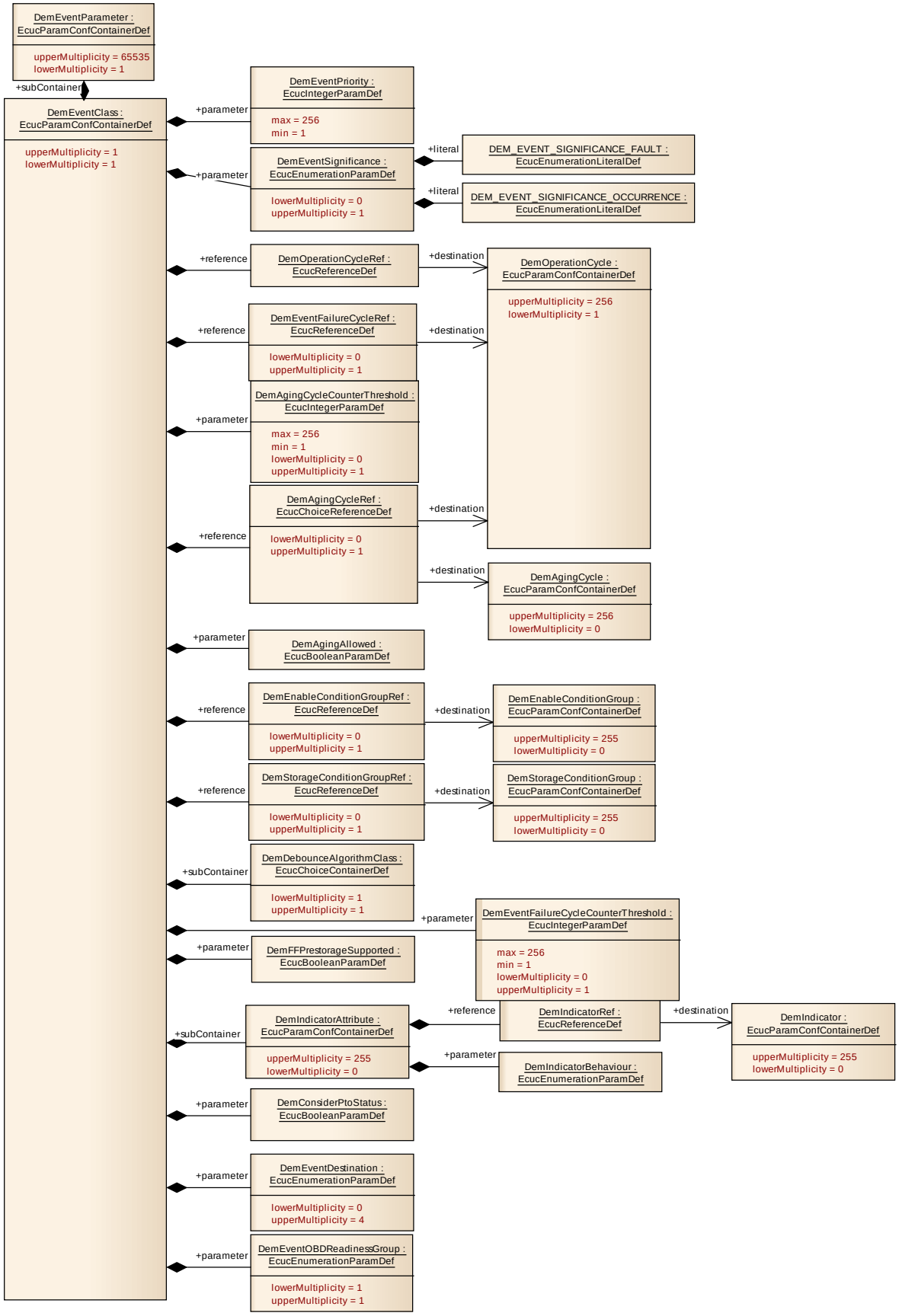

Figure 5 Event class configuration

7.1.1 Event priority

Event priority is defined as a ranking of events based upon level of importance. It is used to determine which fault entries may be removed from the event memory in the case of the number of stored events exceeds the maximum number of memory entries (event memory is full).

[Dem382] ⌈Each supported event shall have a priority assigned to it (refer to parameter DemEventPriority in DemEventClass).⌋()

[Dem383] ⌈A priority value of 1 shall be the highest priority. Larger priority value shall define lower importance.⌋()

7.1.2 Event occurrence

[Dem011] ⌈The Dem module shall provide an occurrence counter per event memory entry.⌋(BSW04067)

[Dem523] ⌈The Dem module shall initialize the occurrence counter with the value one if the related event is entered in the respective event memory (refer to Dem184).⌋()

[Dem524] ⌈The Dem module shall increment the occurrence counter by one if the related event is already stored in the event memory and the UDS DTC status bit 0 (TestFailed) changes from 0 to 1.⌋()

[Dem580] ⌈If the configuration parameter DemOccurrenceCounterProcessing (refer to DemGeneral) is DEM_PROCESS_OCCCTR_CDTC, the Dem module shall only increment the occurrence counter if the fault confirmation (refer to chapter 7.3.4) has been successfully finished.⌋()

[Dem625] ⌈The Dem module shall not increment the event-specific occurrence counter anymore, if it has reached its maximum value (e.g. 255, if data type is mapped to one byte, refer to Dem471).⌋()

7.1.3 Event kind

There are two different types of events:

- BSW-related events (reported via C-API – Dem_ReportErrorStatus)
- SW-C-related events (reported via RTE operation – SetEventStatus)

This kind is configurable per event (refer to DemEventKind in DemEventParameter).

This is necessary because BSW-events may be reported prior to full Dem initialization and need to be buffered (refer to chapter 7.6).

7.1.4 Event significance

There are two different significance levels of events:

- “fault”: classifies a failure, which relates to the component/ECU itself (and requires for example a repair action)
- “occurrence”: classifies an issue, which indicates additional information concerning insufficient system behavior (and relates for example to a condition out of the ECU’s control)

This significance level is configurable per event (refer to DemEventSignificance in DemEventClass) and can be mapped as a data element (refer to chapter 7.3.7.4, DEM_SIGNIFICANCE).

7.1.5 Event destination

The configuration parameter DemEventDestination (refer to DemEventClass) defines the dedicated storage location(s) of the event and its related data (refer to chapter 7.3.7).

The “permanent event memory” assignment is implicitly derived from the related DTC kind (refer to chapter 7.2). Emission-related events are automatically assigned to the permanent event memory, since the storage of an event as “permanent DTC” is dynamically derived from its current status (handling is described in chapter 7.7.1.4).
In this context the term "permanent" relates to an attribute of emission-related events and does not relate only to persistent storage via NvM, which is done for each event memory type anyway.

The definition and use of the different memory types is OEM specific.

For the Dcm-Dem interface the parameter DTCOrigin is used to distinguish between the different memory areas. The intention is to allow specific operations on the different memory areas (primary, secondary, permanent and mirror memory).

7.1.6 Diagostic monitor definition

A diagnostic monitor is a routine entity determining the proper functionality of a component. This monitoring function identifies a specific fault type (e.g. short to ground, open load, etc.) for a monitoring path.
A monitoring path represents the physical system or a circuit, that is being monitored (e.g. sensor input). Each monitoring path is associated to exactly one diagnostic event.

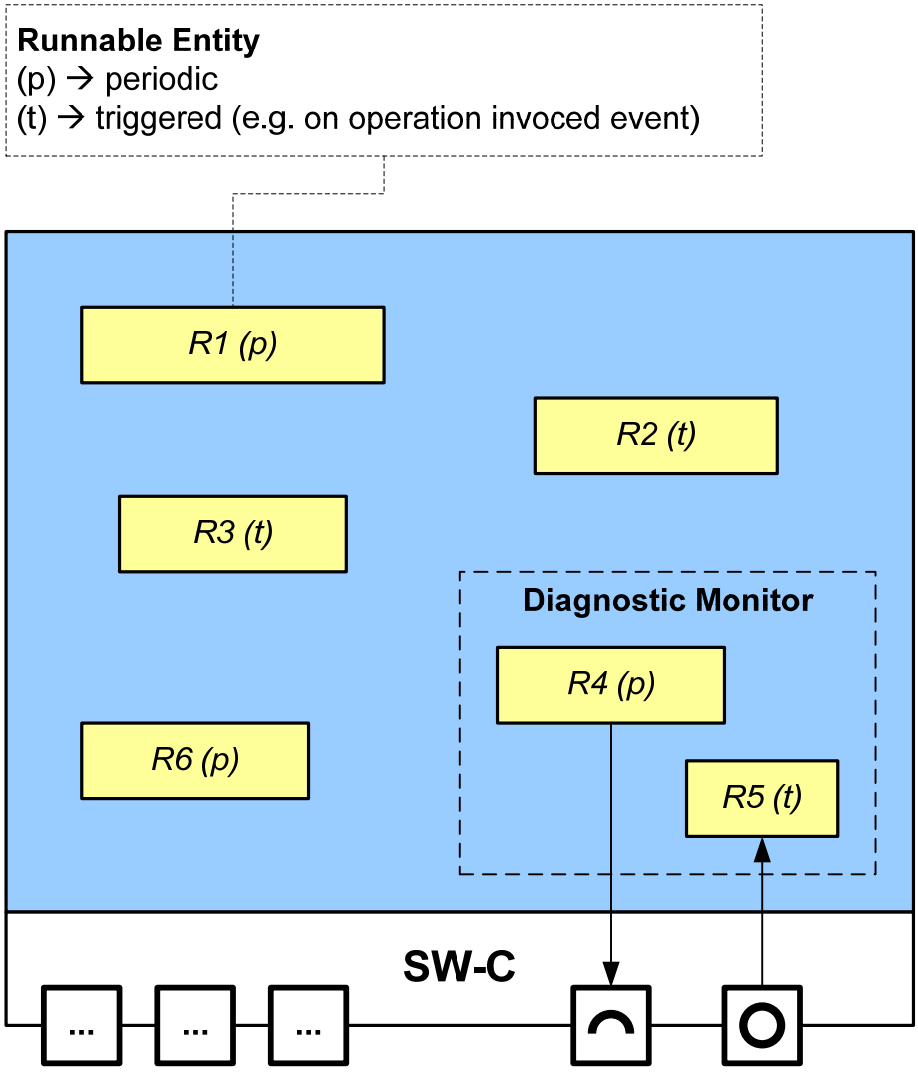

Figure 6 Example for a monitor embedded within a SW-C

If the monitor debounces on its own, the reporting API is called only after a qualified result (passed or failed) is available. A report is necessary at least if the result changes. However, usually it is computationally more efficient for the monitor, to always call the Dem and should therefore be preferred.
Hence, it is implementation specific for the Dem to deal with that reports with unchanged results.

If the monitor uses Dem-internal debouncing mechanism (refer to chapter 7.3.3), the reporting API is called whenever the code with the functional check is executed.

7.2 Diagnostic trouble code definition

There are two different kinds of DTCs:

- non OBD-relevant DTCs (UDS DTCs)
- OBD-relevant DTCs

This kind is derived implicitly from the DemDTCClass configuration. If the parameter DemObdDTC is existing, the DTC and all releated events are OBD-relevant.

[Dem013] ⌈The Dem module shall support DTC formats according to:

- ISO 14229-1
- ISO 15031-6
- SAE J1939-73
- ISO 11992-4⌋()

The configuration parameter DemTypeOfDTCSupported (refer to DemGeneral) is used to select one of the supported DTC formats defined in Dem013 of the ECU (refer to Dem_GetTranslationType). This is required to determine the DTC format value to be reported for ISO 14229-1 service Read DTC Information (0x19).

[Dem645] ⌈The Dem module shall support UDS and OBD DTC format for each DTC based on the configuration (refer to parameters DemUdsDTC and DemObdDC in DemDTCClass).⌋()

Note: A DTC can have either two formats (UDS and OBD), or one format only. The Dem will therefore handle two DTC value lists internally. Which format is reported depends on the Dem_DTCFormatType (refer to chapter 8.2.1.9) or is defined by the context of the related API.

[Dem277] ⌈The Dem shall report DTC values as a uint32 with byte 0 = LowByte, byte 1 = MiddleByte, byte 2 = HighByte and byte 3 is unused. For OBD DTC format there are only two bytes (HighByte, LowByte) used. The Dem services shall report these DTCs as a uint32 with byte 1 = LowByte, byte 2 = HighByte, byte 3 is unused and byte 0 = 0x00.⌋()

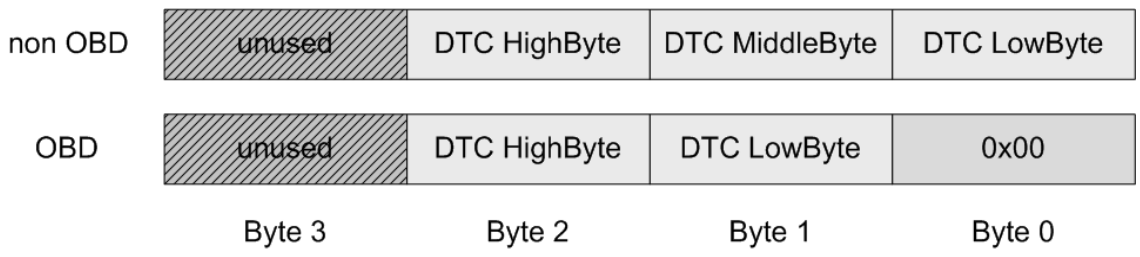

Figure 7 DTC Byte Order

[Dem269] ⌈The function Dem_GetDTCOfEvent (refer to chapter 8.3.3.13) shall get the DTC which is mapped to EventId by the Dem configuration.⌋()

[Dem033] ⌈Severity may be assigned to DTCs regarding the importance of the specific events according to ISO 14229-1, Annex D, DTCSeverityMask and DTCSeverity bit definitions (refer to 8.3.4.1.12 Dem_GetSeverityOfDTC).⌋()

Note:The severity is only available for ISO 14229-1 DTCs.

The severity is configurable per DTC optionally (refer to DemDTCSeverity).

Note: ISO 14229-1, Annex D defines the following severity levels: no severity available, Maintenance, Check at next Halt, Check immediately.

The function Dem_SetDTCFilter (refer to Dem057) allows filtering for DTCs with severity information.

[Dem593] ⌈The Dem module shall provide a functional unit value per DTC (refer to 8.3.4.1.13 Dem_GetFunctionalUnitOfDTC).⌋()

The functional unit value is configurable per DTC (refer to DemDTCFunctionalUnit DemDTCClass).

Note: The functional unit values are required by the Dcm.

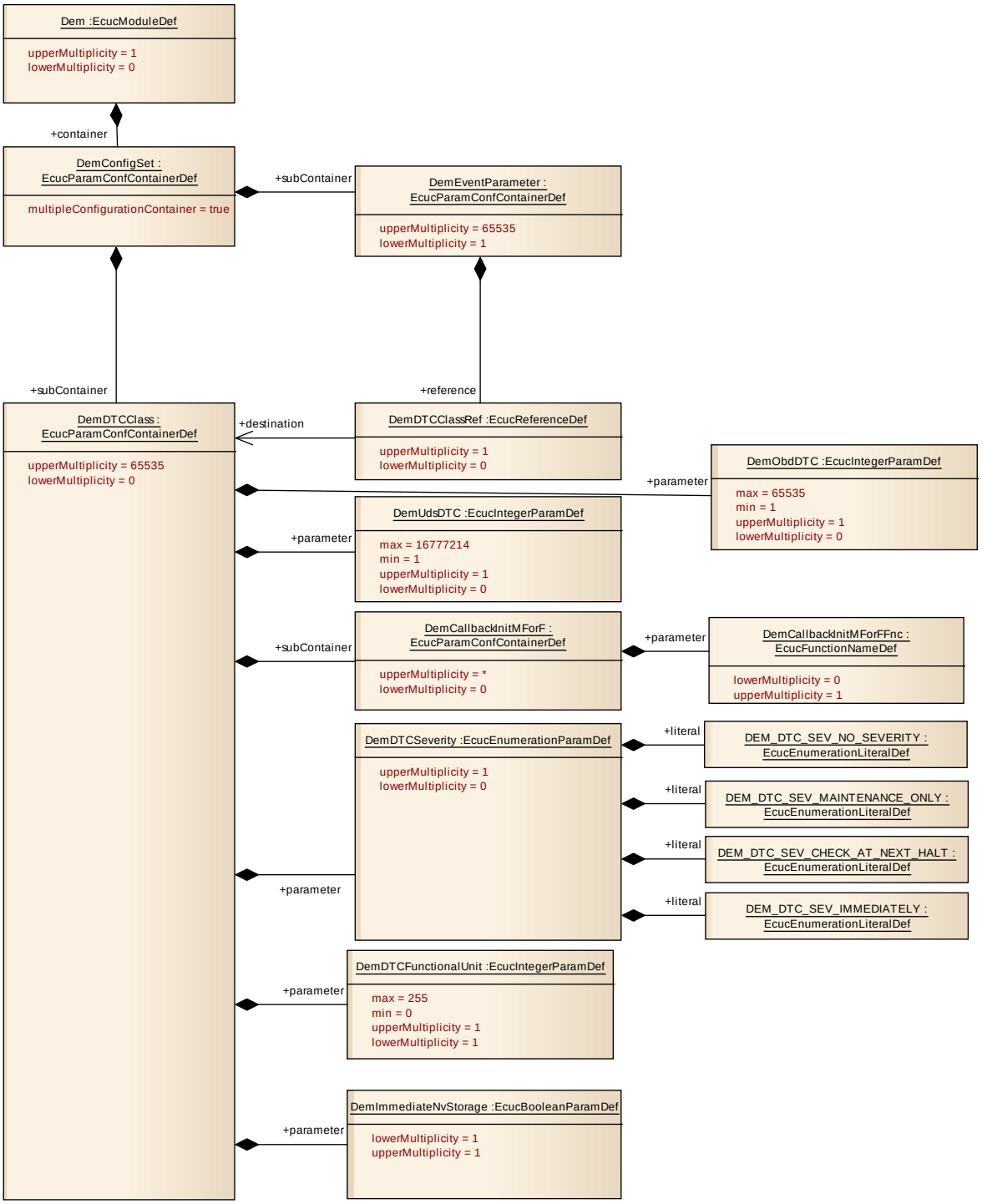

Figure 8 DTC configuration

7.2.1 DTC groups

In addition to single DTC values, groups of DTCs can be configured (refer to DemGroupOfDTC), as defined by ISO 14229-1 [15] – Annex D.1. Each DTC group

has its own DTC group value assigned (which must be unique to any other DTC and DTC group value).

Note: DTC groups are relevant for the diagnostic service ClearDiagnosticInformation (0x14, refer to 8.3.4.3.1 Dem_ClearDTC), as well as for the diagnostic service ControlDTCSetting (0x85, refer to 8.3.4.3.2 Dem_DisableDTCSetting and 8.3.4.3.3 Dem_EnableDTCSetting).

The following DTC groups are provided:

- emission-related DTC group (optional, fixed value = 0x000000)
- powertrain DTC group (optional, configurable value)
- chassis DTC group (optional, configurable value)
- body DTC group (optional, configurable value)
- network communication DTC group (optional, configurable value)
- further user-defined DTC groups (optional, configurable value)
- 'all DTCs' DTC group (mandatory, fixed value = 0xFFFFFF)

Note, that the DTC group 'all DTCs' will not be configured in DemGroupOfDTC, because it has always to be provided by the Dem module.

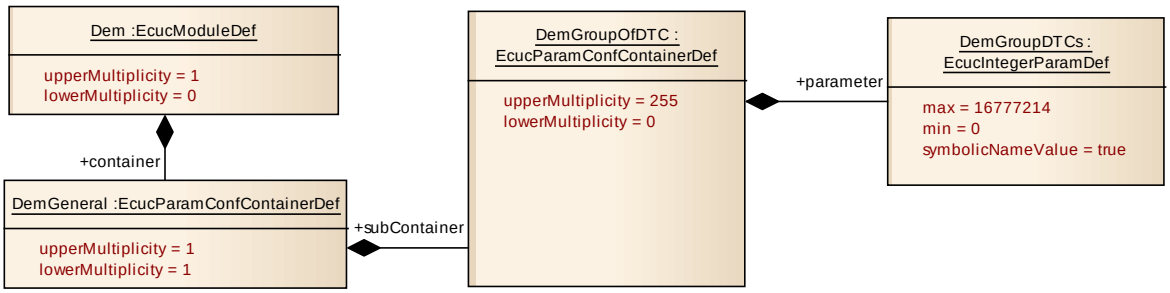

Figure 9 DTC group configuration

7.2.2 DTC suppression

[Dem586] 「The Dem module shall provide the capability to enable or disable the suppression of DTCs dynamically (refer to chapter 8.3.3.21 Dem_SetDTCSuppression), if the configuration parameter DemDTCSuppressionSupport (refer to DemGeneral) is enabled.」()

[Dem587] 「If the suppression for a specific DTC is enabled, this DTC shall be fully invisible.」()

Note: This means the (external) reporting (refer to chapter 7.8.2) of the disabled DTC is suppressed, but the event processing (refer to chapter 7.3) is not affected (e.g. for functional degradation).

[Dem588] ⌈The API Dem_SetDTCSuppression shall reject the request and return E_NOT_OK, if an event memory entry exists already for the requested DTC.⌋()

Note: If this API is used e.g. within a diagnostic service function, E_NOT_OK can be used to create a NRC.

7.3 Event memory description

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

[Dem010] ⌈The Dem module shall support the primary event memory.⌋(BSW04066)

[Dem548] ⌈If configured (refer to Dem162) the Dem module shall support the additional event memories (secondary, mirror, permanent).⌋()

The size of the different event memories is configurable in the Dem configuration.

[Dem162] ⌈The Dem module shall provide configuration parameters to adapt the fault memory size to the ECU memory space available (refer to configuration parameters DemMaxNumberEventEntry<…> in DemGeneral).⌋(BSW04070)

Note: If the size is configured to zero, the respective event memory is deactivated.

[Dem329] ⌈For storing to non-volatile memory the Dem shall use the NVRAM Manager (refer to chapter 7.8.4).⌋()

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

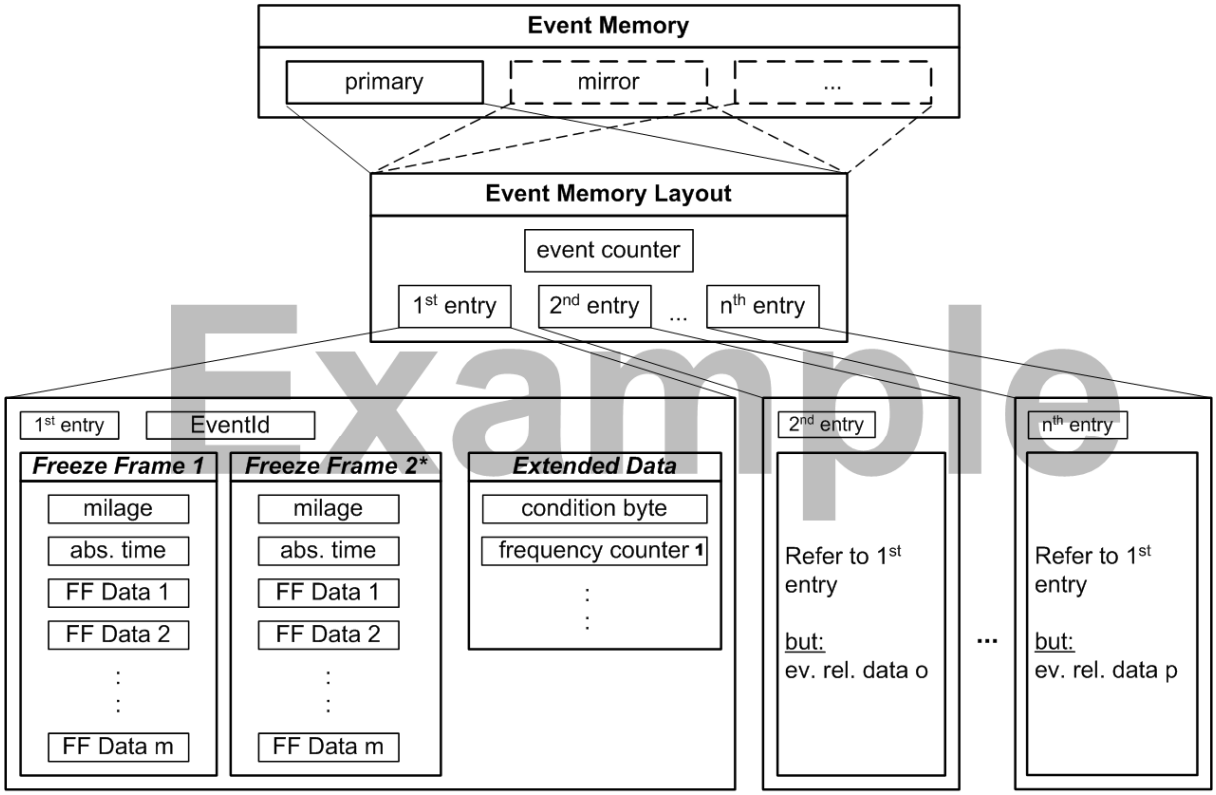

Figure 10 Example of a logical Dem event memory layout

If there are limitations of the memory size, it is necessary to provide overflow indication of the event memory and a displacement strategy (refer to chapter 7.3.2).

7.3.1 Event status management

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

[Dem330] 「The Dem module shall provide the capability to report the status of an event allowing a diagnostic monitor to inform the Dem about the result of the internal diagnostic test (refer to 8.3.3.2 Dem_SetEventStatus and Dem107).」()

The monitors, which are located in the application, should call the function Dem_SetEventStatus to report an event status as soon as a new test result is available. This will be done independently of the current state of the Dem module (refer to Figure 11).

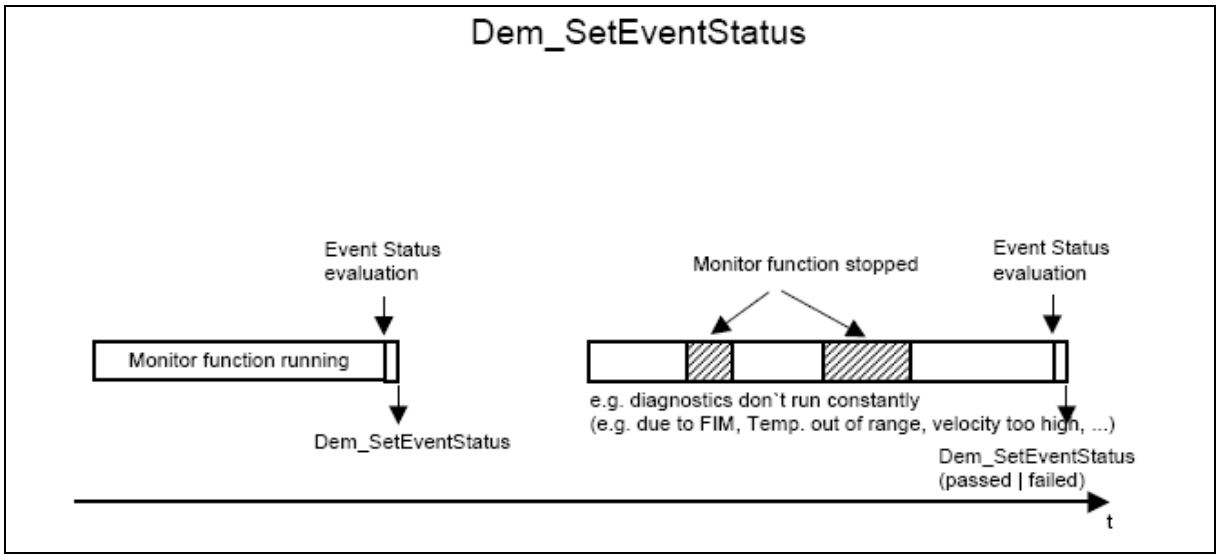

Figure 11 Example for using Dem_SetEventStatus

[Dem331] ⌈The Dem module shall provide the capability to reset the failed status of an event without reporting a passed result (refer to 8.3.3.3 Dem_ResetEventStatus).⌋ ()

Note: Dem_ResetEventStatus does not mean an event status report (like done by Dem_SetEventStatus). After the call, the event status is not qualified or tested.

Monitors will use the function Dem_ResetEventStatus in order to deactivate limp home and switch back to normal operation. At this point in time, the monitor has typically not been OK-tested and therefore Dem_SetEventStatus with tested and passed cannot be used.

[Dem187] ⌈The function Dem_ResetEventStatus shall set the UDS DTC status bit 0 (TestFailed) to 0 and set Dem-internal debounce counters to initial values if configured.⌋()

Note: The function Dem_ResetEventStatus does not change the UDS DTC status bit 6 (TestNotCompletedThisOperationCycle) and does not clear the pre-stored freeze frame.

[Dem638] ⌈The function Dem_ResetEventStatus shall return E_NOT_OK, if the event was already tested this operation cycle (UDS DTC status bit 6 – TestNotCompletedThisOperationCycle is set to 0).⌋()

[Dem051] ⌈The Dem module shall provide the capability to retrieve the current UDS DTC status byte of a specific event (refer to 8.3.3.10 Dem_GetEventStatus).⌋()

Note: The function Dem_GetEventStatus is provided to be used by SW-Cs or other BSW modules (e.g. FiM) on event-level. The Dcm uses the function Dem_GetStatusOfDTC on DTC-level instead.

[Dem333] ⌈The Dem module shall be able to provide the current event failed status (refer to 8.3.3.11 Dem_GetEventFailed) and the current event tested status (refer to 8.3.3.12 Dem_GetEventTested).⌋()

[Dem052] ⌈The function Dem_GetEventFailed shall report the UDS DTC status bit 0 (TestFailed) of the requested diagnostic event.⌋()

[Dem053] ⌈The function Dem_GetEventTested shall read the negated UDS DTC status bit 6 (TestNotCompletedThisOperationCycle) of the requested diagnostic event.⌋()

7.3.1.1 Status bit support

[Dem006] ⌈The Dem module shall implement one set of current UDS DTC status bits according to the definition in ISO14229-1 [15] for each diagnostic event.⌋()

Note: For this specification, the UDS DTC status byte of ISO14229-1 is represented by the Dem data type Dem_EventStatusExtendedType (defined in chapter 8.2.1.4).

7.3.1.2 Status bit update

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

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

[Dem379] ⌈Depending on the design decision (synchronous or asynchronous event processing) the status update of the following bits:
Bit 2 PendingDTC

Bit 3 ConfirmedDTC
Bit 7 WarningIndicatorRequested
may occur at a later point in time.」()

Note: If the status update of the bits described in Dem379 is implemented asynchronously, a queuing mechanism is needed which ensures that all the changes applied to bit 0, 1, 4, 5 and 6 will be considered when calculating bit 2, 3 and 7. Similarly, all processing and data storage associated with bit 2, 3 and 7 needs to be queued (refer to Figure 12 describing different approaches of synchronous and asynchronous event processing design).

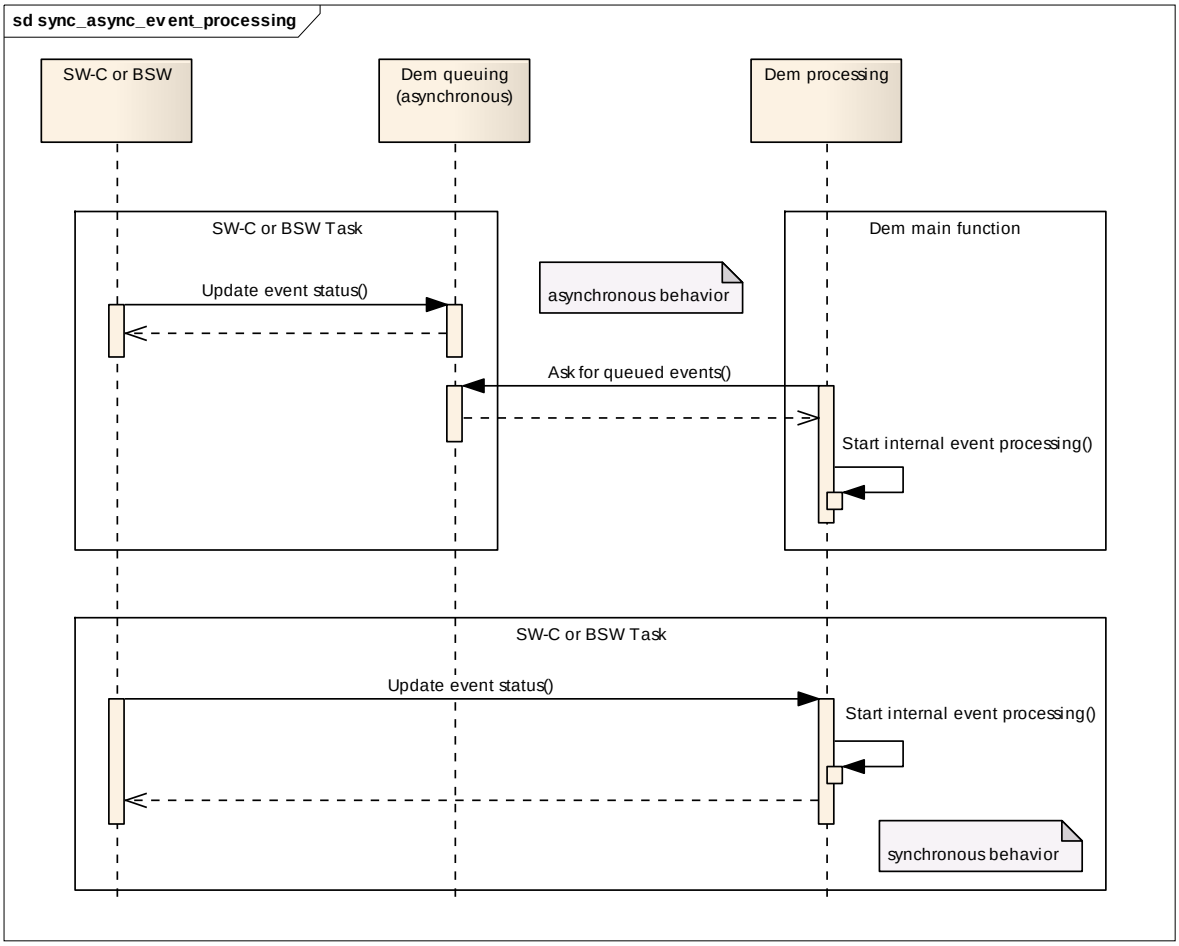

Figure 12 Synchronous and asynchronous event processing

Note: Refer to chapter 7.3.2 for the event memory management.

7.3.1.3 Status bit transitions

This section describes the behavior of the individual status bits of the UDS DTC status byte (refer to [15], UDS Status Byte – bit transitions).

[Dem385] ⌈After a clear command has been applied to a specific DTC (refer to Dem009) the Dem module shall set the UDS status byte to 0x50 (Readiness bits 4 and 6 set to 1, and all others are set to zero).⌋()

Note: The value of the UDS status byte after a clear command has been applied represents the delivery status (initial state) of this byte as well.

[Dem386] ⌈The Dem module shall implement the status bit transitions for UDS DTC status bit 0 (TestFailed) according to Figure 13.⌋()

[Dem387] ⌈ The Dem module shall support the configuration parameter DemStatusBitStorageTestFailed (refer to DemGeneral) used to determine whether the TestFailed status is stored volatile or non-volatile.⌋()

[Dem388] ⌈If the configuration parameter DemStatusBitStorageTestFailed is set to False, the Dem module shall not retain the TestFailed status over power cycles (volatile).⌋(BSW04076)

[Dem525] ⌈If the configuration parameter DemStatusBitStorageTestFailed is set to True, the Dem module shall retain the TestFailed status over power cycles (non-volatile).⌋()

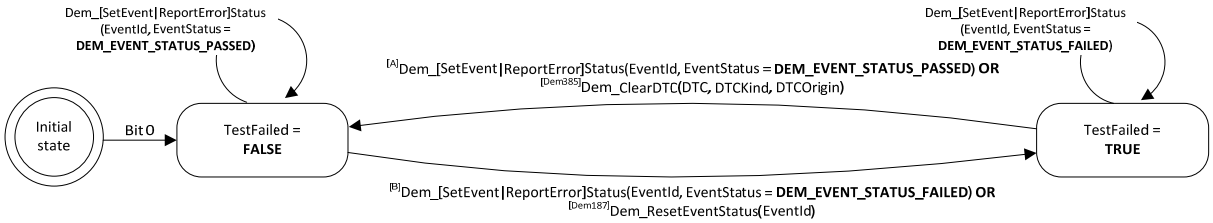

Figure 13 DTC status bit 0 TestFailed logic

[Dem389] ⌈The Dem module shall implement the status bit transition for UDS DTC status bit 1 (TestFailedThisOperationCycle) according to Figure 14.⌋()

Note: The information for UDS DTC status bit 1 (TestFailedThisOperationCycle) needs to be stored non-volatile, if the PendingDTC bit is used (refer to DemDtcStatusAvailabilityMask) or if the Dem module supports operation cycles over power cycles (refer to DemOperationCycleStatusStorage).

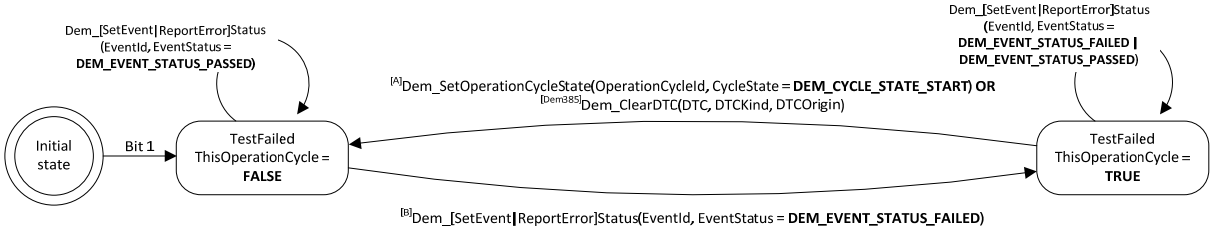

Figure 14 DTC status bit 1 TestFailedThisOperationCycle logic

[Dem390] ⌈The Dem module shall implement the status bit transition for UDS DTC status bit 2 (PendingDTC) according to Figure 15.⌋()

Note: The information for UDS DTC status bit 2 (PendingDTC) needs to be stored non-volatile, if the Dem module supports operation cycles over power cycles (refer to DemOperationCycleStatusStorage).

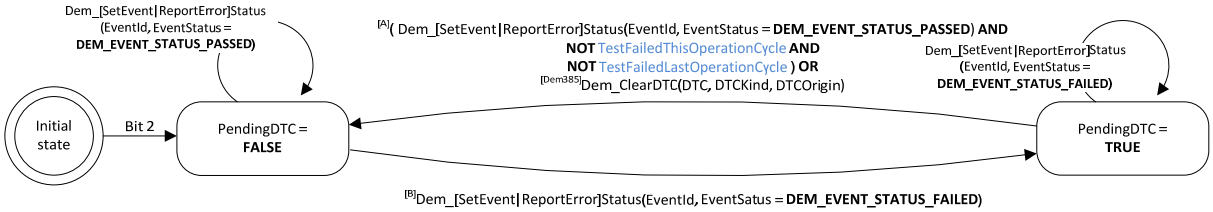

Figure 15 DTC status bit 2 PendingDTC logic

Note: The ISO 14229-1 contains a vague description of the UDS DTC status bit 2 (PendingDTC) transition to 0. Based on the vendor specific interpretation there are different implementations possible.

[Dem391] ⌈The Dem module shall implement the status bit transition for UDS DTC status bit 3 (ConfirmedDTC) according to Figure 16.⌋()

Note: The information for UDS DTC status bit 3 (ConfirmedDTC) needs to be stored non-volatile (but it is also calculable based on the respective event memory entry).

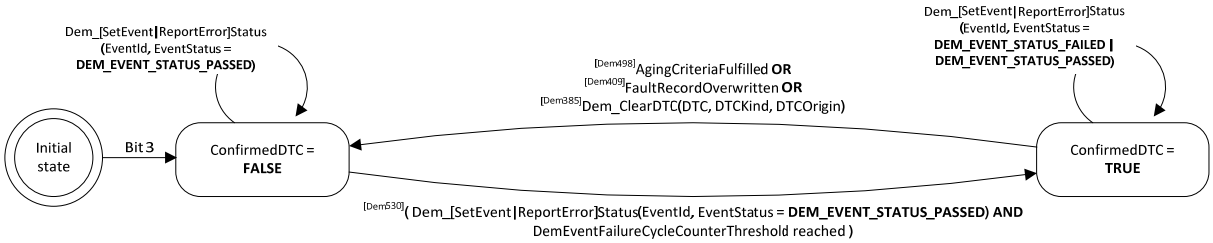

Figure 16 DTC status bit 3 ConfirmedDTC logic

Note: The “AgingCriteriaFulfilled” condition is specified by Dem498. The “FaultRecordOverwritten” condition is specified by Dem409.

[Dem392] ⌈The Dem module shall implement the status bit transition for UDS DTC status bit 4 (TestNotCompletedSinceLastClear) according to Figure 17.⌋()

Note: The information for UDS DTC status bit 4 (TestNotCompletedSinceLastClear) needs to be stored non-volatile.

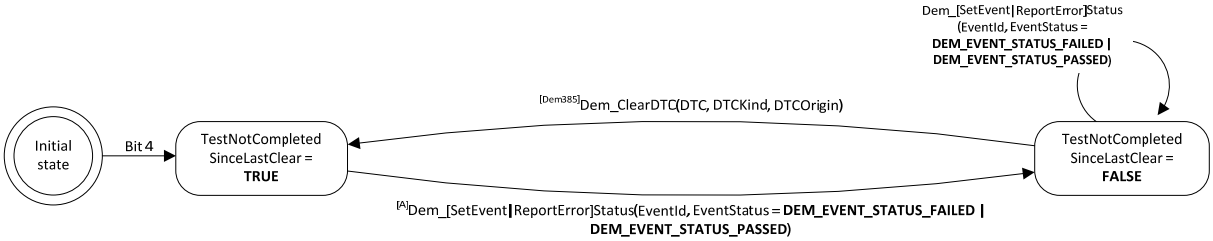

Figure 17 DTC status bit 4 TestNotCompletedSinceLastClear logic

[Dem393] ⌈The Dem module shall implement the status bit transition for UDS DTC status bit 5 (TestFailedSinceLastClear) according to Figure 18.⌋()

Note: The information for UDS DTC status bit 5 (TestFailedSinceLastClear) needs to be stored non-volatile.

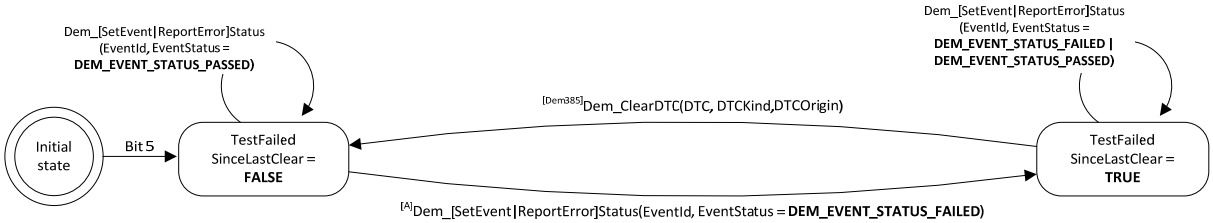

Figure 18 DTC status bit 5 TestFailedSinceLastClear logic

[Dem394] ⌈The Dem module shall implement the status bit transition for UDS DTC status bit 6 (TestNotCompleteThisOperationCycle) according to Figure 19.⌋()

Note: The information for UDS DTC status bit 6 (TestNotCompleteThisOperationCycle) needs to be stored non-volatile, if the PendingDTC bit is used (refer to DemDtcStatusAvailabilityMask), or if the Dem module supports operation cycles over power cycles (refer to DemOperationCycleStatusStorage).

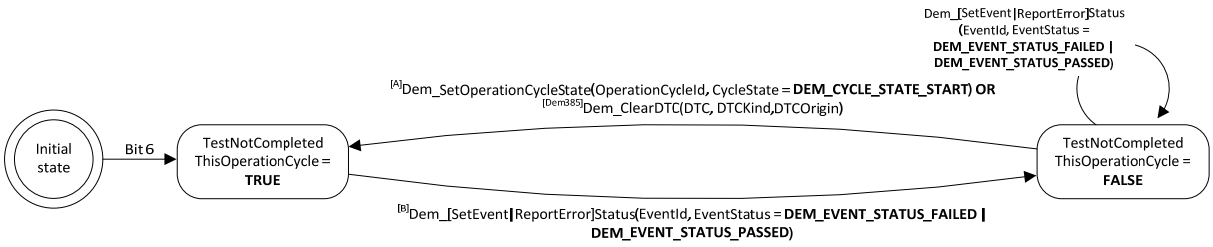

Figure 19 DTC status bit 6 TestNotCompleteThisOperationCycle logic

[Dem395] ⌈The Dem module shall implement the status bit transition for UDS DTC status bit 7 (WarningIndicatorRequested) according to Figure 20.⌋()

Note: The information for UDS DTC status bit 7 (WarningIndicatorRequested) may be stored volatile (because it is calculated based on the assigned warning indicator states).

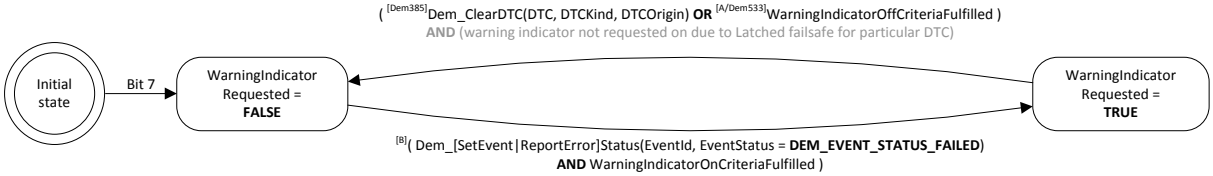

Figure 20 DTC status bit 7 WarningIndicatorRequested logic

Note: The "WarningIndicatorOffCriteriaFulfilled" and the "WarningIndicatorOnCriteriaFulfilled" condition are specified in chapter 7.3.10.

Note: ISO 14229-1 additionally specifies "warning indicator not requested on due to latched failsafe for particular DTC" as condition. This has to be ensured by the monitor.

7.3.1.4 Active/Passive status

If an event gets qualified as failed, it becomes active. If the event gets qualified as passed, it becomes passive. This status can be derived from the UDS DTC status byte.

As the TestFailed bit (UDS DTC status bit 0) is configurable in persistent storage ability (refer to configuration parameter DemStatusBitStorageTestFailed), also the meaning of active/passive is influenced:

- If the TestFailed bit is stored non-volatile, "event active" equals to TestFailed = 1 and "event passive" equals to TestFailed = 0.
- If the TestFailed bit is only stored volatile, additionally the information, if the event was already tested/reported this power cycle, is required. As long, as this information is not present, the active/passive status is undefined.

Note: For events, which are assigned to the operation cycle type DEM_OPCYC_POWER, the inverted TestNotCompletedThisOperationCycle bit represents the information about "already tested/reported this power cycle".

Note: There are also ECUs, where all monitors run during startup phase. In this case, it is also sufficient, to map the active/passive status directly to the TestFailed bit.

7.3.1.5 Notification of status bit changes

The Dem module can inform the monitor and/or other components about the status change of the event or DTC.

Note: For asynchronous event processing (refer to Dem379), the Dem module will trigger the respective notification functions twice per event/DTC qualification.

[Dem016] ⌈The Dem module shall trigger the event-specific callback-function EvenStatusChanged (refer to chapter 8.4.3.1.3) on each event status change.⌋()

Note: The Dem module does not evaluate the return value (e.g. if other than E_OK) of this callback function.

Note: The configuration container DemCallbackEventStatusChanged (refer to chapter) is used to specify one or more ports/c-callbacks per event.

Note: The respective callback-functions for FiM and Dlt are specified in chapter 7.8.

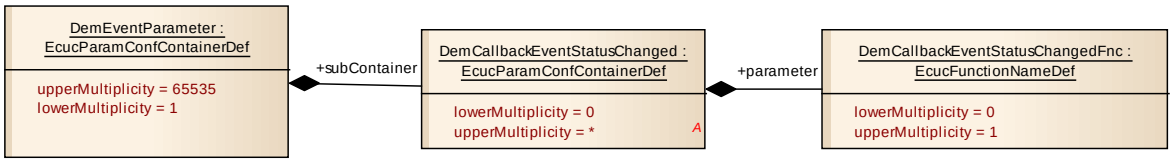

Figure 21 EventStatusChanged callback configuration

[Dem284] ⌈The Dem module shall trigger the callback-function DTCStatusChanged on every status change of an event, which has a DTC number assigned.⌋()

Note: The Dem module does not evaluate the return value (e.g. if other than E_OK) of this callback function.

Note: The result of EventStatusChanged may only be different to the result of DTCStatusChanged in case of event combination (refer to chapter 7.3.5).

Note: The configuration container DemCallbackDTCStatusChanged is used to specify one or more ports/c-callbacks for the Dem module globally.

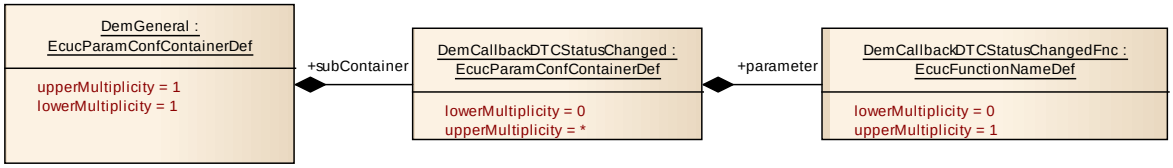

Figure 22 DTCStatusChanged callback configuration

7.3.2 Event memory management

The event memory management is defined as the process of adding, updating and removing event records in and out of the Dem module. The Dem module determines if the event is new or currently exists in the event memory.

Note: The requirements in this section do not determine the software implementation but describe the behavior of the ECU towards requests from the test tool.

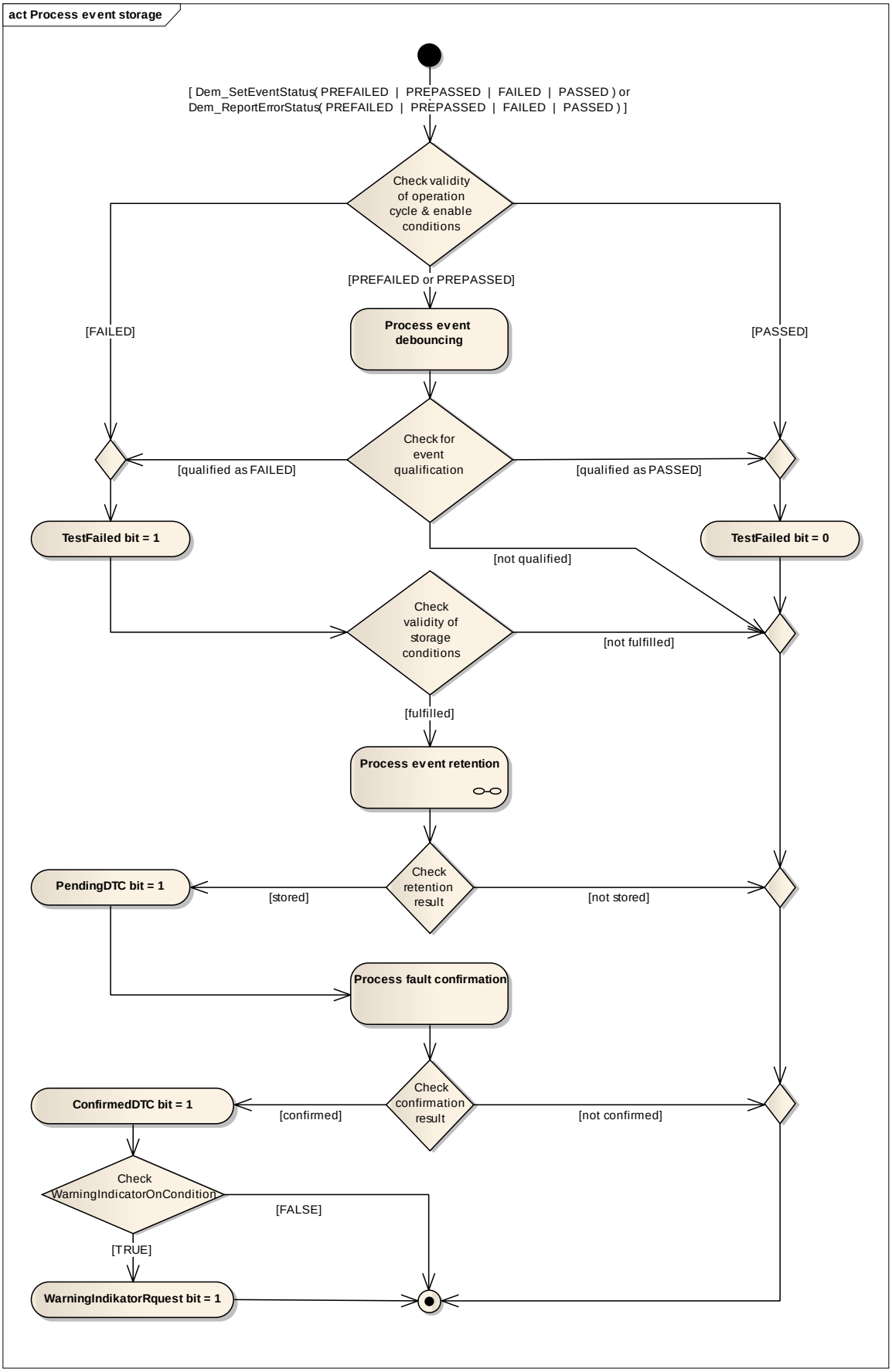

Figure 23 General diagnostic event storage processing

7.3.2.1 Event retention

Event retention is defined as the ability of the Dem module to record and handle events (DTCs), DTC status information and event related data (e.g. freeze frames, extended data).

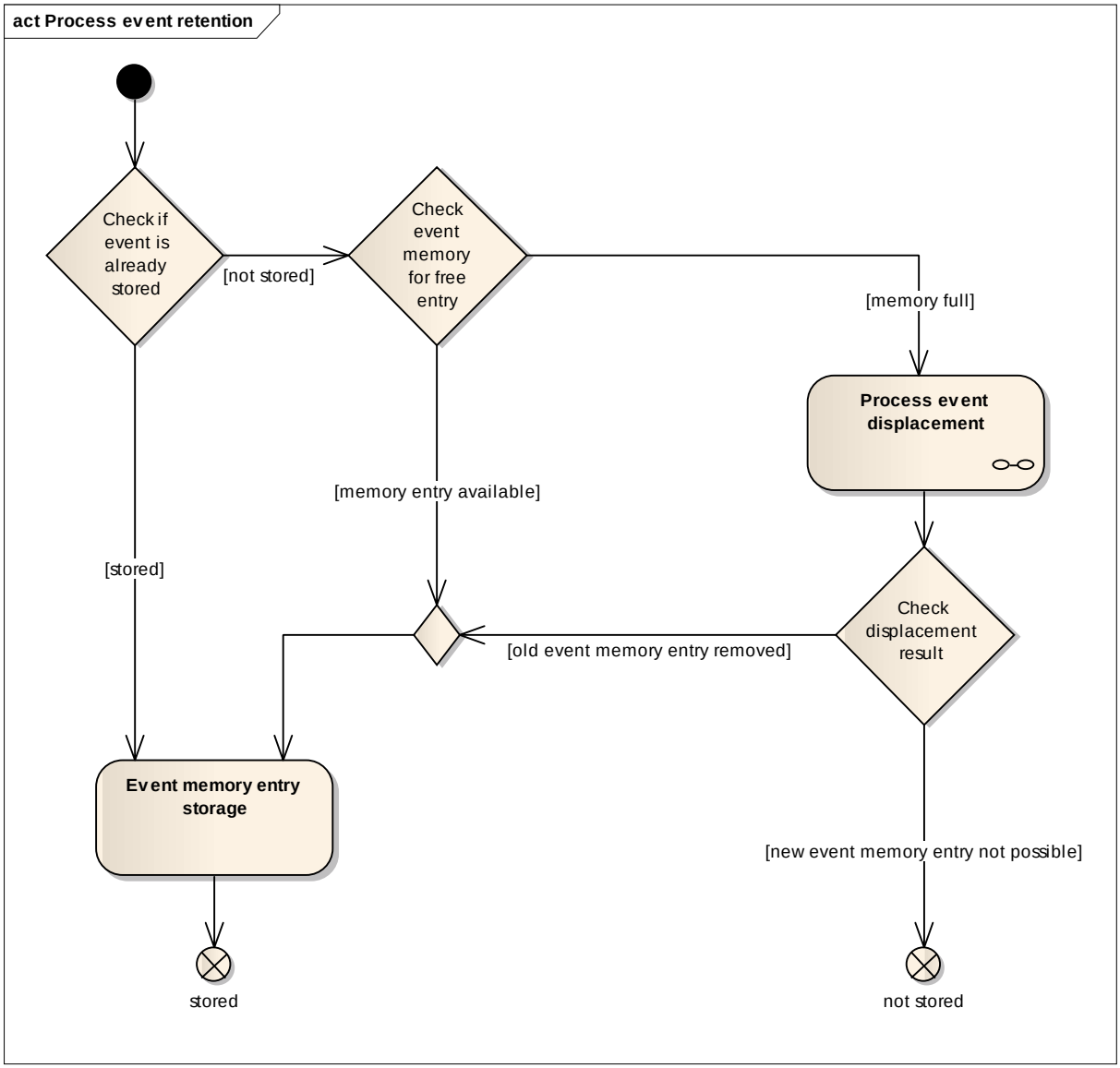

Figure 24 General diagnostic event retention processing

[Dem184] ⌈The Dem module shall store an event which is qualified as failed (UDS DTC status bit 0 changes from 0 to 1) in its configured event memory (refer to DemEventDestination, refer to Dem190).⌋()

[Dem396] ⌈If an event gets qualified as failed (UDS DTC status bit 0 changes from 0 to 1) and a respective event memory entry exists, the Dem module shall update this event memory entry.⌋(BSW04070)

Note: The storage of a new event memory entry or the update of an already existing event memory entry further includes the handling / update of event related data (refer to chapter 7.3.7).

7.3.2.2 Event memory overflow indication

[Dem397] ⌈The Dem module shall indicate for each event memory if the event memory is full and there was an attempt to store an additional event in this event memory.⌋()

This overflow indication can be used to trigger further internal behavior of the Dem module (e.g. displacement strategy). Furthermore, it can be used for additional fault analysis in workshops.

[Dem398] ⌈The Dem module shall provide the API Dem_GetEventMemoryOverflow (refer to chapter 8.3.3.20) to provide access to the event memory overflow indication status of the respective event memory.⌋()

Note: This API can be used for vendor-specific Diagnostic Services or other vendor-specific handling, and is provided to SW-Cs, as well as complex device drivers.

[Dem399] ⌈The event memory overflow indication of the respective event memory shall be reset, if all DTCs of this memory are deleted by Dem_ClearDTC.⌋()

Note: In case of aging and deleting single DTCs, the overflow indication of the event memory is not reset to keep this status information until the ECU receives an explicit clear command of this specific event memory. The ECU itself should not change this status information during normal operation (because it is used in workshops).

7.3.2.3 Event displacement

Event displacement means, that the most insignificant, already existing event memory entry is replaced by a new event, which needs to be stored. During displacement, the most insignificant entry gets lost.

[Dem400]「If an event gets qualified as failed (UDS DTC status bit 0 changes from 0 to 1) and the event memory is full, the Dem module shall check for entered events to be displaced by the new event.」()

Note: If the event memory size is configured to cover all possible events, no displacement will occur.

[Dem401]「The Dem module shall provide the configuration parameter DemEventDisplacementSupport (refer to DemGeneral) defining whether the existing event entry can be displaced or not.」()

[Dem402]「If event displacement is disabled (DemEventDisplacementSupport selects FALSE), the Dem module shall not displace existing event memory entries if the event memory is full.」()

[Dem406]「If event displacement is enabled (DemEventDisplacementSupport selects TRUE), the Dem module shall perform the following sequence by combination of the different displacement criteria (refer to Figure 25):

1. Priority
2. Status (active/passive)
3. Occurrence」()

[Dem403]「If displacement is enabled, the Dem module shall displace lower prioritized events by higher prioritized events.」()

[Dem404]「If displacement is enabled, the Dem module shall displace passive events preferably (refer to chapter 7.3.1.4).」()

[Dem405]「If displacement is enabled, the Dem module shall displace older events by newer events preferably.」()

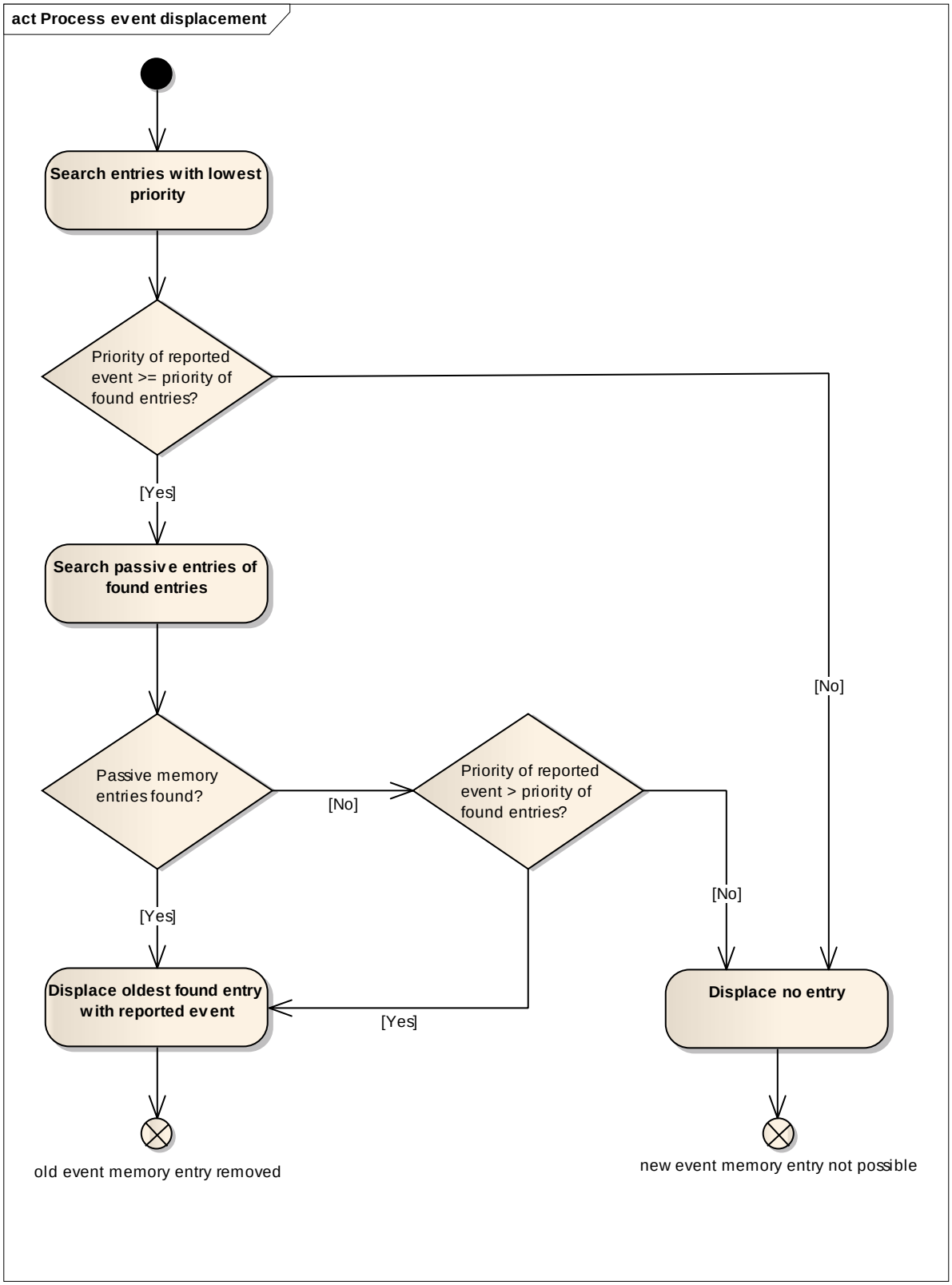

Figure 25 Combined displacement criteria processing

[Dem407] 「If no event memory entry for displacement was identified, the Dem module shall discard the reported event.」()

[Dem408] ⌈If the event memory entry for displacement was identified, the Dem module shall remove this event including its event related data from the event memory and shall store the reported event in the event memory.⌋()

[Dem409] ⌈If an event memory entry was removed during displacement, the Dem module shall not modify the UDS status bits of the removed event, except the UDS status bit 3 (ConfirmedDTC) which is set to 0.⌋()

7.3.2.4 Reporting order of event memory entries

[Dem410] ⌈The Dem module shall report DTCs in the chronological order of the event storage (refer to API Dem_GetNextFilteredDTC), if the DTC status mask parameter is set to “pending DTC” or “confirmed DTC” bit, or both bits (no other status bits are allowed to be set).⌋(BSW04070)

Note: The chronological order is for reporting purposes only and does not imply explicit memory structure, which is implementation specific.

Note: The reporting order may vary with the customer specific attributes used by the algorithm for sorting the DTC records in case of not confirmed or pending. If the chronological order is required for further DTC status mask parameters, additional resources may be necessary.

[Dem411] ⌈If the Dem module is requested to report in chronological order, the most recent event memory entry shall be reported at first.⌋()

[Dem412] ⌈The Dem module shall derive the chronological reporting order from the point in time when the event memory entry is stored.⌋()

Note: If there are more than one event memory entries stored at the same point in time the chronological reporting order of these elements is undefined. It depends on the implementation which of these elements is reported at first or will be displaced.

7.3.3 Debouncing of diagnostic events

In general, an ECU may implement several types of debouncing improving signal quality. This section describes methods, how the Dem module shall implement basic algorithms used for fault maturation (diagnostic event debouncing).

If the Dem module is configured to implement the debounce algorithm for a specific event, one of the following debounce algorithms are to be performed Dem-internally.

Otherwise, the respective monitor implemented in a SW-C or BSW will report the status of a certain event, after diagnostic event debouncing has been completed. For interaction between the Dem and SW-C please refer to the API InitMonitorForEvent (refer to chapter 8.4.3.1.1).

If there are any requirements to get the passed or failed status within a certain amount of time, the diagnostic monitor is responsible.

[Dem413] ⌈ The Dem module shall support the event-specific configuration of debounce algorithms by using the configuration container DemDebounceAlgorithmClass. ⌋()

Note: That means for each individual event a specific algorithm can be specified.

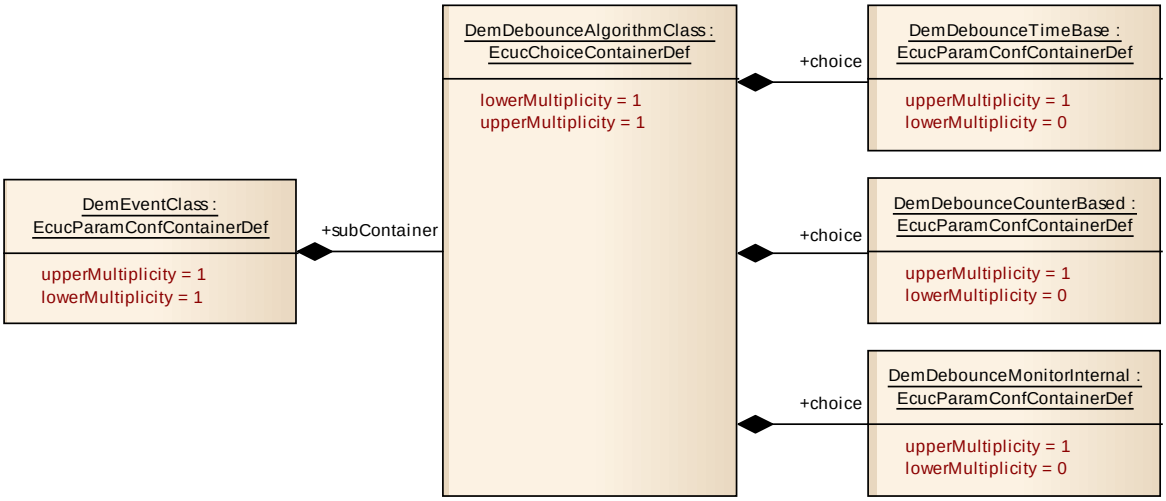

Figure 26 Event-specific debounce algorithms

7.3.3.1 Counter based debounce algorithm

[Dem526] ⌈ The Dem module shall provide a configuration parameter DemDebounceCounterBasedSupport (refer to DemGeneral) to enable or disable the Dem-internal counter based debouncing. ⌋()

[Dem414] ⌈ If the configuration container DemDebounceAlgorithmClass is set to DemDebounceCounterBased, the Dem module shall provide an internal debounce counter for each individual event, to qualify the reported event. ⌋()

For huge debounce ranges, the maximal range of the internal debounce counter is defined as sint16. This does not imply any explicit implementation.

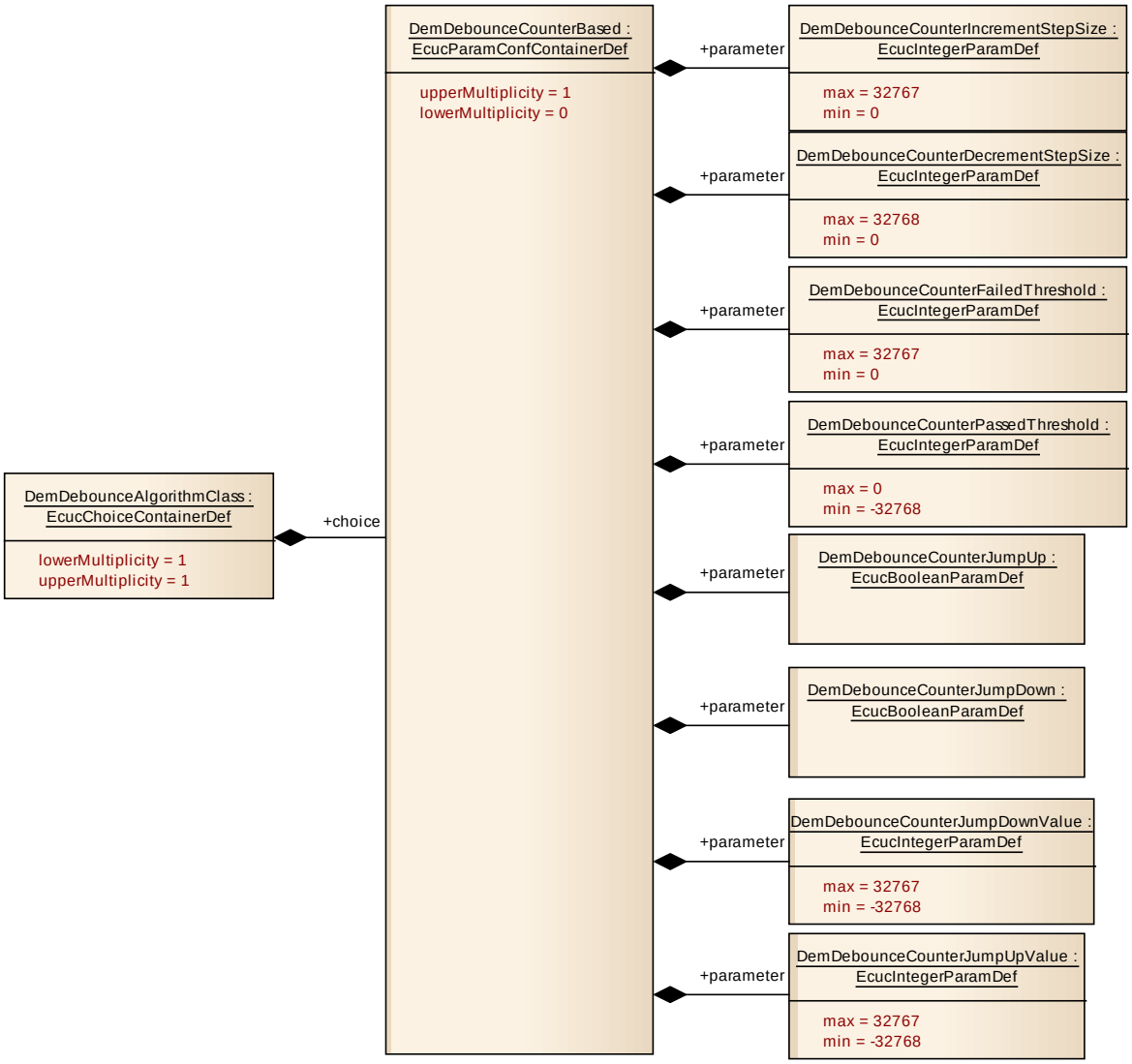

Figure 27 Counter based debounce algorithm

[Dem415] ⌈The Dem module shall calculate the fault detection counter (-128 …+127 according to UDS) based on the value and range of the internal debounce counter to map the internal counter values linearly to the external values.⌋()

[Dem416] ⌈ The Dem module shall provide the configuration parameter DemDebounceCounterFailedThreshold used to define the event-specific limit indicating the failed status (active).⌋()

[Dem417] ⌈ The Dem module shall provide the configuration parameter DemDebounceCounterPassedThreshold used to define the event-specific limit indicating the passed status (passive).⌋()

[Dem418] ⌈The Dem module shall increment the internal debounce counter with its configured step-size (refer to DemDebounceCounterIncrementStepSize), when the monitor reports DEM_EVENT_STATUS_PREFAILED (refer to EventStatus).⌋()

[Dem419] ⌈The Dem module shall decrement the internal debounce counter with its configured step-size (refer to DemDebounceCounterDecrementStepSize), when the monitor reports DEM_EVENT_STATUS_PREPASSED (refer to EventStatus).⌋()

[Dem420] ⌈If the monitor reports DEM_EVENT_STATUS_FAILED, the Dem module shall set the internal debounce counter value to its configured threshold being the failed criteria.⌋()

[Dem421] ⌈ If the monitor reports DEM_EVENT_STATUS_PASSED, the Dem module shall set the internal debounce counter value to its configured threshold being the passed criteria.⌋()

[Dem422] ⌈ The Dem module shall provide the configuration parameter DemDebounceCounterJumpDown for activating or deactivating the jump down behavior.⌋()

[Dem423] ⌈If the jump down behavior is active, the Dem module shall provide the configuration parameter DemDebounceCounterJumpDownValue defining the new internal debounce counter init value if the counting direction changes from incrementing to decrementing.⌋()

[Dem424] ⌈ The Dem module shall provide the configuration parameter DemDebounceCounterJumpUp for activating or deactivating the jump up behavior.⌋()

[Dem425] ⌈If the jump up behavior is active, the Dem module shall provide the configuration parameter DemDebounceCounterJumpUpValue defining the new internal debounce counter init value if the counting direction changes from decrementing to incrementing.⌋()

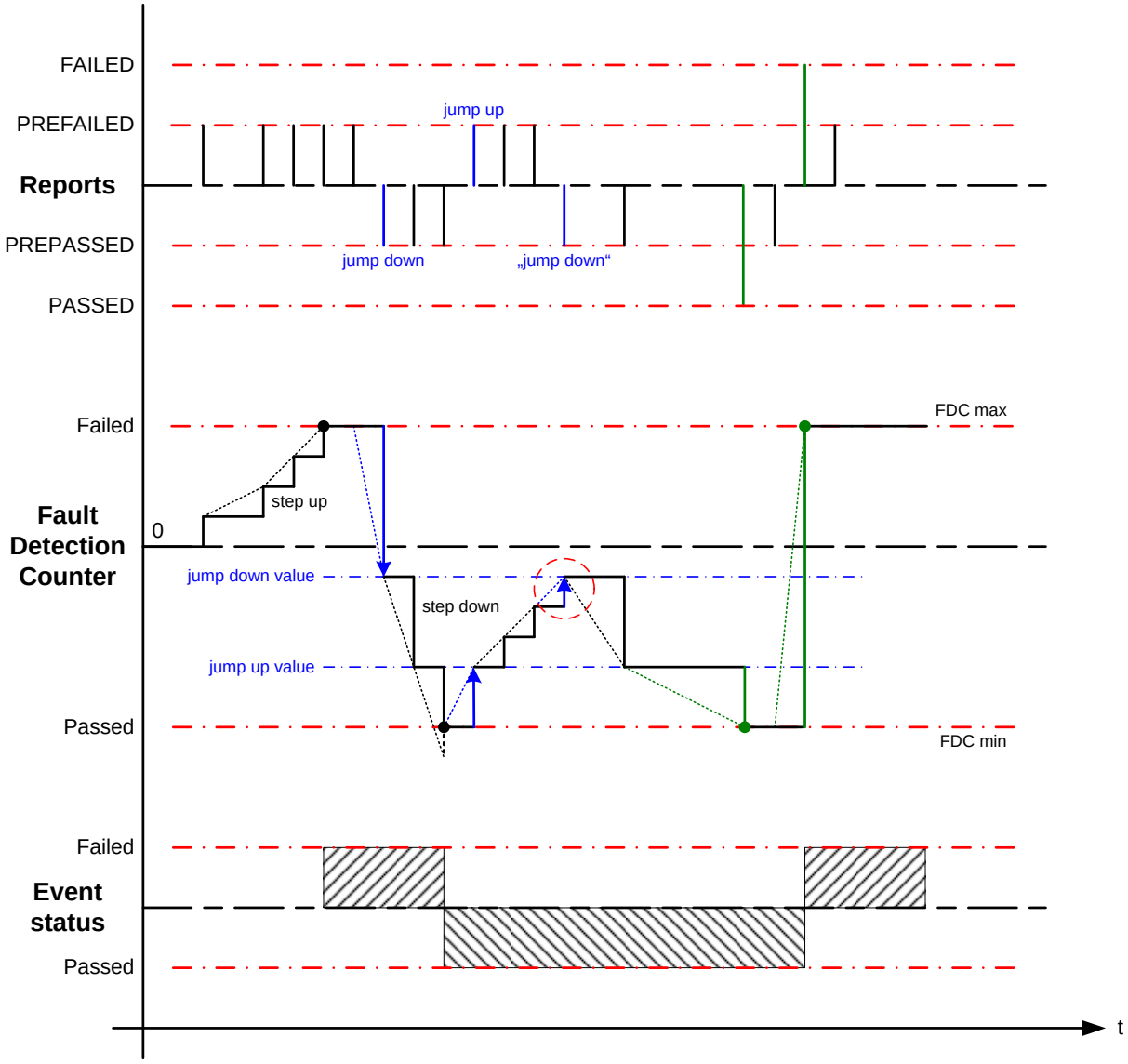

Figure 28 Example of counter based debouncing (including jump behaviour)

Note: In Figure 28, the value of Fault Detection Counter maps linearly to the internal debounce counter value.

7.3.3.2 Time based debounce algorithm

[Dem527] ⌈ The Dem module shall provide a configuration parameter DemDebounceTimeBasedSupport (refer to DemGeneral) to enable or disable the Dem-internal time based debouncing. ⌋()

[Dem426] ⌈ If the configuration container DemDebounceAlgorithmClass is set to DemDebounceTimeBased, the Dem module shall provide an internal debounce timer for each individual event, to qualify the reported event. ⌋(BSW04059)

For long debounce periods, the maximal range of the internal debounce timer is defined as sint16. This does not imply any explicit implementation.

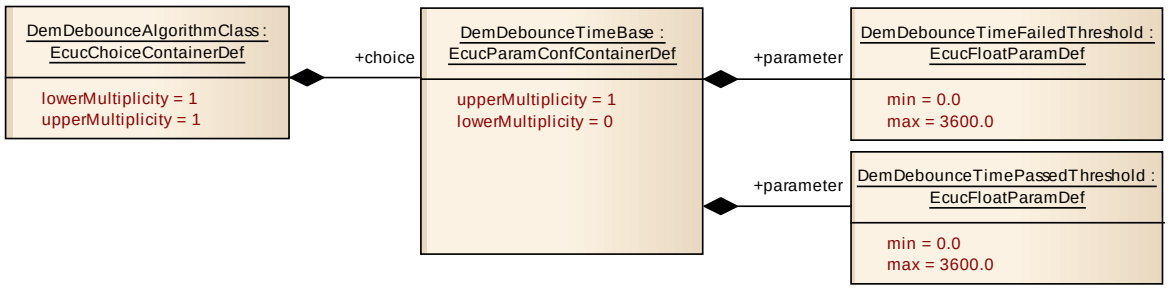

Figure 29 Time based debounce algorithm

[Dem427] ⌈The Dem module shall calculate the fault detection counter (-128 …+127 according to UDS) based on the value and range of the internal debounce timer, to map the internal timer values linearly to the external values (refer to Figure 30).⌋()

[Dem428] ⌈The Dem module shall start the internal debounce timer to qualify the reported event as failed when the monitor reports DEM_EVENT_STATUS_PREFAILED (refer to EventStatus).⌋()

[Dem429] ⌈If the internal debounce timer of a specific event was already triggered and the monitor reports consecutively DEM_EVENT_STATUS_PREFAILED again, the Dem module shall not restart the internal debounce timer.⌋()

[Dem430] ⌈ The Dem module shall provide the configuration parameter DemDebounceTimeFailedThreshold in [ms] used to define the event-specific delay indicating the failed status (active).⌋()

[Dem431] ⌈If the monitor reports DEM_EVENT_STATUS_FAILED, the Dem module shall set the internal debounce timer value to its configured threshold being the failed criteria (refer to DemDebounceTimeFailedThreshold).⌋()

[Dem432] ⌈The Dem module shall start the internal debounce timer to qualify the reported event as passed when the monitor reports DEM_EVENT_STATUS_PREPASSED (refer to EventStatus).⌋()

[Dem433] ⌈If the internal debounce timer of a specific event was already triggered and the monitor reports consecutively DEM_EVENT_STATUS_PREPASSED again, the Dem module shall not restart the internal debounce timer.⌋()

[Dem434] ⌈ The Dem module shall provide the configuration parameter DemDebounceTimePassedThreshold in [ms] used to define the event-specific delay indicating the passed status (not active).⌋()

[Dem435] ⌈If the monitor reports DEM_EVENT_STATUS_PASSED the Dem module shall set the internal debounce timer value to its configured threshold being the passed criteria (refer to DemDebounceTimePassedThreshold).⌋()

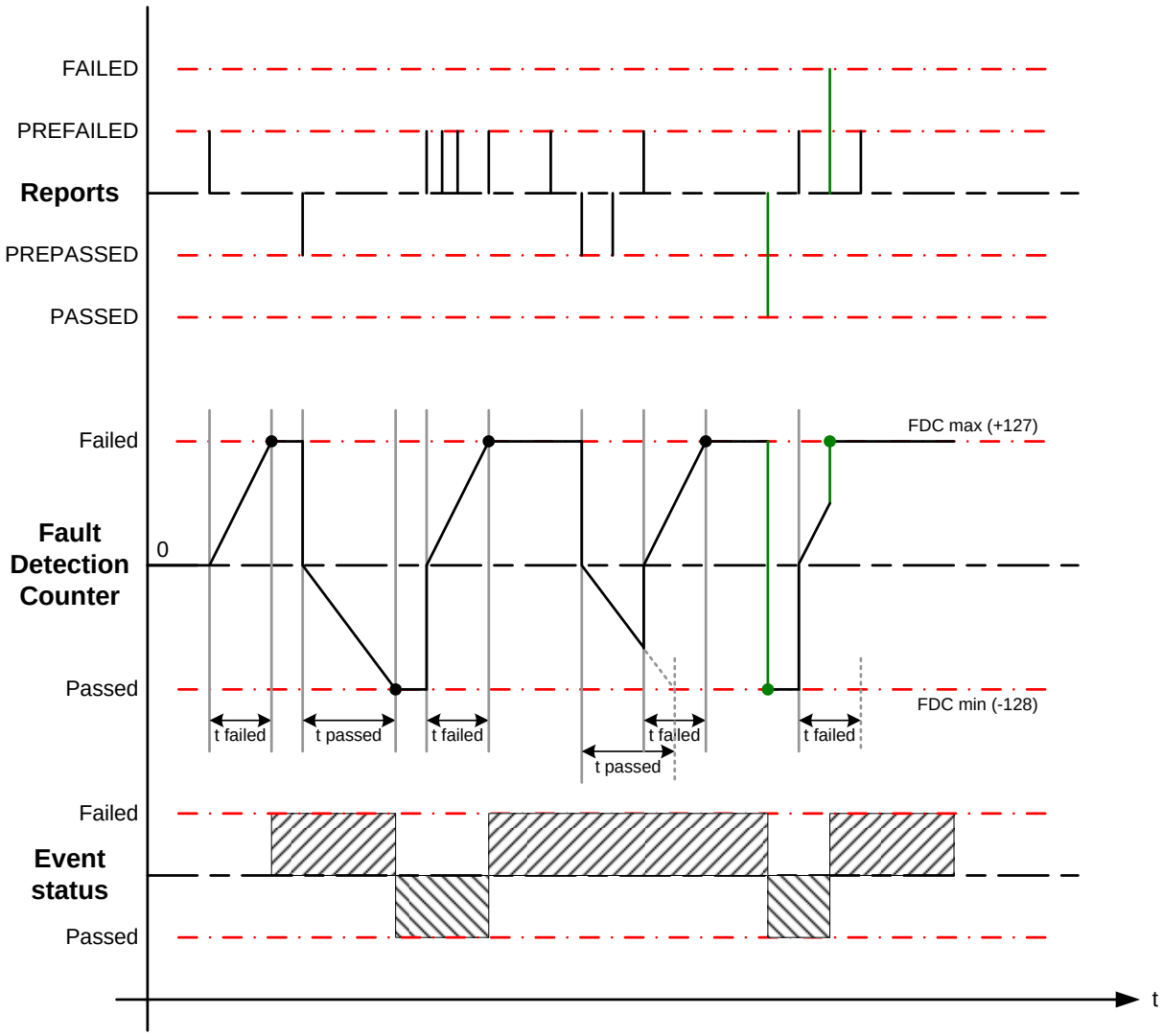

Figure 30 Example of time based debouncing

7.3.3.3 Further specific debounce algorithms

The Dem module may be extended with other specific debounce algorithms, if an application requires particular debounce algorithms implemented Dem-internally.

[Dem436] ⌈If the Dem module implements application-specific debounce algorithms, the event states DEM_EVENT_STATUS_PREFAILED and DEM_EVENT_STATUS_PREPASSED shall be used within Dem_SetEventStatus indicating the maturation process.⌋()

7.3.3.4 Monitor internal debounce algorithm

[Dem437] ⌈If the configuration container DemDebounceAlgorithmClass is set to DemDebounceMonitorInternal, the Dem module shall not use a Dem-internal debounce mechanism for each individual event, to qualify the reported event.⌋()

Note: The monitor is not allowed to report the event states DEM_EVENT_STATUS_PREFAILED and DEM_EVENT_STATUS_PREPASSED for monitor internal debouncing.

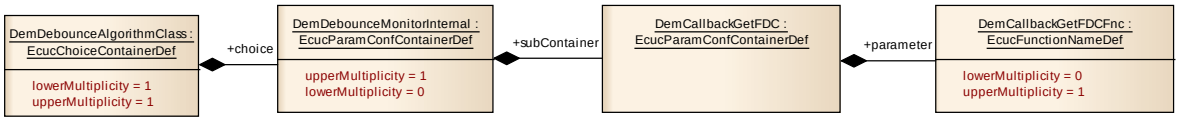

Figure 31 Monitor internal debounce algorithm

7.3.3.5 Fault detection counter initialization and reset conditions

[Dem438] ⌈If Dem-internal debouncing is configured, the Dem module shall set all event-specific fault detection counters to 0 (zero) when Dem_PreInit has been called.⌋()

[Dem343] ⌈After receiving a command for clearing the event memory (refer to Dem_ClearDTC), the respective fault detection counters shall be initialized with 0, presuming event debouncing is handled Dem-internally.⌋()

[Dem344] ⌈If Dem-internal debouncing is configured, the Dem module shall reset the fault detection counter upon starting a new monitoring cycle / operation cycle.⌋()

7.3.3.6 Fault detection counter retrieval

[Dem204] ⌈The event-specific fault detection counter shall be accessible by using the API Dem_GetFaultDetectionCounter (refer to chapter 8.3.3.16).⌋()

[Dem264]「If debouncing is performed by a SW-C (not handled Dem-internally), the Dem module shall use the callback-function GetFaultDetectionCounter (refer to chapter 8.4.3.1.8) to request the current value of the fault detection counter for a given event.」()

[Dem439]「If the callback-function GetFaultDetectionCounter returns other then E_OK, this return value shall also be returned by the API Dem_GetFaultDetectionCounter.」()

Note: For resetting the fault detection counter implemented in a monitor, the Dem module uses the callback-function InitMonitorForEvent (refer to chapters 7.5 and 8.4.3.1.1).

7.3.4 Fault confirmation

After reporting, a fault and entering the pending status, the fault confirmation process begins within the Dem (refer to Figure 23 General diagnostic event storage processing).

This fault confirmation process results in the confirmed state. For that purpose, respective counter thresholds and counter types are specified.

[Dem528]「The Dem module shall provide the configuration parameter DemEventFailureCycleCounterThreshold per event (refer to DemEventClass) defining the maximum number of tested and failed cycles, before the stored event becomes “confirmed”, i.e. enters the confirmed state.」()

[Dem529]「The Dem module shall provide the configuration parameter DemEventFailureCycleRef per event (refer to DemEventClass) defining the specific cycle type, which represents the trigger, that causes an update of the failure counter provided there is failed result reported for the event.」()

[Dem530]「The Dem module shall set the UDS DTC status bit 3 (ConfirmedDTC) to 1, if the respective failure counter of the event memory entry has been processed (incremented or decremented) DemEventFailureCycleCounterThreshold times.」()

Note: This cycle type could be for example equal to DEM_OPCYC_OBD_DCY or to DEM_OPCYC_POWER (refer to DemOperationCycle) and the combination of handling the counters for a specific cycle type is called “fault confirmation”.

The healing confirmation process (described in chapter 7.3.10) handles reported OK / passed results, which yields in the deactivation of a specific warning indicator. For that purpose, respective counter thresholds and counter types are specified.

7.3.5 Combination of diagnostic events

Event combination defines the ability of the Dem module to merge several events to one specific event/DTC. It is used to adapt different monitor results to one significant fault, which is clearly evaluable in a service station. The essential part of implementing a combined event is the calculation of its status information. The combined UDS DTC status byte results from a bitwise logical operation of all associated events.

[Dem025] ⌈ The Dem module shall provide the configuration parameter DemEventCombinationSupport (refer to DemGeneral) to activate and select event combination (combined type 1 or combined type 2).⌋(BSW04073)

[Dem024] ⌈If the Dem module is requested to support event combination, a unique EventId shall be assigned to each combined event (refer to Dem153).⌋(BSW04073)

[Dem026] ⌈If the Dem module is requested to support event combination, the Dem module shall provide an event specific attribute indicating that this event is used for event combination.⌋(BSW04073)

Based on the configuration of the event related data the Dem module allows two different types of event combination (assign event related data to the combine event/DTC or to the sub-events).

[Dem536] ⌈If the Dem module is requested to support event combination, the Dem module shall support the following two types of event combination:

- Combined type 1: The event related data is assigned to the combined event itself and is not assigned to the sub-events, which will be merged.
- Combined type 2: The event related data is assigned to the sub-events, which will be merged to the combined event. The data is not assigned to the combined event itself.⌋()

Note: The primary use-case of event combination is to map more than one event (represented by an EventId) to only one DTC number. If combined type 1 is used, the Dem module handles only one event memory entry and its event related data per combined DTC. If combined type 2 is used, there may be several event memory entries and several sets of event related data. The Dem module handles the combined DTC in consideration of its related event memory entries.

The Table 1 below shows an example of a Dem configuration table including combined events. Several events are mapped to one specific combined event and its related DTC. The combined type 1 is represented by the DTC[1]. For the combined event, a freeze frame is configured. The events, which will be merged to the

combined event, do not have any event related data. On the other hand, the DTC[2] shows an example of the combined type 2. The event related data is assigned to the sub-events.

Unique EventId	*DTC Status mask*	*Combination attribute*	*Combined Event*	*Assinged DTC*	*Freeze frame*	*…*
Event[1]	S1	K1	CE[1]	DTC[1]	FF-Id[28]	…
Event[2]	S2	K1			None	
Event[3]	S3	K1			None	
Event[4]	S4	K2	CE[2]	DTC[2]	None	
Event[5]	S5	K2			FF-Id[77]	…
Event[6]	S6	K2			FF-Id[75]	…
Event[7]	S7	K3	CE[3]	DTC[3]	FF-Id[89]	…
Event[8]	S8	K4	CE[4]	DTC[4]	FF-Id[67]	…
…	…	…	…	…	…	…

Table 1 Example of a Dem configuration table including combined events

As apposed to the chapter 7.3.1.3 the calculation of the combined status byte is based on the logical operation of all combined UDS DTC status bits.

[Dem441] ⌈The Dem module shall implement the status bit calculations for the UDS DTC status byte of a combined event according to Table 2 (below).⌋()

UDS DTC status bit description		*Combined event status information logical equation*
0	TestFailed	$CbEvt_{Bit\ 0}$ = $Event[1]_{Bit\ 0}$ | $Event[2]_{Bit\ 0}$ | … | $Event[n]_{Bit\ 0}$
1	TestFailedThis OperationCycle	$CbEvt_{Bit\ 1}$ = $Event[1]_{Bit\ 1}$ | $Event[2]_{Bit\ 1}$ | … | $Event[n]_{Bit\ 1}$
2	PendingDTC	$CbEvt_{Bit\ 2}$ = $Event[1]_{Bit\ 2}$ | $Event[2]_{Bit\ 2}$ | … | $Event[n]_{Bit\ 2}$
3	ConfirmedDTC	$CbEvt_{Bit\ 3}$ = $Event[1]_{Bit\ 3}$ | $Event[2]_{Bit\ 3}$ | … | $Event[n]_{Bit\ 3}$
4	TestNotCompleted SinceLastClear	$CbEvt_{Bit\ 4}$ = ($Event[1]_{Bit\ 4}$ | $Event[2]_{Bit\ 4}$ | … | $Event[n]_{Bit\ 4}$) & ! $CbEvt_{Bit\ 3}$
5	TestFailedSince LastClear	$CbEvt_{Bit\ 5}$ = $Event[1]_{Bit\ 5}$ | $Event[2]_{Bit\ 5}$ | … | $Event[n]_{Bit\ 5}$
6	TestNotCompleted ThisOperationCycle	$CbEvt_{Bit\ 6}$ = ($Event[1]_{Bit\ 6}$ | $Event[2]_{Bit\ 6}$ | … | $Event[n]_{Bit\ 6}$) & ! $CbEvt_{Bit\ 1}$
7	WarningIndicator Requested	$CbEvt_{Bit\ 7}$ = $Event[1]_{Bit\ 7}$ | $Event[2]_{Bit\ 7}$ | … | $Event[n]_{Bit\ 7}$

Table 2 Combined event status bit ($CbEvt_{Bit}$) calculation

In the table above the following logical operators are used:

- ! = logical negation (NOT)
- | = logical bitwise OR-operation
- & = logical bitwise AND-operation

Note: Events, which are merged to one combined event, are equivalent and valid apart.

[Dem440] ⌈If the Dem module is requested to clear the UDS DTC status information of a combined event/DTC (refer to service $14), the Dem module shall set the related UDS status byte of all events, which are merged to this combined event to 0x50 (Readiness bits 4 and 6 set to 1, and all others are set to zero).⌋()

Note: This implies that it is not allowed to reset the status information of an event, which will be merged to a combined event.

The following section describes the behavior of the Dem module in case of the combined type 1 is configured.

[Dem163] ⌈If the Dem module is requested to support event combination (combined type 1), the UDS DTC status bit transitions of the combined event shall trigger the collection, update and storage of its event related data (freeze frames or extended data records).⌋()

[Dem537] ⌈If the Dem module is requested to report event related data of a combined event (combined type 1), the Dem module shall return the assigned event related data of the combined event.⌋()

Note: This implies that it is not allowed to configure freeze frames or extended data records of an event, which will be merged to a combined event. If it is configured, the data is not reported by a diagnostic request (e.g. $19 06).

[Dem442] ⌈If aging of a combined event (combined type 1) occurs, the Dem module shall delete the combined event from the event memory including its event related data.⌋()

Note: If event combination is used, the duration of the aging mechanism can be extended since several monitors restart the aging counter.

[Dem443] ⌈If a combined event (combined type 1) shall be displaced, the Dem module shall remove this event including its event related data from the event memory (refer to Dem408).⌋()

[Dem538] ⌈If a combined event memory entry (combined type 1) was removed during displacement, the Dem module shall not modify the UDS status bits of the removed combined event, except the UDS status bit 3 (ConfirmedDTC) which is set to 0.⌋()

The section below describes the behavior of the Dem module in case of the combined type 2 is used.

[Dem539] ⌈If the Dem module is requested to support event combination (combined type 2), the UDS DTC status bit transition of each sub-event shall trigger the collection, update and storage of the assigned event related data.⌋()

[Dem540] ⌈If the Dem module is requested to report event related data of a combined event (combined type 2), the Dem module shall return the event related data of the assigned sub-events.⌋()

Note: This implies that it is only allowed to configure freeze frames or extended data records for the sub-event(s). It is up to the implementation, which set of event related is chosen.

[Dem541] ⌈If aging of a combined event (combined type 2) occurs, the Dem module shall delete the combined sub-event from the event memory including its event related data.⌋()

[Dem542] ⌈If a combined event (combined type 2) shall be displaced, the Dem module shall remove this sub-event including its event related data from the event memory (refer to Dem408).⌋()

7.3.6 Enable and storage conditions of diagnostic events

In certain cases, the event retention depends on parameters, which are available on operation system level. These parameters are combined to groups and define a certain number of checks (e.g. correct voltage range) before the event report is accepted or the event gets qualified. The checks are done by software components. The Dem module provides the ability to consider the reported result (a specific condition is fulfilled or not) during the event handling. There are two different types of conditions: enable conditions and storage conditions.

The strategy how to use the conditions (e.g. suppression of subsequent faults) and especially the assigning matrix of conditions to the events depends on the OEM.

The enable conditions are defined as a set of parameters, which are assigned to a specific condition. As long as this condition is not fulfilled, the event reports (refer to Dem_ReportErrorStatus and Dem_SetEventStatus) are not valid and therefore will not be accepted. It has no impact on Dem_ResetEventStatus. A similar functionality is used for the function inhibition. In contrast to the mutual exclusion matrix of the FiM, which is based on events, the enable conditions are based on system parameters (e.g. ignition status, local voltage).

The storage conditions are defined as a set of parameters, which are assigned to a specific condition. As long as this condition is not fulfilled, the Dem module does not store the event to the event memory.

The following requirements introduce the handling of the enable conditions.

[Dem444] 「 The Dem module shall provide a configuration parameter DemEnableConditionSupport (refer to DemGeneral) to enable or disable the handling of enable conditions Dem-internally.」()

[Dem202]「If the Dem module is requested to support enable conditions, the Dem module shall provide the API Dem_SetEnableCondition (refer to chapter 8.3.3.14) receiving the current status (condition fulfilled or not) of a specific enable condition.」()

[Dem446]「If the Dem module is requested to support enable conditions, the Dem module shall provide the ability to assign one enable condition group (refer to DemEnableConditionGroup), which includes one or several enable conditions (refer to DemEnableCondition) to a specific event.」()

Note: The enable condition groups are introduced to improve the configuration methodology of the Dem module. If a significant number of events depend always on the same enable conditions, it is less effort to select one of the defined enable condition groups instead of assigning several enable conditions to each of those events.

[Dem447]「If the Dem module is requested to support enable conditions, the Dem module shall check the assigned enable conditions after the diagnostic monitor reports an event (passed/failed or pre-passed/pre-failed, refer to APIs Dem_ReportErrorStatus and Dem_SetEventStatus).」()

[Dem449]「If one enable condition is not fulfilled, all status reports from SW-Cs (Dem_SetEventStatus) and BSW modules (Dem_ReportErrorStatus) for those events being assigned to this condition shall be ignored (no change of UDS DTC status byte) by the Dem.」()

[Dem450]「If all event-specific enable conditions are fulfilled, all status reports from SW-Cs (Dem_SetEventStatus) and BSW modules (Dem_ReportErrorStatus) for those events being assigned to these conditions shall be accepted by the Dem from this point in time on.」()

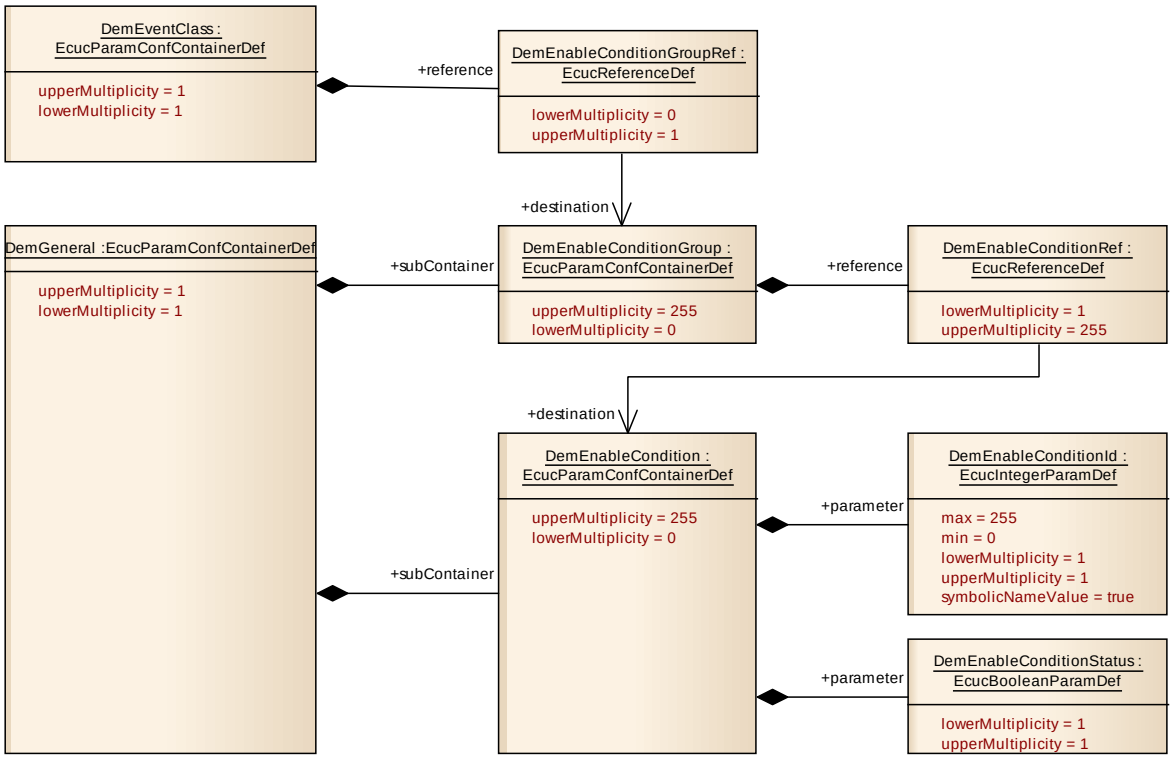

Figure 32 Enable condition assignment configuration

The following requirements introduce the handling of the storage conditions.

[Dem451] 「 The Dem module shall provide a configuration parameter DemStorageConditionSupported (refer to DemGeneral) to enable or disable the handling of storage conditions Dem-internally.」()

[Dem543] 「If the Dem module is requested to support storage conditions, the Dem module shall provide the API Dem_SetStorageCondition (refer to chapter 8.3.3.15) receiving the current status (condition fulfilled or not) of a specific storage condition.」 ()

[Dem453] 「If the Dem module is requested to support storage conditions, the Dem module shall provide the ability to assign one storage condition group (refer to DemStorageConditionGroup), which includes one or several storage conditions (refer to DemStorageCondition) to a specific event.」()

Note: The storage condition groups are introduced to improve the configuration methodology of the Dem module. If a significant number of events depend always on the same storage conditions, it is less effort to select one of the defined storage condition groups instead of assigning several storage conditions to each of those events.

[Dem455] ⌈If the Dem module is requested to support storage conditions, the Dem module shall check the assigned storage conditions after the event gets qualified as failed (UDS DTC status bit 0 changes from 0 to 1).⌋()

[Dem458] ⌈If one storage condition is not fulfilled and no respective event memory entry exists, the Dem module shall not enter the reported event into the event memory (refer to Dem184).⌋()

[Dem591] ⌈If one storage condition is not fulfilled and a respective event memory entry exists, the Dem module shall not update the event memory of the reported event (refer to Dem396).⌋()

[Dem459] ⌈If all event-specific storage conditions are fulfilled, the Dem module shall permit the storage of the reported event.⌋()

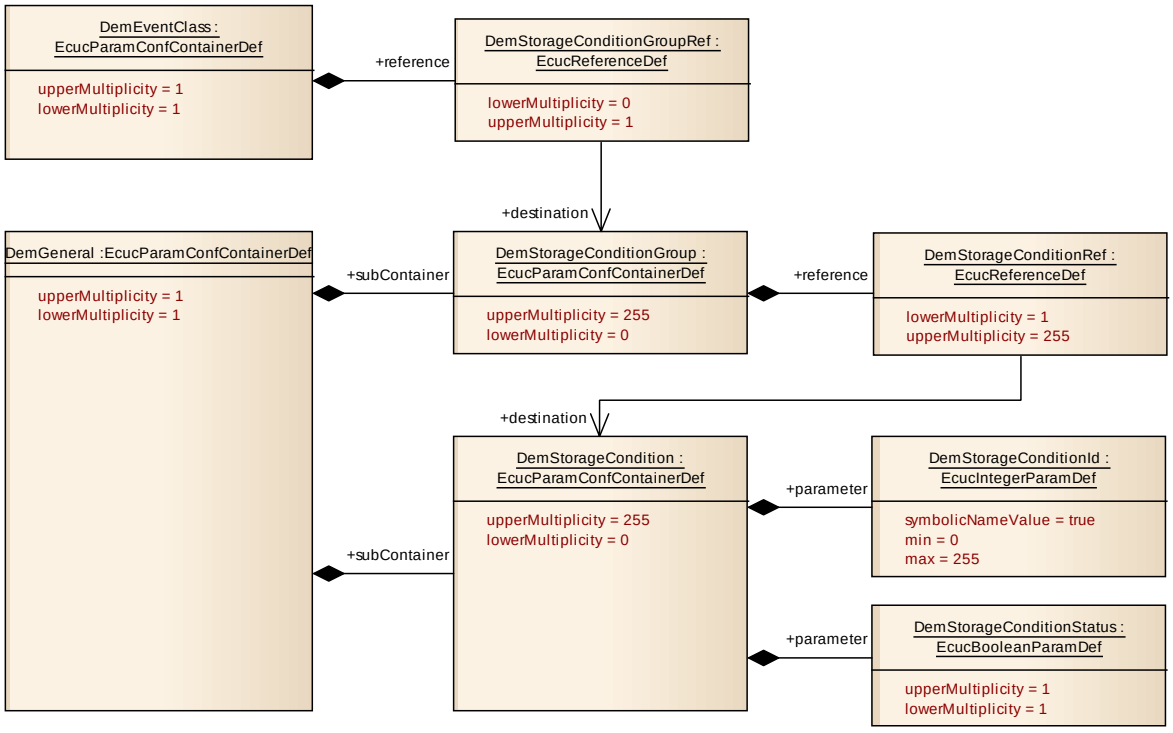

Figure 33 Storage condition assignment configuration

7.3.7 Event related data

´Event related data´ are additional data, e.g. sensor values or time stamp/mileage, which are stored in case of an event. ISO 14229-1 defines two different types of event related data: snapshot data (freeze frames) and extended data. The number or

sets of stored event related data are strongly OEM / failure specific and are therefore configurable. This data is provided by SW-C or other BSW modules.

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

[Dem460] ⌈The Dem module shall provide a configuration table to combine event related data (freeze frames & extended data) with a specific DTC number, etc. (refer to chapter 7.3.7.4).⌋()

Note: This does not define a specific implementation (e.g. look-up table, matrix, etc.). Furthermore it relates to the link between the configured data. An event is characterized by its event Id, DTC value, configured freeze frames and extended data records, etc.

7.3.7.1 Storage of freeze frame data

[Dem039] ⌈The Dem module shall support event-specific freeze frame storage.⌋ (BSW04074)

In general, there are two options for freeze frame configuration: (1) Non-emission related freeze frames are configured specific to one particular event. (2) Emission related freeze frames are configured globally for a particular ECU (OBD legislation requires one single freeze frame class only).

[Dem040] ⌈The Dem module shall support the storage of one or several DIDs per freeze frame record assigned by configuration (refer to DemFreezeFrameClass).⌋ (BSW04074)

Note: A freeze frame is represented as a list of DIDs (refer to DemDidClass) or PIDs (refer to DemDidClass).

Note: Due to implementation reasons, the Dem usually needs to reserve memory for the maximum freeze frame size multiplied by the number of freeze frames.

[Dem337] ⌈ If the Dem module uses calculated record numbers (refer to DemTypeOfFreezeFrameRecordNumeration), the Dem module shall be capable to store the configured number of freeze frames (refer to DemMaxNumberFreezeFrameRecords).⌋()

[Dem581] ⌈If the Dem module uses calculated record numbers, the Dem module shall numerate the event-specific freeze frame records consecutively starting by 1, based on their chronological order.⌋()

[Dem582] ⌈If the Dem module uses dedicated, configured record numbers (refer to DemTypeOfFreezeFrameRecordNumeration), the Dem module shall be capable to store the configured number of configured freeze frame record numbers (refer to DemFreezeFrameRecNumClassRef).⌋()

[Dem583] ⌈If the Dem module uses dedicated, configured record numbers, the Dem module shall numerate the event-specific freeze frame records by their configured values (refer to DemFreezeFrameRecNumClass), with the same order of the configuration for their chronological order.⌋()

[Dem461] ⌈ The Dem module shall provide the configuration parameter DemFreezeFrameCapture (refer to DemGeneral) to define the global trigger to collect the data of an event-specific freeze frame.⌋()

[Dem261] ⌈The Dem module shall use the C-callback ReadDataElement (refer to chapter 8.4.3.1.7) respectively the operation ReadData of the interface DataServices_<SyncDataElement> (refer to chapter 8.4.3.8) to collect all external data elements of the respective freeze frame.⌋()

[Dem463] ⌈If the SW-C or BSW module cannot not provide the requested data (ReadDataElement returns other than E_OK), the Dem shall fill the missing data with the padding value 0xFF, reports the development error DEM_E_NODATAAVAILABLE to the Det and continues its normal operation.⌋()

[Dem190] ⌈If an event is stored or updated in the event memory, the Dem module shall store the collected freeze frame record data (according to Dem581 and Dem583) into the event memory entry.⌋()

[Dem585] ⌈If more than one freeze frame record is configured for a specific event, and this event is updated in the event memory, and all available freeze frame record slots for this event are occupied, the Dem module shall update the most recent record.⌋()

Note: The first freeze frame record slot will always represent the first occurrence.

7.3.7.2 Pre-storage of freeze frame data

The pre-storage of freeze frames can be used for event with highly volatile freeze frame data. With the first indication of the appearance of a specific event, even if the event is not yet de-bounced or qualified, the freeze frame data is captured (e.g.

because of rapid changing of event related data during running failure monitoring phase).
The pre-stored freeze frame functionality is used by monitors.

[Dem334] ⌈For OBD relevant systems (refer to DemOBDSupport in DemGeneral) the Dem module shall provide the API Dem_PrestoreFreezeFrame (refer to chapter 8.3.3.4) and Dem_ClearPrestoreFreezeFrame (refer to chapter 8.3.3.5).⌋()

Note: In case of a non OBD relevant system the Dem module can provide the prestorage of freeze frames.

[Dem002] ⌈ The Dem module shall provide the configuration parameter DemFFPrestorageSupported (refer to DemEventClass) to enable or disable pre-storage handling of event-specific freeze frames.⌋()

Note: The configuration parameter DemFFPrestorageSupported may be used as indicator whether prestorage is required at all by any event (in order to globally activate / deactivate this feature) and how many events require it (in order to derive Dem configuration internal switches).

[Dem189] ⌈The Dem module shall provide the API Dem_PrestoreFreezeFrame (refer to chapter 8.3.3.4) to capture the pre-storage data of an event-specific freeze frame regardless of the UDS DTC status bit changes.⌋()

[Dem464] ⌈If a pre-stored freeze frame is available, the Dem module shall use the data of the pre-stored freeze frame instead of the current data at the point in time when the event is qualified (refer to Dem461 and Dem190).⌋()

Note: This implies that the pre-stored freeze frame is released after the event is qualified.

[Dem191] ⌈If no pre-stored freeze frame is available, the Dem module shall use the defined trigger conditions for capturing and storing freeze frame data (refer to Dem461).⌋()

Note: The captured data while using pre-stored freeze frames can differ from the data, which is collected using the UDS DTC status bit transitions as a trigger.

Note: To ensure absence of reaction to stored freeze frames of qualified events an additional freeze frame 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.

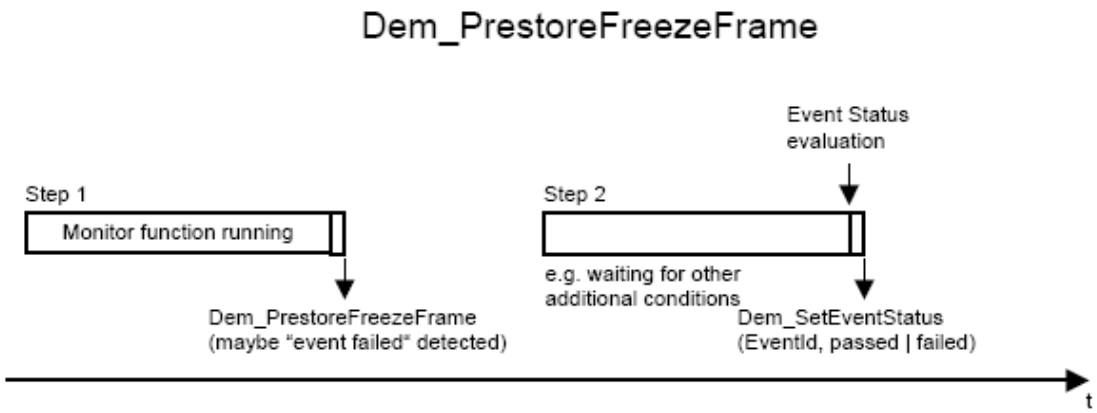

Figure 34 Example to use Dem_PrestoreFreezeFrame to prestore freeze frame data

[Dem050] ⌈ The Dem module shall provide the API Dem_ClearPrestoredFreezeFrame (refer to chapter 8.3.3.5) to release the pre-stored freeze frame for the specific event.⌋()

[Dem465] ⌈If an event gets qualified as passed (UDS DTC status bit 0 changes from 1 to 0) the Dem module shall release the pre-stored freeze frame for the specific event.⌋()

7.3.7.3 Storage of extended data

An extended data record contains additional information associated to a specific event that is not contained in a freeze frame (extended data, e.g. frequency counters, aging counters, etc.). According to the DID- or PID-based configuration of freeze frame data, extended data are divided in extended data records defined by its record numbers.

[Dem041] ⌈The Dem module shall support the storage of one or several extended data records assigned by configuration (refer to DemExtendedDataClass).⌋()

Note: Extended data are represented as a list of records (refer to DemExtendedDataRecordClass).

[Dem466] ⌈ The Dem module shall provide the configuration parameter DemExtendedDataRecordUpdate per extended data record, to define the update-condition (first time or every time) for the data of this record.⌋()

[Dem467] ⌈ The Dem module shall provide the configuration parameter DemExtendedDataCapture (refer to DemGeneral) to define the global trigger to

collect data of an event-specific extended data (according to the specific record update-conditions).」()

[Dem282] 「The Dem module shall use the C-callback ReadDataElement respectively the operation ReadData of the interface DataServices_<SyncDataElement> to collect all external data elements of the respective extended data record.」()

[Dem468] 「If an event is stored or updated in the event memory, the Dem module shall store the collected extended data (according to Dem466) into the event memory entry.」()

7.3.7.4 Configuration of Event related data

This section describes the configuration of event related data and the access of event related data from the SW-Cs/BSW modules.

Note: The configuration model follows a flexible configuration process, but does not imply any explicit implementation.

The event related data of diagnostic events contain none or one set of extended data records (refer to DemExtendedDataClass), and none or one set of freeze frame records (refer to DemFreezeFrameClass) with its calculated or configured record numbers. Therefore a class-concept is used (refer to Figure 35).

An extended data record, a DID, or a PID can contain one ore more data elements (refer to DemDataElementClass).

Note: Asynchronous DIDs, as well as DIDs with a variable length are not supported by the Dem module, and shall therefore not be connected to the Dem.

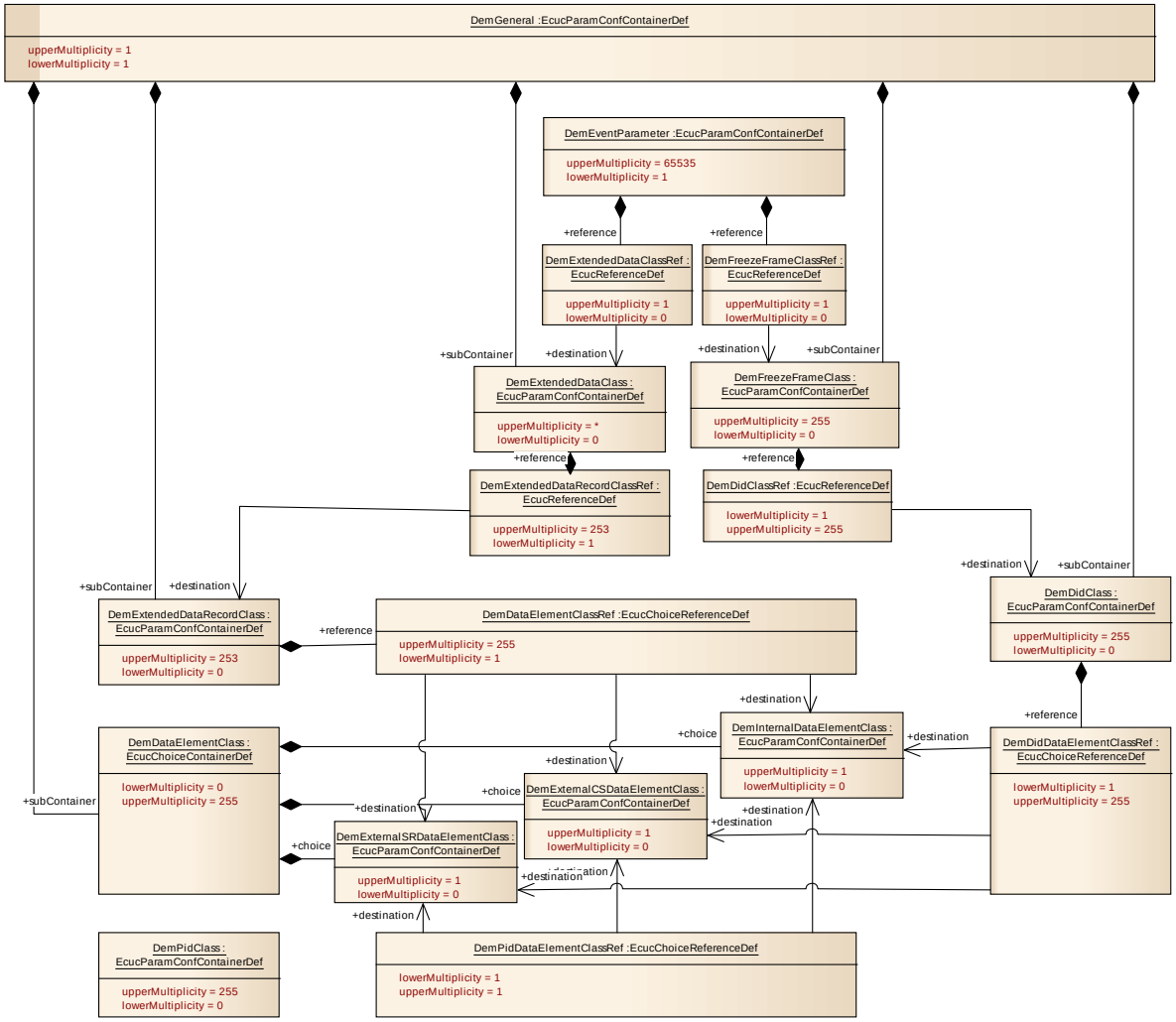

Figure 35 Event related data configuration

A data element is provided from a SW-C or BSW module, or is computed Dem-internally.

For each external data element a according require-port or C-callback is generated based on the configuration (refer to DemExternalCSDataElement). For each internal data element, the respective Dem-internal value is mapped.

Note: These data elements are typically specified in a Diagnostic Data Template.

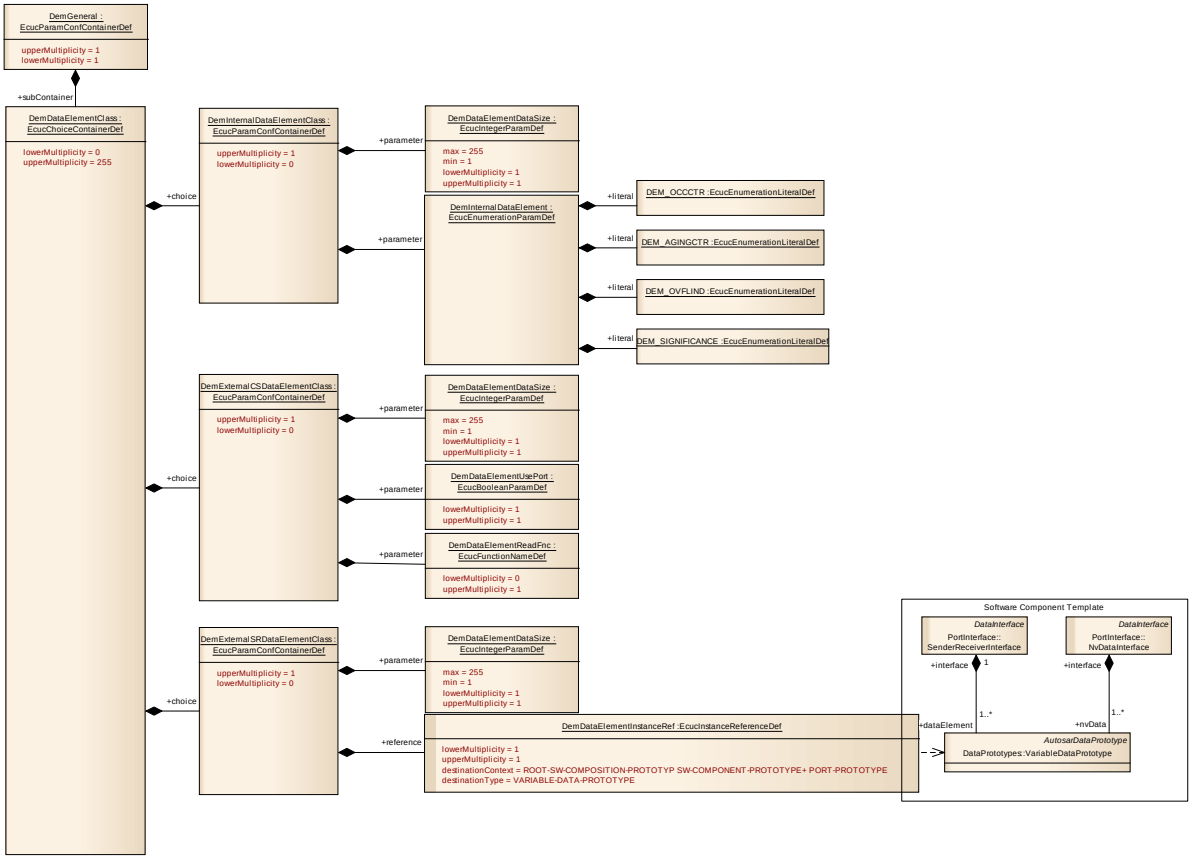

Figure 36 Data element configuration

[Dem469] ⌈The Dem module shall provide the ability to map Dem-internal data values (e.g. aging counter, occurrence counter) to specific data element (refer to DemDataElement in DemDataElementClass) contained in an extended data records, a DID, or a PID.⌋()

Note: If a Dem-internal data element is mapped to e.g. an extended data record (by configuration), this information can simply be requested by UDS Service *ReadDTCInformation* – Sub-Service *reportDTCExtendedDataRecordByDTCNumber* (19, 06 hex).

[Dem471] ⌈ If the configuration parameter DemDataElement is set to DEM_OCCCTR, then the Dem-internal value of the occurrence counter (refer to chapter 7.1.2) shall be mapped to the respective data element.⌋()

[Dem472] ⌈ If the configuration parameter DemDataElement is set to DEM_AGINGCTR, then the Dem-internal value of the aging counter (refer to chapter 7.3.9) shall be mapped to the respective data element (based on Dem643 or Dem644).⌋()

[Dem643] ⌈ If DemAgingCycleCounterProcessing is set to DEM_PROCESS_AGINGCTR_INTERN, the aging counter mapping shall be based on a count-up mechanism from 0 to DemAgingCycleCounterThreshold (refer to ISO 14229-1, Annex D).⌋()

[Dem646] ⌈ If no aging is supported for an event and DemAgingCycleCounterProcessing is set to DEM_PROCESS_AGINGCTR_INTERN, the reported data element shall be 0.⌋()

[Dem644] ⌈ If DemAgingCycleCounterProcessing is set to DEM_PROCESS_AGINGCTR_EXTERN, the aging counter mapping shall be based on the event-specific aging counter value (refer to Dem640, Dem641, Dem642).⌋()

[Dem647] ⌈ If no aging is supported for an event and DemAgingCycleCounterProcessing is set to DEM_PROCESS_AGINGCTR_EXTERN, the reported data element shall be the unavailable value (255).⌋()

[Dem473] ⌈ If the configuration parameter DemDataElement is set to DEM_OVFLIND, then the Dem-internal value of the overflow indication (refer to chapter 7.3.2.2) shall be mapped to the respective data element.⌋()

[Dem592] ⌈ If the configuration parameter DemDataElement is set to DEM_SIGNIFICANCE, then the (static) Dem-internal value of the event significance (refer to chapter 7.1.4) shall be mapped to the respective data element.⌋()

The Dem module may be extended with further specific Dem-internal data elements, if a specific configuration requires particular data values, which are computed Dem-internally.

[Dem470] ⌈If the Dem module implements customer-specific Dem-internal data elements, the configuration parameter DemDataElement shall be extended with the respective enumeration values.⌋()

Note: The computation of any Dem-internal data value, which is not configured as data element, can be discarded (if it is not necessary for other internal behavior handling).

7.3.7.5 Notification of data changes

[Dem474] ⌈The Dem module shall notify other SW-Cs / BSW modules about updates of the event related data in the event memory (refer to 8.4.3.1.5 EventDataChanged).⌋()

An update of the event related data occurs every time, a new event memory entry is done or an existing is updated.

Note: The Dem module does not evaluate the return value (e.g. if other than E_OK) of this callback function.

[Dem475] ⌈The Dem module shall call EventDataChanged if an event is stored in its event memory and if an existing event memory entry is updated.⌋()

Note: The callback is not triggered after each update of statistical data (e.g. fault detection counter) avoiding a high call rate.

Note: The callback EventStatusChanged is also called on update of the event related data (during the event status changes), but this callback is called quite more often (e.g. on operation cycle change, on event-deletion, etc.).

Configuration of EventDataChanged: It is configurable per event, whether a SW-C / BSW module shall be notified from this callback function, when an update of the event related data for this event is performed.

[Dem478] ⌈The Dem module shall provide access on (new) freeze frame data (refer to 8.3.3.18 Dem_GetEventFreezeFrameData).⌋()

[Dem479] ⌈The function Dem_GetEventFreezeFrameData shall report the data of the freeze frame record of the requested diagnostic event.⌋()

The format of the data in the destination buffer (DestBuffer) of the function Dem_GetEventFreezeFrameData is raw hexadecimal values and contains no header-information like RecordNumber or DataId.
The size of the buffer equals to the configuration settings of all respective data elements.

Note: The Dcm uses the functions Dem_GetFreezeFrameDataByRecord, Dem_GetFreezeFrameDataByDTC and Dem_ReadDataOfOBDFreezeFrame instead of the function Dem_GetEventFreezeFrameData. The Dlt uses the function Dem_DltGetMostRecentFreezeFrameRecordData instead.

[Dem476] ⌈The Dem module shall provide access on current extended data (refer to 8.3.3.19 Dem_GetEventExtendedDataRecord).⌋()

[Dem477] 「The function Dem_GetEventExtendedDataRecord shall report the data of the extended data record of the requested diagnostic event.」()

The format of the data in the destination buffer (DestBuffer) of the function Dem_GetEventExtendedDataRecord is raw hexadecimal values and contains no header-information like RecordNumber.
The size of the buffer equals to the configuration settings of all respective data elements.

Note: The Dcm uses the function Dem_GetExtendedDataRecordByDTC instead of the function Dem_GetEventExtendedDataRecord. The Dlt uses the function Dem_DltGetAllExtendedDataRecords instead.

7.3.8 Operation cycle management

The Dem module uses different operation cycles (ref. to ISO 14229-1). Those cycles could either be provided by other BSW modules and SW-C or generated by the Dem module itself.

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

The Dem operation cycle management processes these different types of operation cycle definitions to create Dem-specific operation cycle state information.

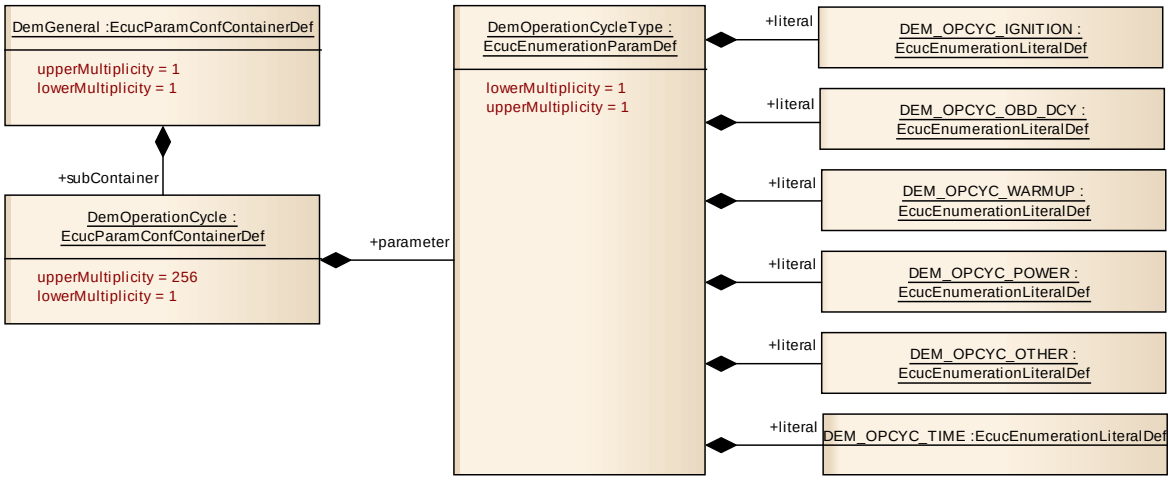

Figure 37 Operation cycle configuration

[Dem577] ⌈ The Dem module shall provide the configuration parameter DemOperationCycleStatusStorage (refer to DemGeneral) to define if the operation cycle state shall be available over the power cycle (stored non-volatile) or not. ⌋()

[Dem480] ⌈ The operation cycle management of the Dem module shall handle different types of operation cycle definitions, which are defined by the configuration parameter DemOperationCycleType (refer to DemOperationCycle). ⌋(BSW04076)

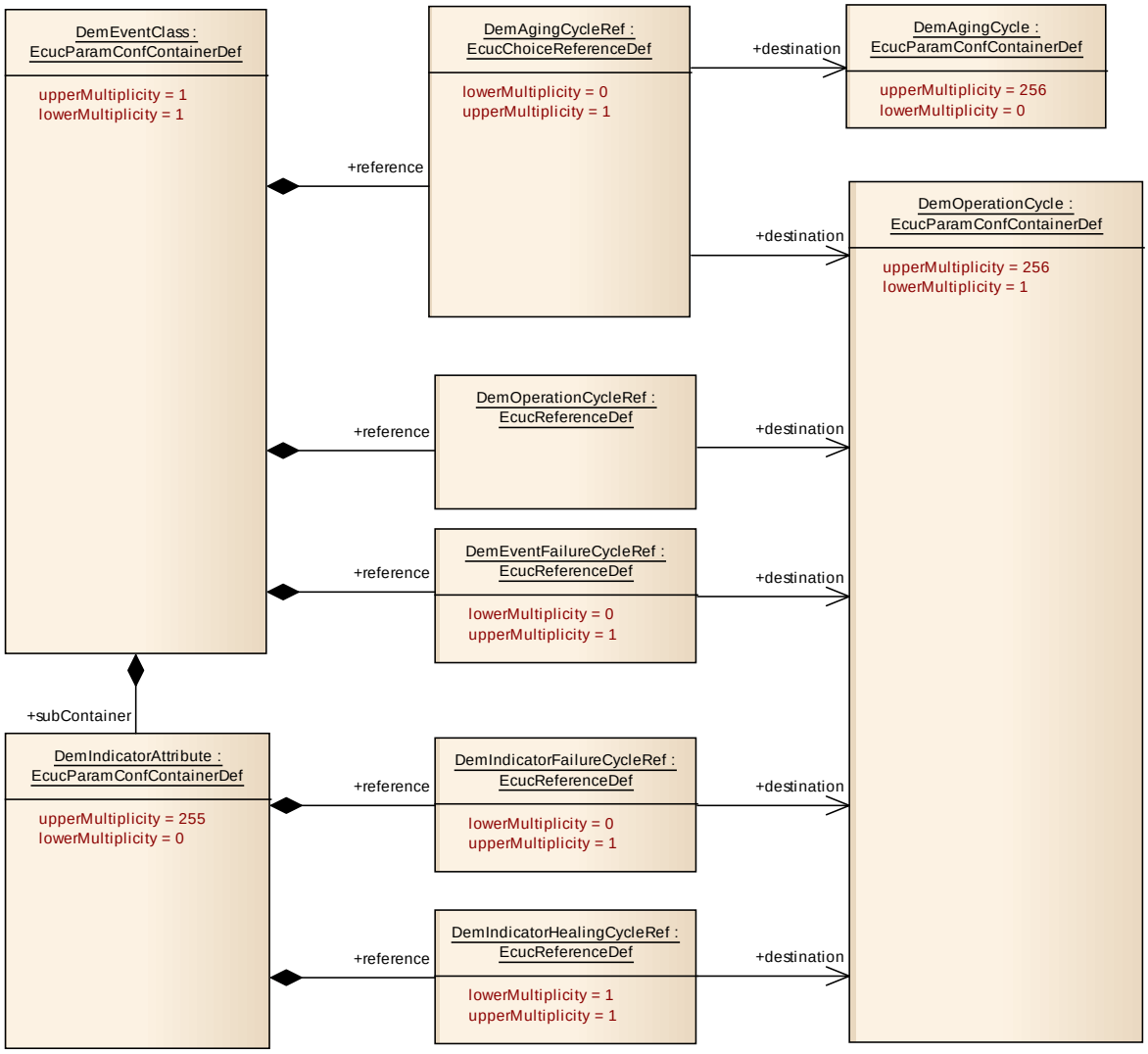

Figure 38 Operation and aging cycle assignment

[Dem481] ⌈ If an operation cycle has started, all status reports from SW-Cs (Dem_SetEventStatus) and BSW modules (Dem_ReportErrorStatus) for those events, being assigned to this cycle shall be accepted by the Dem from this point in time on. ⌋()

[Dem482] ⌈If an operation cycle has ended, all status reports from SW-Cs (Dem_SetEventStatus) and BSW modules (Dem_ReportErrorStatus) for those events being assigned to this cycle shall be ignored (no change of UDS DTC status byte) by the Dem.⌋()

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

[Dem338] ⌈The operation cycle management of the Dem module shall use the reported state (DEM_CYCLE_STATE_START / DEM_CYCLE_STATE_END) of the API Dem_SetOperationCycleState (refer to chapter 8.3.3.6) to set the Dem specific operation cycle state (started / ended).⌋()

Note: This API is called by the SW-Cs / BSW modules, as soon as those detect the status change of the monitored operation cycles.

The operation cycle reporting can be Dem-internal for Dem module self-calculated operation cycles.

[Dem483] ⌈If the API Dem_SetOperationCycleState is called with DEM_CYCLE_STATE_START and the respective operation cycle was already started, the operation cycle shall be restarted (started again).⌋()

Note: This will be the case, if the end- and start-criteria of an operation cycle is fulfilled at exactly the same point in time. Therefore, the caller of Dem_SetOperationCycleState needs only to report “start”.

[Dem484] ⌈If the API Dem_SetOperationCycleState is called with DEM_CYCLE_STATE_END and the respective operation cycle was already ended, the API shall perform no further action.⌋()

[Dem485] ⌈If an external operation cycle counter is configured, the Dem module shall provide the API Dem_SetOperationCycleCntValue (refer to chapter 8.3.3.7) to get the current value of the operation cycle counter reported.⌋()

[Dem486] ⌈If the API Dem_SetOperationCycleCntValue is used, the Dem module shall detect changes of the reported value and update the internal operation cycle counter value of the Dem module.⌋()

Note: There is a use-case to handle an external operation cycle counter value, which is provided by a software component or a software module of a connected ECU. To differentiate between the internal handling triggered by using the API Dem_SetOperationCycleState and the provided operation cycle counter value, the Dem module uses the configuration parameter DemOperationCycleProcessing.

[Dem487] ⌈ The Dem module shall provide the configuration parameter DemOperationCycleProcessing (refer to DemGeneral) configuring whether the operation cycles are triggered by DEM_CYCLE_STATE_START or collecting an external counter value, which results in respective state changes.⌋()

7.3.9 Aging of diagnostic events

The Dem module provides the ability to remove a specific event from the event memory, if its fault conditions are not fulfilled for a certain period of time (operation cycles). This process is called as “aging” or “unlearning”.

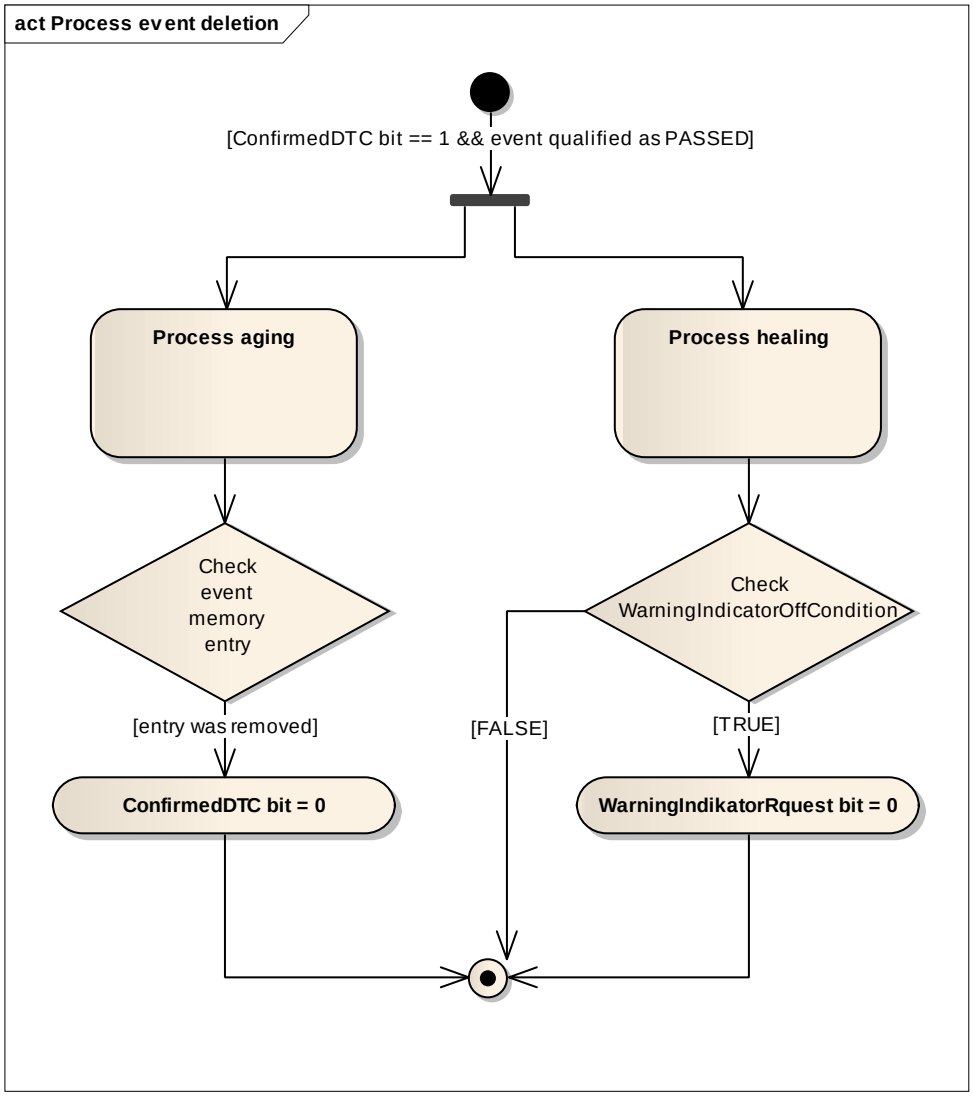

Figure 39 General diagnostic event deletion processing

[Dem019] ⌈The Dem module shall support an event-specific aging counter, which is calculated based on the configured (refer to configuration parameter

DemAgingCycleRef) or an external aging cycle, if aging is enabled (refer to DemAgingAllowed) for this event.」(BSW04068, BSW04106, BSW04076)

[Dem488] 「 The Dem module shall provide the configuration parameter DemAgingCycleCounterProcessing (refer to DemGeneral) defining the handling of the aging cycle counter value, which is internally derived by a status change of the defined aging cycle or provided externally by Dem_SetAgingCycleCounterValue (refer to Dem491).」()

[Dem492] 「The Dem module shall be able to cover the current value of the aging counter of each individual event memory entry, to support an output.」()

Note: For extended fault analysis, it is possible to map the current value of the aging cycle counter to a specific data element of an extended data record (refer to 7.3.7.3 Storage of extended data).

[Dem493] 「 The Dem module shall provide the configuration parameter DemAgingCycleCounterThreshold (refer to DemEventClass) defining the number of passed aging cycles, after which the stored event shall be deleted from the event memory.」()

[Dem495] 「In case of OBD-relevant events, the aging cycle shall be based on cycles defined by OBD legislation.」()

[Dem497] 「If aging of an event occurs, the Dem module shall delete the event from the event memory including its event related data.」()

[Dem498] 「If aging of an event occurs, the Dem module shall not modify the UDS status bits of the removed event, except the UDS status bit 3 (ConfirmedDTC) which is set to 0.」()

[Dem161] 「The Dem module shall handle the reoccurrence of unlearned events like new events, since they were previously deleted from the event memory by aging.」(BSW04070)

7.3.9.1 Internal aging

[Dem489] ⌈The Dem module shall only allow processing (counting further) the value of the aging cycle counter, if the related event is stored in the event memory and is qualified as passed.⌋()

[Dem494] ⌈ The Dem module shall provide the configuration parameter DemAgingCycleRef (refer to DemEventClass) defining the event-specific operation/aging cycle, whose status change triggers the processing (counting further) of the aging cycle counter value.⌋()

Note: Refer to chapter 7.3.8 for the handling of the operation cycle.

[Dem490] ⌈If aging is processed Dem-internally (DemAgingCycleCounterProcessing is set to DEM_PROCESS_AGINGCTR_INTERN), the Dem module shall process (count further) the aging cycle counter value, if the respective aging cycle ends/restarts.⌋()

Note: It is possible to define an event-specific aging cycle (refer to DemAgingCycle), which does not correspond to the event-specific operation cycle. In this case, the Dem module is not able to trigger the event-specific aging cycle counter by using the API Dem_SetOperationCycleState (refer to chapter 8.3.3.6) or Dem_SetOperationCycleCntValue (refer to chapter 8.3.3.7).

[Dem496] ⌈If any configured event-specific aging cycle differs from the event-specific operation cycle, the Dem module shall provide the API Dem_SetAgingCycleState (refer to chapter 8.3.3.8) to indicate the status change of the defined aging cycle.⌋()

7.3.9.2 External aging

[Dem491] ⌈ If an external aging cycle counter value is configured (DemAgingCycleCounterProcessing is set to DEM_PROCESS_AGINGCTR_EXTERN), the Dem module shall provide the API Dem_SetAgingCycleCounterValue (refer to chapter 8.3.3.8) to get the current value of the aging cycle counter reported centrally.⌋()

[Dem639] ⌈If Dem_SetAgingCycleCounterValue() is called the Dem module shall process aging with the handed parameter AgingCycleCounterValue (refer to Dem493) for all events, which are stored in the event memory and qualified as passed.⌋()

Note: If DemAgingCycleCounterProcessing is set to DEM_PROCESS_AGINGCTR_EXTERN, no event-specific aging cycle (DemAgingCycleRef) and no usage of the API Dem_SetAgingCycleState is available.

[Dem640] ⌈ If DemAgingCycleCounterProcessing is set to DEM_PROCESS_AGINGCTR_EXTERN and Dem_SetAgingCycleCounterValue was never invoked since the first start of the Dem module, the aging cycle counter shall be set to be unavailable (255).⌋()

[Dem641] ⌈ If DemAgingCycleCounterProcessing is set to DEM_PROCESS_AGINGCTR_EXTERN and the UDS DTC Status Bit 0 (TestFailed) changes from 0 to 1, the event-specific aging counter shall be set to the latest reported external aging cycle counter value.⌋()

[Dem642] ⌈ If DemAgingCycleCounterProcessing is set to DEM_PROCESS_AGINGCTR_EXTERN and the UDS DTC Status Bit 0 (TestFailed) changes from 1 to 0, the event-specific aging counter shall be set according to the following calculation:
event-specific aging counter = (“latest reported external aging cycle counter value” + DemAgingCycleCounterThreshold) % (“maximum value (254)” + 1).⌋()

Note: For external aging, the value 256 cannot be used for DemAgingCycleCounterThreshold.

7.3.10 Healing of diagnostic events

The Dem module provides the ability to activate and deactivate indicators per event stored in the event memory. This process is defined as healing of a diagnostic event (refer to Figure 39).

7.3.10.1 Warning indicator handling

The detailed configuration of the warning indicator handling within the Dem is shown in Figure 40.

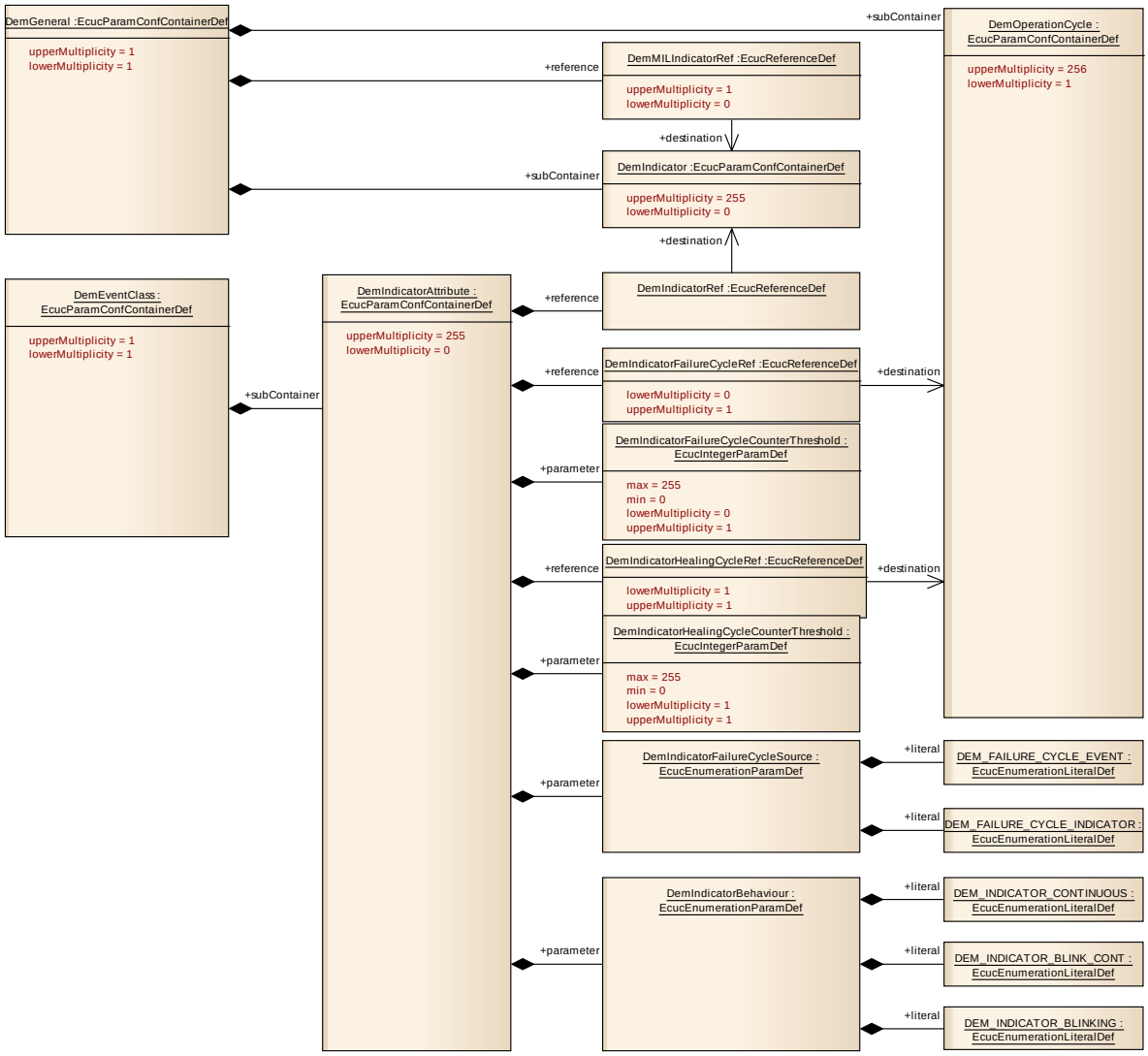

Figure 40 Warning indicator configuration

[Dem499] ⌈The Dem module shall support event specific counters to activate and deactivate indicators, which are calculated based on the configured failure and healing cycles (e.g. to turn on the MIL upon fault confirmation, and turn off the MIL after subsequent healing over three OBD-driving cycles).⌋()

[Dem509] ⌈ The Dem module shall provide the configuration parameter DemIndicatorId (refer to DemIndicator) to cover the identifier of a specific indicator.⌋()

[Dem510] ⌈The Dem module shall be able to assign no indicator, one indicator, or several indicators to a specific event (refer to DemIndicatorAttribute).⌋()

[Dem511] ⌈ The Dem module shall provide the configuration parameter DemIndicatorBehaviour (refer to DemIndicatorAttribute) to assign a specific behavior (e.g. illuminate continuously or blinking) to each indicator and per event.⌋()

Note: During the integration process of the Dem module, different indicators and behaviors (e.g. indicator lamps, text messages or icons) can be assigned to an event.

[Dem566] ⌈If more than one indicator is configured for a specific event, the Dem module shall use a logical OR operation of all combined warning indicators assigned to this event to calculate the UDS DTC status bit 7 (WarningIndicator).⌋()

[Dem500] ⌈ The Dem module shall provide the configuration parameter DemIndicatorFailureCycleCounterThreshold per indicator per event (refer to DemIndicatorAttribute) to define the maximum number of tested and failed cycles, before the respective indicator is activated (i.e. generates a specific condition WarningIndicatorOnCriteriaFulfilled, refer to Figure 20).⌋()

[Dem504] ⌈ The Dem module shall provide the configuration parameter DemIndicatorFailureCycleRef per indicator per event (refer to DemIndicatorAttribute) defining the indicator specific cycle, which represents the event or trigger, that causes an update of the failure counter provided there is failed result reported for the event.⌋()

[Dem501] ⌈ The Dem module shall generate the condition WarningIndicatorOnCriteriaFulfilled specific for the assigned warning indicator, if the respective indicator failure counter of the event entry has been processed (incremented or decremented) DemIndicatorFailureCycleCounterThreshold times.⌋()

[Dem568] ⌈ The Dem module shall provide the configuration parameter DemIndicatorFailureCycleSource (refer to DemIndicatorAttribute) to define which failure cycle (event or indicator based) is used for the WarningIndicatorOnCriteria handling.⌋()

Note: For emission related events the fault confirmation given by the DemEventClass also leads to the MIL on i.e. in that case, the IndicatorOnCriteria bases on the same failure cycle settings (trigger and threshold) of the event. Then the source would be DEM_FAILURE_CYCLE_EVENT and the indicator specific settings would be ignored (grayed out). If an additional indicator needs to be activated e.g. via time for this particular event, then the DemIndicatorSource would be DEM_FAILURE_CYCLE_INDICATOR. That means the trigger and the threshold are defined for this indicator.

[Dem502] ⌈ The Dem module shall provide the configuration parameter DemIndicatorHealingCycleCounterThreshold per indicator per event (refer to DemIndicatorAttribute) to define the maximum number of tested and passed healing cycles, before the respective indicator is deactivated (i.e. generates a specific condition WarningIndicatorOffCriteriaFulfilled, refer to Figure 20).⌋()

[Dem505] ⌈ The Dem module shall provide the configuration parameter DemIndicatorHealingCycleRef per indicator per event (refer to DemIndicatorAttribute) defining the indicator specific cycle, which represents the event or trigger, that causes an update of the healing counter provided there is OK / passed result reported for the EventId.⌋()

[Dem503] ⌈ The Dem module shall generate the condition WarningIndicatorOffCriteriaFulfilled specific for the assigned warning indicator, if the respective indicator healing counter of the event entry has been processed (incremented or decremented) DemIndicatorHealingCycleCounterThreshold times.⌋()

[Dem533] ⌈If more than one indicator is configured for a specific event and each assigned indicator healing counter has been processed (incremented or decremented) DemIndicatorHealingCycleCounterThreshold times, the Dem module shall reset the UDS DTC status bit 7 (WarningIndicatorRequested).⌋()

[Dem506] ⌈The Dem module shall process all indicator failure or indicator healing counters, if the cycle is started (refer to chapter 7.3.8) according to the current status of the stored events.⌋()

Note: This cycle type could be for example equal to DEM_OPCYC_OBD_DCY or to DEM_OPCYC_POWER (refer to DemOperationCycle).

It is possible to define a failure cycle or healing cycle, which does not correspond to an operation cycle in the sense, e.g. of a period of time with a beginning and an end. For that reason, the operation cycles can be extended with new types, e.g. DEM_OPCYC_TIME_200MS interpreted as triggers. In this example, the trigger is set every 200ms, i.e. is started every 200ms, to initiate the indicator counters to be processed.

7.3.10.2 Handling of the warning indicator lamp (MIL)

[Dem546] ⌈For OBD-relevant ECUs, the Dem module shall provide the configuration parameter DemMILIndicotorRef (refer to DemGeneral) to indicate that the configured indicator controls the MIL activation and deactivation.⌋()

[Dem544] ⌈If an indicator is configured for controlling the MIL of an OBD-relevant ECU, the transition of the UDS DTC status bit 3 (ConfirmedDTC) from 0 to 1 shall be activate the MIL.⌋()

[Dem567] ⌈If an indicator is configured for controlling the MIL of an OBD-relevant ECU, the Dem module shall use the configured event failure cycle counter of this event (refer to 7.3.4 Fault confirmation) to define the maximum number of tested and failed cycles, before the stored event activates the respective indicator.⌋()

Note: For OBD systems, the activation of the Malfunction Indicator Lamp (MIL) is linked with the entry to confirmed state. Therefore the configuration of the event specific fault confirmation counter (refer to the configuration parameter DemEventFailureCycleCounterThreshold and DemEventFailureCycleRef) is used instead of the indicator failure cycle counter.

Note: Leaving Pending state and the deactivation of the MIL is controlled by the configuration of the indicator healing cycle counter.

[Dem535] ⌈In case of OBD-relevant events the indicator cycles shall be based on cycles defined by OBD legislation.⌋()

7.3.10.3 Notification of the warning indicator status

[Dem046] ⌈The Dem module shall provide the API Dem_GetIndicatorStatus (refer to chapter 8.3.3.17) to inform a software component about the calculated indicator status.⌋()

Note: The Dem module derives the indicator status internally based on the event status as a summary of all assigned events. One indicator can be activated by several events and one event can also have more than one indicator assigned to it (refer to DemIndicatorAttribute).

[Dem508] ⌈If the warning indicator is not supported the Dem module shall report E_NOT_OK (refer to Dem_GetIndicatorStatus).⌋()

7.4 Startup behavior

[Dem169] ⌈The Dem module shall distinguish between a pre-initialization mode and a full-initialized mode (operation mode).⌋(BSW00406)

[Dem180] ⌈The function Dem_PreInit (refer to chapter 8.3.2.1) shall initialize the internal states of the Dem module necessary to process events reported by BSW modules by using Dem_ReportErrorStatus.⌋()

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

The function Dem_Init (refer to chapter 8.3.2.2) is called during the startup phase of the ECU, after the NVRAM Manager has finished the restore of NVRAM data. SW-Components including monitors are initialized afterwards. The function Dem_Init is also used to reinitialize the Dem module after the Dem_Shutdown was called.

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

[Dem368] ⌈After Dem_Shutdown (refer to chapter 8.3.2.3) has been called the Dem module shall ignore all function calls until Dem_Init is called again.⌋()

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

7.5 Monitor re-initialization

The primary initialization of the monitors in the application is done via Rte_Start. The initialization of the event-specific part of the monitor can be triggered by the Dem.

[Dem003] ⌈The Dem module shall provide the interface InitMonitorForEvent (refer to chapter 8.4.3.1.1) to trigger the initalization of a diagnostic monitor.⌋()

[Dem376] ⌈The Dem module shall trigger the callback function InitMonitorForEvent to initialize the monitor of a specific event (refer also to chapter 7.3.3 and Dem573) in case of starting or restarting the operation cycle of the event (DEM_INIT_MONITOR_RESTART), or in case of clearing the event via Dem_ClearDTC (DEM_INIT_MONITOR_CLEAR).⌋()

The function parameter InitMonitorReason indicates the reason / trigger of initialization.

Note: The Dem module does not evaluate the return value (e.g. if other than E_OK) of this callback function.

The link between the event and the corresponding initialization function is configured within the Dem module (refer to DemCallbackInitMForE in DemEventParameter).

The following figures show two examples:

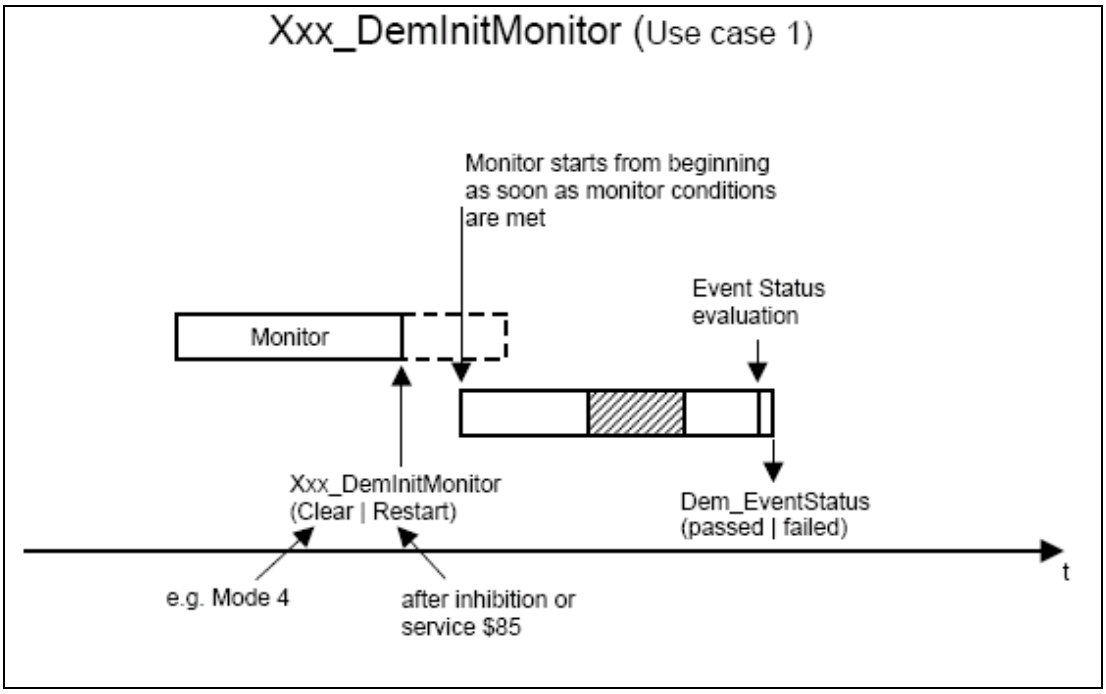

Figure 41 Use-case of the interface InitMonitorForEvent for a specific event

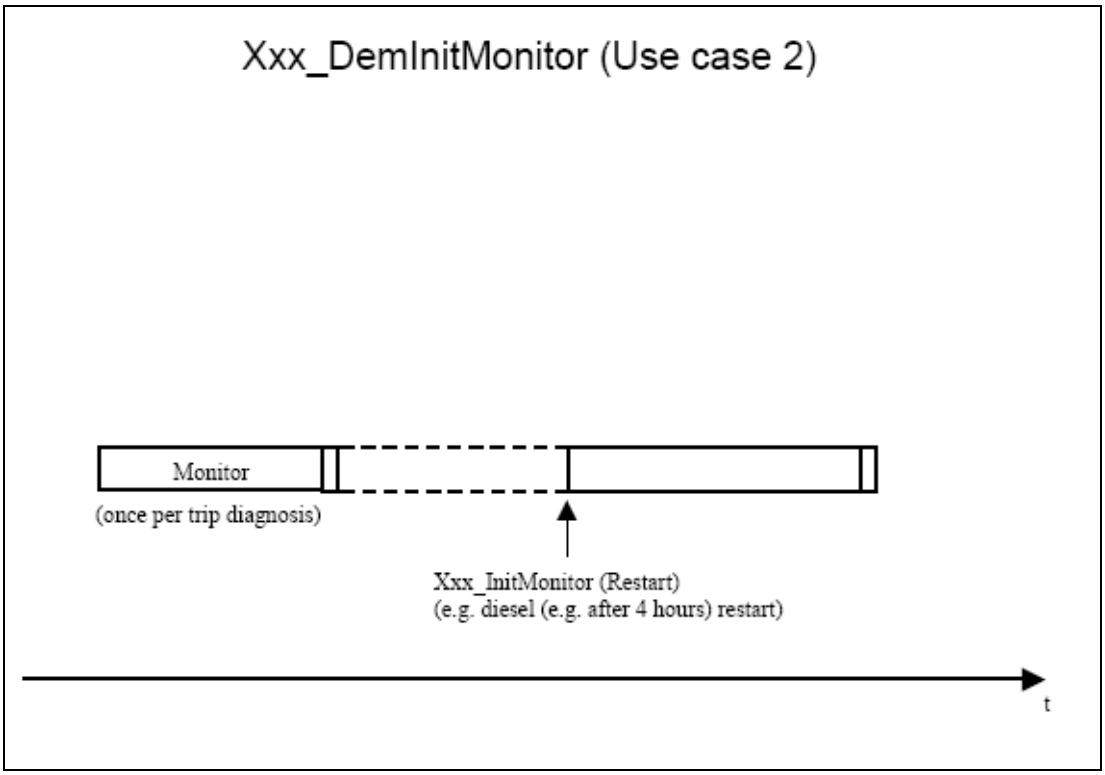

Figure 42 Use-case of the interface InitMonitorForEvent for a specific event

The initialization of any function (which may relate to the monitor) can also be triggered by the Dem.

[Dem335] ⌈The Dem module shall provide the interface InitMonitorForFunction (refer to 8.4.3.1.2) to trigger the initialization of an assigned functionality, which is not a monitor.⌋()

[Dem049] ⌈ The Dem module shall trigger the callback function InitMonitorForFunction to initialize the respective functionality of a specific SW-C or BSW module.⌋()

Note: The Dem module does not evaluate the return value (e.g. if other than E_OK) of this callback function.

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

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

7.6 BSW Error Handling

Beside application software components also the basic software (BSW) can detect errors (e.g. hardware driver faults), especially during startup (ref. to document [4] for further details). For these errors (only a small number compared to application specific events), some additional aspects apply:

- Errors can be detected before Dem is fully initialized
- Errors can be reported during startup, information needs to be buffered until Dem is fully available
- Errors can be reported between startup and shutdown, information needs to be buffered and need to be processed by the Dem main function (RTE related call tree requirement)
- Entries in the event memory can have a different configuration (e.g. no emphasis on freeze frame data for the workshop)

Since BSW events are treated as normal SW-C events in the event memory, they can also be classified (availability in workshop tester), aged (refer to chapter 7.3.9) and prioritized (displacement handling).

[Dem107] ⌈The Dem module shall provide the interface Dem_ReportErrorStatus (refer to chapter 8.3.3.1) to the BSW modules, to report BSW events which are processed like event reports by Dem_SetEventStatus.⌋(BSW00417, BSW00420, BSW00421)

[Dem167] ⌈The Dem module shall provide a buffer mechanism (FIFO) for the API Dem_ReportErrorStatus to queue events which are reported during startup (if the Dem module cannot access the event memory) and during normal operation (after the Dem module has been already initialized).⌋()

Dem_ReportErrorStatus is used by BSW modules to report errors from the point in time when the Dem module is pre-initialized. Within Dem_Init, the queued events are processed. During normal operation (after full initialization), the queuing mechanism of the API Dem_ReportErrorStatus is necessary to process the reported fault within the main function of the Dem module.

During initialization of the Dem module, the API Dem_ReportErrorStatus supports debouncing (pre-failed and pre-passed). The corresponding debounce counters will be initialized by calling Dem_PreInit.

[Dem207] ⌈The size of the queue of the function Dem_ReportErrorStatus is configurable by the configuration parameter DemBswErrorBufferSize (refer to DemGeneral).⌋()

Note: During startup phase, not all event-related data might be available.
SW-C events cannot be stored before complete initialization of the Dem.

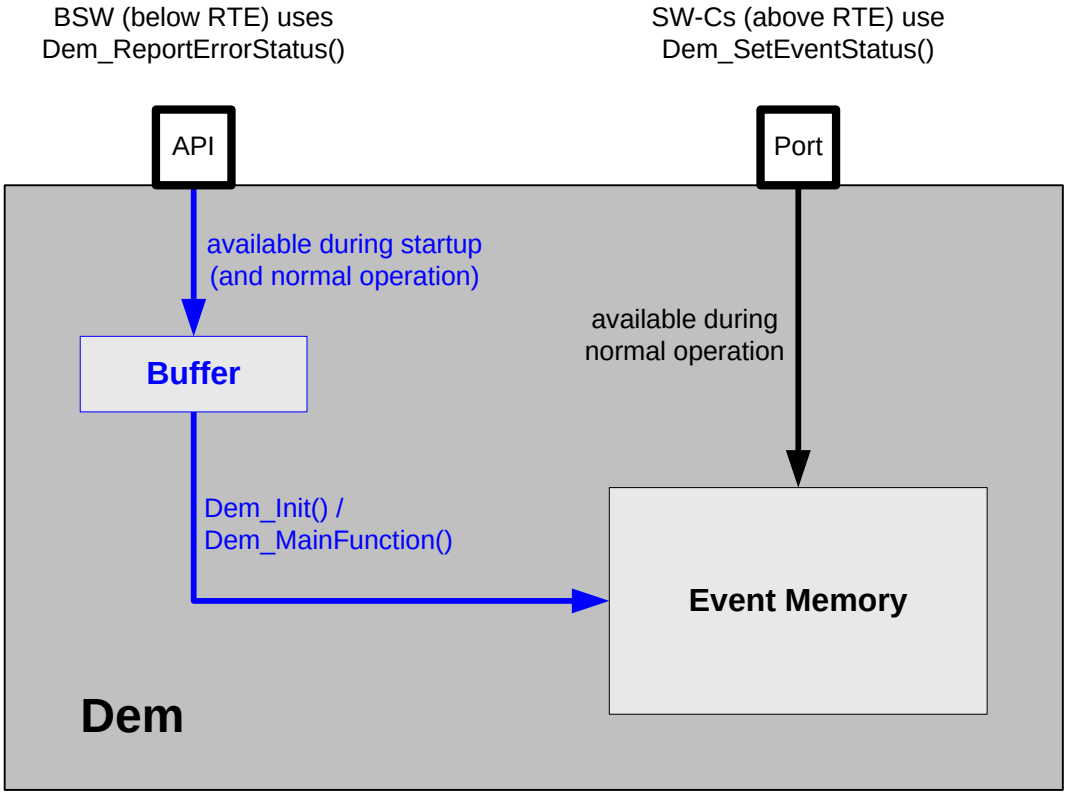

Figure 43 Dem_ReportErrorStatus buffering behavior

7.7 OBD-specific functionality

7.7.1 General overview and restrictions

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

Together with the Dcm, this Dem SWS provides standardized AUTOSAR interfaces to support the OBD services $01 - $0A defined in SAE J1979 Rev May 2007 [17]. With these services, Autosar OBD functionality shall be capable of meeting all light duty OBD regulations worldwide (California OBDII, EOBD, Japan OBD, and all others.)

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

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

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

In the following chapters, OBD relevant functionality and interfaces are described. It is important to note, that independent of standard Autosar mechanisms (e.g. communication via RTE), response codes and timing constraints need to fulfill OBD requirements (refer to [16] and [18]).

[Dem584] ⌈While applying standard mechanisms (like Std_ReturnType), the Dem module shall only return values ensuring OBD compliant behavior with regard to permitted response codes and timing constraints.⌋()

Note: The usage of E_NOT_OK as response value for API Dem_DcmReadDataOfPID<NN> and Dem_GetInfoTypeValue<08/0B> is not permitted.

7.7.1.1 Service $01, $02 and OBD PID data

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

[Dem291] ⌈The Dem module shall support only the legislative freeze frame (record number 0). This will be a single list of PIDs assigned to this freeze frame (refer to DemPIDClass).⌋()

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

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

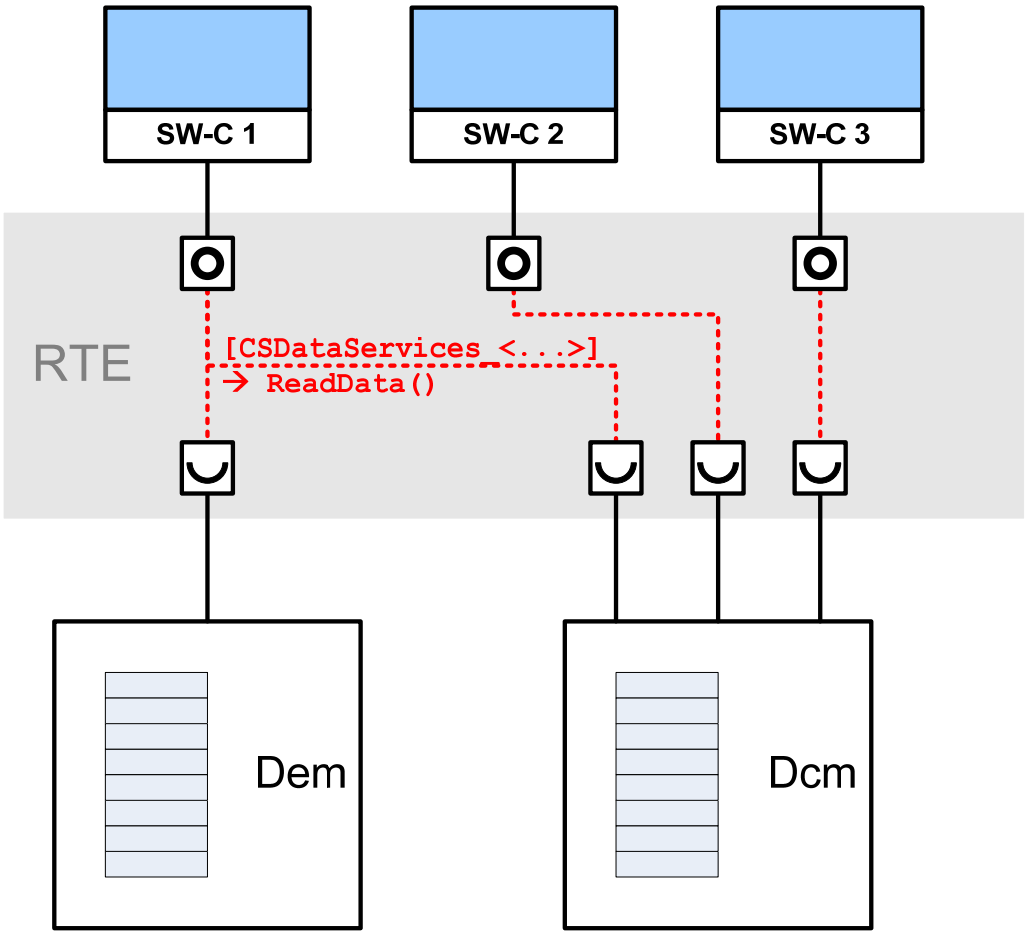

Figure 44 Dem and Dcm module requests PID data elements of SW-C via ReadData operation

[Dem596] ⌈The Dem module shall provide access on PID data elements of the most important freeze frame being selected for the output of service $02 (OBD freeze frame of the event which caused MIL on) to the Dcm module (refer to 8.3.6.15 Dem_ReadDataOfOBDFreezeFrame).⌋()

Note: The Dem processes and stores only the raw data for service $02. Any PID header-information and fill bytes are added by the Dcm. The Dcm configuration defines the individual PID layout, which also defines an order of the contained data elements. Each data element references to its linked element in the Dem configuration (refer to Figure 45).

The individual data elements of a PID are selected via an index by the Dcm.

[Dem597] ⌈The index values (refer to API parameter DataElementIndexOfPID) of Dem_ReadDataOfOBDFreezeFrame shall be assigned zero-based and consecutive. Their order shall be derived from the position of the data elements (refer to DcmDspPidDataPos, of the referencing DcmDspPidData containers) in the Dcm's PID layout.⌋()

Note: This index is not configured explicitly (to be able to avoid resource overhead for e.g. mapping tables in the implementation).

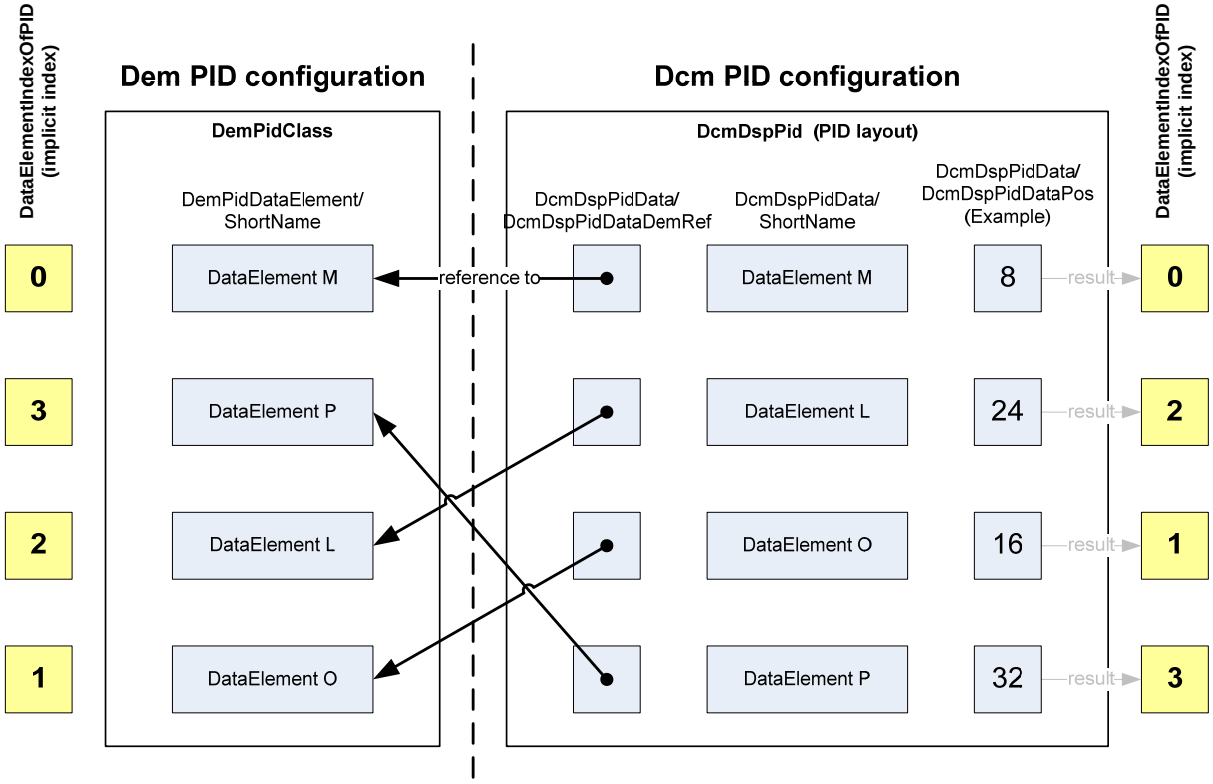

Figure 45 PID layout configuration within Dcm (master) and Dem

[Dem623] ⌈The function Dem_GetDTCOfOBDFreezeFrame (refer to chapter 8.3.6.16) shall return the DTC associated with the most important freeze frame being selected for the output of service $02 (PID $02).⌋()

7.7.1.2 PIDs for service $01 computed by Dem

[Dem293] ⌈There are some PIDs, which shall be computed directly within the Dem.

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

(1) PID $02 is only required for service $02 and therefore no interface (like Dem_DcmReadDataOfPID02) is necessary. Instead the API Dem_GetDTCOfOBDFreezeFrame is used (refer to Dem623).

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

[Dem346] ⌈The Dem module shall use PID $0D (refer to chapter 0) to calculate PID $21 and PID $31.⌋()

[Dem304] ⌈A Dem delivery shall provide function call interfaces to the Dcm and the respective ServiceNeeds to declare to the Dcm that these PIDs are supported.⌋()

[Dem347] ⌈If PID $1E (auxiliary input status) is supported the PTO (Power Take Off) related event status handling shall implemented inside the Dem module (refer to [22]).⌋()

[Dem377] ⌈The Dem module shall provide the interface Dem_SetPtoStatus (refer to chapter 8.3.6.17) allowing a SW-C implementing the PTO functionality to notify the Dem module if PTO is active or inactive (refer to section 8).⌋()

[Dem378] ⌈ The Dem module shall support the configuration parameter DemConsiderPtoStatus (refer to DemEventClass) indicating that a certain event is affected by the Dem PTO handling.⌋()

The Dem module provides the configuration switch DemPTOSupport (refer to Dem704_Conf) to enable or disable the usage of PID $1E.

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

[Dem349] ⌈ The Dem module shall support the configuration parameter DemEventOBDReadinessGroup (refer to DemEventClass) for OBD systems, to assign individual events to one specific readiness group.⌋()

According to SAEJ1979 [18], the group AirCondition Component shall not be supported anymore. However, it is still included in ISO 15031-5 [16].
The groups distinguish between spark ignition engines (spark) and compression ignition engines (compr.). However, it is necessary to configure per event, to which readiness group the monitor contributes (if at all).

[Dem351] ⌈The Dem module shall compute and provide the number of confirmed faults (PID $01, Byte A).⌋()

[Dem352] ⌈The Dem module shall provide the MIL status.⌋()

[Dem354] ⌈The Dem module shall compute the readiness status (if all events of a group are reported as OK tested since last clear, or the event has caused MIL on).⌋()

[Dem355] ⌈The Dem module shall compute the readiness group complete for current driving cycle (if all events of a group are tested in the current driving cycle).⌋()

Note: For calculation of the group readiness, the Dem module considers all events assigned to the specific readiness group.

[Dem356] ⌈The Dem module shall compute the readiness group disabled (if the disabled status is reported by the monitor for any event of a group).⌋()

[Dem348] ⌈The Dem module shall provide the disabling of events (refer to 8.3.6.1 Dem_SetEventDisabled).⌋()

[Dem294] ⌈In order to allow a monitor to report that the event cannot be computed in the driving cycle (aborted e.g. due to physical reasons), the Dem shall provide the API Dem_SetEventDisabled.⌋()

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

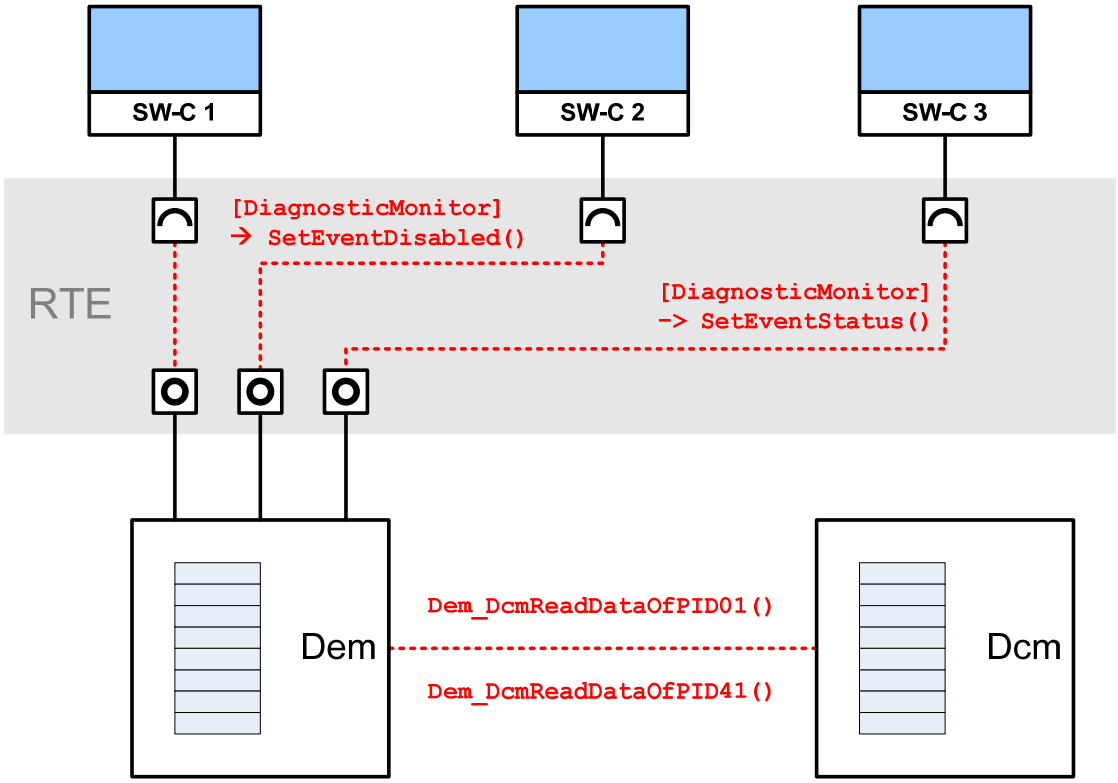

Figure 46 Dem calculates PID$01 and PID$41 data based on specific port operations

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

The IUMPR-data collected need to be provided upon a service $09 request. For gasoline engines, the Info Type $08 is used and for diesel engines the Info Type $0B is used (refer to [16] and [18]).

[Dem298] ⌈In order to support the data requests in service $09 as described above, the Dem shall provide the API Dem_GetInfoTypeValue08 or Dem_GetInfoTypeValue0B to the Dcm.⌋()

[Dem357] ⌈If the OBD system is a spark ignition system the Dem module shall provide the API Dem_GetInfoTypeValue08 (refer to chapter 8.3.6.5) for Info Type $08 IUMPR data.⌋()

The service Dem_GetInfoTypeValue08 shall be used by the Dcm, to request the IUMPR-data according to the InfoType $08 data format used for the output to service $09.

[Dem358] ⌈If the OBD system is a compression ignition system the Dem module shall provide the API Dem_GetInfoTypeValue0B (refer to chapter 8.3.6.6) for Info Type $0B IUMPR data.⌋()

The service Dem_GetInfoTypeValue0B shall be used by the Dcm, to request the IUMPR-data according to the InfoType $0B data format used for the output to service $09.

The type of OBD system is defined by using the configuration switch DemOBDSupport (refer to DemGeneral).

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

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

Typically, one or more events serve for the monitoring of these components, e.g. the oxygen sensor. Hence, in order to record the in-use performance of the OBD-system, the test performance of all the relevant events for all the IUMPR-groups needs to be recorded. For that purpose, certain data structures need to be configured by the Dem.

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

This leads to a combination of an event to be recorded and a primary FID (and optionally multiple secondary FIDs) representing all the events stopping the computation of the IUMPR-event. For the purpose of classifying the events into groups, an IUMPR-group is also part of this combination.

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

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

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

[Dem359] ⌈Only for monitors being configured with the option “observer”, the Dem module shall increment the numerator of the corresponding monitor, if the assigned event gets tested/qualified (as passed or failed).⌋()

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

[Dem360] ⌈For OBD relevant systems the Dem module shall provide the API Dem_RepIUMPRFaultDetect (refer to chapter 8.3.6.2).⌋()

[Dem296] ⌈The Dem module shall provide the API Dem_RepIUMPRFaultDetect (refer to chapter 8.3.6.2) for the asymmetric monitor to report that a malfunction could have been found.⌋()

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

[Dem361] ⌈The Dem module shall provide the configuration parameter DemRatioIdType (refer to DemRatioId), to indicate per RatioId if the numerator is calculated based on the TESTED-status or the API call.⌋()

For some particular monitors (e.g. for secondary air system, comprehensive components), there are additional conditions defined on the denominator: Their denominator will only be incremented if certain temperature or activity conditions are met. Then, the monitor needs to lock the denominator per API at the beginning of the driving cycle and will unlock it when the additional conditions on the denominator are met.

[Dem362] ⌈The Dem module shall provide the APIs for locking (refer to 8.3.6.3 Dem_RepIUMPRDenLock) and unlocking (refer to 8.3.6.4 Dem_RepIUMPRDenRelease) of the denominator under special conditions.⌋()

[Dem297] ⌈The Dem shall provide the API Dem_RepIUMPRDenLock (refer to chapter 8.3.6.3) to IUMPR-relevant SW-C, to control the denominator specific to the respective RatioId.⌋()

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

[Dem308] ⌈The Dem shall provide the API Dem_RepIUMPRDenRelease (refer to chapter 8.3.6.4) to IUMPR-relevant SW-C, to control the denominator specific to the respective RatioId.⌋()

Note: This service shall be used by the monitor to report, that a denominator of a specific monitor (represented by FID and EventId) is released for physical reasons.

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

[Dem299] ⌈As long as an event has Pending status, the Dem module shall stop increasing the numerator and denominator.⌋()

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

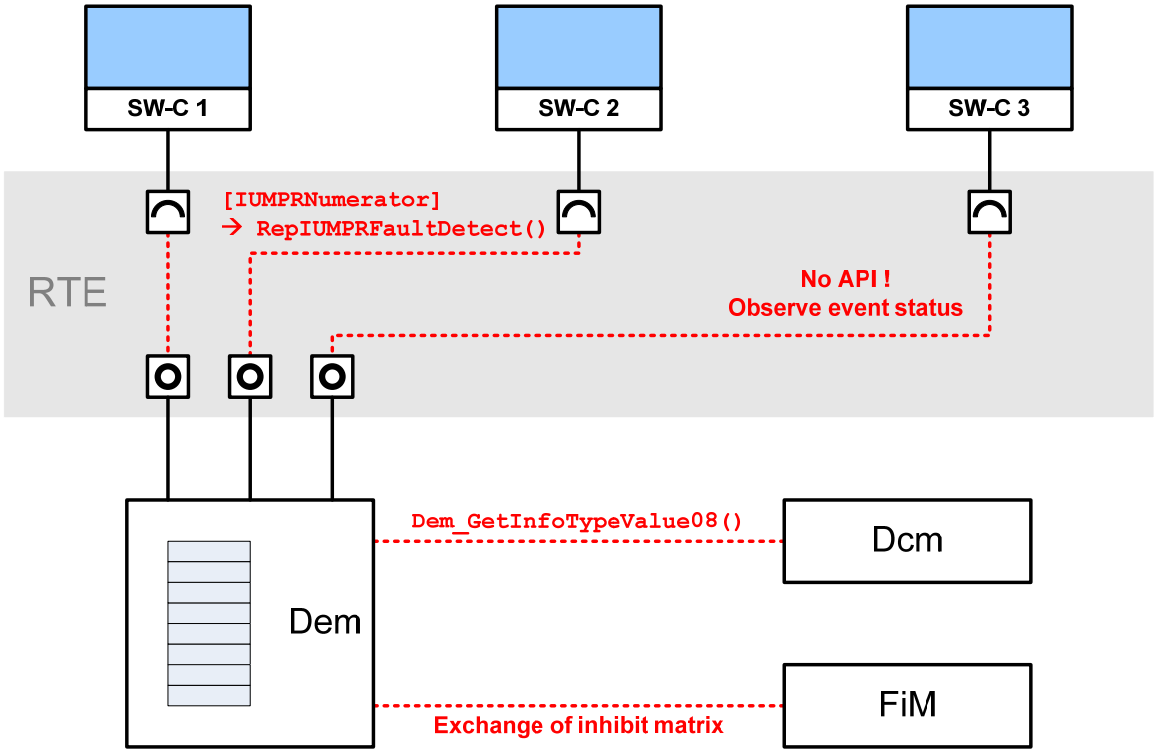

Figure 47 Dem calculates IUMPR data based on specific interface operations

7.7.1.4 Service $0A – Permanent DTC

[Dem300] ⌈The Dem shall be able to handle Permanent DTCs according to regulations (refer to [19]) within the specific event memory type “permanent” (refer to chapter 7.1.5).⌋()

[Dem590] ⌈Events that have been confirmed and activate the MIL (refer to OBD-relevant DTCs, chapter 7.2), shall be stored robustly against ClearDTC or powerfail (for details confer [19]).⌋()

[Dem301] ⌈The Dem shall provide the ability to access the stored Permanent DTC by filtering for permanent DTCs to the Dcm (refer to 8.3.4.1.1 Dem_SetDTCFilter and 8.3.4.1.6 Dem_GetNextFilteredDTC).⌋()

Note: For service $0A, the Dcm uses the DTCOrigin DEM_DTC_ORIGIN_PERMANENT_MEMORY.

7.8 Interaction with other Software Modules

7.8.1 Interaction with Software Components (SW-C)

In the AUTOSAR ECU architecture, the Diagnostic Event Manager implements an AUTOSAR Service. This means that the Dem module does not exclusively communicate with other BSW modules, but also with SW-C via the RTE (refer to Figure 1).

From the viewpoint of the BSW “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 (by accessing the Dem Service Component).
- The application is optionally notified upon the outcome of requested asynchronous activity (via direct/indirect RTE API 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 Service Component Description of the Dem Service.

The service component description of the Dem Service will define the ports below the RTE. Each SW-C, which uses the Dem Service, must contain respective ports in its own SW-C description, which will be typed by the same (or compatible) interfaces and must be connected to the ports of the Dem, so that the RTE can be generated.
Ids used in these Dem-ports are abstracted with port-defined arguments.

[Dem512] ⌈The name of the Dem Service Component shall be “Dem”.⌋()

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

In contrast to that, there are some special SW-Cs, that need to handle event status/data changes uniformly. These SW-Cs are notified by the Dem, which provides then also the EventId for the application, to be able, to request the respective data from the Dem.
Therefore, general interfaces are provided (refer to chapters 8.3.3.23, 8.4.3.4, and 8.4.3.6), which shall only be used in the way, that the EventIds (given by the Dem) shall not be interpreted by SW-Cs in any way, as well as the EventId values (used by the SW-Cs to request the event-specific data) have always to be provided by the Dem. Here only a hand-through mechanism is allowed to be used by SW-Cs for EventIds.

In the current Autosar release, the RTE analyzes the complete call-tree of triggered runnables. Therefore, some limitations in the Dem API are introduced to avoid function calls from BSW modules, which rely on callbacks to SW-Cs via RTE.

7.8.2 Interaction with Diagnostic Communication Manager (Dcm)

The Dcm accesses the Dem module, in order to delegate those diagnostic services, which are related to the event memory.

A detailed 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 [7]. There, especially the handling of freeze frame data is described.

[Dem171] 「If the Dem module's environment has requested an unavailable event memory/origin, the Dem module functions with the return code Dem_ReturnGetStatusOfDTCType shall return the value WRONG_DTCORIGIN.」()

[Dem172] 「If the Dem module's environment has requested a DTC that is available but has a different origin than the requested one, the Dem module functions with the return code Dem_ReturnGetStatusOfDTCType shall return the value WRONG_DTC.」()

[Dem565] 「If the Dem module informs the Dcm module (refer to [7]) about a status change of a DTC (used for the ROE service OnDTCStatusChanged handling), then the Dem module shall use the function Dcm_DemTriggerOnDTCStatus.」()

Note: The configuration parameter DemTriggerDcmReports enables this function. The function Dcm_DemTriggerOnDTCStatus uses the same prototype as <Module>_DemTriggerOnDTCStatus (refer to chapter 8.4.3.1.4). This configuration parameter is for simplicity (so the generic parameter DemCallbackDTCStatusChangedFnc needs not to be used).

7.8.2.1 Access DTCs and Status Information

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

[Dem231] ⌈The function Dem_GetTranslationType (refer to chapter 8.3.4.1.11) shall provide the capability to get the configured translation format of the ECU (refer to DemTypeOfDTCSupported in DemGeneral).⌋()

[Dem014] ⌈The Dem module shall be able to return the DTC status availability mask (refer to 8.3.4.1.4 Dem_GetDTCStatusAvailabilityMask) in accordance to ISO 14229-1 [15].⌋()

[Dem059] ⌈The function Dem_GetStatusOfDTC (refer to chapter 8.3.4.1.3) shall copy the status of a DTC to the API parameter DTCStatus according to ISO 14229-1 [15].⌋()

It is possible, that a DTC with different states exists depending on the location. If the secondary memory is used as a kind of protocol stack that gives information which services have been performed on the primary memory, different DTC states might appear (e.g. DTC has been deleted from the primary memory and is written to the secondary memory with its latest status).

[Dem060] ⌈The function Dem_GetDTCStatusAvailabilityMask shall provide the DTC status availability mask, that means the DTC status information (according to ISO 14229-1 [15]) supported by the Dem module (refer to DemDtcStatusAvailabilityMask in DemGeneral).⌋()

[Dem057] ⌈The function Dem_SetDTCFilter (refer to chapter 8.3.4.1.1) shall set the filter mask attributes to be used for the sub-sequent calls of Dem_GetNumberOfFilteredDTC, Dem_GetNextFilteredDTC, Dem_GetNextFilteredDTCAndFDC, as well as Dem_GetNextFilteredDTCAndSeverity, and reset an internal counter to the first event. The filter mask attributes shall be used, until the next call of Dem_SetDTCFilter or Dem initialization.⌋(BSW04057)

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

[Dem061] ⌈The function Dem_GetNumberOfFilteredDTC shall get the number of DTCs matching the defined filter mask, which is configured with Dem_SetDTCFilter (refer to Dem057).⌋()

Dem_SetDTCFilter has to be called prior to Dem_GetNumberOfFilteredDTC.

[Dem216] ⌈The function Dem_GetNextFilteredDTC shall return the current DTC and its associated status from the Dem module matching the filter criteria defined by the function call of Dem_SetDTCFilter.⌋()

[Dem217] ⌈The function Dem_GetNextFilteredDTC shall skip to the next DTC matching the filter criteria, after having returned the requested data.⌋()

The Dcm calls the function Dem_GetNextFilteredDTC continuously until the return value of the function is DEM_FILTERED_NO_MATCHING_DTC, to receive all DTCs matching the filter criteria.

[Dem228] ⌈The function Dem_GetNextFilteredDTCAndFDC shall return the current DTC and its associated fault detection counter (FDC) from the Dem matching the filter criteria defined by the function call Dem_SetDTCFilter.⌋()

[Dem229] ⌈The function Dem_GetNextFilteredDTCAndFDC shall skip to the next DTC matching the filter criteria, after having returned the requested data.⌋()

[Dem513] ⌈If the callback-function GetFaultDetectionCounter (refer to chapter 8.4.3.1.8) returns other then E_OK for a DTC filtered by the function Dem_GetNextFilteredDTCAndFDC, this DTC shall not match the filter criteria (i.e. assume FDC is 0).⌋()

The Dcm calls the function Dem_GetNextFilteredDTCAndFDC continuously until the return value of the function is DEM_FILTERED_NO_MATCHING_DTC, to receive all DTCs matching the filter criteria.

Note: Non-Dem-internal calculated fault detection counters are typically requested from SW-Cs through the RTE. To indicate an equivalent call-tree for these runnables, a work-around is used: The Dcm main function specifies a trigger to the Dem interface GeneralEvtInfo (operation GetFaultDetectionCounter), which triggers the respective runnable (refer to RunnableEntity GetFaultDetectionCounter, chapter 8.3.3.23).

[Dem287] ⌈The function Dem_GetNextFilteredDTCAndSeverity shall return the

current DTC and its associated fault severity from the Dem matching the filter criteria defined by the function call Dem_SetDTCFilter.」()

[Dem288] 「The function Dem_GetNextFilteredDTCAndSeverity shall skip to the next DTC matching the filter criteria, after having returned the requested data.」()

The Dcm calls the function Dem_GetNextFilteredDTCAndSeverity continuously until the return value of the function is DEM_FILTERED_NO_MATCHING_DTC, to receive all DTCs matching the filter criteria.

For information about the functions Dem_GetSeverityOfDTC and Dem_GetFunctionalUnitOfDTC, refer to chapter 7.1.6.

[Dem595] 「 The function Dem_SetFreezeFrameRecordFilter (refer to chapter 8.3.4.1.2) shall set the static filter mask attribute “all freeze frame records currently stored in the event memory” to be used for the sub-sequent calls of Dem_GetNextFilteredRecord, and reset an internal counter to the first freeze frame record.」()

[Dem210] 「 The function Dem_SetFreezeFrameRecordFilter shall retrieve the number of filtered freeze frame records. This filter always belongs to primary memory.」()

[Dem225] 「The function Dem_GetNextFilteredRecord (refer to chapter 8.3.4.1.8) shall return the current DTC and its associated relative addressed freeze frame record numbers from the Dem module matching the filter criteria defined by the function call Dem_SetFreezeFrameRecordFilter.」()

[Dem226] 「The function Dem_GetNextFilteredRecord shall skip to the next freeze frame record matching the filter criteria, after having returned the requested data.」()

The Dcm calls the function Dem_GetNextFilteredRecord continuously until the return value of the function is DEM_FILTERED_NO_MATCHING_DTC, to receive all records matching the filter criteria.

[Dem219] 「 The function Dem_GetDTCByOccurrenceTime (refer to chapter 8.3.4.1.7) shall provide the capability to get one DTC stored in the primary event memory according to the API parameter DTCRequest, which specifies the relevant occurrence time.」(BSW04070)

[Dem221] 「 The function Dem_GetDTCByOccurrenceTime shall return DEM_OCCURR_NOT_AVAILABLE, if no DTC is matching the requested occurrence time (DTCRequest).」(BSW04070)

7.8.2.2 Access event related data

This section defines the APIs, to get access to the event related data stored with the DTCs in the event memory of the Dem via diagnostics. Furthermore, access to OBD-relevant PIDs stored in a freeze frame is made available. Details concerning freeze frame handling can be found in ISO 14229-1 and ISO 15031-5.

The freeze frames can be addressed either using absolute numbers or relative numbers. If absolute addressing is used (emission relevant ECUs) a unique record number for a freeze frame exists throughout the whole ECU (refer to Dem_GetFreezeFrameDataByRecord). In case of relative addressing, the record number of the freeze frames are unique to a DTC (refer to Dem_GetFreezeFrameDataByDTC).

[Dem574]「The Dem module shall support global record numbers (unique per ECU) or relative record numbers (unique per DTC) based on the respective configuration.」()

[Dem070] 「The function Dem_GetFreezeFrameDataByRecord (refer to chapter 8.3.4.2.3) shall return the DTC associated with the freeze frame selected via its absolute record number.」(BSW04074)

Note: This is currently limited to the absolute record number 0x00 only (refer to chapter 4.1), also from Dcm side.

[Dem575] 「 If the API parameter RecordNumber of Dem_GetFreezeFrameDataByRecord is set to 0x00, the Dem module shall respond the complete OBD freeze frame.」()

Note: The record number used in the API Dem_GetFreezeFrameDataByRecord is unique throughout the whole ECU (refer to ISO 14229-1 [15]). The value 0x00 is only available, if the Dem module supports OBD (refer to DemOBDSupport).

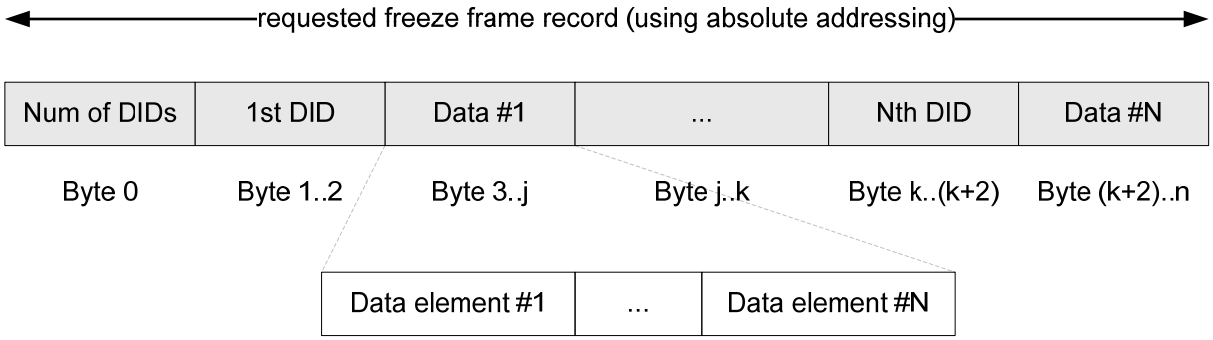

Figure 48 Buffer format used from Dem_GetFreezeFrameDataByRecord

[Dem071] ⌈ The function Dem_GetFreezeFrameDataByDTC (refer to chapter 8.3.4.2.4) shall copy the specific data identifiers (DIDs) of the requested freeze frame record to the destination buffer (DestBuffer). The function shall transmit these data as a complete record with the format shown in Figure 49.⌋(BSW04074)

[Dem576] ⌈ If the API parameter RecordNumber of Dem_GetFreezeFrameDataByDTC is set to 0x00, the Dem module shall respond the existing event/DTC-specific OBD freeze frame.⌋()

Note: The data returned includes the DTCSnapshotRecord (RecordNum) and DTCSnapshotNumberOfIdentifiers (Num of DIDs) as defined by ISO 14229-1 for the response message to service 0x19 sub-function 0x05 or sub-function 0x04.

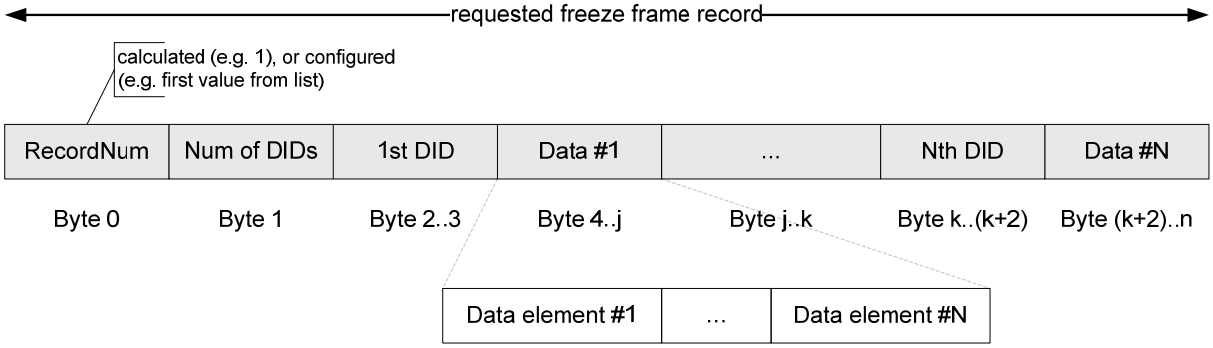

Figure 49 Buffer format used from Dem_GetFreezeFrameDataByDTC

Note: In contradiction to the Dcm PID-structure handling, the DID-structure is handled within the Dem module. Therefore, the Dem returns already full freeze frame records according to the response layout. This results in an optimized Dem/Dcm interface.

[Dem630] ⌈If Dem_GetFreezeFrameDataByDTC is called with a valid DTC and a valid freeze frame record number which is not stored, the Dem shall return DEM_GET_FFDATABYDTC_OK and BufSize 0 (empty buffer).⌋(BSW04074)

[Dem074] ⌈The function Dem_GetSizeOfFreezeFrameByDTC (refer to chapter 8.3.4.2.5) shall return the size of the requested freeze frame record, which represents the number of user data bytes including any freeze frame header-information (according to the format defined in Dem071).⌋(BSW04074, BSW04079)

Note: If the record number value 0xFF is requested, the Dem considers the size of all stored freeze frame records according to Dem074.

[Dem075] ⌈The function Dem_GetExtendedDataRecordByDTC (refer to chapter 8.3.4.2.6) shall copy the complete data of the requested extended data record to the destination buffer (DestBuffer). The function shall transmit these data as a complete record with the format shown in Figure 50.⌋(BSW04074)

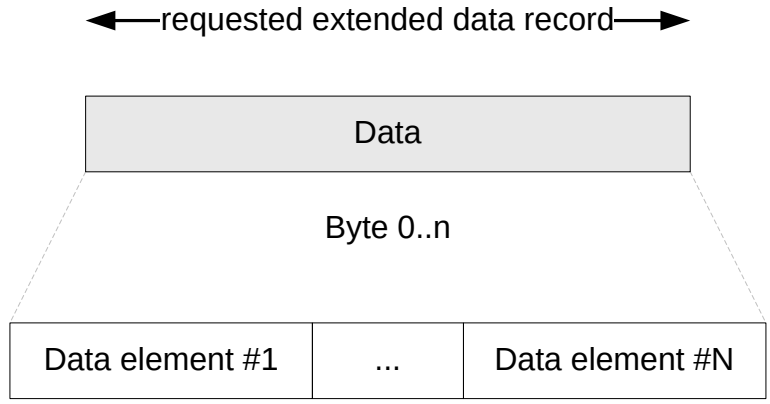

Figure 50 Buffer format used from Dem_GetExtendedDataRecordByDTC

[Dem631] ⌈If Dem_GetExtendedDataRecordByDTC is called with a valid DTC and a valid extended data record number which is not stored, the Dem shall return DEM_RECORD_OK and BufSize 0 (empty buffer).⌋(BSW04074)

[Dem076] ⌈The function Dem_GetSizeOfExtendedDataRecordByDTC (refer to chapter 8.3.4.2.7) shall return the size of the requested extended data record, which represents the number of user data bytes stored in the extended data record (according to the format defined in Dem075).⌋(BSW04074)

Note: If the record number value 0xFE is requested, the Dem considers the size of all OBD stored extended data records in the range of 0x90 to 0xEF according to Dem076.

Note: If the record number value 0xFF is requested, the Dem considers the size of all stored extended data records (in the range of 0x01 to 0xEF) according to Dem076.

The Dcm uses the function Dem_DisableDTCRecordUpdate, if the freeze frame or extended data of a specific DTC is about to be accessed by subsequent API-calls.

This is done to ensure, that the requested data of this DTC are not changed while the Dcm accesses those by several API-calls.

[Dem270]「The function Dem_DisableDTCRecordUpdate (refer to chapter 8.3.4.2.1) shall protect the event related data of the specified DTC within the specified origin from updating or deleting, to allow a consistent read for the following subsequent API-calls:

- Dem_GetSizeOfFreezeFrameByDTC and Dem_GetFreezeFrameDataByDTC
- Dem_GetSizeOfExtendedDataRecordByDTC and Dem_GetExtendedDataRecordByDTC.」()

New and other events including their associated freeze frames and extended data records can still be added to and changed in the event memory as long as space is available.

Note: Event related data might still be updated in background (e.g. Dem-internal data elements).

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

[Dem271]「The function Dem_EnableDTCRecordUpdate (refer to chapter 8.3.4.2.2) shall release the currently disabled DTC which has been protected by the function Dem_DisableDTCRecordUpdate, so that the data can be updated again.」()

The function Dem_EnableDTCRecordUpdate is the counterpart to the function Dem_DisableDTCRecordUpdate.

The Dcm calls the function Dem_EnableDTCRecordUpdate after the freeze frame and extended data of the specific DTC were protected by the function Dem_DisableDTCRecordUpdate, after the access by subsequent API-calls is finished.

[Dem648]「If development error detection is enabled and the function Dem_DisableDTCRecordUpdate is called while another DTC is locked, the Dem module shall set the error code DEM_E_WRONG_CONDITION (refer also to Dem370).」()

7.8.2.3 Clear diagnostic information

[Dem009]「The Dem module shall provide the API Dem_ClearDTC (refer to chapter 8.3.4.3.1) to the Dcm for deleting single DTCs, DTC groups, and all DTC from the event memory. This shall trigger also initializing of related SW-Cs / BSW modules according to Dem003.」(BSW04065)

The service ClearDiagnosticInformation (14 hex) of ISO 14229-1 [15] defines and covers the required actions and the deletion of related memory areas like freeze frames. One DTC value is transmitted, which can represent either a single DTC, or a group of DTCs. The groups are defined in ISO 14229-1, Annex D1.

Note: Dem_ClearDTC typically triggers further callbacks through the RTE. To indicate the respective call-tree for these runnables, a work-around is used: The Dcm triggers the DTC deletion using the Dem interface DcmIf (operation ClearDTC, refer to chapter 8.3.4.5) instead of a direct C call.

[Dem077] ⌈The function Dem_ClearDTC shall clear the status of all event(s) related to the specified DTC(s), as well as all associated event memory entries for these event(s).⌋(BSW04065)

Note: Exceptions may apply due to Dem514.

[Dem569] ⌈ The Dem module shall provide a configuration parameter DemClearDTCBehavior (refer to DemGeneral) to define the clearing process of diagnostic information concerning volatile and non-volatile memory and the positive response handling for the Dcm module.⌋(BSW04065)

Note: The interaction with the NvRAM Manager is described in chapter 7.8.4. Due to the hardware design in some cases, the direct access to the non-volatile memory is only possible during shutdown phase.

[Dem570] ⌈If the Dem module is requested to clear diagnostic information and the configuration parameter DemClearDTCBehavior is set to DEM_CLRRESP_VOLATILE, the Dem module shall return DEM_CLEAR_OK (refer to 8.2.2.5 Dem_ReturnClearDTCType) after the volatile memory is cleared. ⌋ (BSW04065)

[Dem571] ⌈If the Dem module is requested to clear diagnostic information and the configuration parameter DemClearDTCBehavior is set to DEM_CLRRESP_NONVOLATILE_TRIGGER, the Dem module shall return DEM_CLEAR_OK (refer to 8.2.2.5 Dem_ReturnClearDTCType) after the volatile memory is cleared and clearing of the non-volatile memory is triggered.⌋(BSW04065)

[Dem572] ⌈If the Dem module is requested to clear diagnostic information and the configuration parameter DemClearDTCBehavior is set to DEM_CLRRESP_NONVOLATILE_FINISH, the Dem module shall return DEM_CLEAR_OK (refer to 8.2.2.5 Dem_ReturnClearDTCType) after the volatile memory and the non-volatile memory is cleared.⌋(BSW04065)

Note: If the Dcm module receives the return type DEM_CLEAR_OK of the API Dem_ClearDTC, the Dcm module sends the positive response (refer to [7]).

[Dem573] ⌈ The Dem module shall provide the configuration parameter DemTriggerMonitorInitBeforeClearOk (refer to DemGeneral) to configure, if the function InitMonitorForEvent has to be called before or after the Dem returns DEM_CLEAR_OK.⌋(BSW04065)

[Dem629] ⌈ If clear diagnostic information is in progress and Dem_DcmCancelOperation is received, new Dem_ClearDTC calls shall henceforth return DEM_CLEAR_FAILED until the currently running clear operation is finished.⌋ (BSW04065)

Rationale: Clearing the error memory involves asynchronous writing to the NvM and calling of multiple callbacks, which cannot be canceled for physical reasons or without risks for data consistency.

Note: This ensures that Dem_ClearDTC can react to calls from different services in a safe way (e.g. preemptive protocol changes from ISO14229 service $14 to ISO15031 mode $04).

[Dem514] ⌈ The Dem module shall be able to suppress the DTC-deletion in Dem_ClearDTC. The callback ClearEventAllowed (refer to chapter 8.4.3.1.6), which is configurable/optional per event, shall realize this suppression handling. ⌋ (BSW04092)

Note: Event displacement (mainly controlled by the event priority) and aging are not influenced by this callback.

This functionality is intended for some special events, which must never be cleared from event memory within specific states, like for example special operation modes of the ECU (e.g. assembly-, transport-, or flash-mode) or Dcm sessions (which can be determined, using API Dcm_GetSesCtrType).

It is configurable per event (refer to DemCallbackClearEventAllowed), whether a SW-C / BSW module shall be requested about the event-deletion allowance from this callback function. One callback (representing a specific condition) can be assigned per event. If multiple conditions apply, they need to be summarized by one callback individually.

[Dem515] ⌈The Dem module shall call the callback ClearEventAllowed for each event it is configured for, before clearing this event (via Dem_ClearDTC). If the out-parameter ‘Allowed’ returns “false” (and the return value is equal to E_OK) after the execution of this callback function, the respective event must not be cleared (and the event status shall not be modified).⌋(BSW04092)

[Dem516] ⌈If the return value of the callback ClearEventAllowed is not equal to E_OK, the event-deletion is allowed anyway (because of safety aspects).⌋ (BSW04092)

7.8.2.4 Control DTC setting

[Dem035] ⌈The Dem module shall be capable of disabling (refer to 8.3.4.3.2 Dem_DisableDTCSetting) and enabling (refer to 8.3.4.3.3 Dem_EnableDTCSetting) the DTC setting of a specific DTC group.⌋()

[Dem626] ⌈When DTC setting is disabled, all status reports from SW-Cs (refer to 8.3.3.2 Dem_SetEventStatus) and BSW modules (refer to 8.3.3.1 Dem_ReportErrorStatus) for those events being assigned to this specific DTC group shall be ignored (no change of UDS DTC status byte) by the Dem.⌋()

Note: This is similar like the enable condition handling (refer to Dem449).

[Dem079] ⌈The function Dem_DisableDTCSetting shall disable the storage of all events assigned to specific a DTC group in the event memory including the respective UDS DTC status byte updates.⌋()

The function Dem_DisableDTCSetting is used in case of an induced failure situation in a system, e.g. during flash-reprogramming of one ECU in a network. In that case all the ECUs are commanded via diagnostic request (forwarded from Dcm by using Dem_DisableDTCSetting / Dem_EnableDTCSetting) to ignore DTC reports, as the flashed ECU is not participating in the normal communication anymore.
Note: If one of the other networked ECUs needs one of the signals, which are now missing, for this case a failsafe-reaction of the ECU cannot be assigned to the UDS DTC status byte updates, as these are also suppressed during disabled DTC setting.

[Dem080] ⌈The function Dem_EnableDTCSetting shall enable the storage of all events assigned to specific a DTC group in the event memory including the respective UDS DTC status byte updates.⌋()

7.8.2.5 Asynchronous Dcm operations

Most of the APIs of the Dem/Dcm interface may depend on NVRAM data. Therefore, an asynchronous processing of these APIs can be realized by using the additional “PENDING” value of the respective API return types (refer to chapter 8.2.2).

[Dem547] ⌈If asynchronous processing of the Dem/Dcm interface is used, the Dem module shall provide cancelation functionality for pending Dcm operations (refer to 8.3.4.4 Dem_DcmCancelOperation).⌋()

7.8.3 Interaction with Function Inhibition Manager (FiM)

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

The Dem contribution to the above functionality is to provide event status information to the FiM.

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

[Dem029] ⌈If DemTriggerFimReports (refer to DemGeneral) is enabled, the Dem module shall notify the FiM module (refer to [10]) on each event status change of all events (refer also to Dem016), by calling the function FiM_DemTriggerOnEventStatus.⌋(BSW04031)

Note: The function FiM_DemTriggerOnEventStatus is equal to <Module>_DemTriggerOnEventStatus (refer to chapter 8.4.3.1.3). This configuration parameter is for simplicity to avoid a reference of the function to each individual EventId.

The Dem module provides the function Dem_GetEventStatus for possible plausibility checks of the FiM, re-building, startup etc. of inhibition relations by the FiM.

7.8.4 Interaction with NVRAM Manager (NvM)

Typically, the Dem module uses non-volatile memory blocks (configurable in size by the NVRAM Manager [6]) to achieve permanent storage of event status information and event related data (e.g. retrieve status at startup).
Each non-volatile memory block used from the Dem, needs also to be configured (refer to DemNvRamBlockId).
The number, the type, and the content of the used non-volatile memory blocks are not prescribed. These shall be handled implementation-specific.
The NvM usage can also be deactivated by configuration (refer to multiplicity of DemNvRamBlockId), so that the Dem will work based on RAM only.

[Dem339] ⌈The Dem module shall verify the validity (which relates to block states), integrity (which relates to CRC results), and for general NvM-reading errors of its non-volatile blocks (before using the respective data).⌋(BSW04107)

Usually this verification is done in the API Dem_Init (refer to chapter 8.3.2.2) by using NvM_GetErrorStatus for these blocks, which are read by the ECU State Manager (refer to API NvM_ReadAll).

Note: For the non-volatile data of the Dem module, it is recommended to configure a CRC in the NvM.

[Dem578] ⌈If the NVM module was not able to read some non-volatile data of the Dem module, the Dem module shall initialize all non-volatile data with their initial values.⌋()

Note: To avoid inconsistencies between readable blocks and erroneous blocks, all non-volatile data are initialized. The initialization is done to allow the fault detection mechanism of the NvM module, to report the respective reading error(s) to the Dem module (refer to Dem_ReportErrorStatus). These errors denote the defective NVRAM.

[Dem340] ⌈After the API Dem_Init has finished, the Dem shall be fully operational.⌋()

[Dem550] ⌈ The Dem module shall provide the configuration parameter DemImmediateNvStorage per DTC (refer to DemDTCClass) to trigger the storage of the respective event memory entry including its event related data in non-volatile memory (refer to NvM_WriteBlock) immediately.⌋()

[Dem551] ⌈If immediate non-volatile storage is enabled for a specific DTC, the Dem module shall trigger the storage for new event memory entries (refer to Dem184) and after every change of the event related data (event memory entry was updated, refer to Dem396).⌋()

Note: For event memory entries, which are stored immediately, it is necessary to ensure data consistency (e.g. with the event status byte) during Dem_Init.

Note: If the immediate non-volatile storage is disabled, the event memory entry and its event related data are stored persistently during the shutdown phase (refer to Dem102 and the note below).

[Dem552] ⌈If immediate non-volatile storage is enabled for a specific DTC, the Dem module shall not perform further updates of an event memory entry, if its occurrence counter has reached the value defined by the configuration parameter DemImmediateNvStorageLimit (refer to DemGeneral), until this event memory entry is removed.⌋()

Note: This requirement is intended, to prevent the Dem module from writing too often to the NVRAM, if immediate non-volatile storage of an event memory entry is enabled.

[Dem102] ⌈The API Dem_Shutdown shall finalize all pending operations in the Dem module to prepare the internal states and data for transfer to the NVRAM. The event memory shall be locked and not modified until the API Dem_Init is recalled.⌋ (BSW101)

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

[Dem341] ⌈For individual non-volatile blocks the API NvM_WriteBlock shall be called within the API Dem_Shutdown.⌋()

If the ECU power supply is disconnected before the NvM has finished copying all data to NVRAM, these data will be incomplete/inconsistent or not stored. At next startup, the events of the last operating cycle could not be found anymore. Therefore, the NVRAM Manager configuration provides mechanisms for data consistency, like redundant data blocks.

[Dem164] ⌈ The Dem module shall use the APIs NvM_WriteBlock and NvM_ReadBlock of the NVRAM Manager, if there is the necessity to store and restore data between Dem_Init and Dem_Shutdown.⌋()

Note: The NvM module realizes a retry mechanism for block reading and writing. Therefore, the Dem module does not implement any retry mechanism for its non-volatile blocks.

[Dem579] ⌈If the NVM module was not able to write (some) non-volatile data of the Dem module, the Dem module shall ignore the reported negative return values by the NvM.⌋()

Note: If writing of non-volatile Dem data fails, the Dem module is not able to perform any adequate reaction.

7.8.5 Interaction with Diagnostic Error Tracer (Det)

The interaction with the Det is described in chapters 7.10, 7.11 and 7.12.

7.8.6 Interaction with Diagnostic Log & Trace (Dlt)

[Dem517] ⌈If DemTriggerDltReports (refer to DemGeneral) is enabled, the Dem module shall notify the Dlt module (refer to [11]) on each event status change of all

events (refer also to Dem016), by calling the function Dlt_DemTriggerOnEventStatus.
」(BSW04099)

Note: The function Dlt_DemTriggerOnEventStatus is equal to <Module>_DemTriggerOnEventStatus (refer to chapter 8.4.3.1.3). This configuration parameter is for simplicity to avoid a reference of the function to each individual EventId.

After the Dlt module has been notified by Dlt_DemTriggerOnEventStatus, it will make use of the functions Dem_DltGetMostRecentFreezeFrameRecordData (refer to chapter 8.3.5.1) and Dem_DltGetAllExtendedDataRecords (refer to chapter 0) to collect the current event related data.

[Dem632] 「If DemTriggerDltReports is enabled, the Dem module shall provide access on the most recent freeze frame record data (refer to 8.3.5.1 Dem_DltGetMostRecentFreezeFrameRecordData) to the Dlt module.」(BSW04099)

[Dem633] 「The function Dem_DltGetMostRecentFreezeFrameRecordData shall report the data of the most recent freeze frame record of the requested diagnostic event.」(BSW04099)

The format of the data in the destination buffer (DestBuffer) of the function Dem_DltGetMostRecentFreezeFrameRecordData is raw hexadecimal values and contains no header-information like RecordNumber or DataId.
The size of the buffer equals to the configuration settings of all respective data elements.

[Dem634] 「If DemTriggerDltReports is enabled, the Dem module shall provide access on all extended data records (refer to 0 Dem_DltGetAllExtendedDataRecords) to the Dlt module.」(BSW04099)

[Dem635] 「The function Dem_DltGetAllExtendedDataRecords shall report the data of all extended data records of the requested diagnostic event.」(BSW04099)

The format of the data in the destination buffer (DestBuffer) of the function Dem_DltGetAllExtendedDataRecords is raw hexadecimal values and contains no header-information like RecordNumber.
The size of the buffer equals to the configuration settings of all respective data elements.

7.8.7 Required data by the Dem module

The Dem module requires different information for internal computation/processing. If this information is provided by SW-Cs (as provide-ports), they are described in the respective ServiceNeeds (diagnostic capabilities).

One kind of information required by the Dem module are event related data (represented by freeze frames and extended data records, refer to chapter 7.3.7).

[Dem302] 「For OBD-relevant ECUs, the Dem module shall access (via the data element interface) on the following data values:

- engine temperature
- engine speed
- vehicle speed (refer to Dem346, PID $0D)
- distance information
- programming event
- ambient temperature
- ambient pressure
- accelerator paddle information」()

However, the list of data elements and their required size and resolution are implementation-specific and will be configured in DemGeneralOBD and handled during integration process.
For example, the vehicle speed, accelerator paddle information, ambient temperature, etc. are necessary to evaluate the cycle conditions of the IUMPR General Denominator.
Similar inputs are necessary for the PID computation (e.g. the engine temperature for the computation of the Warm-Up cycle or the Warm-Up cycle condition itself). These variables are typically accessed through the RTE.

7.9 Version check

[Dem067] 「The Dem module shall perform Inter Module Checks to avoid the integration of incompatible files. The imported included files shall be checked by preprocessing directives.」(BSW4)

The following version numbers shall be verified:

- `<MODULENAME>_AR_RELEASE_MAJOR_VERSION`
- `<MODULENAME>_AR_RELEASE_MINOR_VERSION`

If the values are not identical to the values expected, an error shall be reported.

7.10 Error classification

[Dem115] ⌈Values for event Ids (typically production codes, refer to DemEventId in DemEventParameter) shall be assigned by the configuration tool of the Dem automatically. They shall be published by the Dem module and are included via Dem.h by other modules.⌋()

Note, that also SW-C event Ids are published especially for complex device drivers, which may access on the status of specific SW-C events. SW-Cs use different ports to distinguish between different events.

[Dem116] ⌈Development error values are of type uint8.⌋(BSW00386)

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

Type or error	*Relevance*	*Related error code*	*Value [hex]*
API function called with a parameter value, which is not allowed by active configuration	Development	DEM_E_PARAM_CONFIG	0x10
API function called with a NULL pointer	Development	DEM_E_PARAM_POINTER	0x11
API function called with wrong parameter value	Development	DEM_E_PARAM_DATA	0x12
API function called with wrong length parameter value	Development	DEM_E_PARAM_LENGTH	0x13
API function called before the Dem module has been full initialized (refer to Dem124, Dem364)	Development	DEM_E_UNINIT	0x20
No valid data available by the SW-C	Development	DEM_E_NODATAAVAILABLE	0x30
Required conditions for the respective API call are not fulfilled (e.g. an invalid status change was initiated, or a filter was not set correctly, etc. – refer to Dem518).	Development	DEM_E_WRONG_CONDITION	0x40

Table 3 Types of errors which can be detected by the Dem module⌋()

[Dem124] ⌈If development error detection is enabled and any instance calls any Dem API, excluding Dem_ReportErrorStatus, before the Dem was not fully initialized, the Dem module shall set the error code DEM_E_UNINIT.⌋(BSW00406)

[Dem364] ⌈If development error detection is enabled and any instance calls Dem_ReportErrorStatus before the Dem was not pre-initialized, the Dem module shall set the error code DEM_E_UNINIT.⌋()

[Dem518] ⌈If development error detection is enabled and a Dem function is called with required pre-conditions not fulfilled, the Dem module shall set the error code DEM_E_WRONG_CONDITION.⌋()

Note: For example, Dem_GetNextFilteredDTCAndFDC is called, after Dem_SetDTCFilter with FilterForFaultDetectionCounter = FALSE was called.

[Dem370] ⌈If development error detection is enabled and a Dem function detects a development error, excluding DEM_E_NODATAAVAILABLE, then the Dem function shall return immediately with E_NOT_OK (in case of Std_ReturnType) or an appropriate negative return value, after the development error was reported.⌋()

7.11 Error detection

[Dem113] ⌈The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch *DemDevErrorDetect* (refer to DemGeneral) shall activate or deactivate the detection of all development errors.⌋(BSW00385)

[Dem114] ⌈If the *DemDevErrorDetect* switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.10 and chapter 8.⌋(BSW00385)

[Dem174] ⌈The detection of production code errors cannot be switched off.⌋()

7.12 Error notification

[Dem117] ⌈ Detected development errors shall be reported to the *Det_ReportError* service of the Development Error Tracer (Det) if the pre-processor switch *DEM_DEV_ERROR_DETECT* is set.⌋(BSW00418)

7.13 Support for Debugging

[Dem519] ⌈Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable.⌋()

[Dem520] ⌈All type definitions of variables, which shall be debugged, shall be accessible by the header file Dem.h.⌋()

[Dem521] ⌈The declaration of variables in the header file shall be such, that it is possible to calculate the size of the variables by C-"sizeof".⌋()

[Dem522] ⌈Variables available for debugging shall be described in the respective Basic Software Module Description.⌋()

8 API specification

The figures below show the interfaces between Dem and its surrounding SW-Cs and BSW modules. The description of the interface shall give a simple overview of these relations.

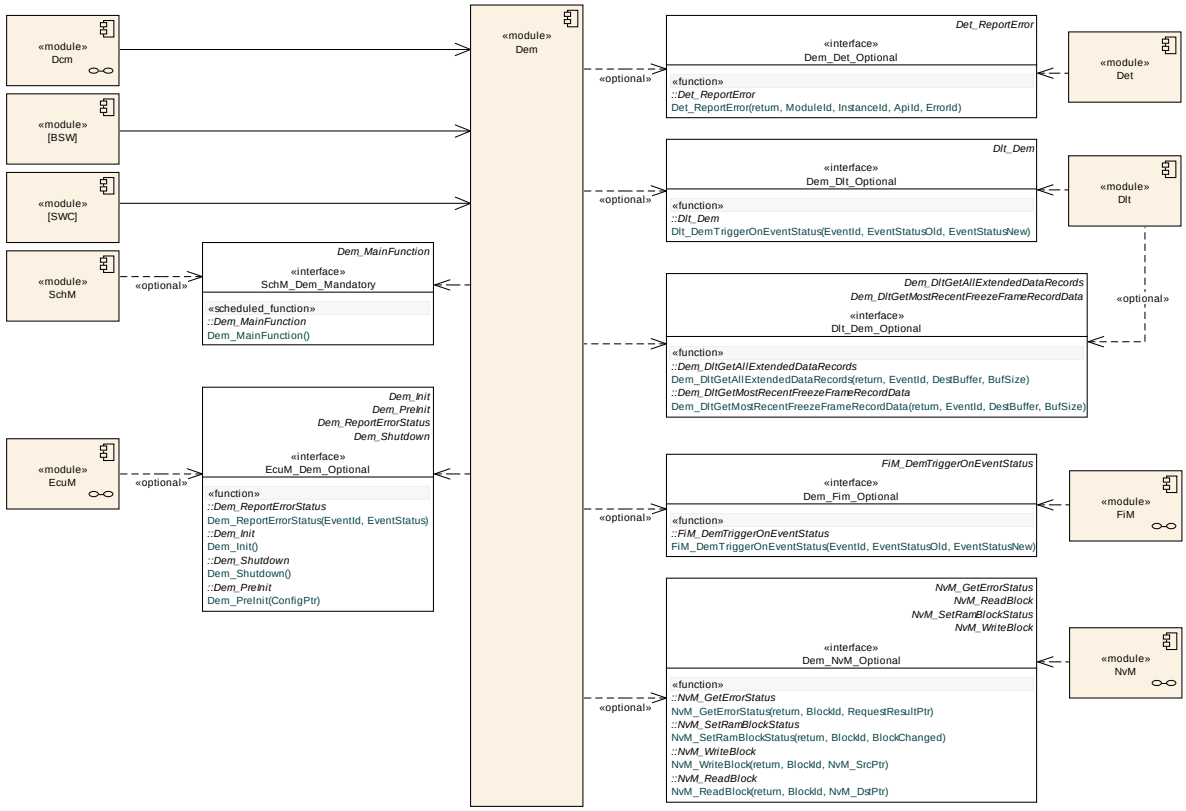

Figure 51 Overview of interfaces between the Dem and other BSW modules

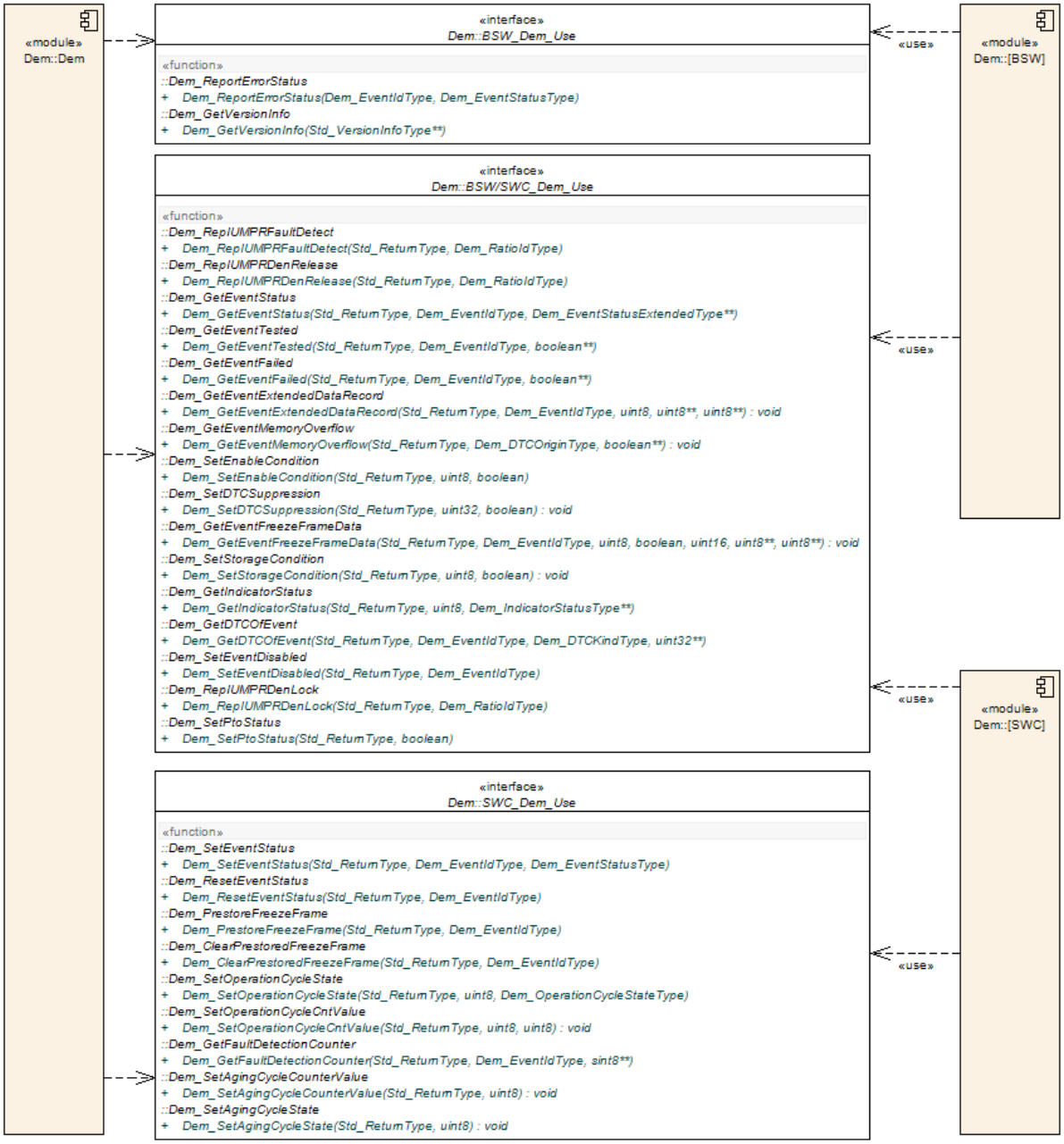

Figure 52 Overview of interfaces between the Dem and other modules (in general)

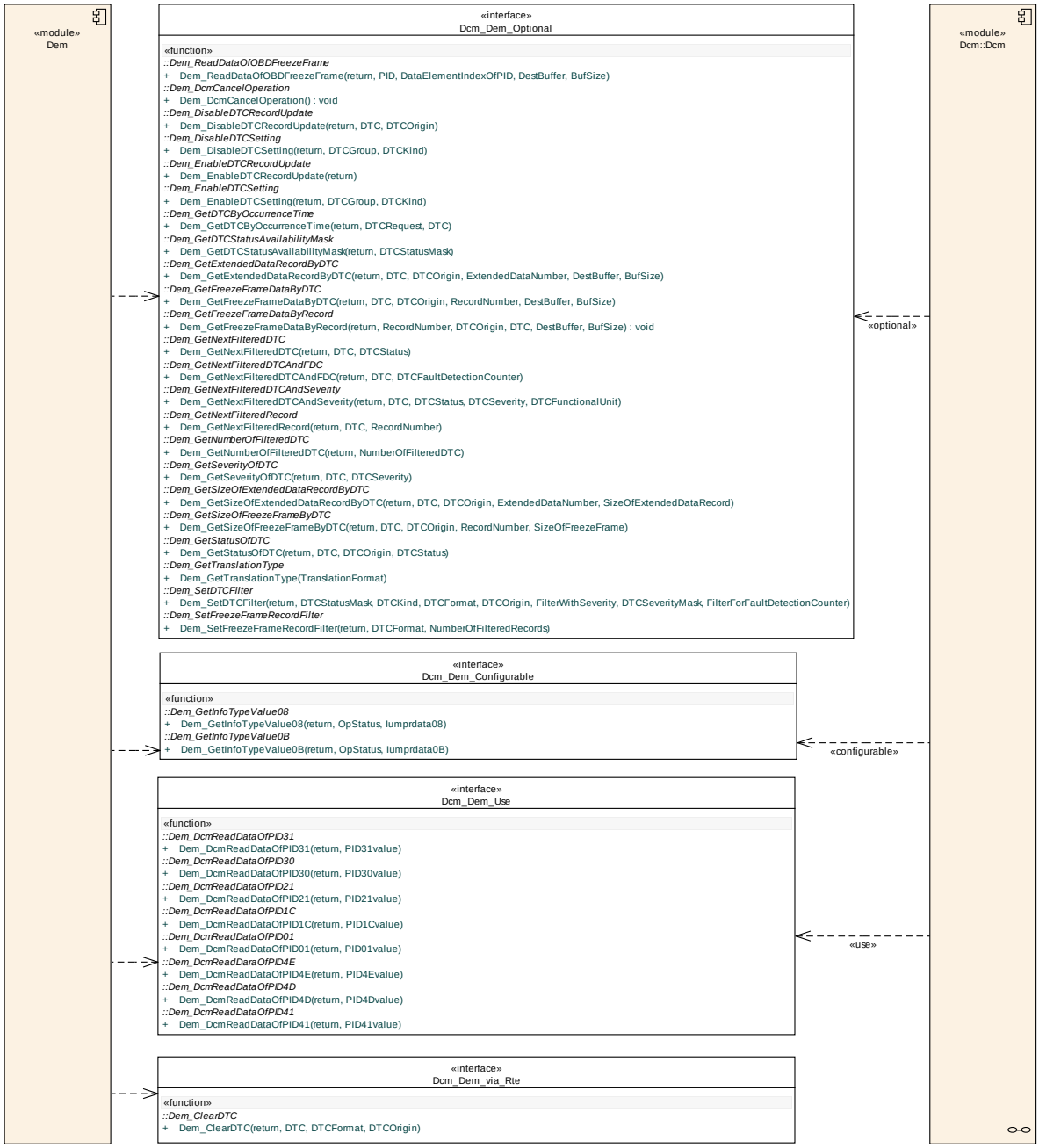

Figure 53 Overview of interfaces between the Dem and Dcm

8.1 Imported types

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

[Dem176]

⌈

Module	*Imported Type*

Dcm	Dcm_OpStatusType
NvM	NvM_BlockIdType
	NvM_RequestResultType
Std_Types	Std_ReturnType
	Std_VersionInfoType

]()

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

```
// refer to chapter 8.2.1.2
ImplementationDataType Dem_EventIdType {
   LOWER-LIMIT = 1;
   UPPER-LIMIT = <N>;
};

// refer to chapter 8.2.1.3
ImplementationDataType Dem_EventStatusType {
   LOWER-LIMIT=0;
   UPPER-LIMIT=3;
   0 -> DEM_EVENT_STATUS_PASSED
   1 -> DEM_EVENT_STATUS_FAILED
   2 -> DEM_EVENT_STATUS_PREPASSED
   3 -> DEM_EVENT_STATUS_PREFAILED
}

// refer to chapter 8.2.1.4
ImplementationDataType Dem_EventStatusExtendedType {
   LOWER-LIMIT = 0;
   UPPER-LIMIT = 255;
   0x01 -> DEM_UDS_STATUS_TF
   0x02 -> DEM_UDS_STATUS_TFTOC
   0x04 -> DEM_UDS_STATUS_PDTC
   0x08 -> DEM_UDS_STATUS_CDTC
   0x10 -> DEM_UDS_STATUS_TNCSLC
   0x20 -> DEM_UDS_STATUS_TFSLC
   0x40 -> DEM_UDS_STATUS_TNCTOC
   0x80 -> DEM_UDS_STATUS_WIR
}

// refer to chapter 8.2.1.9
ImplementationDataType Dem_DTCFormatType {
   LOWER-LIMIT=0;
   UPPER-LIMIT=1;
   0 -> DEM_DTC_FORMAT_OBD
   1 -> DEM_DTC_FORMAT_UDS
}

// refer to chapter 8.2.1.17
ImplementationDataType Dem_InitMonitorReasonType {
   LOWER-LIMIT=1;
   UPPER-LIMIT=2;
   1 -> DEM_INIT_MONITOR_CLEAR
   2 -> DEM_INIT_MONITOR_RESTART
}

// refer to chapter 8.2.1.5
ImplementationDataType Dem_OperationCycleStateType {
```

```
   LOWER-LIMIT=0;
   UPPER-LIMIT=1;
   0 -> DEM_CYCLE_STATE_START
   1 -> DEM_CYCLE_STATE_END
}

ImplementationDataType Dem_DTCStatusMaskType {
   LOWER-LIMIT = 0;
   UPPER-LIMIT = 255;
}

// refer to chapter 8.2.1.6
ImplementationDataType Dem_IndicatorStatusType {
   LOWER-LIMIT=0;
   UPPER-LIMIT=3;
   0 -> DEM_INDICATOR_OFF
   1 -> DEM_INDICATOR_CONTINUOUS
   2 -> DEM_INDICATOR_BLINKING
   3 -> DEM_INDICATOR_BLINK_CONT
}

ImplementationDataType Dem_MaxDataValueType {
   CATEGORY = ARRAY;
   subElement {
      CATEGORY = TYPE_REFERENCE;
      implementationDataType ELEMENT-TYPE-REF = uint8;
      ARRAY-SIZE = [size of largest Extended data record / DID];
   }
}
```

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

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

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

// refer to chapter 8.2.1.16
ImplementationDataType Dem_RatioIdType {
   LOWER-LIMIT = 0;
   UPPER-LIMIT = <N - 1>;
};

// refer to chapter 8.2.1.9
ImplementationDataType Dem_DTCOriginType {
   LOWER-LIMIT = 1;
   UPPER-LIMIT = 4;
   1 -> DEM_DTC_ORIGIN_PRIMARY_MEMORY
   2 -> DEM_DTC_ORIGIN_MIRROR_MEMORY
   3 -> DEM_DTC_ORIGIN_PERMANENT_MEMORY
   4 -> DEM_DTC_ORIGIN_SECONDARY_MEMORY
```

```
}
```

8.2 Type definitions

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

8.2.1 Dem data types

8.2.1.1 Dem_ConfigType

Name:	`Dem_ConfigType`		
Type:	`Structure`		
Element:	`void`	`implementation specific`	--
Description:	This type of the external data structure shall contain the post build initialization data for the Dem.		

8.2.1.2 Dem_EventIdType

Name:	`Dem_EventIdType`		
Type:	`uint16`		
Range:	`1...65535`	`--`	Internal identifier of a diagnostic event Remark: 0 is not a valid value
Description:	Identification of an event by assigned EventId. The EventId is assigned by the Dem. Example: 1 refers to monitor x, 2 refers to monitor y, etc.		

8.2.1.3 Dem_EventStatusType

Name:	`Dem_EventStatusType`		
Type:	`uint8`		
Range:	`DEM_EVENT_STATUS_PASSED`	`0x00`	Monitor reports qualified test result passed.
	`DEM_EVENT_STATUS_FAILED`	`0x01`	Monitor reports qualified test result

			failed.
	DEM_EVENT_STATUS_PREPASSED	0x02	Monitor reports non-qualified test result pre-passed (debounced
	DEM_EVENT_STATUS_PREFAILED	0x03	Monitor reports non-qualified test result pre-failed (debounced Dem-internally).
		0x04 - 0xFF	reserved
Description:	This type contains all monitor test result values, which can be reported via Dem_ReportErrorStatus() and Dem_SetEventStatus().		

8.2.1.4 Dem_EventStatusExtendedType

Name:	Dem_EventStatusExtendedType		
Type:	uint8		
Range:	range	0x00..0xFF	the possible range from 0x00 to 0xFF allows for all combinations of the individual status bits
	DEM_UDS_STATUS_TF	0x01	bit 0: TestFailed
	DEM_UDS_STATUS_TFTOC	0x02	bit 1: TestFailedThisOperationCycle
	DEM_UDS_STATUS_PDTC	0x04	bit 2: PendingDTC
	DEM_UDS_STATUS_CDTC	0x08	bit 3: ConfirmedDTC
	DEM_UDS_STATUS_TNCSLC	0x10	bit 4: TestNotCompletedSinceLastClear
	DEM_UDS_STATUS_TFSLC	0x20	bit 5: TestFailedSinceLastClear
	DEM_UDS_STATUS_TNCTOC	0x40	bit 6: TestNotCompletedThisOperationCycle
	DEM_UDS_STATUS_WIR	0x80	bit 7: WarningIndicatorRequested
Description:	In this data-type each bit has an individual meaning. The bit is set to 1 when the condition holds. For example, if the 2nd bit (0x02) is set to 1, this means that the test failed this operation cycle. If the bit is set to 0, it has not yet failed this cycle.		

8.2.1.5 Dem_OperationCycleStateType

Name:	Dem_OperationCycleStateType		
Type:	uint8		
Range:	DEM_CYCLE_STATE_START	0x00	Start/restart the operation cycle
	DEM_CYCLE_STATE_END	0x01	End the operation cycle
Description:	This type contains operation cycle state values, which can be reported via Dem_SetOperationCycleState().		

8.2.1.6 Dem_IndicatorStatusType

Name:	Dem_IndicatorStatusType		
Type:	uint8		
Range:	DEM_INDICATOR_OFF	0x00	Indicator off mode
	DEM_INDICATOR_CONTINUOUS	0x01	Indicator continuously on mode
	DEM_INDICATOR_BLINKING	0x02	Indicator blinking mode
	DEM_INDICATOR_BLINK_CONT	0x03	Indicator blinking or continuously on mode
Description:	Indicator mode returned by Dem_GetIndicatorStatus()		

8.2.1.7 Dem_DTCGroupType

Name:	Dem_DTCGroupType		
Type:	uint32		
Range:	DEM_DTC_GROUP_BODY_DTCS	--	selects group of body DTCs
	DEM_DTC_GROUP_CHASSIS_DTCS	--	selects group of chassis DTCs
	DEM_DTC_GROUP_NETWORK_COM_DTCS	--	selects group of network communication DTCs
	DEM_DTC_GROUP_POWERTRAIN_DTCS	--	selects group of powertrain DTCs
	DEM_DTC_GROUP_EMISSION_REL_DTCS	0x000000	selects group of OBD-relevant DTCs
	DEM_DTC_GROUP_ALL_DTCS	0xffffff	0xFFFFFF: selects all DTCs
Description:	Selects a group of DTCs.		

8.2.1.8 Dem_DTCKindType

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

8.2.1.9 Dem_DTCFormatType

Name:	Dem_DTCFormatType		
Type:	uint8		
Range:	DEM_DTC_FORMAT_OBD	0	selects the 2-byte OBD DTC format (refer to DemObdDTC)
	DEM_DTC_FORMAT_UDS	1	selects the 3-byte UDS DTC format (refer to DemUdsDTC)
Description:	Selects/specifies the format of the DTC value.		

8.2.1.10 Dem_DTCOriginType

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

8.2.1.11 Dem_DTCRequestType

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

8.2.1.12 Dem_DTCTranslationFormatType

Name:	Dem_DTCTranslationFormatType

Type:	uint8		
Range:	DEM_DTC_TRANSLATION_ISO15031_6	0x00	ISO15031-6 DTC format
	DEM_DTC_TRANSLATION_ISO14229_1	0x01	ISO14229-1 DTC format
	DEM_DTC_TRANSLATION_SAEJ1939_73	0x02	SAEJ1939-73 DTC format
	DEM_DTC_TRANSLATION_ISO11992_4	0x03	ISO11992-4 DTC format
Description:	DTC translation format as defined in ISO14229-1 Service 0x19 returned by Dem_GetTranslationType().		

8.2.1.13 Dem_DTCSeverityType

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

8.2.1.14 Dem_FilterForFDCType

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

8.2.1.15 Dem_FilterWithSeverityType

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

8.2.1.16 Dem_RatioIdType

Name:	Dem_RatioIdType		
Type:	uint8, uint16		
Range:	0..255, 0..65535	- -	Configurable, size depends on system complexity (refer to multiplicity of the configuration container DemRatioId).
Description:	OBD specific ratio Id (related to a specific event, a FID, and an IUMPR group). This type depends on the Dem configuration.		

8.2.1.17 Dem_InitMonitorReasonType

Name:	Dem_InitMonitorReasonType		
Type:	uint8		
Range:	DEM_INIT_MONITOR_CLEAR	0x01	Event was cleared and all internal values and states are reset.
	DEM_INIT_MONITOR_RESTART	0x02	Operation cycle of the event was (re-)started.
Description:	(Re-)Initialization reason returned by the callback <Module>_DemInitMonitorFor<EventName>().		

8.2.2 Dem return types

8.2.2.1 Dem_ReturnSetFilterType

Name:	Dem_ReturnSetFilterType		
Type:	uint8		
Range:	DEM_FILTER_ACCEPTED	0x00	Filter was accepted
	DEM_WRONG_FILTER	0x01	Wrong filter selected
Description:	Used to return the status of (re-)setting a specific filter.		

8.2.2.2 Dem_ReturnGetStatusOfDTCType

Name:	Dem_ReturnGetStatusOfDTCType		
Type:	uint8		
Range:	DEM_STATUS_OK	0x00	Status of DTC is OK

	DEM_STATUS_WRONG_DTC	0x01	DTC value not existing (in this format)
	DEM_STATUS_WRONG_DTCORIGIN	0x02	Wrong DTC origin
	DEM_STATUS_FAILED	0x03	DTC failed
	DEM_STATUS_PENDING	0x04	The requested value is calculated asynchronously and currently not available. The caller can retry later.
Description:	Used to return the status of Dem_GetStatusOfDTC.		

8.2.2.3 Dem_ReturnGetNextFilteredDTCType

Name:	Dem_ReturnGetNextFilteredDTCType		
Type:	uint8		
Range:	DEM_FILTERED_OK	0x00	Returned next filtered DTC
	DEM_FILTERED_NO_MATCHING_DTC	0x01	No further DTC (matching the filter criteria) found
	DEM_FILTERED_PENDING	0x02	The requested value is calculated asynchronously and currently not available. The caller can retry later. Only used by asynchronous interfaces.
Description:	Used to return the status of the Dem_GetNextFiltered<...> interfaces.		

8.2.2.4 Dem_ReturnGetNumberOfFilteredDTCType

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

8.2.2.5 Dem_ReturnClearDTCType

Name:	Dem_ReturnClearDTCType		
Type:	uint8		
Range:	DEM_CLEAR_OK	0x00	DTC successfully cleared
	DEM_CLEAR_WRONG_DTC	0x01	DTC value not existing (in this format)
	DEM_CLEAR_WRONG_DTCORIGIN	0x02	Wrong DTC origin
	DEM_CLEAR_FAILED	0x03	DTC not cleared
	DEM_CLEAR_PENDING	0x04	Clearing of DTC is pending
Description:	Used to return the status of Dem_ClearDTC.		

8.2.2.6 Dem_ReturnControlDTCSettingType

Name:	Dem_ReturnControlDTCSettingType		
Type:	uint8		
Range:	DEM_CONTROL_DTC_SETTING_OK	0x00	DTC setting control successful
	DEM_CONTROL_DTC_SETTING_N_OK	0x01	DTC setting control not successful
	DEM_CONTROL_DTC_WRONG_DTCGROUP	0x02	DTC setting control not successful because group of DTC was wrong
Description:	Used to return the status of Dem_DisableDTCSetting and Dem_EnableDTCSetting.		

8.2.2.7 Dem_ReturnDisableDTCRecordUpdateType

Name:	Dem_ReturnDisableDTCRecordUpdateType		
Type:	uint8		
Range:	DEM_DISABLE_DTCRECUP_OK	0x00	Event memory update of DTC successfully disabled
	DEM_DISABLE_DTCRECUP_WRONG_DTC	0x01	DTC value not existing (in UDS format)
	DEM_DISABLE_DTCRECUP_WRONG_DTCORIGIN	0x02	Wrong DTC origin
	DEM_DISABLE_DTCRECUP_PENDING	0x03	Disabling is currently not possible. The caller can retry later.
Description:	Used to return the status of Dem_DisableDTCRecordUpdate		

8.2.2.8 Dem_ReturnGetFreezeFrameDataByRecordType

Name:	Dem_ReturnGetFreezeFrameDataByRecordType
Type:	uint8

Range:	DEM_GET_FFBYRECORD_OK	0x00	DTC successfully returned
	DEM_GET_FFBYRECORD_WRONG_RECORD	0x01	Wrong record
	DEM_GET_FFBYRECORD_NO_DTC_FOR_RECORD	0x02	No DTC for record available
Description:	Used to return the status of Dem_GetFreezeFrameDataByRecord.		

8.2.2.9 Dem_ReturnGetExtendedDataRecordByDTCType

Name:	Dem_ReturnGetExtendedDataRecordByDTCType		
Type:	uint8		
Range:	DEM_RECORD_OK	0x00	Extended data record successfully returned
	DEM_RECORD_WRONG_DTC	0x01	DTC value not existing (in UDS format)
	DEM_RECORD_WRONG_DTCORIGIN	0x02	Origin wrong
	DEM_RECORD_WRONG_NUMBER	0x03	Record number is not supported by configuration and therefore invalid
	DEM_RECORD_WRONG_BUFFERSIZE	0x04	Provided buffer to small
	DEM_RECORD_PENDING	0x05	The requested value is calculated asynchronously and currently not available. The caller can retry later.
Description:	Used to return the status of Dem_GetExtendedDataRecordByDTC.		

8.2.2.10 Dem_ReturnGetDTCByOccurrenceTimeType

Name:	Dem_ReturnGetDTCByOccurrenceTimeType		
Type:	uint8		
Range:	DEM_OCCURR_OK	0x00	matching DTC available
	DEM_OCCURR_NOT_AVAILABLE	0x01	no DTC is matching the requested occurrence time
Description:	Used to return the status of Dem_GetDTCByOccurrenceTime.		

8.2.2.11 Dem_ReturnGetFreezeFrameDataByDTCType

Name:	Dem_ReturnGetFreezeFrameDataByDTCType		
Type:	uint8		
Range:	DEM_GET_FFDATABYDTC_OK	0x00	Size successfully returned.
	DEM_GET_FFDATABYDTC_WRONG_DTC	0x01	DTC value not existing (in UDS format)
	DEM_GET_FFDATABYDTC_WRONG_DTCORIGIN	0x02	Wrong DTC origin

	DEM_GET_FFDATABYDTC_WRONG_RECORDNUMBER	0x03	Record number is not supported by configuration and therefore invalid
	DEM_GET_FFDATABYDTC_WRONG_BUFFERSIZE	0x04	provided buffer size to small
	DEM_GET_FFDATABYDTC_PENDING	0x05	The requested value is calculated asynchronously and currently not available. The caller can retry later.
Description:	Used to return the status of Dem_GetFreezeFrameDataByDTC.		

8.2.2.12 Dem_ReturnGetSizeOfExtendedDataRecordByDTCType

Name:	Dem_ReturnGetSizeOfExtendedDataRecordByDTCType		
Type:	uint8		
Range:	DEM_GET_SIZEOFEDRBYDTC_OK	0x00	Size successfully returned.
	DEM_GET_SIZEOFEDRBYDTC_W_DTC	0x01	DTC value not existing (in UDS format)
	DEM_GET_SIZEOFEDRBYDTC_W_DTCOR	0x02	Wrong DTC origin
	DEM_GET_SIZEOFEDRBYDTC_W_RNUM	0x03	Record number is not supported by configuration and therefore invalid
	DEM_GET_SIZEOFEDRBYDTC_PENDING	0x04	The requested value is calculated asynchronously and currently not available. The caller can retry later.
Description:	Used to return the status of Dem_GetSizeOfExtendedDataRecordByDTC.		

8.2.2.13 Dem_ReturnGetSizeOfFreezeFrameByDTCType

Name:	Dem_ReturnGetSizeOfFreezeFrameByDTCType		
Type:	uint8		
Range:	DEM_GET_SIZEOFFF_OK	0x00	Size successfully returned.
	DEM_GET_SIZEOFFF_WRONG_DTC	0x01	DTC value not existing (in UDS format)
	DEM_GET_SIZEOFFF_WRONG_DTCOR	0x02	Wrong DTC origin
	DEM_GET_SIZEOFFF_WRONG_RNUM	0x03	Record number is not supported by configuration and therefore invalid
	DEM_GET_SIZEOFFF_PENDING	0x04	The requested value is calculated asynchronously and currently not available. The caller can retry later.
Description:	Used to return the status of Dem_GetSizeOfFreezeFrameByDTC.		

8.2.2.14 Dem_ReturnGetSeverityOfDTCType

Name:	Dem_ReturnGetSeverityOfDTCType		
Type:	uint8		
Range:	DEM_GET_SEVERITYOFDTC_OK	0x00	Severity successfully returned.
	DEM_GET_SEVERITYOFDTC_WRONG_DTC	0x01	DTC value not existing (in UDS format)
	DEM_GET_SEVERITYOFDTC_NOSEVERITY	0x02	Severity information is not available
	DEM_GET_SEVERITYOFDTC_PENDING	0x03	The requested value is calculated asynchronously and currently not available. The caller can retry later.
Description:	Used to return the status of Dem_GetSeverityOfDTC.		

8.2.2.15 Dem_ReturnGetFunctionalUnitOfDTCType

Name:	Dem_ReturnGetFunctionalUnitOfDTCType		
Type:	uint8		
Range:	DEM_GET_FUNCTIONALUNITOFDTC_OK	0x00	Functional unit successfully returned.
	DEM_GET_FUNCTIONALUNITOFDTC_WRONG_DTC	0x01	DTC value not existing (in UDS format)
Description:	Used to return the status of Dem_GetFunctionalUnitOfDTC.		

8.3 Function definitions

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

8.3.1 Dem_GetVersionInfo

[Dem177]

⌈

Service name:	Dem_GetVersionInfo
Syntax:	void Dem_GetVersionInfo(Std_VersionInfoType* versioninfo

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

」()

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

- Module Id
- Vendor Id
- Vendor specific version numbers (BSW00407).」(BSW3, BSW00407, BSW00411)

[Dem111]「The function Dem_GetVersionInfo shall be precompile time configurable (ON/OFF) by the configuration parameter DemVersionInfoApi.」(BSW3, BSW00407, BSW00411)

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

8.3.2 Interface ECU State Manager ⇔ Dem

8.3.2.1 Dem_PreInit

[Dem179]

「

Service name:	Dem_PreInit

Syntax:	`void Dem_PreInit(`
	`  const Dem_ConfigType* ConfigPtr`
	`)`

Service ID[hex]:	0x01
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	ConfigPtr --
Parameters (inout):	None
Parameters (out):	None

Return value:	None
Description:	Initializes the internal states necessary to process events reported by BSW-modules.

⌋()

8.3.2.2 Dem_Init

[Dem181]

⌈

Service name:	Dem_Init

Syntax:	`void Dem_Init(`
	`  void`
	`)`

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

⌋()

8.3.2.3 Dem_Shutdown

[Dem182]

⌈

Service name:	Dem_Shutdown

Syntax:	`void Dem_Shutdown(`
	`  void`
	`)`

Service ID[hex]:	0x03
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Shuts down this module.

⌋(BSW336)

8.3.3 Interface BSW modules / SW-Components ⇔ Dem

8.3.3.1 Dem_ReportErrorStatus

[Dem206]

⌈

Service name:	Dem_ReportErrorStatus	

Syntax:	`void Dem_ReportErrorStatus(`
	`  Dem_EventIdType EventId,`
	`  Dem_EventStatusType EventStatus`
	`)`	

Service ID[hex]:	0x0f	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an event by assigned Event ID.
	EventStatus	Monitor test result
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function.	

⌋()

8.3.3.2 Dem_SetEventStatus

[Dem183]

⌈

Service name:	Dem_SetEventStatus	

Syntax:	`Std_ReturnType Dem_SetEventStatus(`
	`  Dem_EventIdType EventId,`
	`  Dem_EventStatusType EventStatus`
	`)`	

Service ID[hex]:	0x04	
Sync/Async:	Synchronous/Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
	EventStatus	Monitor test result
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: set of event status was successful E_NOT_OK: set of event status failed or could not be accepted (e.g.: the operation cycle configured for this event has not been started, an according enable condition has been disabled)
Description:	Processes the events reported by SW-Cs via RTE. This API can only be used through the RTE, and therefore no declaration is exported via Dem.h. Some bits of	

	the UDS DTC status byte changes synchronously or asynchronously (refer to Dem036 and Dem379).

⌋()

8.3.3.3 Dem_ResetEventStatus

[Dem185]

⌈

Service name:	Dem_ResetEventStatus	
Syntax:	`Std_ReturnType Dem_ResetEventStatus( Dem_EventIdType EventId )`	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: reset of event status was successful E_NOT_OK: reset of event status failed or is not allowed, because the event is already tested in this operation cycle
Description:	Resets the event failed status. This API can only be used through the RTE and therefore no declaration is exported via Dem.h.	

⌋()

8.3.3.4 Dem_PrestoreFreezeFrame

[Dem188]

⌈

Service name:	Dem_PrestoreFreezeFrame	
Syntax:	`Std_ReturnType Dem_PrestoreFreezeFrame( Dem_EventIdType EventId )`	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK Freeze frame prestorage was successful E_NOT_OK Freeze frame prestorage failed
Description:	Captures the freeze frame data for a specific event. This API can only be used through the RTE and therefore no declaration is exported via Dem.h.	

⌋()

8.3.3.5 Dem_ClearPrestoredFreezeFrame

[Dem193]

⌈

Service name:	Dem_ClearPrestoredFreezeFrame	
Syntax:	`Std_ReturnType Dem_ClearPrestoredFreezeFrame( Dem_EventIdType EventId )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Clear prestored freeze frame was successful E_NOT_OK: Clear prestored freeze frame failed
Description:	Clears a prestored freeze frame of a specific event. This API can only be used through the RTE and therefore no declaration is exported via Dem.h.	

⌋()

8.3.3.6 Dem_SetOperationCycleState

[Dem194]

⌈

Service name:	Dem_SetOperationCycleState	
Syntax:	`Std_ReturnType Dem_SetOperationCycleState( uint8 OperationCycleId, Dem_OperationCycleStateType CycleState )`	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	OperationCycleId	Identification of operation cycle, like power cycle, driving cycle.
	CycleState	New operation cycle state: (re-)start or end
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: set of operation cycle was successful E_NOT_OK: set of operation cycle failed
Description:	Sets an operation cycle state. This API can only be used through the RTE and therefore no declaration is exported via Dem.h.	

⌋()

8.3.3.7 Dem_SetOperationCycleCntValue

[Dem553]

⌜

Service name:	Dem_SetOperationCycleCntValue	
Syntax:	`Std_ReturnType Dem_SetOperationCycleCntValue( uint8 OperationCycleId, uint8 CounterValue )`	
Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	OperationCycleId	Identification of operation cycle, like power cycle, driving cycle.
	CounterValue	Current external counter value of the according opertion cycle.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: set of operation cycle counter was successful E_NOT_OK: set of operation cycle counter failed
Description:	Provides the value of the external operation cycle counter. This API can only be used through the RTE, and therefore no declaration is exported via Dem.h.	

⌟()

8.3.3.8 Dem_SetAgingCycleState

[Dem554]

⌜

Service name:	Dem_SetAgingCycleState	
Syntax:	`Std_ReturnType Dem_SetAgingCycleState( uint8 AgingCycleId )`	
Service ID[hex]:	0x11	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	AgingCycleId	Identification of aging cycle.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: set of aging cycle was successful E_NOT_OK: set of aging cycle failed
Description:	Triggers the next aging cycle state. This API can only be used through the RTE, and therefore no declaration is exported via Dem.h.	

⌟()

8.3.3.9 Dem_SetAgingCycleCounterValue

[Dem555]

⌜

Service name:	Dem_SetAgingCycleCounterValue	
Syntax:	`Std_ReturnType Dem_SetAgingCycleCounterValue( uint8 CounterValue )`	
Service ID[hex]:	0x12	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CounterValue	Current external aging cycle counter value.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: set of aging cycle counter was successful E_NOT_OK: set of aging cycle counter failed
Description:	Provides the value of the external aging cycle counter. This API can only be used through the RTE, and therefore no declaration is exported via Dem.h.	

⌟()

8.3.3.10 Dem_GetEventStatus

[Dem195]

⌜

Service name:	Dem_GetEventStatus	
Syntax:	`Std_ReturnType Dem_GetEventStatus( Dem_EventIdType EventId, Dem_EventStatusExtendedType* EventStatusExtended )`	
Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
Parameters (inout):	None	
Parameters (out):	EventStatusExtended	UDS DTC status byte of the requested event (refer to chapter "Status bit support"). If the return value of the function call is E_NOT_OK, this parameter does not contain valid data.
Return value:	Std_ReturnType	E_OK: get of event status was successful E_NOT_OK: get of event status failed
Description:	Gets the current extended event status of an event.	

⌟()

8.3.3.11 Dem_GetEventFailed

[Dem196]

⌜

Service name:	Dem_GetEventFailed

Syntax:	`Std_ReturnType Dem_GetEventFailed(`
	`  Dem_EventIdType EventId,`
	`  boolean* EventFailed`
	`)`	

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
Parameters (inout):	None	
Parameters (out):	EventFailed	TRUE - Last Failed FALSE - not Last Failed
Return value:	Std_ReturnType	E_OK: get of “EventFailed” was successful E_NOT_OK: get of “EventFailed” was not successful
Description:	Gets the event failed status of an event.	

⌋()

8.3.3.12 Dem_GetEventTested

[Dem197]

⌈

Service name:	Dem_GetEventTested	

Syntax:	`Std_ReturnType Dem_GetEventTested(`
	`  Dem_EventIdType EventId,`
	`  boolean* EventTested`
	`)`	

Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
Parameters (inout):	None	
Parameters (out):	EventTested	TRUE - event tested this cycle FALSE - event not tested this cycle
Return value:	Std_ReturnType	E_OK: get of event state “tested” successful E_NOT_OK: get of event state “tested” failed
Description:	Gets the event tested status of an event.	

⌋()

8.3.3.13 Dem_GetDTCOfEvent

[Dem198]

⌈

Service name:	Dem_GetDTCOfEvent

Syntax:	`Std_ReturnType Dem_GetDTCOfEvent(`
	`  Dem_EventIdType EventId,`
	`  Dem_DTCFormatType DTCFormat,`
	`  uint32* DTCOfEvent`

	)	
Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
	DTCFormat	Defines the output-format of the requested DTC value.
Parameters (inout):	None	
Parameters (out):	DTCOfEvent	Receives the DTC value in respective format returned by this function. If the return value of the function is other than E_OK this parameter does not contain valid data.
Return value:	Std_ReturnType	E_OK: get of DTC was successful E_NOT_OK: the call was not successful DEM_E_NO_DTC_AVAILABLE: there is no DTC configured in the requested format
Description:	Gets the DTC of an event.	

⌋()

8.3.3.14 Dem_SetEnableCondition

[Dem201]

⌈

Service name:	Dem_SetEnableCondition	
Syntax:	`Std_ReturnType Dem_SetEnableCondition( uint8 EnableConditionID, boolean ConditionFulfilled )`	
Service ID[hex]:	0x39	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EnableConditionID	This parameter identifies the enable condition.
	ConditionFulfilled	This parameter specifies whether the enable condition assigned to the EnableConditionID is fulfilled (TRUE) or not fulfilled (FALSE).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	In case the enable condition could be set successfully the API call returns E_OK. If the setting of the enable condition failed the return value of the function is E_NOT_OK.
Description:	Sets an enable condition.	

⌋()

8.3.3.15 Dem_SetStorageCondition

[Dem556]

⌈

Service name:	Dem_SetStorageCondition

Syntax:	`Std_ReturnType Dem_SetStorageCondition(`
	`  uint8 StorageConditionID,`
	`  boolean ConditionFulfilled`
	`)`	

Service ID[hex]:	0x38	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	StorageConditionID	This parameter identifies the storage condition.
	ConditionFulfilled	This parameter specifies whether the storage condition assigned to the StorageConditionID is fulfilled (TRUE) or not fulfilled (FALSE).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	In case the storage condition could be set successfully the API call returns E_OK. If the setting of the storage condition failed the return value of the function is E_NOT_OK.
Description:	Sets a storage condition.	

⌋()

8.3.3.16 Dem_GetFaultDetectionCounter

[Dem203]

⌈

Service name:	Dem_GetFaultDetectionCounter	

Syntax:	`Std_ReturnType Dem_GetFaultDetectionCounter(`
	`  Dem_EventIdType EventId,`
	`  sint8* FaultDetectionCounter`
	`)`	

Service ID[hex]:	0x3e	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
Parameters (inout):	None	
Parameters (out):	FaultDetectionCounter	This parameter receives the Fault Detection Counter information of the requested EventId. If the return value of the function call is other than E_OK this parameter does not contain valid data. -128dec...127dec PASSED... FAILED according to ISO 14229-1
Return value:	Std_ReturnType	E_OK: request was successful E_NOT_OK: request failed
Description:	Gets the fault detection counter of an event. This API can only be used through the RTE, and therefore no declaration is exported via Dem.h.	

⌋()

8.3.3.17 Dem_GetIndicatorStatus

[Dem205]

⌈

Service name:	Dem_GetIndicatorStatus	

Syntax:	`Std_ReturnType Dem_GetIndicatorStatus(`
	`  uint8 IndicatorId,`
	`  Dem_IndicatorStatusType* IndicatorStatus`
	`)`	

Service ID[hex]:	0x29	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	IndicatorId	Number of indicator
Parameters (inout):	None	
Parameters (out):	IndicatorStatus	Status of the indicator, like off, on, or blinking.
Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Description:	Gets the indicator status derived from the event status.	

⌋()

8.3.3.18 Dem_GetEventFreezeFrameData

[Dem558]

⌈

Service name:	Dem_GetEventFreezeFrameData	

Syntax:	`Std_ReturnType Dem_GetEventFreezeFrameData(`
	`  Dem_EventIdType EventId,`
	`  uint8 RecordNumber,`
	`  boolean ReportTotalRecord,`
	`  uint16 DataId,`
	`  uint8* DestBuffer`
	`)`	

Service ID[hex]:	0x31	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
	RecordNumber	This parameter is a unique identifier for a freeze frame record as defined in ISO15031-5 and ISO14229-1. **0xFF means most recent freeze frame record is returned.**
	ReportTotalRecord	TRUE: total freeze frame record (all PIDs/DIDs) data are requested FALSE: a dedicated PID/DID is requested by the parameter DataId
	DataId	This parameter specifies the PID (ISO15031-5) or data identifier (ISO14229-1) that shall be copied to the destination buffer. If ReportTotalRecord is TRUE, the value of DataId is ignored.
Parameters (inout):	None	

Parameters (out):	DestBuffer	This parameter contains a byte pointer that points to the buffer, to which the freeze frame data record shall be written to. The format is raw hexadecimal values and contains no header-information.
Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	Gets the data of a freeze frame by event.	

⌋()

8.3.3.19 Dem_GetEventExtendedDataRecord

[Dem557]

⌈

Service name:	Dem_GetEventExtendedDataRecord	

Syntax:	`Std_ReturnType Dem_GetEventExtendedDataRecord(`
	`  Dem_EventIdType EventId,`
	`  uint8 RecordNumber,`
	`  uint8* DestBuffer`
	`)`	

Service ID[hex]:	0x30	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
	RecordNumber	Identification of requested Extended data record. Valid values are between 0x01 and 0xEF as defined in ISO14229-1. 0xFF means data of all extended data records are returned.
Parameters (inout):	None	
Parameters (out):	DestBuffer	This parameter contains a byte pointer that points to the buffer, to which the extended data shall be written to. The format is raw hexadecimal values and contains no header-information.
Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	Gets the data of an extended data record by event.	

⌋()

8.3.3.20 Dem_GetEventMemoryOverflow

[Dem559]

⌈

Service name:	Dem_GetEventMemoryOverflow

Syntax:	`Std_ReturnType Dem_GetEventMemoryOverflow(`
	`  Dem_DTCOriginType DTCOrigin,`
	`  boolean* OverflowIndication`
	`)`

Service ID[hex]:	0x32
Sync/Async:	Synchronous
Reentrancy:	Reentrant

Parameters (in):	DTCOrigin	If the Dem supports more than one event memory this parameter is used to select the source memory the overflow indication shall be read from.
Parameters (inout):	None	
Parameters (out):	OverflowIndication	This parameter returns TRUE if the according event memory was overflowed, otherwise it returns FALSE.
Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Description:	Gets the event memory overflow indication status.	

」(BSW04066)

8.3.3.21 Dem_SetDTCSuppression

[Dem589]

「

Service name:	Dem_SetDTCSuppression	

Syntax:	`Std_ReturnType Dem_SetDTCSuppression(`
	`  uint32 DTC,`
	`  Dem_DTCFormatType DTCFormat,`
	`  boolean SuppressionStatus`
	`)`	

Service ID[hex]:	0x33	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	Diagnostic Trouble Code
	DTCFormat	Defines the input-format of the provided DTC value.
	SuppressionStatus	This parameter specifies whether the respective DTC shall be disabled (TRUE) or enabled (FALSE).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK (Operation was successful), E_NOT_OK (operation failed or event entry for this DTC still exists)
Description:	Set the suppression status of a specific DTC.	

」()

8.3.3.22 Service Interface DiagnosticMonitor

[Dem598] 「 The *Dem Service Component* shall provide the interface *DiagnosticMonitor* as defined below (to provide the capability to modify the event information).」()

One port of this interface type is provided per application-related diagnostic event by the *Dem Service Component*. It has EventId as a port-defined argument.

Note: Each port of the *DiagnosticMonitor* interface is only connected to one monitor port.

```
ClientServerInterface DiagnosticMonitor {
   PossibleErrors {
      E_NOT_OK = 1
   }
   SetEventStatus(
      IN Dem_EventStatusType EventStatus,
      ERR{E_NOT_OK});
   ResetEventStatus(
      ERR{E_NOT_OK});
   PrestoreFreezeFrame(
      ERR{E_NOT_OK}); // OPTIONAL for non-OBD Dem
   ClearPrestoredFreezeFrame(
      ERR{E_NOT_OK}); // OPTIONAL for non-OBD Dem
   // SetEventDisabled shall only exist, if DemOBDSupport is enabled
   SetEventDisabled(
      ERR{E_NOT_OK});
}

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

// for each port providing the interface DiagnosticMonitor:
PortArgument {port=Event_<EventName>, value.type=Dem_EventIdType,
              value.value=<n>, where <n> = 1..<N>}

RunnableEntity SetEventStatus
      symbol "Dem_SetEventStatus"
      canbeInvokedConcurrently = TRUE
      SSCP = port CBStatusEvt_<EventName>_*, EventStatusChanged
      SSCP = port GeneralCBStatusEvt, EventStatusChanged
      SSCP = port CBStatusDTC_*, DTCStatusChanged
      optional SSCP = port CBDataEvt_<EventName>, EventDataChanged
      optional SSCP = port GeneralCBDataEvt, EventDataChanged
      optional SSCP = port CBReadData_*, ReadData
RunnableEntity ResetEventStatus
      symbol "Dem_ResetEventStatus"
      canbeInvokedConcurrently = TRUE
      SSCP = port CBStatusEvt_<EventName>_*, EventStatusChanged
      SSCP = port GeneralCBStatusEvt, EventStatusChanged
      SSCP = port CBStatusDTC_*, DTCStatusChanged
RunnableEntity PrestoreFreezeFrame
      symbol "Dem_PrestoreFreezeFrame"
      canbeInvokedConcurrently = TRUE
      SSCP = port CBReadData_*, ReadData
RunnableEntity ClearPrestoredFreezeFrame
      symbol "Dem_ClearPrestoredFreezeFrame"
      canbeInvokedConcurrently = TRUE
RunnableEntity SetEventDisabled
      symbol "Dem_SetEventDisabled"
      canbeInvokedConcurrently = TRUE
```

8.3.3.23 Service Interface DiagnosticInfo & General

[Dem599]「The *Dem Service Component* shall provide the interface *DiagnosticInfo* as defined below (to provide the capability to obtain the event information).」()

One port of this interface type is provided per diagnostic event by the *Dem Service Component*. It has EventId as a port-defined argument.

Note: These DiagnosticInfo ports are also available for each BSW event (so that also SW-Cs can access on).

[Dem600]「 The *Dem Service Component* shall provide the interface *GeneralDiagnosticInfo* as defined below (to also provide the capability to obtain event information, but in comparision with the *DiagnosticInfo* interface, it is provided for those SW-Cs, which must access on these data uniformly, refer to chapter 7.8.1).」()

One global port of this interface type is provided by the *Dem Service Component*.

```
ClientServerInterface DiagnosticInfo {
   PossibleErrors {
      E_NOT_OK = 1,
      DEM_E_NO_DTC_AVAILABLE = 2
   }
   GetEventStatus(
      OUT Dem_EventStatusExtendedType EventStatusExtended,
      ERR{E_NOT_OK});
   GetEventFailed(
      OUT boolean EventFailed,
      ERR{E_NOT_OK});
   GetEventTested(
      OUT boolean EventTested,
      ERR{E_NOT_OK});
   GetDTCOfEvent(
      IN Dem_DTCFormatType DTCFormat,
      OUT uint32 DTCOfEvent,
      ERR{E_NOT_OK, DEM_E_NO_DTC_AVAILABLE});
   GetFaultDetectionCounter(
      OUT sint8 FaultDetectionCounter,
      ERR{E_NOT_OK});
   GetEventFreezeFrameData(
      IN uint8 RecordNumber,
      IN boolean ReportTotalRecord,
      IN uint16 DataId,
      OUT Dem_MaxDataValueType DestBuffer,
      ERR{E_NOT_OK});
   GetEventExtendedDataRecord(
      IN uint8 RecordNumber,
      OUT Dem_MaxDataValueType DestBuffer,
      ERR{E_NOT_OK});
}

ProvidePort DiagnosticInfo EvtInfo_<EventName>;
ProvidePort DiagnosticInfo EvtInfo_<EventName>;
…
```

```
// for each port providing the interface DiagnosticInfo:
PortArgument {port= EvtInfo_<EventName>, value.type=Dem_EventIdType,
              value.value=<n>, where <n> = 1..<N>}

ClientServerInterface GeneralDiagnosticInfo {
   PossibleErrors {
      E_NOT_OK = 1,
      DEM_E_NO_DTC_AVAILABLE = 2
   }
   GetEventStatus(
      IN Dem_EventIdType EventId,
      OUT Dem_EventStatusExtendedType EventStatusExtended,
      ERR{E_NOT_OK});
   GetEventFailed(
      IN Dem_EventIdType EventId,
      OUT boolean EventFailed,
      ERR{E_NOT_OK});
   GetEventTested(
      IN Dem_EventIdType EventId,
      OUT boolean EventTested,
      ERR{E_NOT_OK});
   GetDTCOfEvent(
      IN Dem_EventIdType EventId,
      IN Dem_DTCFormatType DTCFormat,
      OUT uint32 DTCOfEvent,
      ERR{E_NOT_OK, DEM_E_NO_DTC_AVAILABLE});
   GetFaultDetectionCounter(
      IN Dem_EventIdType EventId,
      OUT sint8 FaultDetectionCounter,
      ERR{E_NOT_OK});
   GetEventFreezeFrameData(
      IN Dem_EventIdType EventId,
      IN uint8 RecordNumber,
      IN boolean ReportTotalRecord,
      IN uint16 DataId,
      OUT Dem_MaxDataValueType DestBuffer,
      ERR{E_NOT_OK});
   GetEventExtendedDataRecord(
      IN Dem_EventIdType EventId,
      IN uint8 RecordNumber,
      OUT Dem_MaxDataValueType DestBuffer,
      ERR{E_NOT_OK});
}

ProvidePort GeneralDiagnosticInfo GeneralEvtInfo;

RunnableEntity GetEventStatus
      symbol "Dem_GetEventStatus"
      canbeInvokedConcurrently = TRUE
RunnableEntity GetEventFailed
      symbol "Dem_GetEventFailed"
      canbeInvokedConcurrently = TRUE
RunnableEntity GetEventTested
      symbol "Dem_GetEventTested"
      canbeInvokedConcurrently = TRUE
RunnableEntity GetDTCOfEvent
      symbol "Dem_GetDTCOfEvent"
      canbeInvokedConcurrently = TRUE
RunnableEntity GetFaultDetectionCounter
      symbol "Dem_GetFaultDetectionCounter"
```

```
      canbeInvokedConcurrently = TRUE
      SSCP = port CBFaultDetectCtr_*, GetFaultDetectionCounter
RunnableEntity GetEventFreezeFrameData
      symbol "Dem_GetEventFreezeFrameData"
      canbeInvokedConcurrently = TRUE
RunnableEntity GetEventExtendedDataRecord
      symbol "Dem_ GetEventExtendedDataRecord"
      canbeInvokedConcurrently = TRUE
```

8.3.3.24 Service Interface OperationCycle

[Dem601] ⌈The *Dem Service Component* shall provide the interface *OperationCycle* as defined below (to provide the capability to set either the state or the counter value of an operation cycle).⌋()

One port of this interface type is provided per operation cycle by the *Dem Service Component*. It has OperationCycleId as a port-defined argument.

```
ClientServerInterface OperationCycle {
   PossibleErrors {
      E_NOT_OK = 1
   }
   // SetOperationCycleState shall only exist,
   // if DemOperationCycleProcessing = DEM_PROCESS_OPCYC_STATE
   SetOperationCycleState(
      IN Dem_OperationCycleStateType CycleState,
      ERR{E_NOT_OK});
   // SetOperationCycleCntValue shall only exist,
   // if DemOperationCycleProcessing = DEM_PROCESS_OPCYC_COUNTER
   SetOperationCycleCntValue(
      IN uint8 CounterValue,
      ERR{E_NOT_OK});
}

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

// for each port providing the interface OperationCycle:
PortArgument {port=OpCycle_<CycleName>,
              value.type=Dem_OperationCycleIdType,
              value.value=<n>, where <n> = 0..<N - 1>}

RunnableEntity SetOperationCycleState
      symbol "Dem_SetOperationCycleState"
      canbeInvokedConcurrently = TRUE
      SSCP = port CBInitEvt_<EventName>, InitMonitorForEvent
      SSCP = port CBStatusEvt_*, EventStatusChanged
      SSCP = port GeneralCBStatusEvt, EventStatusChanged
      SSCP = port CBStatusDTC_*, DTCStatusChanged
RunnableEntity SetOperationCycleCntValue
      symbol "Dem_SetOperationCycleCntValue"
      canbeInvokedConcurrently = TRUE
      SSCP = port CBInitEvt_<EventName>, InitMonitorForEvent
      SSCP = port CBStatusEvt_*, EventStatusChanged
```

```
      SSCP = port GeneralCBStatusEvt, EventStatusChanged
      SSCP = port CBStatusDTC_*, DTCStatusChanged
```

8.3.3.25 Service Interface AgingCycle

[Dem602] ⌈The *Dem Service Component* shall provide the interface *AgingCycle* as defined below (to provide the capability to set an aging cycle state).⌋()

One port of this interface type is provided per aging cycle by the *Dem Service Component*. It has AgingCycleId as a port-defined argument.

```
ClientServerInterface AgingCycle {
   PossibleErrors {
      E_NOT_OK = 1
   }
   SetAgingCycleState(
      ERR{E_NOT_OK});
}

ProvidePort AgingCycle AgingCycle_<CycleName>;
ProvidePort AgingCycle AgingCycle_<CycleName>;
…

// for each port providing the interface AgingCycle:
PortArgument {port=AgingCycle_<CycleName>,
              value.type=Dem_OperationCycleIdType,
              value.value=<n>, where <n> = 0..<N - 1>}

RunnableEntity SetAgingCycleState
      symbol "Dem_SetAgingCycleState"
      canbeInvokedConcurrently = TRUE
      SSCP = port CBStatusEvt_*, EventStatusChanged
      SSCP = port GeneralCBStatusEvt, EventStatusChanged
      SSCP = port CBStatusDTC_*, DTCStatusChanged
```

8.3.3.26 Service Interface ExternalAgingCycle

[Dem603] ⌈ The *Dem Service Component* shall provide the interface *ExternalAgingCycle* as defined below (to provide the capability to set the current value of the aging cycle counter Dem-externally).⌋()

One global port of this interface type is provided by the *Dem Service Component*.

```
ClientServerInterface ExternalAgingCycle {
   PossibleErrors {
      E_NOT_OK = 1
   }
   SetAgingCycleCounterValue(
      IN uint8 CounterValue,
      ERR{E_NOT_OK});
```

```
}

ProvidePort ExternalAgingCycle ExtAgingCycle;

RunnableEntity SetAgingCycleCounterValue
      symbol "Dem_SetAgingCycleCounterValue"
      canbeInvokedConcurrently = FALSE
      SSCP = port CBStatusEvt_*, EventStatusChanged
      SSCP = port GeneralCBStatusEvt, EventStatusChanged
      SSCP = port CBStatusDTC_*, DTCStatusChanged
```

8.3.3.27 Service Interface EnableCondition

[Dem604] ⌈The *Dem Service Component* shall provide the interface *EnableCondition* as defined below (to provide the capability to set an enable condition), if at least one enable condition is configured.⌋()

One port of this interface type is provided per enable condition by the *Dem Service Component*. It has EnableConditionId as a port-defined argument.

```
ClientServerInterface EnableCondition {
   PossibleErrors {
      E_NOT_OK = 1
   }
   SetEnableCondition(
      IN boolean ConditionFulfilled,
      ERR{E_NOT_OK});
}

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

// for each port providing the interface EnableCondition:
PortArgument {port=EnableCond_<ConditionName>, value.type=uint8,
              value.value=<n>, where <n> = 0..<N - 1>}

RunnableEntity SetEnableCondition
      symbol "Dem_SetEnableCondition"
      canbeInvokedConcurrently = TRUE
```

8.3.3.28 Service Interface StorageCondition

[Dem605] ⌈ The *Dem Service Component* shall provide the interface *StorageCondition* as defined below (to provide the capability to set an enable condition) if at least one storage condition is configured.⌋()

One port of this interface type is provided per storage condition by the *Dem Service Component*. It has StorageConditionId as a port-defined argument.

```
ClientServerInterface StorageCondition {
   PossibleErrors {
      E_NOT_OK = 1
   }
   SetStorageCondition(
      IN boolean ConditionFulfilled,
      ERR{E_NOT_OK});
}

ProvidePort StorageCondition StorageCond_<ConditionName>;
ProvidePort StorageCondition StorageCond_<ConditionName>;
…

// for each port providing the interface StorageCondition:
PortArgument {port= StorageCond_<ConditionName>, value.type=uint8,
              value.value=<n>, where <n> = 0..<N - 1>}

RunnableEntity SetStorageCondition
      symbol "Dem_SetStorageCondition"
      canbeInvokedConcurrently = TRUE
```

8.3.3.29 Service Interface IndicatorStatus

[Dem606] ⌈The *Dem Service Component* shall provide the interface *IndicatorStatus* as defined below (to provide the capability to set the status of an indicator), if at least one indicator is configured.⌋()

One port of this interface type is provided per indicator by the *Dem Service Component*. It has IndicatorId as a port-defined argument.

```
ClientServerInterface IndicatorStatus {
   PossibleErrors {
      E_NOT_OK = 1
   }
   GetIndicatorStatus(
      OUT Dem_IndicatorStatusType IndicatorStatus,
      ERR{E_NOT_OK});
}

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

// for each port providing the interface IndicatorStatus:
PortArgument {port=IndStatus_<IndicatorName>,
              value.type=Dem_IndicatorIdType,
              value.value=<n>, where <n> = 0..<N - 1>}

RunnableEntity GetIndicatorStatus
      symbol "Dem_GetIndicatorStatus"
      canbeInvokedConcurrently = TRUE
```

8.3.3.30 Service Interface EvMemOverflowIndication

[Dem607] ⌈ The *Dem Service Component* shall provide the interface *EvMemOverflowIndication* as defined below (to provide the status of the event memory), if the respective event memory is configured.⌋()

One port of this interface type is provided per event memory by the *Dem Service Component*. It has DTCOrigin as a port-defined argument.

```
ClientServerInterface EvMemOverflowIndication {
   PossibleErrors {
      E_NOT_OK = 1
   }
   GetEventMemoryOverflow(
      OUT boolean OverflowIndication,
      ERR{E_NOT_OK});
}

ProvidePort EvMemOverflowIndication OverflowIndPrimaryMemory
ProvidePort EvMemOverflowIndication OverflowIndSecondaryMemory
ProvidePort EvMemOverflowIndication OverflowIndMirrorMemory
ProvidePort EvMemOverflowIndication OverflowIndPermanentMemory

// for each port providing the interface EvMemOverflowIndication:
PortArgument {port=OverflowIndPrimaryMemory,
              value.type=Dem_DTCOriginType,
              value.value=DEM_DTC_ORIGIN_PRIMARY_MEMORY}
PortArgument {port=OverflowIndSecondaryMemory,
              value.type=Dem_DTCOriginType,
              value.value=DEM_DTC_ORIGIN_SECONDARY_MEMORY}
PortArgument {port=OverflowIndMirrorMemory,
              value.type=Dem_DTCOriginType,
              value.value=DEM_DTC_ORIGIN_MIRROR_MEMORY}
PortArgument {port=OverflowIndPermanentMemory,
              value.type=Dem_DTCOriginType,
              value.value=DEM_DTC_ORIGIN_PERMANENT_MEMORY}

RunnableEntity GetEventMemoryOverflow
      symbol "Dem_GetEventMemoryOverflow"
      canbeInvokedConcurrently = TRUE
```

8.3.3.31 Service Interface DTCSuppression

[Dem608] ⌈ The *Dem Service Component* shall provide the interface *DTCSuppression* as defined below (to provide the capability to control the suppression of DTCs).⌋()

One port of this interface type is provided globally by the *Dem Service Component*.

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

```
   SetDTCSuppression(
      IN uint32 DTC,
      IN Dem_DTCFormatType DTCFormat,
      IN boolean SuppressionStatus,
      ERR{E_NOT_OK});
}

ProvidePort DTCSuppression ControlDTCSuppression

RunnableEntity SetDTCSuppression
      symbol "Dem_SetDTCSuppression"
      canbeInvokedConcurrently = FALSE
```

8.3.4 Interface Dcm ⇔ Dem

8.3.4.1 Access DTCs and Status Information

8.3.4.1.1 Dem_SetDTCFilter

[Dem208]

⌈

Service name:	Dem_SetDTCFilter	

Syntax:	`Dem_ReturnSetFilterType Dem_SetDTCFilter(`
	`  uint8 DTCStatusMask,`
	`  Dem_DTCKindType DTCKind,`
	`  Dem_DTCFormatType DTCFormat,`
	`  Dem_DTCOriginType DTCOrigin,`
	`  Dem_FilterWithSeverityType FilterWithSeverity,`
	`  Dem_DTCSeverityType DTCSeverityMask,`
	`  Dem_FilterForFDCType FilterForFaultDetectionCounter`
	`)`	

Service ID[hex]:	0x13	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTCStatusMask	Status-byte mask for DTC status-byte filtering Values: 0x00: Autosar-specific value to deactivate the status-byte filtering (different meaning than in ISO 14229-1) to report all supported DTCs (used for service 0x19 subfunctions 0x0A/0x15) 0x01..0xFF: Status-byte mask according to ISO 14229-1 DTCStatusMask (handed over by Dcm from service request directly) to filter for DTCs with at least one status bit set matching this status-byte mask
	DTCKind	Defines the functional group of DTCs to be reported (e.g. all DTC, OBD-relevant DTC)
	DTCFormat	Defines the output-format of the requested DTC values for the sub-sequent API calls.

	DTCOrigin	If the Dem supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.
	FilterWithSeverity	This flag defines whether severity information (ref. to parameter below) shall be used for filtering. This is to allow for coexistence of DTCs with and without severity information.
	DTCSeverityMask	This parameter contains the DTCSeverityMask according to ISO14229-1 (see for example Service 0x19, subfunction 0x08)
	FilterForFaultDetectionCounter	This flag defines whether the 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 use the debouncing inside Dem, then the Dem must request this information via GetFaultDetectionCounter.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_ReturnSetFilterType	Status of the operation to (re-)set a DTC filter.
Description:	The server shall perform a bit-wise logical AND-ing operation between the mask specified in the client's request and the actual status associated with each DTC supported by the server. In addition to the DTCStatusAvailabilityMask, the server shall return all DTCs for which the result of the AND-ing operation is non-zero [i.e. (statusOfDTC & DTCStatusMask) != 0]. If the client specifies a status mask that contains bits that the server does not support, then the server shall process the DTC information using only the bits that it does support. If no DTCs within the server match the masking criteria specified in the client's request, no DTC or status information shall be provided following the DTCStatusAvailabilityMask byte in the positive response message (statusOfDTC & DTCStatusMask) & (severity & DTCSeverityMask) != 0	

⌋(BSW04066)

8.3.4.1.2 Dem_SetFreezeFrameRecordFilter

[Dem209]

⌈

Service name:	Dem_SetFreezeFrameRecordFilter

Syntax:	`Dem_ReturnSetFilterType Dem_SetFreezeFrameRecordFilter(`
	`  Dem_DTCFormatType DTCFormat,`
	`  uint16* NumberOfFilteredRecords`
	`)`

Service ID[hex]:	0x3f
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	DTCFormat	Defines the output-format of the requested DTC values for the sub-sequent API calls.
Parameters (inout):	None	
Parameters (out):	NumberOfFilteredRecords	Number of freeze frame records currently stored in the event memory.
Return value:	Dem_ReturnSetFilterType	Status of the operation to (re-)set a freeze frame record filter.
Description:	Sets a freeze frame record filter.	

⌋()

8.3.4.1.3 Dem_GetStatusOfDTC

[Dem212]

⌈

Service name:	Dem_GetStatusOfDTC	

Syntax:	`Dem_ReturnGetStatusOfDTCType Dem_GetStatusOfDTC(`
	`  uint32 DTC,`
	`  Dem_DTCOriginType DTCOrigin,`
	`  uint8* DTCStatus`
	`)`	

Service ID[hex]:	0x15	
Sync/Async:	Synchronous or Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	Diagnostic Trouble Code in UDS format.
	DTCOrigin	If the Dem supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.
Parameters (inout):	None	
Parameters (out):	DTCStatus	This parameter receives the status information of the requested DTC. If the return value of the function call is other than DEM_STATUS_OK this parameter does not contain valid data. 0x00...0xFF match DTCStatusMask as defined in ISO14229-1
Return value:	Dem_ReturnGetStatusOfDTCType	Status of the operation of type Dem_ReturnGetStatusOfDTCType.
Description:	Gets the status of a DTC. For large configurations and DTC-calibration, the interface behavior can be asynchronous (splitting the DTC-search into segments).	

⌋(BSW04066)

8.3.4.1.4 Dem_GetDTCStatusAvailabilityMask

[Dem213]

⌈

Service name:	Dem_GetDTCStatusAvailabilityMask
Syntax:	`Std_ReturnType Dem_GetDTCStatusAvailabilityMask(`

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

⌋()

8.3.4.1.5 Dem_GetNumberOfFilteredDTC

[Dem214]

⌈

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

⌋()

8.3.4.1.6 Dem_GetNextFilteredDTC

[Dem215]

⌈

Service name:	Dem_GetNextFilteredDTC
Syntax:	Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredDTC(uint32* DTC, uint8* DTCStatus)
Service ID[hex]:	0x18

Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	DTC	Receives the DTC value in respective format of the filter returned by this function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.
	DTCStatus	This parameter receives the status information of the requested DTC. It follows the format as defined in ISO14229-1 If the return value of the function call is other than DEM_FILTERED_OK this parameter does not contain valid data.
Return value:	Dem_ReturnGetNextFilteredDTCType	Status of the operation to retrieve a DTC from the Dem. The value DEM_FILTERED_PENDING is never returned (because of synchronous behavior).
Description:	Gets the next filtered DTC.	

⌋()

8.3.4.1.7 Dem_GetDTCByOccurrenceTime

[Dem218]

⌈

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

⌋()

8.3.4.1.8 Dem_GetNextFilteredRecord

[Dem224]

⌈

Service name:	Dem_GetNextFilteredRecord	

Syntax:	`Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredRecord(`
	`  uint32* DTC,`
	`  uint8* RecordNumber`
	`)`	

Service ID[hex]:	0x3a	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	DTC	Receives the DTC value in respective format of the filter returned by this function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.
	RecordNumber	Freeze frame record number of the reported DTC (relative addressing). If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.
Return value:	Dem_ReturnGetNextFilteredDTCType	Status of the operation to retrieve a DTC and its associated snapshot record number from the Dem.
Description:	Gets the next freeze frame record number and its associated DTC stored in the event memory. The interface has an asynchronous behavior, because NvRAM access might be required.	

⌋()

8.3.4.1.9 Dem_GetNextFilteredDTCAndFDC

[Dem227]

⌈

Service name:	Dem_GetNextFilteredDTCAndFDC

Syntax:	`Dem_ReturnGetNextFilteredDTCType`
	`Dem_GetNextFilteredDTCAndFDC(`
	`  uint32* DTC,`
	`  sint8* DTCFaultDetectionCounter`
	`)`

Service ID[hex]:	0x3b
Sync/Async:	Asynchronous
Reentrancy:	Non Reentrant
Parameters (in):	None

Parameters (inout):	None	
Parameters (out):	DTC	Receives the DTC value in respective format of the filter returned by this function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.
	DTCFaultDetectionCounter	This parameter receives the Fault Detection Counter information of the requested DTC. If the return value of the function call is other than DEM_FILTERED_OK this parameter does not contain valid data. -128dec...127dec PASSED...FAILED according to ISO 14229-1
Return value:	Dem_ReturnGetNextFilteredDTCType	Status of the operation to retrieve a DTC from the Dem.
Description:	Gets the current DTC and its associated Fault Detection Counter (FDC) from the Dem. The interface has an asynchronous behavior, because the FDC might be received asynchronously from a SW-C, too.	

⌋()

8.3.4.1.10 Dem_GetNextFilteredDTCAndSeverity

[Dem281]

⌈

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

	DTCFunctionalUnit	Receives the functional unit value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.
Return value:	Dem_ReturnGetNextFilteredDTCType	Status of the operation to retrieve a DTC from the Dem. The value DEM_FILTERED_PENDING is never returned (because of synchronous behavior).
Description:	Gets the current DTC and its Severity from the Dem.	

⌋()

8.3.4.1.11 Dem_GetTranslationType

[Dem230]

⌈

Service name:	Dem_GetTranslationType	
Syntax:	`Dem_DTCTranslationFormatType Dem_GetTranslationType( void )`	
Service ID[hex]:	0x3c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_DTCTranslationFormatType	Returns the configured DTC translation format. A combination of different DTC formats is not possible.
Description:	Gets the supported DTC formats of the ECU. The supported formats are configured via DemTypeOfDTCSupported.	

⌋()

8.3.4.1.12 Dem_GetSeverityOfDTC

[Dem232]

⌈

Service name:	Dem_GetSeverityOfDTC	
Syntax:	`Dem_ReturnGetSeverityOfDTCType Dem_GetSeverityOfDTC( uint32 DTC, Dem_DTCSeverityType* DTCSeverity )`	
Service ID[hex]:	0x0e	
Sync/Async:	Synchronous or Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	Diagnostic Trouble Code in UDS format.
Parameters	None	

(inout):		
Parameters (out):	DTCSeverity	This parameter contains the DTCSeverityMask according to ISO14229-1.
Return value:	Dem_ReturnGetSeverityOfDTCType	Status of the operation of type Dem_ReturnGetSeverityOfDTCType.
Description:	Gets the severity of the requested DTC. For large configurations and DTC-calibration, the interface behavior can be asynchronous (splitting the DTC-search into segments).	

⌋()

8.3.4.1.13 Dem_GetFunctionalUnitOfDTC

[Dem594]

⌈

Service name:	Dem_GetFunctionalUnitOfDTC	

Syntax:	`Dem_ReturnGetFunctionalUnitOfDTCType Dem_GetFunctionalUnitOfDTC(`
	`  uint32 DTC,`
	`  uint8* DTCFunctionalUnit`
	`)`	

Service ID[hex]:	0x34	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	Diagnostic Trouble Code in UDS format.
Parameters (inout):	None	
Parameters (out):	DTCFunctionalUnit	Functional unit value of this DTC
Return value:	Dem_ReturnGetFunctionalUnitOfDTCType	Status of the operation of type Dem_ReturnGetFunctionalUnitOfDTCType.
Description:	Gets the functional unit of the requested DTC.	

⌋()

8.3.4.2 Access extended data records and FreezeFrame data

8.3.4.2.1 Dem_DisableDTCRecordUpdate

[Dem233]

⌈

Service name:	Dem_DisableDTCRecordUpdate

Syntax:	`Dem_ReturnDisableDTCRecordUpdateType`
	`Dem_DisableDTCRecordUpdate(`
	`  uint32 DTC,`
	`  Dem_DTCOriginType DTCOrigin`
	`)`

Service ID[hex]:	0x1a

Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	Selects the DTC in UDS format, for which DTC record update shall be disabled.
	DTCOrigin	If the Dem supports more than one event memory, this parameter is used to select the source memory for which DTC record update shall be disabled.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_ReturnDisableDTCRecordUpdateType	Status of the operation to disable the event memory update of a specific DTC.
Description:	Disables the event memory update of a specific DTC (only one at one time).	

」()

8.3.4.2.2 Dem_EnableDTCRecordUpdate

[Dem234]

「

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

」()

8.3.4.2.3 Dem_GetFreezeFrameDataByRecord

[Dem235]

「

Service name:	Dem_GetFreezeFrameDataByRecord
Syntax:	`Dem_ReturnGetFreezeFrameDataByRecordType Dem_GetFreezeFrameDataByRecord( uint8 RecordNumber,`

	Dem_DTCOriginType DTCOrigin, uint32* DTC, uint8* DestBuffer, uint16* BufSize)	
Service ID[hex]:	0x1c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	RecordNumber	This parameter is a unique identifier for a freeze frame record as defined in ISO 15031-5 and ISO 14229-1. This record number is unique per ECU (absolute addressing). The value 0xFF is not allowed. The value 0x00 indicates the complete OBD freeze frame.
	DTCOrigin	If the Dem supports more than one event memory, this parameter is used to select the source memory the DTCs shall be read from.
Parameters (inout):	BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Parameters (out):	DTC	Receives the DTC value in UDS format returned by this function. If the return value of the function is other than DEM_GET_FFBYRECORD_OK this parameter does not contain valid data.
	DestBuffer	This parameter contains a byte pointer that points to the buffer, to which the freeze frame data record shall be written to. The format is: {NumOfDIDs, DID[1], data[1], ..., DID[N], data[N]}.
Return value:	Dem_ReturnGetFreezeFrameDataByRecordType	Status of the operation to retrieve the DTC and its associated freeze frame record
Description:	Gets DTC and its associated freeze frame record by absolute record number. The function stores the data in the provided DestBuffer.	

⌋(BSW04066)

8.3.4.2.4 Dem_GetFreezeFrameDataByDTC

[Dem236]

⌈

Service	Dem_GetFreezeFrameDataByDTC

name:		
Syntax:	`Dem_ReturnGetFreezeFrameDataByDTCType Dem_GetFreezeFrameDataByDTC( uint32 DTC, Dem_DTCOriginType DTCOrigin, uint8 RecordNumber, uint8* DestBuffer, uint16* BufSize )`	
Service ID[hex]:	0x1d	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	Diagnostic Trouble Code in UDS format.
	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 freeze frame record as defined in ISO 15031-5 and ISO 14229-1. This record number is unique per DTC (relative addressing). The value 0xFF is not allowed. The value 0x00 indicates the DTC-specific OBD freeze frame.
Parameters (inout):	BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Parameters (out):	DestBuffer	This parameter contains a byte pointer that points to the buffer, to which the freeze frame data record shall be written to. The format is: {RecordNumber, NumOfDIDs, DID[1], data[1], ..., DID[N], data[N]}
Return value:	Dem_ReturnGetFreezeFrameDataByDTCType	Status of the operation to retrieve freeze frame data by DTC.
Description:	Gets freeze frame data by DTC. The function stores the data in the provided DestBuffer.	

⌋(BSW04066)

8.3.4.2.5 Dem_GetSizeOfFreezeFrameByDTC

[Dem238]

⌈

Service name:	Dem_GetSizeOfFreezeFrameByDTC
Syntax:	`Dem_ReturnGetSizeOfFreezeFrameByDTCType Dem_GetSizeOfFreezeFrameByDTC( uint32 DTC, Dem_DTCOriginType DTCOrigin, uint8 RecordNumber, uint16* SizeOfFreezeFrame )`
Service ID[hex]:	0x1f

Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	Diagnostic Trouble Code in UDS format.
	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 freeze frame record as defined in ISO 15031-5 and ISO 14229-1. This record number is unique per DTC (relative addressing). The value 0xFF is explicitly allowed to request the overall size.
Parameters (inout):	None	
Parameters (out):	SizeOfFreezeFrame	Number of bytes in the requested freeze frame record.
Return value:	Dem_ReturnGetSizeOfFreezeFrameByDTCType	Status of the operation to retrieve the size of freeze frame data.
Description:	Gets the size of freeze frame data by DTC.	

⌋(BSW04066)

8.3.4.2.6 Dem_GetExtendedDataRecordByDTC

[Dem239]

⌈

Service name:	Dem_GetExtendedDataRecordByDTC	
Syntax:	`Dem_ReturnGetExtendedDataRecordByDTCType Dem_GetExtendedDataRecordByDTC( uint32 DTC, Dem_DTCOriginType DTCOrigin, uint8 ExtendedDataNumber, uint8* DestBuffer, uint16* BufSize )`	
Service ID[hex]:	0x20	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	Diagnostic Trouble Code in UDS format.
	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/Number of requested

		extended data record. The values 0xFE and 0xFF are not allowed. Valid values are from 0x01 to 0xEF.
Parameters (inout):	BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Parameters (out):	DestBuffer	This parameter contains a byte pointer that points to the buffer, to which the extended data record shall be written to. The format is raw hexadecimal values and contains no header-information.
Return value:	Dem_ReturnGetExtendedDataRecordByDTCType	Status of the operation to retrieve extended data by DTC.
Description:	Gets extended data by DTC. The function stores the data in the provided DestBuffer.	

⌋(BSW04066)

8.3.4.2.7 Dem_GetSizeOfExtendedDataRecordByDTC

[Dem240]

⌈

Service name:	Dem_GetSizeOfExtendedDataRecordByDTC		
Syntax:	`Dem_ReturnGetSizeOfExtendedDataRecordByDTCType Dem_GetSizeOfExtendedDataRecordByDTC( uint32 DTC, Dem_DTCOriginType DTCOrigin, uint8 ExtendedDataNumber, uint16* SizeOfExtendedDataRecord )`		
Service ID[hex]:	0x21		
Sync/Async:	Asynchronous		
Reentrancy:	Non Reentrant		
Parameters (in):	DTC	Diagnostic Trouble Code in UDS format.	
	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/Number of requested extended data record. Valid values are from 0x01 to 0xEF. Additionally the values 0xFE and 0xFF are explicitly allowed to request the overall size of all OBD records / all records.	
Parameters (inout):	None		
Parameters (out):	SizeOfExtendedDataRecord		Pointer to Size of the requested data record
Return value:	Dem_ReturnGetSizeOfExtendedDataRecordByDTCType		Status of the operation to retrieve the size of extended data.

Description:	Gets the size of extended data by DTC.

⌋(BSW04066)

8.3.4.3 DTC storage

8.3.4.3.1 Dem_ClearDTC

[Dem241]

⌈

Service name:	Dem_ClearDTC	

Syntax:	`Dem_ReturnClearDTCType Dem_ClearDTC(`
	`  uint32 DTC,`
	`  Dem_DTCFormatType DTCFormat,`
	`  Dem_DTCOriginType DTCOrigin`
	`)`	

Service ID[hex]:	0x22	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	Defines the DTC in respective format, 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.
	DTCFormat	Defines the input-format of the provided DTC value.
	DTCOrigin	If the Dem supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_ReturnClearDTCType	Status of the operation of type Dem_ReturnClearDTCType.
Description:	Clears single DTCs as well as groups of DTCs. This API can only be used through the RTE, and therefore no declaration is exported via Dem.h.	

⌋(BSW04066)

8.3.4.3.2 Dem_DisableDTCSetting

[Dem242]

⌈

Service name:	Dem_DisableDTCSetting

Syntax:	`Dem_ReturnControlDTCSettingType Dem_DisableDTCSetting(`
	`  Dem_DTCGroupType DTCGroup,`
	`  Dem_DTCKindType DTCKind`
	`)`

Service ID[hex]:	0x24
Sync/Async:	Synchronous

Reentrancy:	Non Reentrant	
Parameters (in):	DTCGroup	Defines the group of DTC that shall be disabled to store in event memory.
	DTCKind	This parameter defines the requested DTC kind, either only OBD-relevant DTCs or all DTCs
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dem_ReturnControlDTCSettingType	Returns status of the operation
Description:	Disables the DTC setting for a DTC group.	

⌋()

8.3.4.3.3 Dem_EnableDTCSetting

[Dem243]

⌈

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

⌋()

8.3.4.4 Dem_DcmCancelOperation

[Dem560]

⌈

Service name:	Dem_DcmCancelOperation
Syntax:	`void Dem_DcmCancelOperation( void )`
Service ID[hex]:	0x2a
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Cancel pending operation started from Dcm.

」()

8.3.4.5 Service Interface DcmIf

[Dem609] 「The *Dem Service Component* shall provide the interface *DcmIf* as defined below (to provide the operations only related to the Dcm).」()

One port of this interface type is provided globally by the *Dem Service Component*.

Note: The port of the *DcmIf* interface is only connected to the respective Dcm port.

```
ClientServerInterface DcmIf {
   PossibleErrors {
      DEM_CLEAR_WRONG_DTC = 1
      DEM_CLEAR_WRONG_DTCORIGIN = 2
      DEM_CLEAR_WRONG_DTCKIND = 3
      DEM_CLEAR_FAILED = 4
      DEM_CLEAR_PENDING = 5
   }
   ClearDTC(
      IN uint32 DTC,
      IN Dem_DTCFormatType DTCFormat,
      IN Dem_DTCOriginType DTCOrigin,
      ERR{DEM_CLEAR_WRONG_DTC, DEM_CLEAR_WRONG_DTCORIGIN,
          DEM_CLEAR_WRONG_DTCKIND, DEM_CLEAR_FAILED, DEM_CLEAR_PENDING});
}

ProvidePort DcmIf Dcm;

RunnableEntity ClearDTC
      symbol "Dem_ClearDTC"
      canbeInvokedConcurrently = FALSE
      SSCP = port CBInitEvt_<EventName>, InitMonitorForEvent
      SSCP = port CBStatusEvt_<EventName>_*, EventStatusChanged
      SSCP = port GeneralCBStatusEvt, EventStatusChanged
      SSCP = port CBStatusDTC_*, DTCStatusChanged
      SSCP = port CBClrEvt_<EventName>, ClearEventAllowed
```

8.3.5 Interface Dlt ⇔ Dem

8.3.5.1 Dem_DltGetMostRecentFreezeFrameRecordData

[Dem636]

⌜

Service name:	Dem_DltGetMostRecentFreezeFrameRecordData	
Syntax:	`Std_ReturnType Dem_DltGetMostRecentFreezeFrameRecordData( Dem_EventIdType EventId, uint8* DestBuffer, uint8* BufSize )`	
Service ID[hex]:	0x41	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
Parameters (inout):	BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Parameters (out):	DestBuffer	This parameter contains a byte pointer that points to the buffer, to which the freeze frame record shall be written to. The format is raw hexadecimal values and contains no header-information.
Return value:	Std_ReturnType	E_OK: Operation was successful. E_NOT_OK: Operation failed.
Description:	Gets the data of an most recent freeze frame record by event.	

⌟(BSW04099)

8.3.5.2 Dem_DltGetAllExtendedDataRecords

[Dem637]

⌜

Service name:	Dem_DltGetAllExtendedDataRecords	
Syntax:	`Std_ReturnType Dem_DltGetAllExtendedDataRecords( Dem_EventIdType EventId, uint8* DestBuffer, uint8* BufSize )`	
Service ID[hex]:	0x40	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
Parameters (inout):	BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Parameters (out):	DestBuffer	This parameter contains a byte pointer that points to the buffer, to which the extended data shall be written to. The format is raw hexadecimal values and contains no header-information.
Return value:	Std_ReturnType	E_OK: Operation was successful. E_NOT_OK: Operation failed.
Description:	Gets the data of all extended data records of an event.	

⌟(BSW04099)

8.3.6 OBD-specific Interfaces

8.3.6.1 Dem_SetEventDisabled

[Dem312]

⌈

Service name:	Dem_SetEventDisabled	

Syntax:	`Std_ReturnType Dem_SetEventDisabled(`
	`  Dem_EventIdType EventId`
	`)`	

Service ID[hex]:	0x51	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK set of event to disabled was successfull. E_NOT_OK set of event disabled failed
Description:	Service for reporting the event as disabled to the Dem for the PID $41 computation.	

⌋()

8.3.6.2 Dem_ReplUMPRFaultDetect

[Dem313]

⌈

Service name:	Dem_ReplUMPRFaultDetect	

Syntax:	`Std_ReturnType Dem_RepIUMPRFaultDetect(`
	`  Dem_RatioIdType RatioID`
	`)`	

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

⌋()

8.3.6.3 Dem_ReplUMPRDenLock

[Dem314]

⌈

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

⌋()

8.3.6.4 Dem_ReplUMPRDenRelease

[Dem315]

⌈

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

⌋()

8.3.6.5 Dem_GetInfoTypeValue08

[Dem316]

⌈

Service name:	Dem_GetInfoTypeValue08	
Syntax:	`Std_ReturnType Dem_GetInfoTypeValue08( Dcm_OpStatusType OpStatus, uint8* Iumprdata08 )`	
Service ID[hex]:	0x6b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	OpStatus	Parameter is required for interface compatibility to Dcm. Only DCM_INITIAL will appear, because this API behaves synchronous.
Parameters (inout):	None	
Parameters (out):	Iumprdata08	Buffer containing the contents of InfoType $08. The buffer is provided by the Dcm.
Return value:	Std_ReturnType	Always E_OK is returned, as E_PENDING and E_NOT_OK will never appear.
Description:	Service is used to request for IUMPR data accoirding InfoType $08.	

⌋()

8.3.6.6 Dem_GetInfoTypeValue0B

[Dem317]

⌈

Service name:	Dem_GetInfoTypeValue0B	
Syntax:	`Std_ReturnType Dem_GetInfoTypeValue0B( Dcm_OpStatusType OpStatus, uint8* Iumprdata0B )`	
Service ID[hex]:	0x6c	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	OpStatus	Parameter is required for interface compatibility to Dcm. Only DCM_INITIAL will appear, because this API behaves synchronous.
Parameters (inout):	None	
Parameters (out):	Iumprdata0B	Buffer containing the contents of InfoType $0B. The buffer is provided by the Dcm.
Return value:	Std_ReturnType	Always E_OK is returned, as E_PENDING and E_NOT_OK will never appear.
Description:	Service is used to request for IUMPR data accoirding InfoType $0B.	

⌋()

8.3.6.7 Dem_DcmReadDataOfPID01

[Dem318]

⌈

Service name:	Dem_DcmReadDataOfPID01	
Syntax:	`Std_ReturnType Dem_DcmReadDataOfPID01( uint8* PID01value )`	
Service ID[hex]:	0x61	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID01value	Buffer containing the contents of PID $01 computed by the Dem. The buffer is provided by the Dcm with the appropriate size, i.e. during configuration, the Dcm identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	Always E_OK is returned, as E_NOT_OK will never appear.
Description:	Service to report the value of PID $01 computed by the Dem.	

⌋()

8.3.6.8 Dem_DcmReadDataOfPID1C

[Dem325]

⌈

Service name:	Dem_DcmReadDataOfPID1C	
Syntax:	`Std_ReturnType Dem_DcmReadDataOfPID1C( uint8* PID1Cvalue )`	
Service ID[hex]:	0x63	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID1Cvalue	Buffer containing the contents of PID $1C computed by the Dem. The buffer is provided by the Dcm with the appropriate size, i.e. during configuration, the Dcm identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	Always E_OK is returned, as E_NOT_OK will never appear.
Description:	Service to report the value of PID $1C computed by the Dem.	

⌋()

8.3.6.9 Dem_DcmReadDataOfPID21

[Dem319]

⌜

Service name:	Dem_DcmReadDataOfPID21	
Syntax:	`Std_ReturnType Dem_DcmReadDataOfPID21( uint8* PID21value )`	
Service ID[hex]:	0x64	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID21value	Buffer containing the contents of PID $21 computed by the Dem. The buffer is provided by the Dcm with the appropriate size, i.e. during configuration, the Dcm identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	Always E_OK is returned, as E_NOT_OK will never appear.
Description:	Service to report the value of PID $21 computed by the Dem.	

⌟()

8.3.6.10 Dem_DcmReadDataOfPID30

[Dem320]

⌜

Service name:	Dem_DcmReadDataOfPID30	
Syntax:	`Std_ReturnType Dem_DcmReadDataOfPID30( uint8* PID30value )`	
Service ID[hex]:	0x65	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID30value	Buffer containing the contents of PID $30 computed by the Dem. The buffer is provided by the Dcm with the appropriate size, i.e. during configuration, the Dcm identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	Always E_OK is returned, as E_NOT_OK will never appear.
Description:	Service to report the value of PID $30 computed by the Dem.	

⌟()

8.3.6.11 Dem_DcmReadDataOfPID31

[Dem321]

⌜

Service name:	Dem_DcmReadDataOfPID31
Syntax:	`Std_ReturnType Dem_DcmReadDataOfPID31(`

	`uint8* PID31value`)	
Service ID[hex]:	0x66	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID31value	Buffer containing the contents of PID $31 computed by the Dem. The buffer is provided by the Dcm with the appropriate size, i.e. during configuration, the Dcm identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	Always E_OK is returned, as E_NOT_OK will never appear.
Description:	Service to report the value of PID $31 computed by the Dem.	

⌋()

8.3.6.12 Dem_DcmReadDataOfPID41

[Dem322]

⌈

Service name:	Dem_DcmReadDataOfPID41	

Syntax:	`Std_ReturnType Dem_DcmReadDataOfPID41(`
	`uint8* PID41value`
	)	

Service ID[hex]:	0x67	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID41value	Buffer containing the contents of PID $41 computed by the Dem. The buffer is provided by the Dcm with the appropriate size, i.e. during configuration, the Dcm identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	Always E_OK is returned, as E_NOT_OK will never appear.
Description:	Service to report the value of PID $41 computed by the Dem.	

⌋()

8.3.6.13 Dem_DcmReadDataOfPID4D

[Dem323]

⌈

Service name:	Dem_DcmReadDataOfPID4D

Syntax:	`Std_ReturnType Dem_DcmReadDataOfPID4D(`
	`uint8* PID4Dvalue`
	)

Service ID[hex]:	0x68
Sync/Async:	Synchronous

Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID4Dvalue	Buffer containing the contents of PID $4D computed by the Dem. The buffer is provided by the Dcm with the appropriate size, i.e. during configuration, the Dcm identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	Always E_OK is returned, as E_NOT_OK will never appear.
Description:	Service to report the value of PID $4D computed by the Dem.	

⌋()

8.3.6.14 Dem_DcmReadDataOfPID4E

[Dem324]

⌈

Service name:	Dem_DcmReadDataOfPID4E	

Syntax:	`Std_ReturnType Dem_DcmReadDataOfPID4E(`
	`  uint8* PID4Evalue`
	`)`	

Service ID[hex]:	0x69	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PID4Evalue	Buffer containing the contents of PID $4E computed by the Dem. The buffer is provided by the Dcm with the appropriate size, i.e. during configuration, the Dcm identifies the required size from the largest PID in order to configure a PIDBuffer.
Return value:	Std_ReturnType	Always E_OK is returned, as E_NOT_OK will never appear.
Description:	Service to report the value of PID $4E computed by the Dem.	

⌋()

8.3.6.15 Dem_ReadDataOfOBDFreezeFrame

[Dem327]

⌈

Service name:	Dem_ReadDataOfOBDFreezeFrame

Syntax:	`Std_ReturnType Dem_ReadDataOfOBDFreezeFrame(`
	`  uint8 PID,`
	`  uint8 DataElementIndexOfPID,`
	`  uint8* DestBuffer,`
	`  uint8* BufSize`
	`)`

Service ID[hex]:	0x52
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	PID	This parameter is an identifier for a PID as defined in ISO15031-5.
	DataElementIndexOfPID	Data element index of this PID according to the Dcm configuration of service $02. It is zero-based and consecutive, and ordered by the data element positions (configured in Dcm, refer to Dem597).
Parameters (inout):	DestBuffer	This parameter contains a byte pointer that points to the buffer, to which the data element of the PID shall be written to. The format is raw hexadecimal values and contains no header-information.
	BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK Freeze frame data was successfully reported E_NOT_OK Freeze frame data was not successfully reported
Description:	Gets data element per PID and index of the most important freeze frame being selected for the output of service $02. The function stores the data in the provided DestBuffer.	

⌋()

8.3.6.16 Dem_GetDTCOfOBDFreezeFrame

[Dem624]

⌈

Service name:	Dem_GetDTCOfOBDFreezeFrame	

Syntax:	`Std_ReturnType Dem_GetDTCOfOBDFreezeFrame(`
	`  uint8 FrameNumber,`
	`  uint32* DTC`
	`)`	

Service ID[hex]:	0x53	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrameNumber	Unique identifier for a freeze frame record as defined in ISO 15031-5. The value 0x00 indicates the complete OBD freeze frame. Other values are reserved for future functionality.
Parameters (inout):	None	
Parameters (out):	DTC	Diagnostic Trouble Code in ODB format. If the return value of the function is other than E_OK this parameter does not contain valid data.
Return value:	Std_ReturnType	E_OK: operation was successful E_NOT_OK: no DTC available
Description:	Gets DTC by freeze frame record number.	

⌋()

8.3.6.17 Dem_SetPtoStatus

[Dem627]

⌈

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

⌋()

8.3.6.18 Service Interface IUMPRNumerator

[Dem610] ⌈ The *Dem Service Component* shall provide the interface *IUMPRNumerator* as defined below (to provide the capability to define the number of times a fault could have been found), if OBD support is configured. ⌋()

One port of this interface type is provided per ratio Id by the *Dem Service Component*. It has RatioID as a port-defined argument.

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

ProvidePort IUMPRNumerator IUMPRNumerator_<RatioName>;
ProvidePort IUMPRNumerator IUMPRNumerator_<RatioName>;
…

// for each port providing the interface IUMPRNumerator:
PortArgument {port=IUMPRNumerator_<RatioName>,
              value.type=Dem_RatioIdType,
              value.value=<n>, where <n> = 0..<N - 1>}

RunnableEntity RepIUMPRFaultDetect
      symbol "Dem_RepIUMPRFaultDetect"
      canbeInvokedConcurrently = TRUE
```

8.3.6.19 Service Interface IUMPRDenominator

[Dem611] 「 The *Dem Service Component* shall provide the interface *IUMPRDenominator* as defined below (to provide the capability to define the number of times the vehicle operation has been fulfilled), if OBD support is configured.」()

One port of this interface type is provided per ratio Id by the *Dem Service Component*. It has RatioID as a port-defined argument.

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

ProvidePort IUMPRDenominator IUMPRDenominator_<RatioName>;
ProvidePort IUMPRDenominator IUMPRDenominator_<RatioName>;
…

// for each port providing the interface IUMPRDenominator:
PortArgument {port= IUMPRDenominator_<RatioName>,
              value.type=Dem_RatioIdType,
              value.value=<n>, where <n> = 0..<N - 1>}

RunnableEntity RepIUMPRDenLock
      symbol "Dem_RepIUMPRDenLock"
      canbeInvokedConcurrently = TRUE
RunnableEntity RepIUMPRDenRelease
      symbol "Dem_RepIUMPRDenRelease"
      canbeInvokedConcurrently = TRUE
```

8.3.6.20 Service Interface PowerTakeOff

[Dem612]「The *Dem Service Component* shall provide the interface *PowerTakeOff* as defined below (to provide the capability to set the PTO status), if OBD support is configured.」()

One port of this interface type is provided by the *Dem Service Component*.

```
ClientServerInterface PowerTakeOff {
   PossibleErrors {
      E_NOT_OK = 1
   }
   SetPtoStatus(
      boolean PtoStatus,
      ERR{E_NOT_OK});
}

ProvidePort PowerTakeOff PowerTakeOffStatus;
```

```
RunnableEntity SetPtoStatus
      symbol "Dem_SetPtoStatus"
      canbeInvokedConcurrently = TRUE
```

8.4 Expected Interfaces

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

8.4.1 Mandatory Interfaces

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

API function	*Description*

8.4.2 Optional Interfaces

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

[Dem255]

⌈

API function	*Description*
Det_ReportError	Service to report development errors.
Dlt_DemTriggerOnEventStatus	This service is provided by the Dem in order to call Dlt upon status changes.
FiM_DemTriggerOnEventStatus	This service is provided by the Dem in order to call FiM upon status changes.
NvM_GetErrorStatus	Service to read the block dependent error/status information.
NvM_ReadBlock	Service to copy the data of the NV block to its corresponding RAM block.
NvM_SetRamBlockStatus	Service for setting the RAM block status of an NVRAM block.
NvM_WriteBlock	Service to copy the data of the RAM block to its corresponding NV block.

⌋()

Note: Based on implementation strategy either NvM_[Read|Write]Block or NvM_SetRamBlockStatus can be omitted, or the NvM usage is deactivated by configuration completely (refer also to chapter 7.8.4).

8.4.3 Configurable interfaces

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

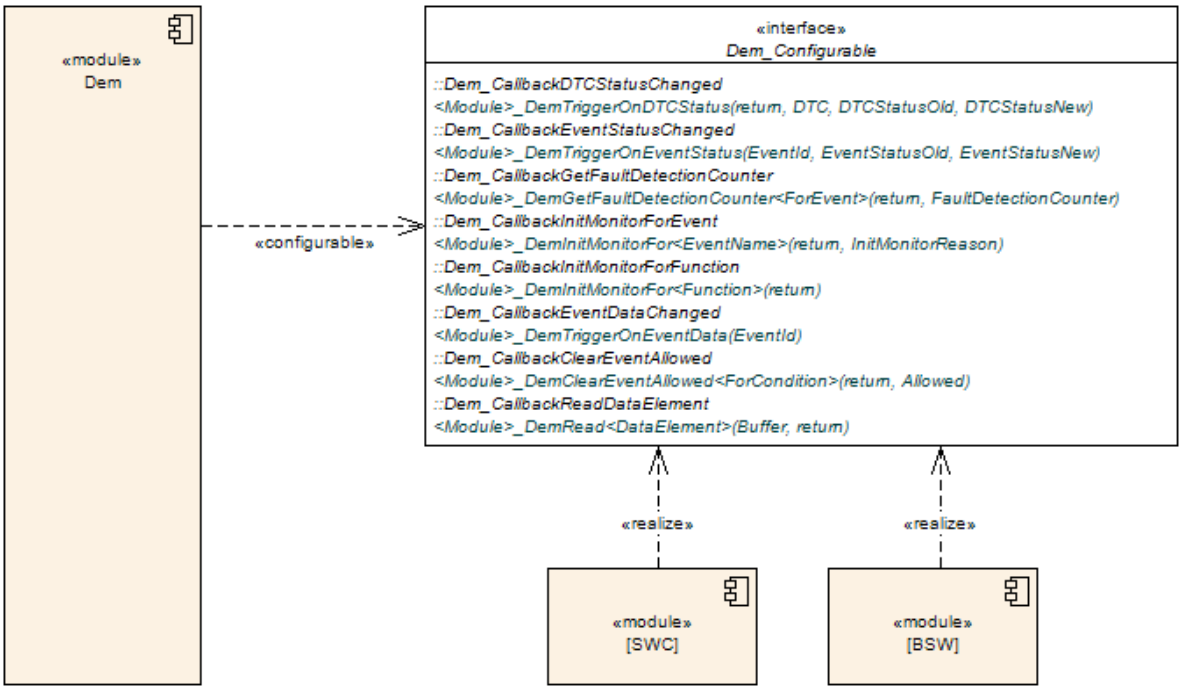

Figure 54 Configuration interfaces of the Dem module

8.4.3.1 Interface BSW modules / SW-Components ⇔ Dem

The callback interface from Dem to SW-Components is realized via RTE port interfaces. The following callback descriptions address the c-callbacks of other BSW modules.

8.4.3.1.1 InitMonitorForEvent

[Dem256]

⌈

Service name:	<Module>_DemInitMonitorFor<EventName>	

Syntax:	`Std_ReturnType <Module>_DemInitMonitorFor<EventName>(`
	`  Dem_InitMonitorReasonType InitMonitorReason`
	`)`	

Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	InitMonitorReason	Specific (re-)initialization reason evaluated from the monitor to identify the initialization kind to be performed.

Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	Return value unused - only for compatibility with according RTE operation.
Description:	Inits the diagnostic monitor of a specific event. There is one separate callback per event (if configured), if no port interface is provided by the Dem.	

⌋()

8.4.3.1.2 InitMonitorForFunction

[Dem258]

⌈

Service name:	<Module>_DemInitMonitorFor<Function>	
Syntax:	`Std_ReturnType <Module>_DemInitMonitorFor<Function>( void )`	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	Return value unused - only for compatibility with according RTE operation.
Description:	Resets the <Function> of the according module.	

⌋()

8.4.3.1.3 EventStatusChanged

[Dem259]

⌈

Service name:	<Module>_DemTriggerOnEventStatus	
Syntax:	`void <Module>_DemTriggerOnEventStatus( Dem_EventIdType EventId, Dem_EventStatusExtendedType EventStatusOld, Dem_EventStatusExtendedType EventStatusNew )`	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
	EventStatusOld	UDS DTC status byte of event before change (refer to chapter "Status bit support").
	EventStatusNew	UDS DTC status byte of event after change (refer to chapter "Status bit support").
Parameters (inout):	None	
Parameters (out):	None	

Return value:	None
Description:	Triggers on changes of the UDS DTC status byte. This prototype differs intentionally from the according RTE operations.

⌋()

8.4.3.1.4 DTCStatusChanged

[Dem260]

⌈

Service name:	<Module>_DemTriggerOnDTCStatus	

Syntax:	`Std_ReturnType <Module>_DemTriggerOnDTCStatus(`
	`  uint32 DTC,`
	`  uint8 DTCStatusOld,`
	`  uint8 DTCStatusNew`
	`)`	

Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DTC	Diagnostic Trouble Code in UDS format.
	DTCStatusOld	DTC status before change
	DTCStatusNew	DTC status after change
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	Return value unused - only for compatibility with according RTE operation.
Description:	Triggers on changes of the UDS DTC status byte.	

⌋()

8.4.3.1.5 EventDataChanged

[Dem562]

⌈

Service name:	<Module>_DemTriggerOnEventData	

Syntax:	`void <Module>_DemTriggerOnEventData(`
	`  Dem_EventIdType EventId`
	`)`	

Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Triggers on changes of the event related data in the event memory. The prototype differs intentionally from the according RTE operations.	

⌋()

8.4.3.1.6 ClearEventAllowed

[Dem563]

⌈

Service name:	<Module>_DemClearEventAllowed<ForCondition>	

Syntax:	`Std_ReturnType <Module>_DemClearEventAllowed<ForCondition>(`
	`  boolean* Allowed`
	`)`	

Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	Allowed	True – clearance of event is allowed False – clearance of event is not allowed
Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	Triggers on DTC-deletion, which is not allowed if the out-parameter returns False. There is one separate callback per condition, which can be assigned to one or several events, if no port interface is provided by the Dem.	

⌋()

8.4.3.1.7 ReadDataElement

[Dem564]

⌈

Service name:	<Module>_DemRead<DataElement>	

Syntax:	`Std_ReturnType <Module>_DemRead<DataElement>(`
	`  uint8* Buffer`
	`)`	

Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	Buffer	Buffer containing the value of the data element
Return value:	Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Description:	Requests the current value of the data element. There is one separate callback per data element, if no port interface is provided by the Dem.	

⌋()

8.4.3.1.8 GetFaultDetectionCounter

[Dem263]

⌈

Service name:	<Module>_DemGetFaultDetectionCounter<ForEvent>	
Syntax:	`Std_ReturnType <Module>_DemGetFaultDetectionCounter<ForEvent>( sint8* FaultDetectionCounter )`	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	FaultDetectionCounter	This parameter receives the fault detection counter information of the requested EventId. If the return value of the function call is other than E_OK this parameter does not contain valid data. -128dec...127dec PASSED...FAILED according to ISO 14229-1
Return value:	Std_ReturnType	E_OK: request was successful E_NOT_OK: request failed
Description:	Gets the current fault detection counter value. There is one c-callback per event using monitor-internal debouncing, if no port interface is provided by the Dem.	

」()

8.4.3.2 Service Interface CallbackInitMonitorForEvent

[Dem613] 「 The *Dem Service Component* shall provide the interface *CallbackInitMonitorForEvent* as defined below (to trigger an event-specific initialization of the monitor part of the SW-C), if configured.」()

For each event, there can be one port of this interface type.

```
ClientServerInterface CallbackInitMonitorForEvent {
   PossibleErrors {
      E_NOT_OK = 1
   }
   InitMonitorForEvent(
      IN Dem_InitMonitorReasonType InitMonitorReason,
      ERR{E_NOT_OK});
}

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

8.4.3.3 Service Interface CallbackInitMonitorForFunction

[Dem614] 「 The *Dem Service Component* shall provide the interface *CallbackInitMonitorForFunction* as defined below, if configured.」()

```
ClientServerInterface CallbackInitMonitorForFunction {
```

```
   PossibleErrors {
      E_NOT_OK = 1
   }
   InitMonitorForFunction(
      ERR{E_NOT_OK});
}

RequirePort CallbackInitMonitorForFunction CBInitFct_1;
RequirePort CallbackInitMonitorForFunction CBInitFct_2;
…
RequirePort CallbackInitMonitorForFunction CBInitFct_<N>;
```

8.4.3.4 Service Interface CallbackEventStatusChange & General

[Dem615] ⌈ The *Dem Service Component* shall provide the interface *CallbackEventStatusChange* as defined below (to trigger SW-Cs on event status byte changes), if configured.⌋()

For each event, there can be several ports of this interface type.

[Dem616] ⌈ The Dem module shall provide the interface *GeneralCallbackEventStatusChange* as defined below (to also trigger SW-Cs on event status byte changes, which are using this information in combination with the interface *GeneralDiagnosticInfo*), if at least on port CBStatusEvt_<EventName>_<SWC> is configured.⌋()

One global port of this interface type is provided by the *Dem Service Component*.

```
ClientServerInterface CallbackEventStatusChange {
   PossibleErrors {
      E_NOT_OK = 1
   }
   EventStatusChanged(
      IN Dem_EventStatusExtendedType EventStatusOld,
      IN Dem_EventStatusExtendedType EventStatusNew,
      ERR{E_NOT_OK});
}

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

ClientServerInterface GeneralCallbackEventStatusChange {
   PossibleErrors {
      E_NOT_OK = 1
   }
   EventStatusChanged(
      IN Dem_EventIdType EventId,
      IN Dem_EventStatusExtendedType EventStatusOld,
      IN Dem_EventStatusExtendedType EventStatusNew,
      ERR{E_NOT_OK});
}
```

```
RequirePort GeneralCallbackEventStatusChange GeneralCBStatusEvt;
```

8.4.3.5 Service Interface CallbackDTCStatusChange

[Dem617] ⌈ The *Dem Service Component* shall provide the interface *CallbackDTCStatusChange* as defined below (to trigger SW-Cs on DTC status byte changes), if configured.⌋()

There can be several ports of this interface type, provided globally by the *Dem Service Component*.

```
ClientServerInterface CallbackDTCStatusChange {
   PossibleErrors {
      E_NOT_OK = 1
   }
   DTCStatusChanged(
      IN uint32 DTC,
      IN Dem_DTCStatusMaskType DTCStatusOld,
      IN Dem_DTCStatusMaskType DTCStatusNew,
      ERR{E_NOT_OK});
}

RequirePort CallbackDTCStatusChange CBStatusDTC_1;
RequirePort CallbackDTCStatusChange CBStatusDTC_2;
…
RequirePort CallbackDTCStatusChange CBStatusDTC_<N>;
```

8.4.3.6 Service Interface CallbackEventDataChanged & General

[Dem618] ⌈ The *Dem Service Component* shall provide the interface *CallbackEventDataChanged* as defined below (to trigger SW-Cs on event related data changes), if configured.⌋()

For each event, there can be one port of this interface type.

[Dem619] ⌈ The Dem module shall provide the interface *GeneralCallbackEventDataChanged* as defined below (to also trigger SW-Cs on event related data changes, which are using this information in combination with the interface *GeneralDiagnosticInfo*), if at least on port CBDataEvt_<EventName> is configured.⌋()

One global port of this interface type is provided by the *Dem Service Component*.

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

```
   EventDataChanged(
      ERR{E_NOT_OK});
}

RequirePort CallbackEventDataChanged CBDataEvt_<EventName>;
RequirePort CallbackEventDataChanged CBDataEvt_<EventName>;
…

ClientServerInterface GeneralCallbackEventDataChanged {
   PossibleErrors {
      E_NOT_OK = 1
   }
   EventDataChanged(
      IN Dem_EventIdType EventId,
      ERR{E_NOT_OK});
}

RequirePort GeneralCallbackEventDataChanged GeneralCBDataEvt;
```

8.4.3.7 Service Interface CallbackClearEventAllowed

[Dem620] ⌈ The *Dem Service Component* shall provide the interface *CallbackClearEventAllowed* as defined below (to get the permission before clearing a specific event from the SW-C), if configured.⌋()

For each event, there can be one port of this interface type.

```
ClientServerInterface CallbackClearEventAllowed {
   PossibleErrors {
      E_NOT_OK = 1
   }
   ClearEventAllowed(
      OUT boolean Allowed,
      ERR{E_NOT_OK});
}

RequirePort CallbackClearEventAllowed CBClrEvt_<EventName>;
RequirePort CallbackClearEventAllowed CBClrEvt_<EventName>;
…
```

8.4.3.8 Service Interface DataServices_<SyncDataElement>

[Dem621] ⌈ The *Dem Service Component* shall provide the interface *DataServices_<SyncDataElement>* as defined below (to get the data element value contained in a DID, a PID, or an extended data record from the respective SW-C via client/server or sender/receiver communication, refer to Figure 44), if configured.⌋()

For each data element, one port of this interface type is provided by the SW-Cs.

Note: The Dem- and Dcm-interfaces of this type are compatible, to be connected to the same provide-port of the application.

Note: The *Dem Service Component* supports synchronouse client/server interfaces only (due to the used mechanisms for the event memory) and is compatible with the Dcm interface *DataServices_<Data>* with the setting for USE_DATA_SYNCH_CLIENT_SERVER.
All further operations contained in the Dcm interface *CSDataServices_<Data>* like WriteData, ReadDataLength (relates to data elements with a variable length), ConditionCheckRead, etc. are provided by the SW-C and used by the Dcm, but are not required/considered by the Dem module.

```
ClientServerInterface CSDataServices_<SyncDataElement> {
   PossibleErrors {
      E_NOT_OK = 1,
   }
   // the server is not allowed to return E_NOT_OK, but shall always
   //   provide a valid data value (e.g. a default/replacement value
   //   in an error-case) to Dcm/Dem
   // nevertheless the signature of the operation includes E_NOT_OK to
   //   ensure compatibility between server runnable and RTE Call API,
   //   since the RTE may return negative Std_Return values in certain
   //   cases (e.g. partition of server stopped)
   ReadData(
      OUT uint8 Data[<DemDataElememtDataSize>],
      ERR{E_NOT_OK})
}

RequirePort CSDataServices_<SyncDataElement> CBReadData_<SyncDataElement>;
RequirePort CSDataServices_<SyncDataElement> CBReadData_<SyncDataElement>;
…
```

The following interface for sender/receiver communication is not specified in detail within this document, but it defines at least a hull to allow an improved non-standardized implementation.

Note: The *Dem Service Component* is compatible with the Dcm interface *DataServices_<Data>* with the setting for USE_DATA_SENDER_RECEIVER.

```
SenderReceiverInterface SRDataServices_<SyncDataElement> {
   RECEIVER <datatype> Data
}

RequirePort SRDataServices_<SyncDataElement> CBReadData_<SyncDataElement>;
RequirePort SRDataServices_<SyncDataElement> CBReadData_<SyncDataElement>;
…
```

8.4.3.9 Service Interface CallbackGetFaultDetectCounter

[Dem622] ⌈ The *Dem Service Component* shall provide the interface *CallbackGetFaultDetectionCounter* as defined below (to get the monitor-internal fault detection counter value of a specific event from the SW-C), if configured.⌋()

For each event, there can be one port of this interface type.

```
ClientServerInterface CallbackGetFaultDetectCounter {
   PossibleErrors {
      E_NOT_OK = 1
   }
   GetFaultDetectionCounter(
      OUT sint8 FaultDetectionCounter,
      ERR{E_NOT_OK});
}

RequirePort CallbackGetFaultDetectCounter CBFaultDetectCtr_<EventName>;
RequirePort CallbackGetFaultDetectCounter CBFaultDetectCtr_<EventName>;
…
```

8.5 Scheduled functions

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

8.5.1 Dem_MainFunction

[Dem266]

⌈

Service name:	Dem_MainFunction

Syntax:	`void Dem_MainFunction(`
	`  void`
	`)`

Service ID[hex]:	0x55
Timing:	FIXED_CYCLIC
Description:	Processes all not event based Dem internal functions.

⌋()

[Dem125] ⌈The function Dem_MainFunction shall process all not event based Dem module internal functions.⌋()

[Dem286] ⌈The Dem module's environment (e.g. by operating system) shall call the function Dem_MainFunction periodically as cyclic task.⌋()

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

8.5.2 Runnable Entity MainFunction

```
RunnableEntity MainFunction
      symbol "Dem_MainFunction"
      canbeInvokedConcurrently = FALSE
      SSCP = port CBStatusEvt_*, EventStatusChanged
      SSCP = port GeneralCBStatusEvt, EventStatusChanged
      SSCP = port CBStatusDTC_*, DTCStatusChanged
      SSCP = port CBDataEvt_*, EventDataChanged
      SSCP = port GeneralCBDataEvt, EventDataChanged
      SSCP = port CBReadData_*, ReadData
```

9 Sequence diagrams

9.1 ControlDTCSetting

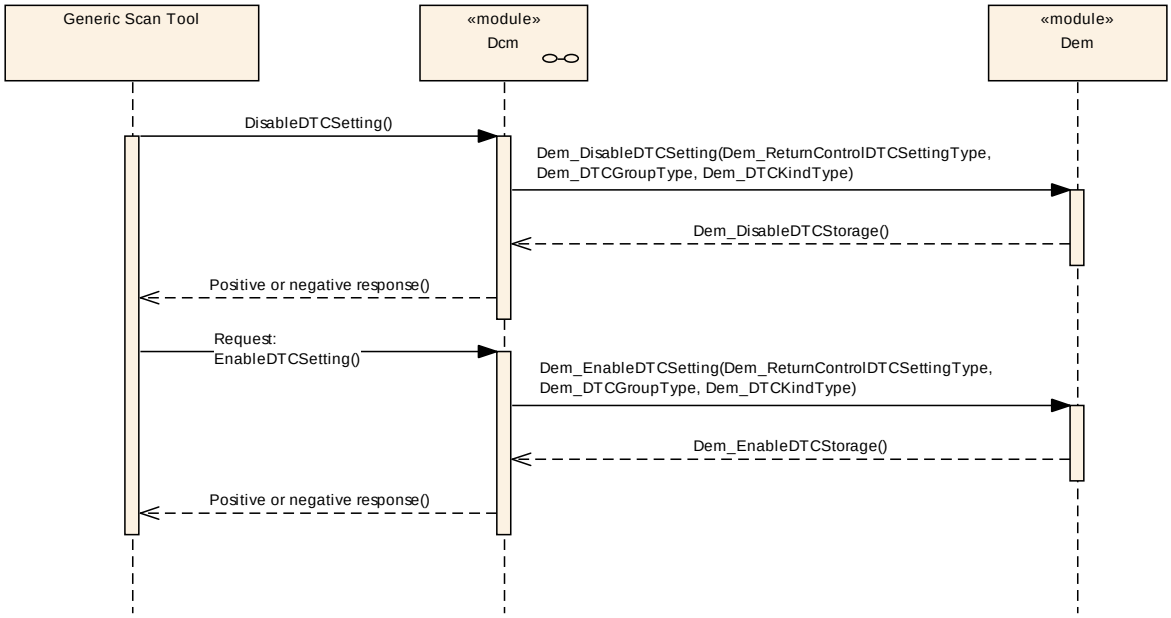

Figure 55 Sequence diagram of Dem_ControlDTCSetting

9.2 Dem_ClearDTC

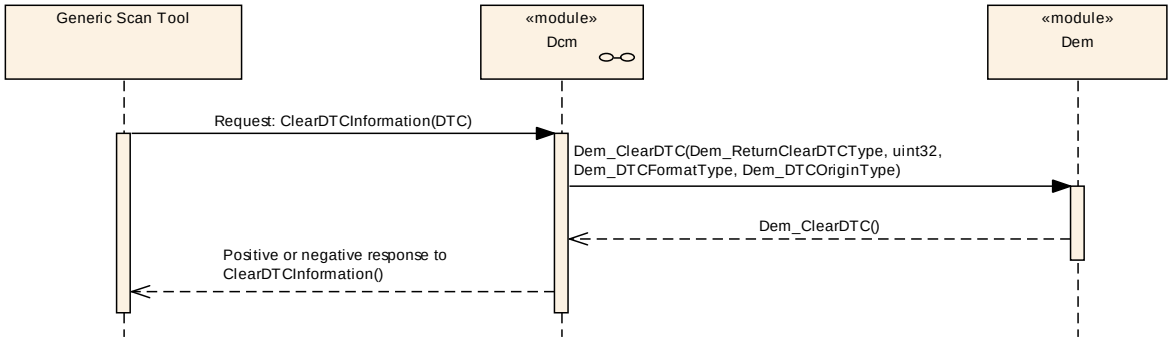

Figure 56 Sequence diagram of Dem_ClearDTC

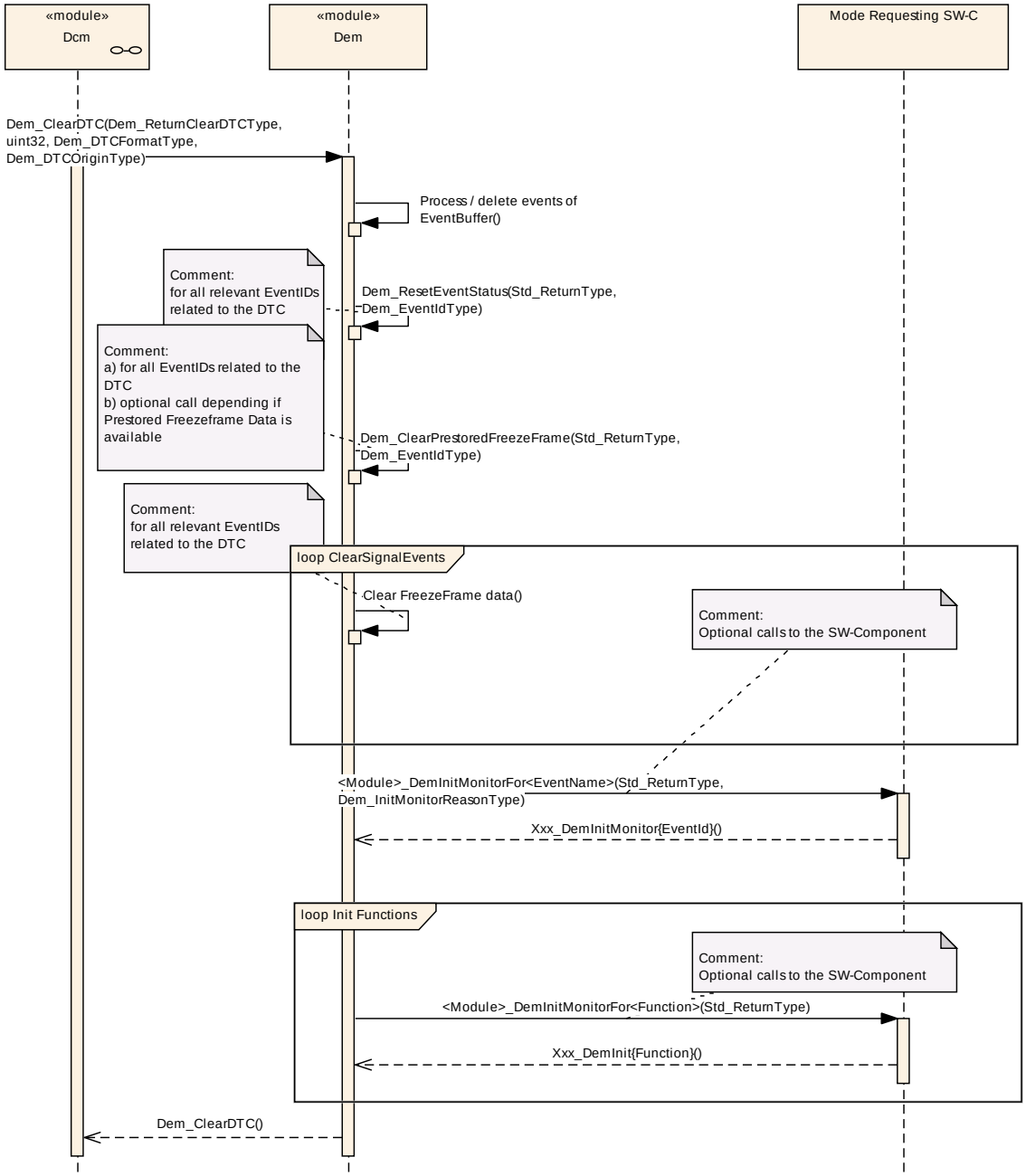

Figure 57 Sequence diagram of Dem_ClearDTC clearing a single DTC Dem-internally

9.3 Dem_GetDTCByOccurrenceTime

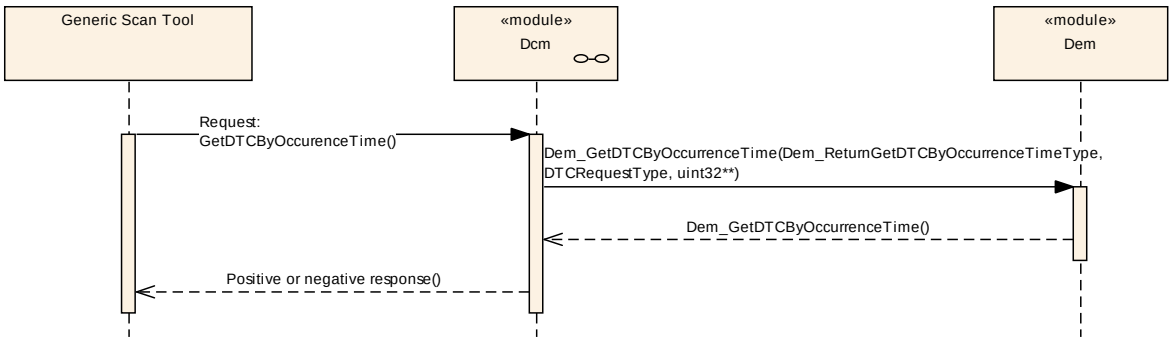

Figure 58 Sequence diagram of Dem_GetDTCByOccurenceTime

9.4 Dem_GetExtendedDataRecordByDTC

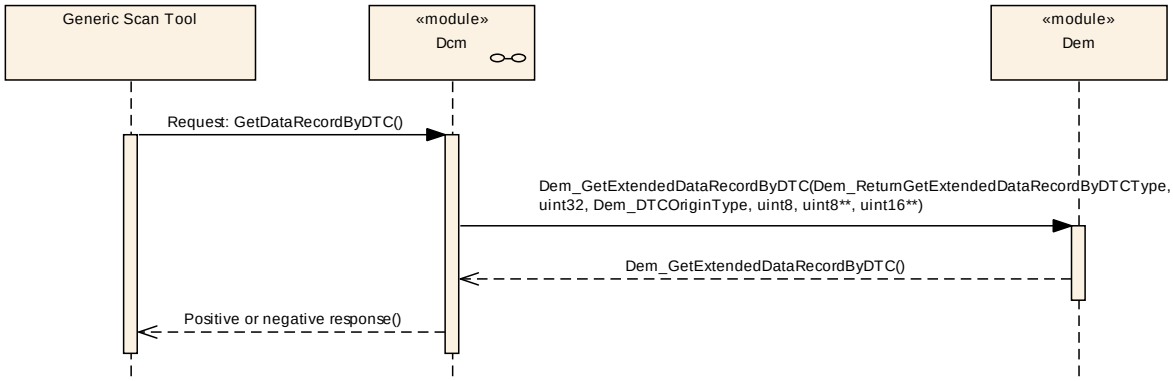

Figure 59 Sequence diagram of Dem_GetExtendedDataRecordByDTC

9.5 Dem_GetStatusOfDTC

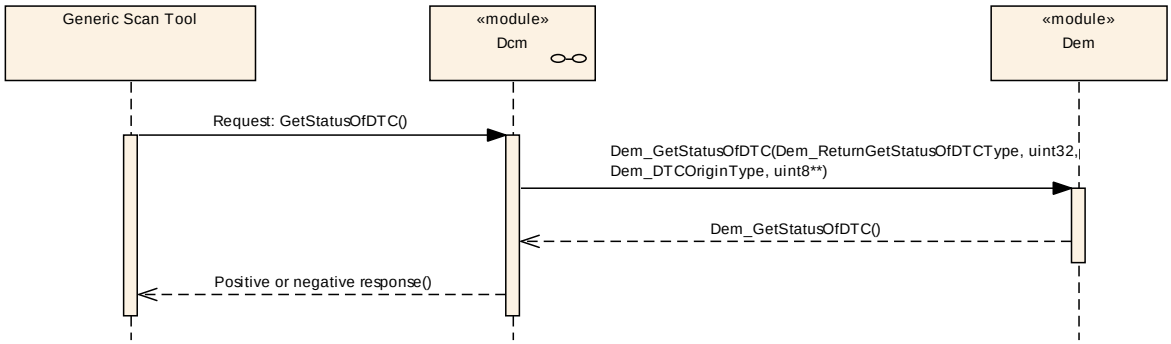

Figure 60 Sequence diagram of Dem_GetStatusOfDTC

9.6 Dem_GetSizeOfFreezeFrameByDTC

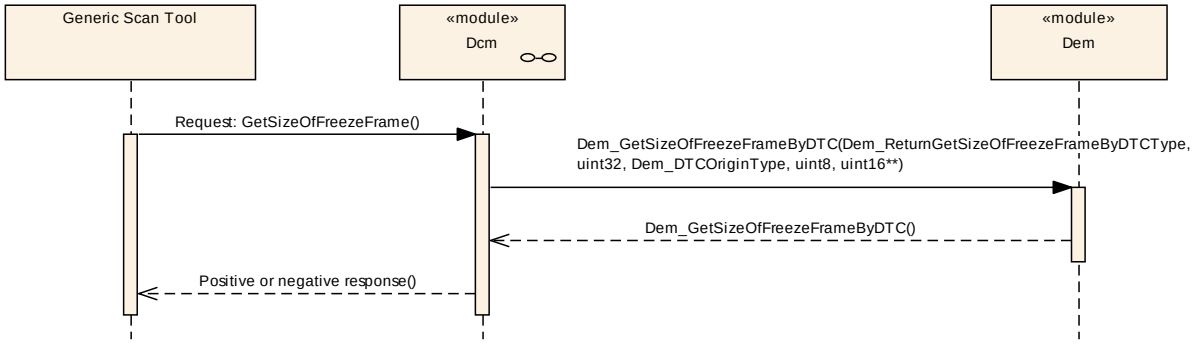

Figure 61 Sequence diagram of Dem_GetSizeOfFreezeFrameByDTC

9.7 GetOBDFaultInformation

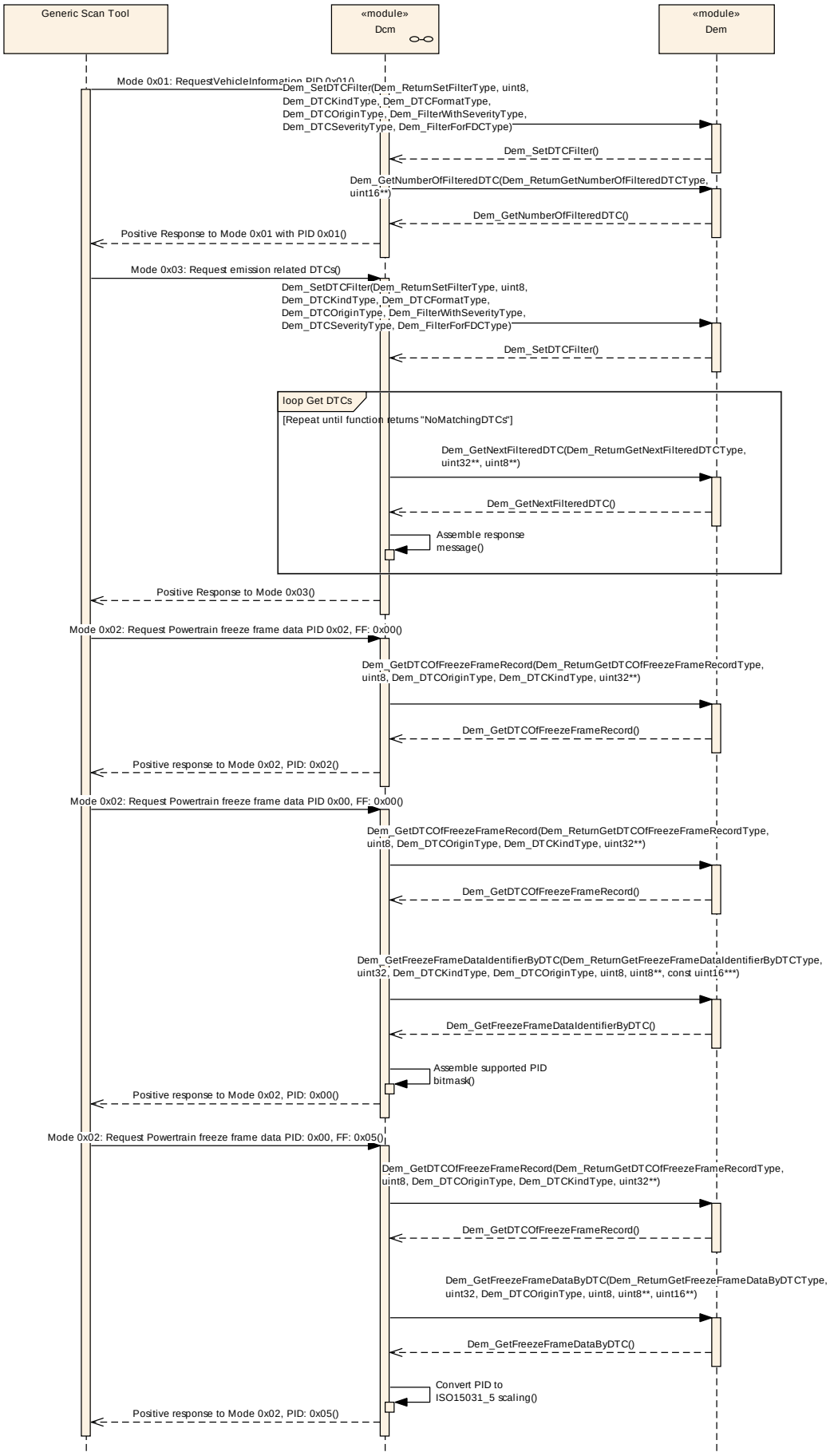
Generic Scan Tool
«module»
Dcm
«module»
Dem
Mode 0x01: RequestVehicleInformation PID 0x010
Dem_SetDTCFilter(Dem_ReturnSetFilterType, uint8,
Dem_DTCKindType, Dem_DTCFormatType,
Dem_DTCOriginType, Dem_FilterWithSeverityType,
Dem_DTCSeverityType, Dem_FilterForFDCType)
Dem_SetDTCFilter()
Dem_GetNumberOfFilteredDTC(Dem_ReturnGetNumberOfFilteredDTCType,
uint16**)
Dem_GetNumberOfFilteredDTC()
Positive Response to Mode 0x01 with PID 0x01()
Mode 0x03: Request emission related DTCs()
Dem_SetDTCFilter(Dem_ReturnSetFilterType, uint8,
Dem_DTCKindType, Dem_DTCFormatType,
Dem_DTCOriginType, Dem_FilterWithSeverityType,
Dem_DTCSeverityType, Dem_FilterForFDCType)
Dem_SetDTCFilter()
loop Get DTCs
[Repeat until function returns "NoMatchingDTCs"]
Dem_GetNextFilteredDTC(Dem_ReturnGetNextFilteredDTCType,
uint32**, uint8**)
Dem_GetNextFilteredDTC()
Assemble response
message()
Positive Response to Mode 0x03()
Mode 0x02: Request Powertrain freeze frame data PID 0x02, FF: 0x00()
Dem_GetDTCOfFreezeFrameRecord(Dem_ReturnGetDTCOfFreezeFrameRecordType,
uint8, Dem_DTCOriginType, Dem_DTCKindType, uint32**)
Dem_GetDTCOfFreezeFrameRecord()
Positive response to Mode 0x02, PID: 0x02()
Mode 0x02: Request Powertrain freeze frame data PID 0x00, FF: 0x00()
Dem_GetDTCOfFreezeFrameRecord(Dem_ReturnGetDTCOfFreezeFrameRecordType,
uint8, Dem_DTCOriginType, Dem_DTCKindType, uint32**)
Dem_GetDTCOfFreezeFrameRecord()
Dem_GetFreezeFrameDataIdentifierByDTC(Dem_ReturnGetFreezeFrameDataIdentifierByDTCType,
uint32, Dem_DTCKindType, Dem_DTCOriginType, uint8, uint8**, const uint16***)
Dem_GetFreezeFrameDataIdentifierByDTC()
Assemble supported PID
bitmask()
Positive response to Mode 0x02, PID: 0x00()
Mode 0x02: Request Powertrain freeze frame data PID: 0x00, FF: 0x05()
Dem_GetDTCOfFreezeFrameRecord(Dem_ReturnGetDTCOfFreezeFrameRecordType,
uint8, Dem_DTCOriginType, Dem_DTCKindType, uint32**)
Dem_GetDTCOfFreezeFrameRecord()
Dem_GetFreezeFrameDataByDTC(Dem_ReturnGetFreezeFrameDataByDTCType,
uint32, Dem_DTCOriginType, uint8, uint8**, uint16**)
Dem_GetFreezeFrameDataByDTC()
Convert PID to
ISO15031_5 scaling()
Positive response to Mode 0x02, PID: 0x05()

Figure 62 Sequence diagram of GetOBDFaultInformation

9.8 ReportDTCByStatusMask

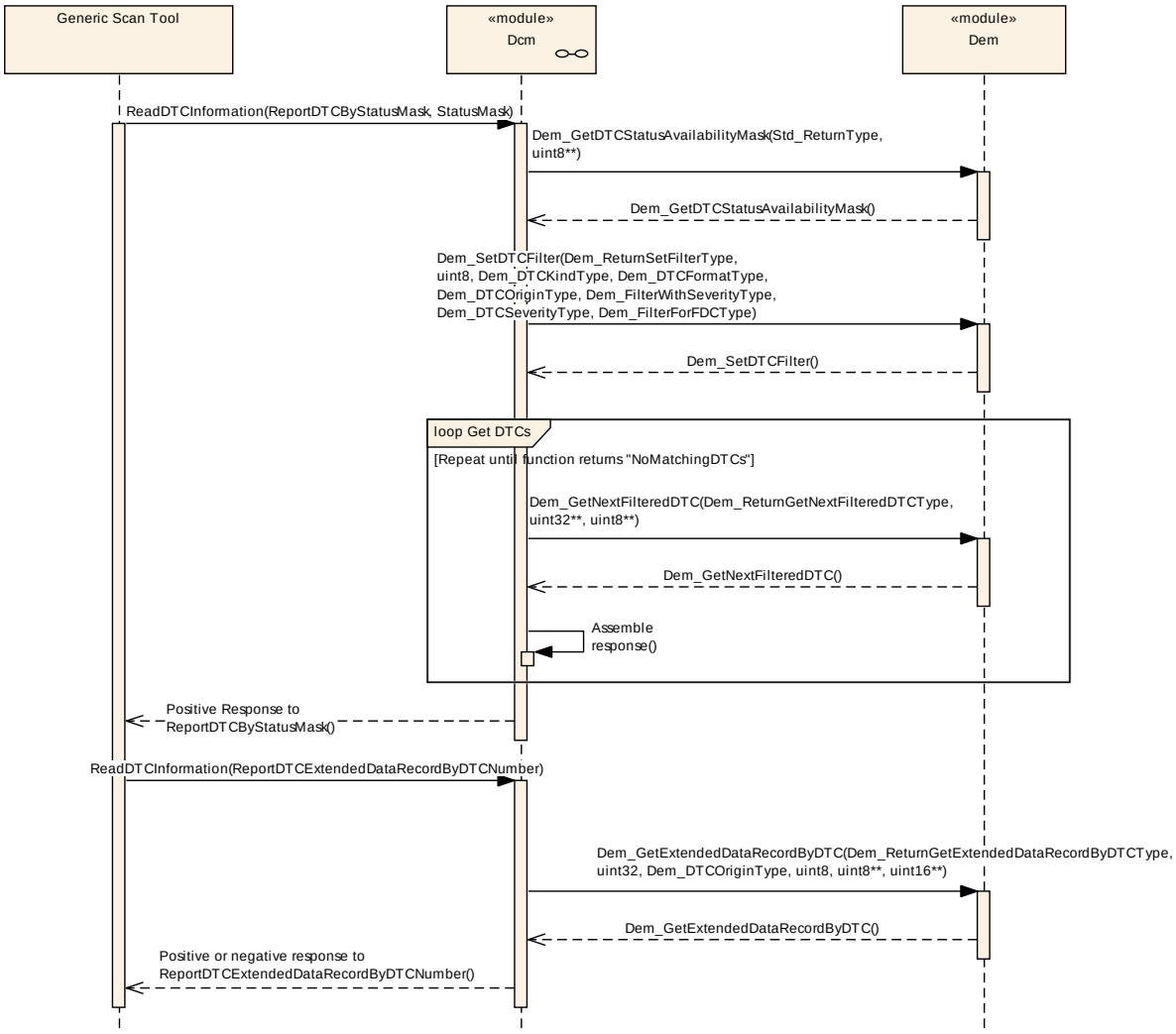

Figure 63 Sequence diagram of ReportDTCStatusMask

9.9 FiM_DemTriggerOnEventStatus

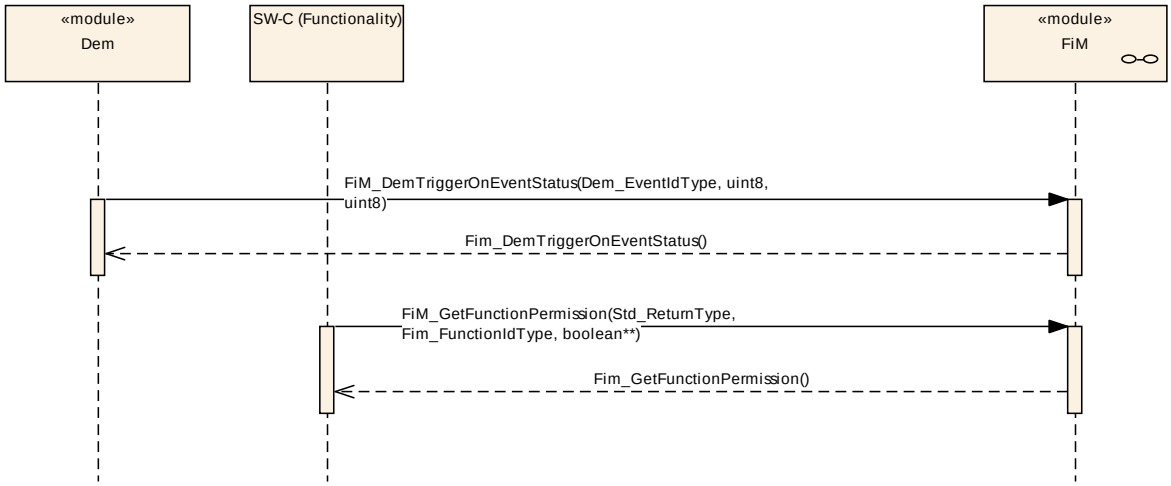

Figure 64 Sequence diagram of FiM_DemTriggerOnEventStatus

9.10 ProcessEvent (Example)

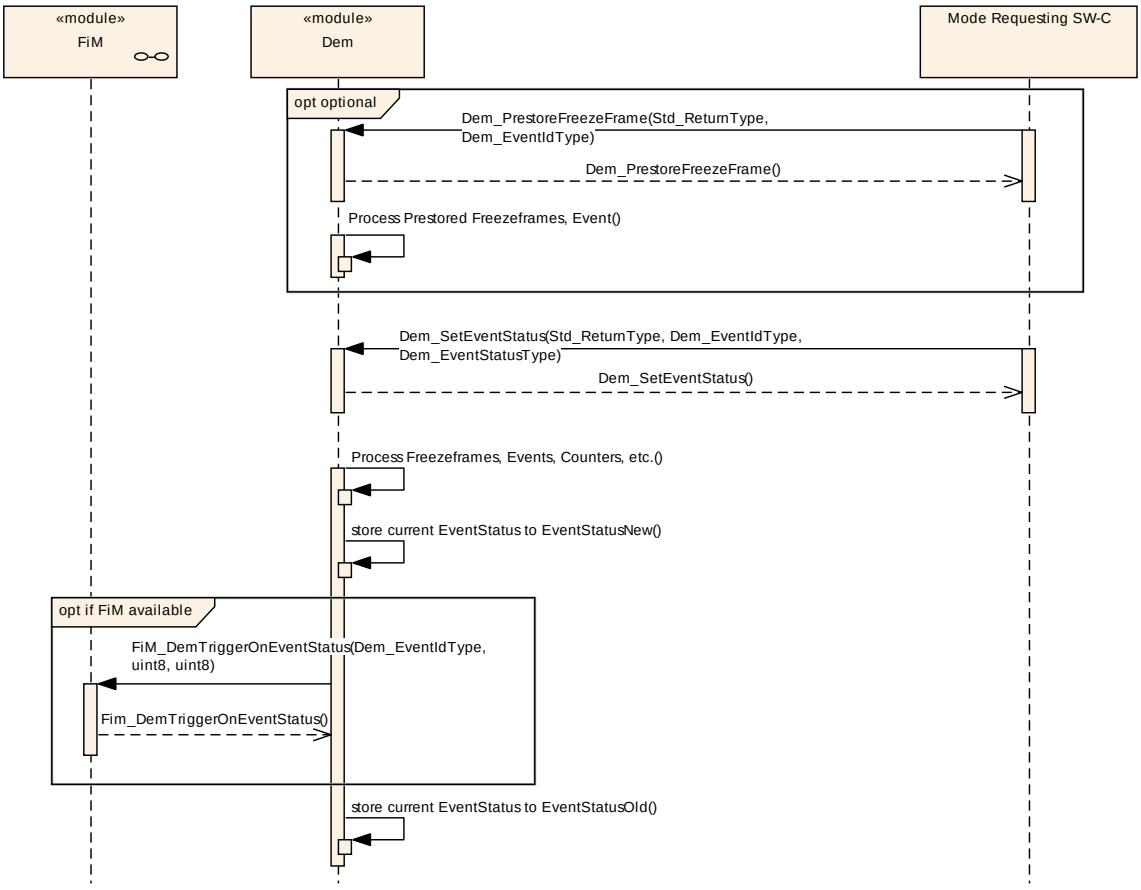

Figure 65 Sequence diagram for an example of ProcessEvent Dem-internally

10 Configuration specification

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

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

Chapter 10.3 specifies published information of the module Dem.

10.1 How to read this chapter

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

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

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

10.1.1 Configuration and configuration parameters

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

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

10.1.2 Variants

Variants describe sets of configuration parameters. . E.g., variant 1: only pre-compile time configuration parameters; variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant a parameter can only be of one configuration class.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

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

10.2 Containers and configuration parameters

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

10.2.1 Variants

The following use-cases and concepts for the Dem are the base for the different Dem configuration variants:

1. Simple ECU configuration, with optimized ROM usage: results in a pre-compile time variant
2. Multiple ECU configuration support: results in a post-built time selectable variant (in combination with a superset method)
   For this use-case, some DTC values have different values in the different configuration sets. Additionally it is possible, to omit events/DTCs (for the outside world) completely in some of the configuration sets.
3. Support of different PID values for Europe/USA results in a post-built time variant (refer to DemPidIdentifier and DemPidDataElementClassRef)
4. Support of different DTC kind for Europe/USA results in a post-built time variant

The following configuration parameters shall be available:

- **[Dem267]** 「VARIANT-PRE-COMPILE: only pre-compile time configuration parameters 」(BSW334, BSW345, BSW00396, BSW00397, BSW00398, BSW00399, BSW00400, BSW00401, BSW00404, BSW00405)
- **[Dem268]** 「VARIANT-POST-BUILD: mix of pre-compile- and post build time-configuration parameters (using post build selectable, refer to chapter 4.1).」(BSW334, BSW345, BSW00396, BSW00397, BSW00398, BSW00399, BSW00400, BSW00401, BSW00404, BSW00405)

Link time configurable parameters (VARIANT-LINK-TIME) are not used in this specification.

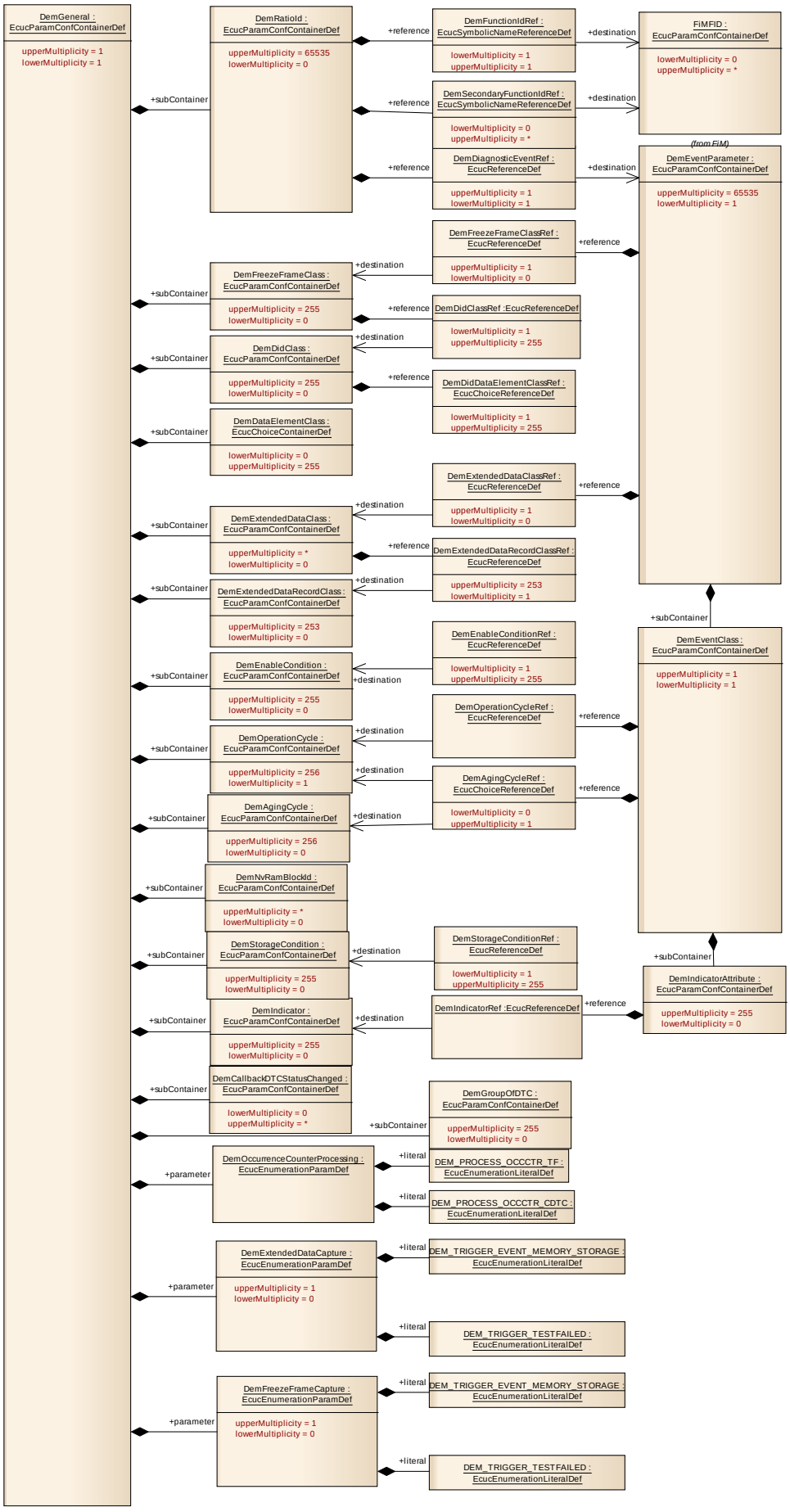
DemGeneral : EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 1
DemRatioId : EcucParamConfContainerDef
upperMultiplicity = 65535
lowerMultiplicity = 0
DemFunctionIdRef : EcucSymbolicNameReferenceDef
lowerMultiplicity = 1
upperMultiplicity = 1
FiMFID : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
DemSecondaryFunctionIdRef : EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = *
DemDiagnosticEventRef : EcucReferenceDef
upperMultiplicity = 1
lowerMultiplicity = 1
(from FiM)
DemEventParameter : EcucParamConfContainerDef
upperMultiplicity = 65535
lowerMultiplicity = 1
DemFreezeFrameClassRef : EcucReferenceDef
upperMultiplicity = 1
lowerMultiplicity = 0
DemFreezeFrameClass : EcucParamConfContainerDef
upperMultiplicity = 255
lowerMultiplicity = 0
DemDidClassRef :EcucReferenceDef
lowerMultiplicity = 1
upperMultiplicity = 255
DemDidClass : EcucParamConfContainerDef
upperMultiplicity = 255
lowerMultiplicity = 0
DemDidDataElementClassRef : EcucChoiceReferenceDef
lowerMultiplicity = 1
upperMultiplicity = 255
DemDataElementClass : EcucChoiceContainerDef
lowerMultiplicity = 0
upperMultiplicity = 255
DemExtendedDataClassRef : EcucReferenceDef
upperMultiplicity = 1
lowerMultiplicity = 0
DemExtendedDataClass : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 0
DemExtendedDataRecordClassRef : EcucReferenceDef
upperMultiplicity = 253
lowerMultiplicity = 1
DemExtendedDataRecordClass : EcucParamConfContainerDef
upperMultiplicity = 253
lowerMultiplicity = 0
DemEventClass : EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 1
DemEnableConditionRef : EcucReferenceDef
lowerMultiplicity = 1
upperMultiplicity = 255
DemEnableCondition : EcucParamConfContainerDef
upperMultiplicity = 255
lowerMultiplicity = 0
DemOperationCycleRef : EcucReferenceDef
DemOperationCycle : EcucParamConfContainerDef
upperMultiplicity = 256
lowerMultiplicity = 1
DemAgingCycleRef : EcucChoiceReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
DemAgingCycle : EcucParamConfContainerDef
upperMultiplicity = 256
lowerMultiplicity = 0
DemNvRamBlockId : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 0
DemStorageConditionRef : EcucReferenceDef
lowerMultiplicity = 1
lowerMultiplicity = 255
DemStorageCondition : EcucParamConfContainerDef
upperMultiplicity = 255
lowerMultiplicity = 0
DemIndicatorAttribute : EcucParamConfContainerDef
upperMultiplicity = 255
lowerMultiplicity = 0
DemIndicatorRef :EcucReferenceDef
DemIndicator : EcucParamConfContainerDef
upperMultiplicity = 255
lowerMultiplicity = 0
DemCallbackDTCStatusChanged : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
DemGroupOfDTC : EcucParamConfContainerDef
upperMultiplicity = 255
lowerMultiplicity = 0
DemOccurrenceCounterProcessing : EcucEnumerationParamDef
DEM_PROCESS_OCCCTR_TF : EcucEnumerationLiteralDef
DEM_PROCESS_OCCCTR_CDTC : EcucEnumerationLiteralDef
DemExtendedDataCapture : EcucEnumerationParamDef
upperMultiplicity = 1
lowerMultiplicity = 0
DEM_TRIGGER_EVENT_MEMORY_STORAGE : EcucEnumerationLiteralDef
DEM_TRIGGER_TESTFAILED : EcucEnumerationLiteralDef
DemFreezeFrameCapture : EcucEnumerationParamDef
upperMultiplicity = 1
lowerMultiplicity = 0
DEM_TRIGGER_EVENT_MEMORY_STORAGE : EcucEnumerationLiteralDef
DEM_TRIGGER_TESTFAILED : EcucEnumerationLiteralDef
+subContainer
+reference
+destination
+parameter
+literal

Figure 66 Configuration overview for DemGeneral

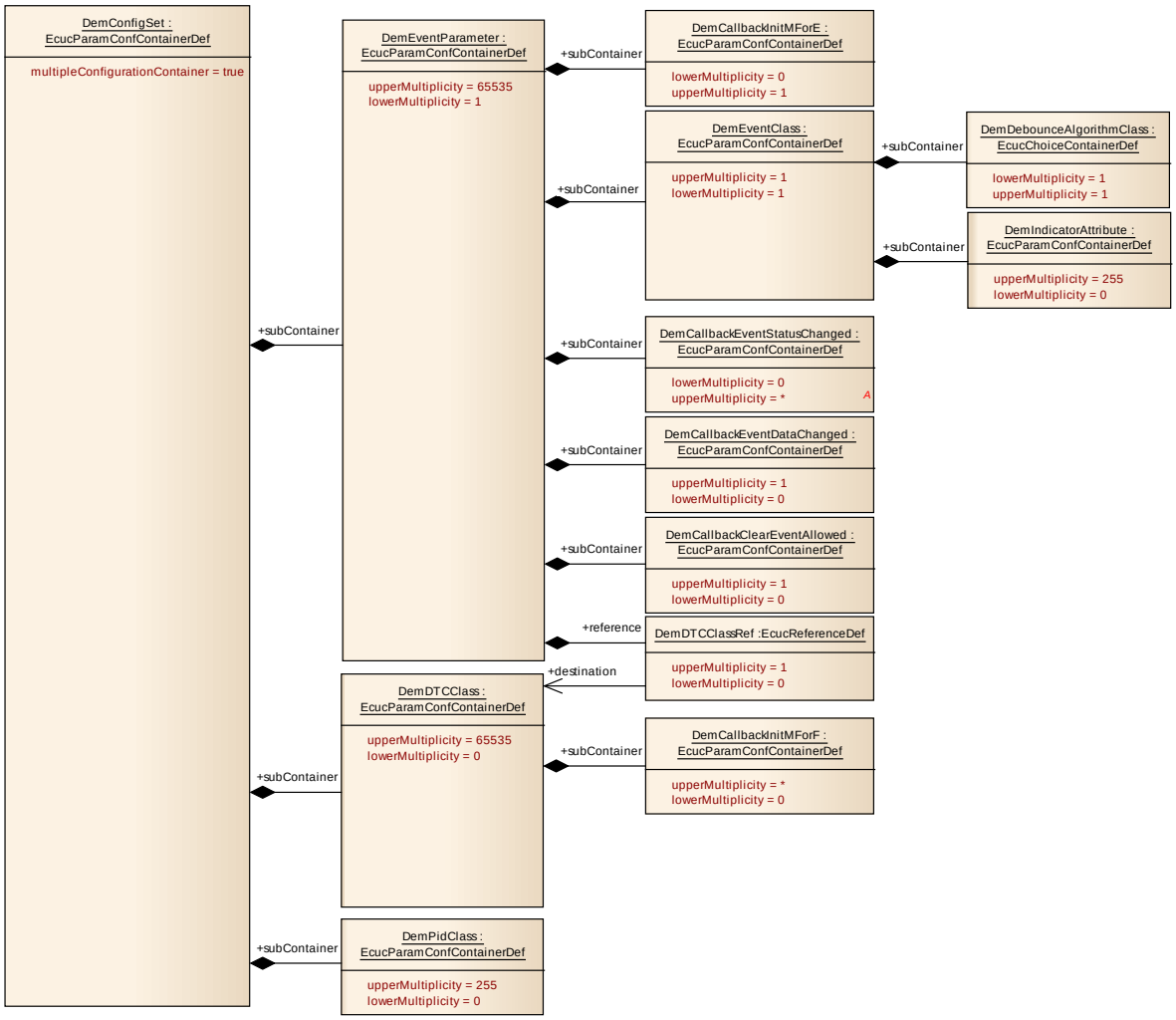

Figure 67 Configuration overview for DemConfigSet

10.2.2 Dem

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

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

10.2.3 DemGeneral

SWS Item	**Dem677_Conf :**
Container Name	DemGeneral
Description	This container contains the configuration (parameters) of the BSW Dem
Configuration Parameters	

SWS Item	**Dem603_Conf :**		
Name	DemAgingCycleCounterProcessing		
Description	This configuration switch defines, whether the aging counter is calculated Dem-internally or provided via Dem_SetAgingCycleCounterValue.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_PROCESS_AGINGCTR_EXTERN	based on API Dem_SetAgingCycleCounterValue	
	DEM_PROCESS_AGINGCTR_INTERN	based on reported cycle states	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem625_Conf :**		
Name	DemBswErrorBufferSize		
Description	Maximum number of elements in buffer for handling of BSW errors (ref. to Dem107).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem766_Conf :**		
Name	DemClearDTCBehavior		
Description	Defines the clearing process of diagnostic information for volatile and non-volatile memory and the positive response handling for the Dcm module.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_CLRRESP_NONVOLATILE_FINISH	Return DEM_CLEAR_OK after volatile and non-volatile event memory data cleared.	
	DEM_CLRRESP_NONVOLATILE_TRIGGER	Return DEM_CLEAR_OK after volatile event memory data cleared and non-volatile event memory clearing is triggered	
	DEM_CLRRESP_VOLATILE	Return DEM_CLEAR_OK after volatile event memory data cleared	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem780_Conf :**		
Name	DemDTCSuppressionSupport		
Description	This configuration switch defines, whether support for DTC suppression is enabled or not. true: DTC suppression support is enabled false: DTC suppression support is disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem724_Conf :**		
Name	DemDebounceCounterBasedSupport		
Description	This configuration switch defines, whether support for counter based debouncing is enabled or not. true: counter based debouncing support is enabled false: counter based debouncing support is disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem725_Conf :**		
Name	DemDebounceTimeBasedSupport		
Description	This configuration switch defines, whether support for time based debouncing is enabled or not. true: time based debouncing support is enabled false: time based debouncing support is disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem648_Conf :**		
Name	DemDevErrorDetect		
Description	Activate/Deactivate the Development Error Detection and Notification. true: Development Error Detection and Notification activated false: Development Error Detection and Notification deactivated		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**Dem652_Conf :**
Name	DemDtcStatusAvailabilityMask
Description	Mask for the supported DTC status bits by the Dem. This mask is used by

	UDS service 0x19.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem726_Conf :**		
Name	DemEnableConditionSupport		
Description	This configuration switch defines, whether support for enable conditions is enabled or not. true: support for enable conditions is enabled false: support for enable conditions is disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem740_Conf :**		
Name	DemEventCombinationSupport		
Description	This parameter defines the type of event combination supported by the Dem.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_EVCOMB_DISABLED	No event combination supported	
	DEM_EVCOMB_TYPE1	Event combination Type 1 enabled	
	DEM_EVCOMB_TYPE2	Event combination Type 2 enabled	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem742_Conf :**		
Name	DemEventDisplacementSupport		
Description	This configuration switch defines, whether support for event displacement is enabled or not. true: event displacement support is enabled false: event displacement support is disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem663_Conf :**
Name	DemExtendedDataCapture
Description	This parameter defines the point in time, when the extended data collection is done for the initial event memory entry.
Multiplicity	0..1
Type	EcucEnumerationParamDef

Range	DEM_TRIGGER_EVENT_MEMORY_STORAGE		Triggers the collection of extended data if the event is stored in fault memory.
	DEM_TRIGGER_TESTFAILED		Triggers the collection of extended data if the UDS DTC status bit 0 (TestedFailed) changes from 0 to 1.
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem672_Conf :**		
Name	DemFreezeFrameCapture		
Description	This parameter defines the point in time, when the freeze frame data collection is done for the initial event memory entry.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	DEM_TRIGGER_EVENT_MEMORY_STORAGE		Triggers the collection of freeze frame data if the event is stored in fault memory.
	DEM_TRIGGER_TESTFAILED		Triggers the collection of freeze frame data if the UDS DTC status bit 0 (TestedFailed) changes from 0 to 1.
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem722_Conf :**		
Name	DemHeaderFileInclusion		
Description	Name of the header file(s) to be included by the Dem module containing the used C-callback declarations.		
Multiplicity	0..*		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	[a-zA-Z0-9_]([a-zA-Z0-9\._])*		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem738_Conf :**	
Name	DemImmediateNvStorageLimit	
Description	This parameter defines the maximum number of occurrences, a specific event memory entry is allowed, to be stored in NVRAM immediately (refer to DemImmediateNvStorage).	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 255	

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

SWS Item	**Dem688_Conf :**		
Name	DemMaxNumberEventEntryMirror		
Description	Maximum number of events which can be stored in the mirror memory		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem689_Conf :**		
Name	DemMaxNumberEventEntryPermanent		
Description	Maximum number of events which can be stored in the permanent memory. The assignment of an event to this memory type is dynamic and used for emission-related events only.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem690_Conf :**		
Name	DemMaxNumberEventEntryPrimary		
Description	Maximum number of events which can be stored in the primary memory		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem691_Conf :**		
Name	DemMaxNumberEventEntrySecondary		
Description	Maximum number of events which can be stored in the secondary memory		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem692_Conf :**		
Name	DemMaxNumberPrestoredFF		
Description	Defines the maximum number for prestored freeze frames. If set to 0, then freeze frame prestorage is not supported by the ECU.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem698_Conf :**		
Name	DemOBDSupport		
Description	This configuration switch defines whether OBD is supported or not		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem767_Conf :**		
Name	DemOccurrenceCounterProcessing		
Description	This configuration switch defines the consideration of the fault confirmation process for the occurrence counter. For OBD and mixed systems (OBD/non OBD, refer to DemOBDSupport) the fault confirmation process must not be considered.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_PROCESS_OCCCTR_CDTC		the occurrence counter is triggered by the TestFailed bit only, but the fault confirmation is not considered
	DEM_PROCESS_OCCCTR_TF		the occurrence counter is triggered after the fault confirmation was successfull
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DemOBDSupport		

SWS Item	**Dem783_Conf :**	
Name	DemOperationCycleProcessing	
Description	This configuration switch defines, whether the operation cycles are triggered by DEM_CYCLE_STATE_START or collecting an external counter value, which results in respective state changes.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DEM_PROCESS_OPCYC_COUNTER	operation cycle processing is based on API Dem_SetOperationCycleCntValue
	DEM_PROCESS_OPCYC_STATE	operation cycle processing is based on

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

SWS Item	**Dem764_Conf :**		
Name	DemOperationCycleStatusStorage		
Description	Defines if the operation cycle state is available over the power cycle (stored non-volatile) or not. true: the operation cycle state is stored non-volatile false: the operation cycle state is only stored volatile		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem704_Conf :**		
Name	DemPTOSupport		
Description	This configuration switch defines, whether PTO support (and therefore PID $1E support) is enabled or not.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem784_Conf :**		
Name	DemStatusBitHandlingTestFailedSinceLastClear		
Description	This configuration switch defines, whether the aging and displacement mechanism shall be applied to the "TestFailedSinceLastClear" status bits.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_STATUS_BIT_AGING_AND_DISPLACEMENT	the "TestFailedSinceLastClear" status bits are reset to 0, if aging or displacement applies (like done for the "ConfirmedDTC" status bits)	
	DEM_STATUS_BIT_NORMAL	aging and displacement has no impact on the "TestFailedSinceLastClear" status bits	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem714_Conf :**
Name	DemStatusBitStorageTestFailed

Description	Activate/Deactivate the permanent storage of the "TestFailed" status bits. true: storage activated false: storage deactivated		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem727_Conf :**		
Name	DemStorageConditionSupport		
Description	This configuration switch defines, whether support for storage conditions is enabled or not. true: support for storage conditions is enabled false: support for storage conditions is disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem715_Conf :**		
Name	DemTaskTime		
Description	Allow to configure the time for the periodic cyclic task. Please note: This configuration value shall be equal to the value in the Basic Software Scheduler configuration of the RTE module. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. Dem configuration tools must convert this float value to the appropriate value format for the use in the software implementation of Dem. min: A negative value is not allowed. max: After event status was reported, processing shall be completed within 100ms in order to have the fault entry status information updated as soon as possible (e.g. for PID $01). upperMultiplicity: Exactly one TaskTime must be specified per configuration. lowerMultiplicity: Exactly one TaskTime must be specified per configuration.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0 .. 0.1`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem754_Conf :**		
Name	DemTriggerDcmReports		
Description	Activate/Deactivate the notification to the Diagnostic Communication Manager for ROE processing. true: Dcm ROE notification activated false: Dcm ROE notification deactivated		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

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

SWS Item	**Dem718_Conf :**		
Name	DemTriggerDltReports		
Description	Activate/Deactivate the notification to the Diagnostic Log and Trace. true: Dlt notification activated false: Dlt notification deactivated		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem719_Conf :**		
Name	DemTriggerFiMReports		
Description	Activate/Deactivate the notification to the Funciton Inhibition Manager. true: FiM notification activated false: FiM notification deactivated		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem765_Conf :**		
Name	DemTriggerMonitorInitBeforeClearOk		
Description	Defines if the monitor re-initialization has to be triggered before or after the Dem module returns DEM_CLEAR_OK. true: trigger re-initialization before DEM_CLEAR_OK false: trigger re-initialization after DEM_CLEAR_OK		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DemClearDTCBehavior		

SWS Item	**Dem720_Conf :**		
Name	DemTypeOfDTCSupported		
Description	This parameter defines the format returned by Dem_GetTranslationType and does not relate to/influence the supported Dem functionality.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_DTC_TRANSLATION_ISO11992_4	ISO11992-4 DTC format	
	DEM_DTC_TRANSLATION_ISO14229_1	ISO14229-1 DTC format (3 byte format)	
	DEM_DTC_TRANSLATION_ISO15031_6	ISO15031-6 DTC format (2 byte format)	
	DEM_DTC_TRANSLATION_SAEJ1939_73	SAEJ1939-73 DTC format	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: ECU

SWS Item	**Dem778_Conf :**		
Name	DemTypeOfFreezeFrameRecordNumeration		
Description	This parameter defines the type of assigning freeze frame record numbers for event-specific freeze frame records.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_FF_RECNUM_CALCULATED	freeze frame records will be numbered consecutive starting by 1 in their chronological order	
	DEM_FF_RECNUM_CONFIGURED	freeze frame records will be numbered based on the given configuration in their chronological order	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem721_Conf :**		
Name	DemVersionInfoApi		
Description	Activate/Deactivate the version information API. true: version information activated false: version information deactivated		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**Dem723_Conf :**		
Name	DemMILIndicatorRef		
Description	This parameter defines the indicator representing the MIL. This paramteter is mandatory for ECUs supporting OBD (refer to DemOBDSupport).		
Multiplicity	0..1		
Type	Reference to [DemIndicator]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DemOBDSupport		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemAgingCycle	0..256	Note that this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the aging cycle name. These aging cycles are reported via API Dem_SetAgingCycleState only.
DemCallbackDTCStatusChanged	0..*	The presence of this container indicates, that the Dem has access to a "DTCStatusChanged" callback, which the Dem will call to notify other components about the change in the status of a DTC. In case there is a

		DemCallbackDTCStatusChangedFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackDTCStatusChangedFnc, the Dem will have an R-Port requiring the interface CallbackDTCStatusChanged whose name is generated by using the unique callback-prefix followed by the event name.
DemDataElementClass	0..255	This container contains the configuration (parameters) for an internal/external data element class.
DemDidClass	0..255	This container contains the configuration (parameters) for a data Id class. It is assembled out of one or several data elements.
DemEnableCondition	0..255	This container contains the configuration (parameters) for enable conditions.
DemEnableConditionGroup	0..255	This container contains the configuration (parameters) for enable condition groups.
DemExtendedDataClass	0..*	This class contains the combinations of extended data records for an extended data class.
DemExtendedDataRecordClass	0..253	This container contains the configuration (parameters) for an extended data record class. It is assembled out of one or several data elements.
DemFreezeFrameClass	0..255	This container contains the combinations of DIDs for a non OBD relevant freeze frame class.
DemFreezeFrameRecNumClass	0..255	This container contains a list of dedicated, different freeze frame record numbers assigned to an event. The order of record numbers in this list is assigned to the chronological order of the according freeze frame records. dependency: DemTypeOfFreezeFrameRecordNumeration
DemGeneralOBD	0..1	This container contains the general OBD-specific configuration (parameters) of the Dem module.
DemGroupOfDTC	0..255	This container contains the configuaration (parameters) for DTC groups.
DemIndicator	0..255	This container contains the configuration (parameters) for Indicators. Note that this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the INDICATOR_NAME.
DemNvRamBlockId	0..*	This container contains the configuaration (parameters) for a non-volatile memory block, which is used from the Dem. If no permanent storage of event memory entries is required, no block needs to be configured. The number of blocks which are necessary depends on the implementation and configuration (e.g. number of used event memories) of the Dem module.
DemOperationCycle	1..256	Note that this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the operation cycle name.
DemRatioId	0..65535	This container contains the OBD specific ratio Id configuraiton. It is related to a specific event, a FID, and an IUMPR group. Note that this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the ratio Id name.
DemStorageCondition	0..255	This container contains the configuration (parameters) for

		storage conditions.
DemStorageConditionGroup	0..255	This container contains the configuration (parameters) for storage condition groups.

10.2.4 DemGeneralOBD

SWS Item	**Dem756_Conf :**
Container Name	DemGeneralOBD
Description	This container contains the general OBD-specific configuration (parameters) of the Dem module.
Configuration Parameters	

SWS Item	**Dem763_Conf :**		
Name	DemOBDInputAcceleratorPaddleInformation		
Description	Input variable for the accelerator paddle information, which is assigned to a specific data element used as interface for the Dem-internal PID calculations.		
Multiplicity	1		
Type	Choice reference to [DemExternalCSDataElementClass , DemExternalSRDataElementClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem762_Conf :**		
Name	DemOBDInputAmbientPressure		
Description	Input variable for the ambient pressure, which is assigned to a specific data element used as interface for the Dem-internal PID calculations.		
Multiplicity	1		
Type	Choice reference to [DemExternalCSDataElementClass , DemExternalSRDataElementClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem761_Conf :**		
Name	DemOBDInputAmbientTemperature		
Description	Input variable for the ambient temperature, which is assigned to a specific data element used as interface for the Dem-internal PID calculations.		
Multiplicity	1		
Type	Choice reference to [DemExternalCSDataElementClass , DemExternalSRDataElementClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem759_Conf :**
Name	DemOBDInputDistanceInformation
Description	Input variable for the distance information, which is assigned to a specific

	data element used as interface for the Dem-internal PID calculations.		
Multiplicity	1		
Type	Choice reference to [DemExternalCSDataElementClass , DemExternalSRDataElementClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem757_Conf :**		
Name	DemOBDInputEngineSpeed		
Description	Input variable for the engine speed, which is assigned to a specific data element used as interface for the Dem-internal PID calculations.		
Multiplicity	1		
Type	Choice reference to [DemExternalCSDataElementClass , DemExternalSRDataElementClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem772_Conf :**		
Name	DemOBDInputEngineTemperature		
Description	Input variable for the engine temperature, which is assigned to a specific data element used as interface for the Dem-internal PID calculations.		
Multiplicity	1		
Type	Choice reference to [DemExternalCSDataElementClass , DemExternalSRDataElementClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem760_Conf :**		
Name	DemOBDInputProgrammingEvent		
Description	Input variable for the programming event, which is assigned to a specific data element used as interface for the Dem-internal PID calculations.		
Multiplicity	1		
Type	Choice reference to [DemExternalCSDataElementClass , DemExternalSRDataElementClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem758_Conf :**		
Name	DemOBDInputVehicleSpeed		
Description	Input variable for the vehicle speed, which is assigned to a specific data element used as interface for the Dem-internal PID calculations.		
Multiplicity	1		
Type	Choice reference to [DemExternalCSDataElementClass , DemExternalSRDataElementClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.5 DemOperationCycle

SWS Item	**Dem701_Conf :**
Container Name	DemOperationCycle
Description	Note that this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the operation cycle name.
Configuration Parameters	

SWS Item	**Dem703_Conf :**		
Name	DemOperationCycleType		
Description	Operation cycles types for the Dem to be supported by cycle-state APIs. Further cycle types can be specified as part of the Dem delivery.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_OPCYC_IGNITION	Ignition ON / OFF cycle	
	DEM_OPCYC_OBD_DCY	OBD Driving cycle	
	DEM_OPCYC_OTHER	further operation cycle	
	DEM_OPCYC_POWER	Power ON / OFF cycle	
	DEM_OPCYC_TIME	Time based operation cycle	
	DEM_OPCYC_WARMUP	OBD OBD Warm up cycle	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.6 DemAgingCycle

SWS Item	**Dem785_Conf :**
Container Name	DemAgingCycle
Description	Note that this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the aging cycle name. These aging cycles are reported via API Dem_SetAgingCycleState only.
Configuration Parameters	

No Included Containers

10.2.7 DemEnableCondition

SWS Item	**Dem653_Conf :**
Container Name	DemEnableCondition
Description	This container contains the configuration (parameters) for enable conditions.
Configuration Parameters	

SWS Item	**Dem654_Conf :**		
Name	DemEnableConditionId		
Description	Defines a unique enable condition Id. This parameter should not be changeable by user, because the Id should be generated by Dem itself to prevent gaps and multiple use of an Id. The enable conditions should be sequentially ordered beginning with 0 and no gaps in between.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem656_Conf :**		
Name	DemEnableConditionStatus		
Description	Defines the inital status for enable or disable of acceptance of event reports of a diagnostic event. The value is the initialization after power up (before this condition is reported the first time). true: acceptance of a diagnostic event enabled false: acceptance of a diagnostic event disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.8 DemEnableConditionGroup

SWS Item	**Dem745_Conf :**
Container Name	DemEnableConditionGroup
Description	This container contains the configuration (parameters) for enable condition groups.
Configuration Parameters	

SWS Item	**Dem655_Conf :**
Name	DemEnableConditionRef
Description	References an enable condition.
Multiplicity	1..255
Type	Reference to [DemEnableCondition]

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

No Included Containers

10.2.9 DemStorageCondition

SWS Item	**Dem728_Conf :**
Container Name	DemStorageCondition
Description	This container contains the configuration (parameters) for storage conditions.
Configuration Parameters	

SWS Item	**Dem730_Conf :**		
Name	DemStorageConditionId		
Description	Defines a unique storage condition Id. This parameter should not be changeable by user, because the Id should be generated by Dem itself to prevent gaps and multiple use of an Id. The storage conditions should be sequentially ordered beginning with 0 and no gaps in between.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem731_Conf :**		
Name	DemStorageConditionStatus		
Description	Defines the initial status for enable or disable of storage of a diagnostic event. The value is the initialization after power up (before this condition is reported the first time). true: storage of a diagnostic event enabled false: storage of a diagnostic event disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

No Included Containers

10.2.10 DemStorageConditionGroup

SWS Item	**Dem773_Conf :**

Container Name	DemStorageConditionGroup
Description	This container contains the configuration (parameters) for storage condition groups.
Configuration Parameters	

SWS Item	**Dem768_Conf :**		
Name	DemStorageConditionRef		
Description	References an enable condition.		
Multiplicity	1..255		
Type	Reference to [DemStorageCondition]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.11 DemIndicator

SWS Item	**Dem680_Conf :**
Container Name	DemIndicator
Description	This container contains the configuration (parameters) for Indicators. Note that this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the INDICATOR_NAME.
Configuration Parameters	

SWS Item	**Dem683_Conf :**		
Name	DemIndicatorID		
Description	Unique identifier of an indicator.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.12 DemNvRamBlockId

SWS Item	**Dem696_Conf :**
Container Name	DemNvRamBlockId
Description	This container contains the configuaration (parameters) for a non-volatile memory block, which is used from the Dem. If no permanent storage of event memory entries is required, no block needs to be configured.

	The number of blocks which are necessary depends on the implementation and configuration (e.g. number of used event memories) of the Dem module.
Configuration Parameters	

SWS Item	**Dem697_Conf :**		
Name	DemNvRamBlockIdRef		
Description	This reference contains the link to a non-volatile memory block. For post build time configurations worst case szenario shall be used.		
Multiplicity	1		
Type	Reference to [NvMBlockDescriptor]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.13 DemGroupOfDTC

SWS Item	**Dem679_Conf :**
Container Name	DemGroupOfDTC
Description	This container contains the configuaration (parameters) for DTC groups.
Configuration Parameters	

SWS Item	**Dem678_Conf :**		
Name	DemGroupDTCs		
Description	DTC value of the selected group of DTC (Range: 3 byte, 0x000000 is only available for the emission-related DTC group, 0xFFFFFF is reserved for 'all DTCs', according to ISO14229-1 Annex D.1,) The DTC group 'all DTCs' is always available and will not be configured.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 16777214		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Vehicle		

No Included Containers

10.2.14 DemRatioId

SWS Item	**Dem734_Conf :**
Container Name	DemRatioId
Description	This container contains the OBD specific ratio Id configuraiton. It is related to a specific event, a FID, and an IUMPR group. Note that this container definition does not explicitly define a symbolic

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

SWS Item	**Dem737_Conf :**		
Name	DemIUMPRGroup		
Description	This parameter specifies the assigned IUMPR group of the ratio Id.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_IUMPR_BOOSTPRS	--	
	DEM_IUMPR_CAT1	--	
	DEM_IUMPR_CAT2	--	
	DEM_IUMPR_EGR	--	
	DEM_IUMPR_EGSENSOR	--	
	DEM_IUMPR_EVAP	--	
	DEM_IUMPR_NMHCCAT	--	
	DEM_IUMPR_NOXADSORB	--	
	DEM_IUMPR_NOXCAT	--	
	DEM_IUMPR_OXS1	--	
	DEM_IUMPR_OXS2	--	
	DEM_IUMPR_PMFILTER	--	
	DEM_IUMPR_PRIVATE	--	
	DEM_IUMPR_SAIR	--	
	DEM_IUMPR_SECOXS1	--	
	DEM_IUMPR_SECOXS2	--	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem741_Conf :**		
Name	DemRatioIdType		
Description	This parameter defines whether the ratio Id will be calculated API or observer based.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_RATIO_API	API based ratio Id	
	DEM_RATIO_OBSERVER	Observer based ratio Id	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem735_Conf :**		
Name	DemDiagnosticEventRef		
Description	This reference contains the link to a diagnostic event.		
Multiplicity	1		
Type	Reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem736_Conf :**

Name	DemFunctionIdRef		
Description	This reference contains the link to a function identifier within the FiM which is used as a primary FID.		
Multiplicity	1		
Type	Reference to [FiMFID]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem782_Conf :**		
Name	DemSecondaryFunctionIdRef		
Description	This reference contains the link to a function identifier within the FiM which is used as a secondary FID. The "primary" and all "secondary" FID inhibitions are combined by "OR".		
Multiplicity	0..*		
Type	Reference to [FiMFID]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.15 DemCallbackDTCStatusChanged

SWS Item	**Dem626_Conf :**
Container Name	DemCallbackDTCStatusChanged
Description	The presence of this container indicates, that the Dem has access to a "DTCStatusChanged" callback, which the Dem will call to notify other components about the change in the status of a DTC. In case there is a DemCallbackDTCStatusChangedFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackDTCStatusChangedFnc, the Dem will have an R-Port requiring the interface CallbackDTCStatusChanged whose name is generated by using the unique callback-prefix followed by the event name.
Configuration Parameters	

SWS Item	**Dem627_Conf :**		
Name	DemCallbackDTCStatusChangedFnc		
Description	Function name of prototype "DTCStatusChanged". Note: If the parameter DemTriggerDcmReports is enabled, this parameter shall not be "Dcm_DemTriggerOnDTCStatus".		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

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

No Included Containers

10.2.16 DemCallbackInitMForF

SWS Item	**Dem600_Conf :**		
Container Name	DemCallbackInitMForF		
Description	The presence of this container indicates, that the Dem has access to an "InitMonitorForFunction" callback, which the Dem will call to initialize a monitor. In case the container has a DemCallbackInitMForFFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackInitMForFFnc, the Dem will have an R-Port requiring the interface CallbackInitMonitorForFunction, whose name is generated by using the unique callback-prefix followed by the event name.		
Configuration Parameters			

SWS Item	**Dem633_Conf :**		
Name	DemCallbackInitMForFFnc		
Description	Function name of prototype "InitMonitorForFunction".		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.17 DemConfigSet

SWS Item	**Dem634_Conf :**
Container Name	DemConfigSet [Multi Config Container]
Description	This container contains the configuration parameters and sub containers of the Dem module supporting multiple configuration sets. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemDTCClass	0..65535	This container contains the configuration (parameters) for DTCClass.

DemEventParameter	1..65535	This container contains the configuaration (parameters) for events. Note that this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the diagonstic event.
DemPidClass	0..255	This container contains the different PIDs for the single global OBD relevant freeze frame class. It is assembled out of one or several data elements.

10.2.18 DemPidClass

SWS Item	**Dem729_Conf :**
Container Name	DemPidClass
Description	This container contains the different PIDs for the single global OBD relevant freeze frame class. It is assembled out of one or several data elements.
Configuration Parameters	

SWS Item	**Dem705_Conf :**		
Name	DemPidIdentifier		
Description	identifier of the PID		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemPidDataElement	1..255	This container contains the different data elements contained in the specific PID.

10.2.19 DemPidDataElement

SWS Item	**Dem896_Conf :**
Container Name	DemPidDataElement
Description	This container contains the different data elements contained in the specific PID.
Configuration Parameters	

SWS Item	**Dem733_Conf :**
Name	DemPidDataElementClassRef
Description	This reference contains the link to a data element class.
Multiplicity	1
Type	Choice reference to [DemExternalCSDataElementClass , DemExternalSRDataElementClass , DemInternalDataElementClass]

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

No Included Containers

10.2.20 DemDTCClass

SWS Item	**Dem641_Conf :**
Container Name	DemDTCClass
Description	This container contains the configuration (parameters) for DTCClass.
Configuration Parameters	

SWS Item	**Dem643_Conf :**		
Name	DemDTCFunctionalUnit		
Description	DTCFuncitonalUnit is a 1-byte value which identifies the corresponding basic vehicle / system function which reports the DTC. This parameter is necessary for the report of severity informations.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem645_Conf :**		
Name	DemDTCSeverity		
Description	DTC severity This parameter depends on automotive manufacturer and is optional.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	DEM_DTC_SEV_CHECK_AT_NEXT_HALT	Check at next halt	
	DEM_DTC_SEV_IMMEDIATELY	Check immediately	
	DEM_DTC_SEV_MAINTENANCE_ONLY	Maintenance required	
	DEM_DTC_SEV_NO_SEVERITY	No severity information available	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem739_Conf :**
Name	DemImmediateNvStorage
Description	Switch to enable immediate storage triggering of an according event memory entriy persistently to NVRAM. true: immediate non-volatile storage triggering enabled false: immediate non-volatile storage triggering disabled
Multiplicity	1
Type	EcucBooleanParamDef
Default value	--

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

SWS Item	**Dem781_Conf :**		
Name	DemObdDTC		
Description	Unique Diagnostic Trouble Code value for OBD		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem640_Conf :**		
Name	DemUdsDTC		
Description	Unique Diagnostic Trouble Code value for UDS (Range: 0x000000 and 0xFFFFFF are reserved for DTC groups by ISO 14229-1)		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 16777214		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemCallbackInitMForF	0..*	The presence of this container indicates, that the Dem has access to an "InitMonitorForFunction" callback, which the Dem will call to initialize a monitor. In case the container has a DemCallbackInitMForFFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackInitMForFFnc, the Dem will have an R-Port requiring the interface CallbackInitMonitorForFunction, whose name is generated by using the unique callback-prefix followed by the event name.

10.2.21 DemEventParameter

SWS Item	**Dem661_Conf :**
Container Name	DemEventParameter
Description	This container contains the configuaration (parameters) for events. Note that this container definition does not explicitly define a symbolic name parameter. Instead, the short name of the container will be used in the Ecu Configuration Description to specify the symbolic name of the diagonstic event.
Configuration Parameters	

SWS Item	**Dem659_Conf :**		
Name	DemEventId		
Description	Unique identifier of a diagnostic event. This parameter should not be changeable by user, because the Id should be generated by Dem itself to prevent gaps and multiple use of an Id. The events should be sequentially ordered beginning with 1 and no gaps in between.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	1 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem660_Conf :**		
Name	DemEventKind		
Description	This parameter is used to distinguish between SW-C and BSW events. SW-C events are reported by Dem_SetEventStatus API and BSW events are reported by Dem_ReportErrorStatus API.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_EVENT_KIND_BSW	The event is a assigned to a BSW module	
	DEM_EVENT_KIND_SWC	The event is a assigned to a SW-C	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem605_Conf :**		
Name	DemMaxNumberFreezeFrameRecords		
Description	This parameter defines the number of according freeze frame records, which can maximal be stored for this event. Therefore all these freeze frame records have the same freeze frame class. This parameter is only required for calculated record numeration (refer to DemTypeOfFreezeFrameRecordNumeration).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DemTypeOfFreezeFrameRecordNumeration		

SWS Item	**Dem642_Conf :**
Name	DemDTCClassRef
Description	This parameter defines the DTC configuration associated with the diagnostic event. It is allowed to have events without a DTC (e.g. for ECU-internal events triggering safety reactions without being reported via diagnostic communication). The same DemDTCClass can be used from several events, to combine these (refer to chapter "Combination of diagonstic event").

Multiplicity	0..1		
Type	Reference to [DemDTCClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem667_Conf :**		
Name	DemExtendedDataClassRef		
Description	This reference defines the link to an extended data class sampler.		
Multiplicity	0..1		
Type	Reference to [DemExtendedDataClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem674_Conf :**		
Name	DemFreezeFrameClassRef		
Description	These references define the links to a freeze frame class sampler.		
Multiplicity	0..1		
Type	Reference to [DemFreezeFrameClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem776_Conf :**		
Name	DemFreezeFrameRecNumClassRef		
Description	This parameter defines the list of dedicated freeze frame record numbers associated with the diagnostic event. These record numbers are assigned to the freeze frame records (instead of calculated record numbers). This parameter is only required for configured record numeration (refer to DemTypeOfFreezeFrameRecordNumeration).		
Multiplicity	1		
Type	Reference to [DemFreezeFrameRecNumClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DemTypeOfFreezeFrameRecordNumeration		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemCallbackClearEventAllowed	0..1	The presence of this container indicates that the Dem has access to a "ClearEventAllowed" callback. In case there is a DemCallbackClearEventAllowedFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackClearEventAllowedFnc, the Dem will have an R-Port requiring the interface CallbackClearEventAllowed whose name is generated by using the unique callback-prefix followed by the event name.
DemCallbackEventDataChanged	0..1	The presence of this container indicates that the Dem has access to an "EventDataChanged" callback. In case there is a DemCallbackEventDataChangedFnc, this parameter

		defines the name of the function that the Dem will call. In case there is no DemCallbackEventDataChangedFnc, the Dem will have an R-Port requiring the interface CallbackEventDataChanged whose name is generated by using the unique callback-prefix followed by the event name.
DemCallbackEventStatusChanged	0..*	The presence of this container indicates, that the Dem has access to an "EventStatusChanged" callback, which the Dem will call to notify other components about the change in the status of an event. In case there is a DemCallbackEvenStatusChangedFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackEvenStatusChangedFnc, the Dem will have an R-Port requiring the interface CallbackEventStatusChanged, whose name is generated by using the unique callback-prefix followed by the event name.
DemCallbackInitMForE	0..1	The presence of this container indicates, that the Dem has access to an "InitMonitorForEvent" callback, which the Dem will call to initialize a monitor. In case the container has a DemCallbackInitMForEFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackInitMForEFnc, the Dem will have an R-Port requiring the interface CallbackInitMonitorForEvent, whose name is generated by using the unique callback-prefix followed by the event name.
DemEventClass	1	This container contains the configuaration (parameters) for EventClass

10.2.22 DemEventClass

SWS Item	**Dem657_Conf :**
Container Name	DemEventClass
Description	This container contains the configuaration (parameters) for EventClass
Configuration Parameters	

SWS Item	**Dem622_Conf :**		
Name	DemAgingAllowed		
Description	Switch to allow aging/unlearning of the event or not. true: aging allowed false: aging not allowed		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem623_Conf :**
Name	DemAgingCycleCounterThreshold
Description	Number ot aging cycles needed to unlearn/delete the event.
Multiplicity	0..1
Type	EcucIntegerParamDef

Range	1 .. 256		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DemAgingAllowed		

SWS Item	**Dem602_Conf :**		
Name	DemConsiderPtoStatus		
Description	This parameter is TRUE, when the event is affected by the Dem PTO handling.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem658_Conf :**		
Name	DemEventDestination		
Description	The event destination assigns events to none, one or multiple origins. If no event destination is assigned to a specific event, the event is handled internally and is not visible externally to the Dcm. If more than one event destination is assigned to a specific event, the event can be present in the corresponding origins.		
Multiplicity	0..4		
Type	EcucEnumerationParamDef		
Range	DEM_DTC_ORIGIN_MIRROR_MEMORY		Event information located in the mirror memory.
	DEM_DTC_ORIGIN_PRIMARY_MEMORY		Event information located in the primary memory.
	DEM_DTC_ORIGIN_SECONDARY_MEMORY		Event information located in the secondary memory.
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem753_Conf :**		
Name	DemEventFailureCycleCounterThreshold		
Description	Defines the number of failure cycles for the event based fault confirmation.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 256		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem755_Conf :**
Name	DemEventOBDReadinessGroup
Description	This parameter specifies the Event OBD Readiness group for PID $01 and PID $41 computation. This parameter is only applicable for emission-related ECUs.

Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_OBD_RDY_AC	A/C system component - spark	
	DEM_OBD_RDY_BOOSTPR	Boost Pressure System - compr.	
	DEM_OBD_RDY_CAT	Catalyst - spark	
	DEM_OBD_RDY_CMPRCMPT	Comprehensive component - spark, compr.	
	DEM_OBD_RDY_EGSENS	Exhaust Gas Sensor - compr.	
	DEM_OBD_RDY_ERG	EGR system - spark, compr.	
	DEM_OBD_RDY_EVAP	Evaporative system - spark	
	DEM_OBD_RDY_FLSYS	Fuel system - spark, compr.	
	DEM_OBD_RDY_HCCAT	Non-Methan HC Catalyst - compr.	
	DEM_OBD_RDY_HTCAT	Heated catalyst - spark	
	DEM_OBD_RDY_MISF	Misfire - spark, compr.	
	DEM_OBD_RDY_NONE	None - spark, compr.	
	DEM_OBD_RDY_NOXCAT	NOx Catalyst - compr.	
	DEM_OBD_RDY_O2SENS	Oxygen sensor - spark	
	DEM_OBD_RDY_O2SENSHT	Oxygen sensor heater - spark	
	DEM_OBD_RDY_PMFLT	Particle Matters Filter - compr.	
	DEM_OBD_RDY_SECAIR	Secondary air system - spark	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem662_Conf :**		
Name	DemEventPriority		
Description	Priority of the event, in view of full event buffer.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 256		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem779_Conf :**		
Name	DemEventSignificance		
Description	Significance of the event, which indicates additional information concerning fault classification and resolution. It can be mapped as Dem-internal data element. It shall be configured, if it is a part of event related data.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	DEM_EVENT_SIGNIFICANCE_FAULT	failure, which affects the component/ECU itself	
	DEM_EVENT_SIGNIFICANCE_OCCURRENCE	issue, which indicates additional information concerning insufficient system behavior	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dem671_Conf :		
Name	DemFFPrestorageSupported		
Description	If this parameter is set to true, then the Prestorage of FreezeFrames is supported by the assigned event. This parameter is useful to calculate the buffer size.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dem624_Conf :		
Name	DemAgingCycleRef		
Description	Reference to the cycle which is triggering the aging of the event. This can either be the same as the operation cycle of the event, or a separate aging cycle reported via API Dem_SetAgingCycleState. If external aging is configured (refer to DemAgingCycleCounterProcessing), this parameter is not used.		
Multiplicity	0..1		
Type	Choice reference to [DemAgingCycle , DemOperationCycle]		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DemAgingAllowed, DemOperationCycleRef, DemAgingCycleCounterProcessing		

SWS Item	Dem746_Conf :		
Name	DemEnableConditionGroupRef		
Description	References an enable condition group.		
Multiplicity	0..1		
Type	Reference to [DemEnableConditionGroup]		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dem752_Conf :		
Name	DemEventFailureCycleRef		
Description	Kind of failure cycle for the event based fault confirmation.		
Multiplicity	0..1		
Type	Reference to [DemOperationCycle]		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DemOperationCycleRef		

SWS Item	Dem702_Conf :		
Name	DemOperationCycleRef		
Description	Kind of operation cycle for the event (e.g. power cycle, driving cycle, ...)		
Multiplicity	1		
Type	Reference to [DemOperationCycle]		
ConfigurationClass	Pre-compile time	X	All Variants

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

SWS Item	**Dem769_Conf :**		
Name	DemStorageConditionGroupRef		
Description	References a storage condition group.		
Multiplicity	0..1		
Type	Reference to [DemStorageConditionGroup]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemDebounceAlgorithmClass	1	Debounce algorithm class: counter based, time based, or monitor internal.
DemIndicatorAttribute	0..255	This container contains the event specific configuration of Indicators.

10.2.23 DemIndicatorAttribute

SWS Item	**Dem681_Conf :**
Container Name	DemIndicatorAttribute
Description	This container contains the event specific configuration of Indicators.
Configuration Parameters	

SWS Item	**Dem682_Conf :**		
Name	DemIndicatorBehaviour		
Description	Behaviour of the linked indicator		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_INDICATOR_BLINKING	The indicator blinks when the event has status FAILED	
	DEM_INDICATOR_BLINK_CONT	The indicator is active and blinks when the event has status FAILED	
	DEM_INDICATOR_CONTINUOUS	The indicator is active when the even has status FAILED	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dem750_Conf :**	
Name	DemIndicatorFailureCycleCounterThreshold	
Description	Defines the number of failure cycles for the WarningIndicatorOnCriteria.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	--	

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

SWS Item	**Dem747_Conf :**		
Name	DemIndicatorFailureCycleSource		
Description	This parameter defines, which failure cycle is used for the WarningIndicatorOnCriteria handling.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEM_FAILURE_CYCLE_EVENT	The event based failure cycle configured in DemEventClass, is used. Therefore, the parameters DemIndicatorFailureCycleRef and DemIndicatorFailureCycleCounterThreshold are not used for this indicator attribute of the event.	
	DEM_FAILURE_CYCLE_INDICATOR	An indicator based failure cycle is used, defined by DemIndicatorFailureCycleRef and DemIndicatorFailureCycleCounterThreshold.	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem748_Conf :**		
Name	DemIndicatorHealingCycleCounterThreshold		
Description	Defines the number of healing cycles for the WarningIndicatorOffCriteria.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem751_Conf :**		
Name	DemIndicatorFailureCycleRef		
Description	Kind of failure cycle for the indicator controlled by the according event used for the WarningIndicatorOnCriteria.		
Multiplicity	0..1		
Type	Reference to [DemOperationCycle]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem749_Conf :**
Name	DemIndicatorHealingCycleRef
Description	Kind of healing cycle for the indicator controlled by the according event used for the WarningIndicatorOffCriteria.
Multiplicity	1

Type	Reference to [DemOperationCycle]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem687_Conf :**		
Name	DemIndicatorRef		
Description	Reference to the used indicator.		
Multiplicity	1		
Type	Reference to [DemIndicator]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.24 DemDebounceAlgorithmClass

SWS Item	**Dem604_Conf :**
Choice container Name	DemDebounceAlgorithmClass
Description	Debounce algorithm class: counter based, time based, or monitor internal.

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
DemDebounceCounterBased	0..1	This container contains the configuration (parameters) for counter based debouncing.
DemDebounceMonitorInternal	0..1	This container contains the configuration (parameters) for monitor internal debouncing.
DemDebounceTimeBase	0..1	This container contains the configuration (parameters) for time based debouncing.

10.2.25 DemDebounceCounterBased

SWS Item	**Dem711_Conf :**
Container Name	DemDebounceCounterBased
Description	This container contains the configuration (parameters) for counter based debouncing.
Configuration Parameters	

SWS Item	**Dem635_Conf :**	
Name	DemDebounceCounterDecrementStepSize	
Description	Defines the step size for decrementation of the internal fault detection counter (PREPASSED).	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 32768	
Default value	--	

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

SWS Item	**Dem618_Conf :**		
Name	DemDebounceCounterFailedThreshold		
Description	Defines the value of the internal fault detection counter, which indicates the failed status.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 32767		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem637_Conf :**		
Name	DemDebounceCounterIncrementStepSize		
Description	Defines the step size for incrementation of the internal fault detection counter (PREFAILED).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 32767		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem685_Conf :**		
Name	DemDebounceCounterJumpDown		
Description	Switch for the activation of Jump-Down. true: Jump-Down activated false: Jump-Down deactivated		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem638_Conf :**		
Name	DemDebounceCounterJumpDownValue		
Description	Jump-Down value of the internal fault detection counter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	-32768 .. 32767		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dem686_Conf :		
Name	DemDebounceCounterJumpUp		
Description	Switch for the activation of Jump-Up. true: Jump-Up activated false: Jump-Up deactivated		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dem639_Conf :		
Name	DemDebounceCounterJumpUpValue		
Description	Jump-Up value of the internal fault detection counter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	-32768 .. 32767		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dem636_Conf :		
Name	DemDebounceCounterPassedThreshold		
Description	Defines the value of the internal fault detection counter, which indicates the passed status.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	-32768 .. 0		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.26 DemDebounceTimeBase

SWS Item	Dem713_Conf :
Container Name	DemDebounceTimeBase
Description	This container contains the configuration (parameters) for time based debouncing.
Configuration Parameters	

SWS Item	Dem716_Conf :
Name	DemDebounceTimeFailedThreshold
Description	Defines the time out duration for "Event Failed" qualification. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. Dem configuration tools must convert

	this float value to the appropriate value format for the use in the software implementation of Dem.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0 .. 3600`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem717_Conf :**		
Name	DemDebounceTimePassedThreshold		
Description	Defines the time out duration for "Event Passed" qualification. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. Dem configuration tools must convert this float value to the appropriate value format for the use in the software implementation of Dem.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0 .. 3600`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.27 DemDebounceMonitorInternal

SWS Item	**Dem712_Conf :**
Container Name	DemDebounceMonitorInternal
Description	This container contains the configuration (parameters) for monitor internal debouncing.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DemCallbackGetFDC	1	The presence of this container indicates, that the Dem has access to a "GetFaultDetectionCounter" callback, which the Dem will call to obtain the value of the fault detection counter. In case the container has a DemCallbackGetFDCFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackGetFDCFnc, the Dem will have a R-Port requiring the interface CallbackGetFaultDetectionCounter, whose name is generated by using the unique callback-prefix followed by the event name.

10.2.28 DemCallbackGetFDC

SWS Item	**Dem630_Conf :**
Container Name	DemCallbackGetFDC
Description	The presence of this container indicates, that the Dem has access to a "GetFaultDetectionCounter" callback, which the Dem will call to obtain the value of the fault detection counter. In case the container has a DemCallbackGetFDCFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackGetFDCFnc, the Dem will have a R-Port requiring the interface CallbackGetFaultDetectionCounter, whose name is generated by using the unique callback-prefix followed by the event name.
Configuration Parameters	

SWS Item	**Dem631_Conf :**		
Name	DemCallbackGetFDCFnc		
Description	Function name of prototype "GetFaultDetectionCounter".		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.29 DemCallbackClearEventAllowed

SWS Item	**Dem607_Conf :**
Container Name	DemCallbackClearEventAllowed
Description	The presence of this container indicates that the Dem has access to a "ClearEventAllowed" callback. In case there is a DemCallbackClearEventAllowedFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackClearEventAllowedFnc, the Dem will have an R-Port requiring the interface CallbackClearEventAllowed whose name is generated by using the unique callback-prefix followed by the event name.
Configuration Parameters	

SWS Item	**Dem609_Conf :**
Name	DemCallbackClearEventAllowedFnc
Description	Function name of prototype "ClearEventAllowed".
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	--
maxLength	--
minLength	--
regularExpression	--

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

No Included Containers

10.2.30 DemCallbackEventDataChanged

SWS Item	**Dem606_Conf :**
Container Name	DemCallbackEventDataChanged
Description	The presence of this container indicates that the Dem has access to an "EventDataChanged" callback. In case there is a DemCallbackEventDataChangedFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackEventDataChangedFnc, the Dem will have an R-Port requiring the interface CallbackEventDataChanged whose name is generated by using the unique callback-prefix followed by the event name.
Configuration Parameters	

SWS Item	**Dem608_Conf :**		
Name	DemCallbackEventDataChangedFnc		
Description	Function name of prototype "EventDataChanged"		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.31 DemCallbackEventStatusChanged

SWS Item	**Dem628_Conf :**
Container Name	DemCallbackEventStatusChanged
Description	The presence of this container indicates, that the Dem has access to an "EventStatusChanged" callback, which the Dem will call to notify other components about the change in the status of an event. In case there is a DemCallbackEvenStatusChangedFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackEvenStatusChangedFnc, the Dem will have an R-Port requiring the interface CallbackEventStatusChanged, whose name is generated by using the unique callback-prefix followed by the event name.

Configuration Parameters

SWS Item	**Dem629_Conf :**		
Name	DemCallbackEventStatusChangedFnc		
Description	Function name of prototype "EventStatusChanged"		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.32 DemCallbackInitMForE

SWS Item	**Dem632_Conf :**
Container Name	DemCallbackInitMForE
Description	The presence of this container indicates, that the Dem has access to an "InitMonitorForEvent" callback, which the Dem will call to initialize a monitor. In case the container has a DemCallbackInitMForEFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackInitMForEFnc, the Dem will have an R-Port requiring the interface CallbackInitMonitorForEvent, whose name is generated by using the unique callback-prefix followed by the event name.
Configuration Parameters	

SWS Item	**Dem601_Conf :**		
Name	DemCallbackInitMForEFnc		
Description	Function name of prototype "InitMonitorForEvent".		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.33 DemFreezeFrameClass

SWS Item	**Dem673_Conf :**
Container Name	DemFreezeFrameClass
Description	This container contains the combinations of DIDs for a non OBD relevant freeze frame class.
Configuration Parameters	

SWS Item	**Dem707_Conf :**		
Name	DemDidClassRef		
Description	For OBD relevant data Multiple PIDs can be relevant per freeze frame.		
Multiplicity	1..255		
Type	Reference to [DemDidClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.34 DemDidClass

SWS Item	**Dem706_Conf :**
Container Name	DemDidClass
Description	This container contains the configuration (parameters) for a data Id class. It is assembled out of one or several data elements.
Configuration Parameters	

SWS Item	**Dem650_Conf :**		
Name	DemDidIdentifier		
Description	Identifier of the Data ID.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Dem617_Conf :**		
Name	DemDidDataElementClassRef		
Description	This reference contains the link to a data element class.		
Multiplicity	1..255		
Type	Choice reference to [DemExternalCSDataElementClass , DemExternalSRDataElementClass , DemInternalDataElementClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.35 DemFreezeFrameRecNumClass

SWS Item	**Dem775_Conf :**
Container Name	DemFreezeFrameRecNumClass
Description	This container contains a list of dedicated, different freeze frame record numbers assigned to an event. The order of record numbers in this list is assigned to the chronological order of the according freeze frame records. dependency: DemTypeOfFreezeFrameRecordNumeration
Configuration Parameters	

SWS Item	**Dem777_Conf :**		
Name	DemFreezeFrameRecordNumber		
Description	This parameter defines a record number for a freeze frame record. This record number is unique per freeze frame record number class. The range of this value is defined by ISO 14229-1 (0x01 .. 0xFE).		
Multiplicity	1..254		
Type	EcucIntegerParamDef		
Range	1 .. 254		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.36 DemExtendedDataClass

SWS Item	**Dem664_Conf :**
Container Name	DemExtendedDataClass
Description	This class contains the combinations of extended data records for an extended data class.
Configuration Parameters	

SWS Item	**Dem774_Conf :**		
Name	DemExtendedDataRecordClassRef		
Description	This reference contains the link to an extended data class record.		
Multiplicity	1..253		
Type	Reference to [DemExtendedDataRecordClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.37 DemExtendedDataRecordClass

SWS Item	**Dem665_Conf :**		
Container Name	DemExtendedDataRecordClass		
Description	This container contains the configuration (parameters) for an extended data record class. It is assembled out of one or several data elements.		
Configuration Parameters			

SWS Item	**Dem666_Conf :**		
Name	DemExtendedDataRecordNumber		
Description	This configuration parameter specifies an unique identifier for an extended data record. One or more extended data records can be assigned to one diagnostic event/DTC. 0xFF and 0xFE are resereved by ISO (therefore the maximal value equals 253).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 253		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem621_Conf :**		
Name	DemExtendedDataRecordUpdate		
Description	This parameter defines the case, when the extended data record is stored/updated.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	DEM_UPDATE_RECORD_NO	This extended data record is only captured for new event memory entries.	
	DEM_UPDATE_RECORD_YES	This extended data record is captured every time.	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem771_Conf :**		
Name	DemDataElementClassRef		
Description	This reference contains the link to a data element class.		
Multiplicity	1..255		
Type	Choice reference to [DemExternalCSDataElementClass , DemExternalSRDataElementClass , DemInternalDataElementClass]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.38 DemDataElementClass

SWS Item	**Dem610_Conf :**
Choice container Name	DemDataElementClass
Description	This container contains the configuration (parameters) for an internal/external data element class.

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
DemExternalCSDataElementClass	0..1	This container contains the configuration (parameters) for an external client/server based data element class. It defines, how the Dem can obtain the value of the data element from either a SW-C or another BSW module. Whether a client/server port or a C function-call is used, is defined by DemDataElementUsePort.
DemExternalSRDataElementClass	0..1	This container contains the configuration (parameters) for an external sender/receiver based data element class. It defines, how the Dem can obtain the value of the data element from a SW-C, by using a sender/receiver port.
DemInternalDataElementClass	0..1	This container contains the configuration (parameters) for an internal data element class.

10.2.39 DemInternalDataElementClass

SWS Item	**Dem684_Conf :**
Container Name	DemInternalDataElementClass
Description	This container contains the configuration (parameters) for an internal data element class.
Configuration Parameters	

SWS Item	**Dem614_Conf :**		
Name	DemDataElementDataSize		
Description	Defines the size of the data element in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem616_Conf :**	
Name	DemInternalDataElement	
Description	This parameter defines the Dem-internal data value, which is mapped to the data element.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DEM_AGINGCTR	map Dem-internal aging counter
	DEM_OCCCTR	map Dem-internal occurrence counter
	DEM_OVFLIND	map Dem-internal overflow indication
	DEM_SIGNIFICANCE	map (static) Dem-internal event significance (refer to

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

No Included Containers

10.2.40 DemExternalCSDataElementClass

SWS Item	**Dem668_Conf :**
Container Name	DemExternalCSDataElementClass
Description	This container contains the configuration (parameters) for an external client/server based data element class. It defines, how the Dem can obtain the value of the data element from either a SW-C or another BSW module. Whether a client/server port or a C function-call is used, is defined by DemDataElementUsePort.
Configuration Parameters	

SWS Item	**Dem646_Conf :**		
Name	DemDataElementDataSize		
Description	Defines the size of the data element in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem619_Conf :**		
Name	DemDataElementReadFnc		
Description	In case of DemDataElementUsePort is false, this parameter defines the prototype of the C function "ReadDataElement" used to get the according value.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem647_Conf :**
Name	DemDataElementUsePort
Description	If the parameter is set to True, a R-Port is generated, to obtain the data element (interface DataServices_<SyncDataElement>). If the parameter is set to False, the information is obtained by C function-call on another BSW

	module specified by the parameter DemDataElementReadFnc.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.41 DemExternalSRDataElementClass

SWS Item	**Dem669_Conf :**
Container Name	DemExternalSRDataElementClass
Description	This container contains the configuration (parameters) for an external sender/receiver based data element class. It defines, how the Dem can obtain the value of the data element from a SW-C, by using a sender/receiver port.
Configuration Parameters	

SWS Item	**Dem615_Conf :**		
Name	DemDataElementDataSize		
Description	Defines the size of the data element in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dem770_Conf :**		
Name	DemDataElementInstanceRef		
Description	Instance Reference to the actual OperationPrototype which shall be traced.		
Multiplicity	1		
Type	Instance reference to [VARIABLE-DATA-PROTOTYPE context: ROOT-SW-COMPOSITION-PROTOTYP SW-COMPONENT-PROTOTYPE+ PORT-PROTOTYPE]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.3 Published Information

[Dem628] 「The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [4] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1].」()

Additional module-specific published parameters are listed below if applicable.

11 Requirement Change History

11.1 Changes to Release 3.1

11.1.1 Deleted SWS Items

SWS Item	*Rationale*
Dem021	Event specific behavior is covered by Dem039
Dem178	Replaced by a note (refer to API Dem_GetVersionInfo)
Dem285	Requirement was duplicated by Dem016
Dem265	Requirement has only described a condition, introduced to Dem264.
Dem047	Requirement regarding SW-C / other modules (no Dem requirement)
Dem275	NvM is responsible for the error handling (and retry mechanism)
Dem276	NvM is responsible for the error handling (and retry mechanism)
Dem283	Requirement was duplicated by Dem261
Dem332	Requirement was duplicated by Dem051
Dem380	Obsolete Requirement (as ‘ConditionCheckRead’ not used any more)
Dem186	Requirement is redundant to Dem187
Dem091	Replaced by reformulation of Dem036 and Dem379
Dem192	Obsolete
Dem237	Removed API Dem_GetFreezeFrameDataIdentifierByDTC
Dem073	Removed functionality of Dem_GetFreezeFrameDataIdentifierByDTC
Dem034	Removed Dem_[En|Dis]ableEventStatusUpdate
Dem081	Removed Dem_[En|Dis]ableEventStatusUpdate
Dem082	Removed Dem_[En|Dis]ableEventStatusUpdate
Dem244	Removed Dem_EnableEventStatusUpdate
Dem245	Removed Dem_DisableEventStatusUpdate
Dem199	Dem_SetValueByOemId is obsolete (no use-case)
Dem200	Dem_SetValueByOemId is obsolete (no use-case)
Dem015	Requirement was duplicated by Dem046
Dem278	Requirement was duplicated by Dem261
Dem363	Requirement was duplicated by Dem282
Dem305	No requirement on the module. Changed to note.
Dem307	No requirement on the module. Changed to note.
Dem309	No requirement on the module. Changed to note.
Dem306	No requirement on the module. Changed to note.
Dem345	No requirement on the module. Changed to note.
Dem310	No requirement on the module. Changed to note.
Dem311	No requirement on the module. Changed to note.
Dem448	Due to review result of MS3 (describes implementation details).
Dem454	Due to review result of MS3 (delay is not needed).
Dem456	Due to review result of MS3 (delay is not needed).
Dem457	Due to review result of MS3 (delay is not needed).
Dem262	Replaced GetExtendedDataRecord by ReadData.
Dem326	Replaced GetPIDValue by ReadData.
Dem445	Due to review result of MS3 (Requirement was duplicated by Dem446)
Dem452	Due to review result of MS3 (Requirement was duplicated by Dem453)
Dem353	Requirement was duplicated by Dem354
Dem531	Due to review result of MS3
Dem532	Due to review result of MS3

Dem534	Due to review result of MS3 (Requirement was duplicated by Dem506)
Dem168	It is covered by Dem167

During Revision 2:

SWS Item	***Rationale***
Dem561	Dem_DcmGetPID02 is coverd by Dem_GetFreezeFrameDataByRecord
Dem549	Functionality was duplicated by Dem349

During Revision 3:

SWS Item	***Rationale***
Dem127	Requirement was duplicated by Dem019
Dem336	Functionality (DTC format) was changed
Dem644_Conf	DTC kind is derived from DemObdDTC multiplicity.

11.1.2 Replaced SWS Items

SWS Item	***replaced by SWS Item***	***Rationale***
none		

11.1.3 Changed SWS Items

SWS Item	***Rationale***
Dem267	Adjusted variant name according to AUTOSAR standard name
Dem268	Adjusted variant name according to AUTOSAR standard name
Dem046	Reworded
Dem189	Reworded because of incorporation of Dem behavior concept
Dem006	Reworded to clarify the UDS DTC status byte support
Dem184	Reworded to clarify the storage of a reported event
Dem190	Reworded to clarify the storage of freeze frames
Dem284	Reworded to formulate it Dem-related (instead of SW-C-related)
Dem261	Changed to data element based interface
Dem282	Changed to data element based interface
Dem162	Reformulated
Dem344	Configuration option removed. FDC always reset.
Dem067	Precised requirement
Dem061	Precised requirement
Dem151	Updated requirement
Dem339	Precised requirement

During Revision 2:

SWS Item	***Rationale***
Dem067	Only Inter Module Version Checks are specified
Dem190	Precised requirement
Dem468	Precised requirement
Dem300	Precised requirement

Dem235	Renamed API
Dem225	Precised requirement
Dem376	Precised requirement
Dem074	Adusted requirement to Dcm needs
Dem076	Adusted requirement to Dcm needs
Dem152	Removed implementation-specific files, Precised requirement
Dem115	Removed implementation-specific files, Precised requirement
Dem447	Considered Dem_ResetEventStatus()

During Revision 3:

SWS Item	***Rationale***
Dem515	Precised requirement
Dem029	Extension of requirement due to FiM
Dem517	Extension of requirement due to Dlt
Dem618	Corrected reference to interface
Dem619	Corrected reference to interface
Dem151	Clarified header-file structure regarding RTE
Dem599	Corrected referenced interface name
Dem600	Corrected referenced interface name
Dem270	Reduced functionality to only one DTC of one origin
Dem271	Reduced functionality to only one DTC of one origin
Dem233	Reduced functionality to only one DTC of one origin
Dem208	Precised API parameter description
Dem219	Precised affected event memory kind
Dem221	Precised requirement
Dem224	Corrected API description
Dem003	Reworked requirement
Dem376	Reworked requirement
Dem663_Conf	Reworked global data capturing configuration
Dem672_Conf	Reworked global data capturing configuration
Dem447	No consideration of enable conditions for Dem_ResetEventStatus
Dem449	No consideration of enable conditions for Dem_ResetEventStatus
Dem450	No consideration of enable conditions for Dem_ResetEventStatus
Dem019	Consideration of external aging
Dem493	Precised requirement
Dem489	Precised requirement
Dem494	Precised requirement
Dem490	Only valid for Dem-internal aging
Dem496	Precised requirement
Dem640_Conf	Extension of requirement on multiple DTC formats
Dem198	Reworked API syntax to support multiple DTC formats
Dem589	Reworked API syntax to support multiple DTC formats
Dem208	Reworked API syntax to support multiple DTC formats
Dem209	Reworked API syntax to support multiple DTC formats
Dem212	Reworked API syntax to support multiple DTC formats
Dem235	Reworked API syntax to support multiple DTC formats
Dem236	Reworked API syntax to support multiple DTC formats
Dem238	Reworked API syntax to support multiple DTC formats
Dem239	Reworked API syntax to support multiple DTC formats
Dem240	Reworked API syntax to support multiple DTC formats
Dem241	Reworked API syntax to support multiple DTC formats
Dem173	Reworked DET error codes
Dem370	Precised DET error code handling
Dem152	Precised requirement
Dem383	Simplified requirement
Dem397	Precised requirement

Dem410	Precised requirement
Dem624_Conf	Extended aging cycle configuration

11.1.4 Added SWS Items

SWS Item	*Rationale*
Dem382	Definition of event priority
Dem383	Definition of event priority
Dem385	Description of UDS DTC status bit transitions according to ISO 14229-1
Dem386	Description of UDS DTC status bit transitions according to ISO 14229-1
Dem387	Description of UDS DTC status bit transitions according to ISO 14229-1
Dem388	Description of UDS DTC status bit transitions according to ISO 14229-1
Dem389	Description of UDS DTC status bit transitions according to ISO 14229-1
Dem390	Description of UDS DTC status bit transitions according to ISO 14229-1
Dem391	Description of UDS DTC status bit transitions according to ISO 14229-1
Dem392	Description of UDS DTC status bit transitions according to ISO 14229-1
Dem393	Description of UDS DTC status bit transitions according to ISO 14229-1
Dem394	Description of UDS DTC status bit transitions according to ISO 14229-1
Dem395	Description of UDS DTC status bit transitions according to ISO 14229-1
Dem396	Requirement of event retention
Dem397	Support of event memory overflow indication
Dem398	Support of event memory overflow indication
Dem399	Support of event memory overflow indication
Dem400	Define mechanism for event displacement
Dem401	Define mechanism for event displacement
Dem402	Define mechanism for event displacement
Dem403	Define mechanism for event displacement
Dem404	Define mechanism for event displacement
Dem405	Define mechanism for event displacement
Dem406	Define mechanism for event displacement
Dem407	Define mechanism for event displacement
Dem408	Define mechanism for event displacement
Dem409	Define mechanism for event displacement
Dem410	Definition of reporting order of events
Dem411	Definition of reporting order of events
Dem412	Definition of reporting order of events
Dem413	Requirement of debouncing of events (generic)
Dem414	Support of event specific debouncing algorithm – counter based
Dem415	Support of event specific debouncing algorithm – counter based
Dem416	Support of event specific debouncing algorithm – counter based
Dem417	Support of event specific debouncing algorithm – counter based
Dem418	Support of event specific debouncing algorithm – counter based
Dem419	Support of event specific debouncing algorithm – counter based
Dem420	Support of event specific debouncing algorithm – counter based
Dem421	Support of event specific debouncing algorithm – counter based
Dem422	Support of event specific debouncing algorithm – counter based
Dem423	Support of event specific debouncing algorithm – counter based
Dem424	Support of event specific debouncing algorithm – counter based
Dem425	Support of event specific debouncing algorithm – counter based
Dem426	Support of event specific debouncing algorithm – time based
Dem427	Support of event specific debouncing algorithm – time based
Dem428	Support of event specific debouncing algorithm – time based
Dem429	Support of event specific debouncing algorithm – time based
Dem430	Support of event specific debouncing algorithm – time based

Dem431	Support of event specific debouncing algorithm – time based
Dem432	Support of event specific debouncing algorithm – time based
Dem433	Support of event specific debouncing algorithm – time based
Dem434	Support of event specific debouncing algorithm – time based
Dem435	Support of event specific debouncing algorithm – time based
Dem436	Description of further debounce mechanisms
Dem437	Requirement of monitor internal debouncing
Dem438	Define fault detection counter initialization
Dem439	Fault detection counter retrieval
Dem440	Description of event combination
Dem441	Description of event combination
Dem442	Description of event combination
Dem443	Description of event combination
Dem444	Support of enable conditions
Dem445	Support of enable conditions
Dem446	Support of enable conditions
Dem447	Support of enable conditions
Dem448	Support of enable conditions (see deleted SWS Items of this release)
Dem449	Support of enable conditions
Dem450	Support of enable conditions
Dem451	Support of storage conditions
Dem452	Support of storage conditions
Dem453	Support of storage conditions
Dem454	Support of storage conditions (see deleted SWS Items of this release)
Dem455	Support of storage conditions
Dem456	Support of storage conditions (see deleted SWS Items of this release)
Dem457	Support of storage conditions (see deleted SWS Items of this release)
Dem458	Support of storage conditions
Dem459	Support of storage conditions
Dem460	Requirements of event related data
Dem461	Define the storage of freeze frame data
Dem462	Define the storage of freeze frame data
Dem463	Define the storage of freeze frame data
Dem464	Support of pre-storage of freeze frame data
Dem465	Support of pre-storage of freeze frame data
Dem466	Storage of extended data records
Dem467	Storage of extended data record
Dem468	Storage of extended data record
Dem469	Define the configuration of event related data
Dem470	Define the configuration of event related data
Dem471	Define the configuration of event related data
Dem472	Define the configuration of event related data
Dem473	Define the configuration of event related data
Dem474	Support of notification of data changes
Dem475	Support of notification of data changes
Dem476	Support of notification of data changes
Dem477	Support of notification of data changes
Dem478	Support of notification of data changes
Dem479	Support of notification of data changes
Dem480	Requirements of operation cycle management
Dem481	Requirements of operation cycle management
Dem482	Requirements of operation cycle management
Dem483	Requirements of operation cycle management
Dem484	Requirements of operation cycle management
Dem485	Requirements of operation cycle management
Dem486	Requirements of operation cycle management
Dem487	Requirements of operation cycle management

Dem488	Define aging/unlearning of diagnostic events
Dem489	Define aging/unlearning of diagnostic events
Dem490	Define aging/unlearning of diagnostic events
Dem491	Define aging/unlearning of diagnostic events
Dem492	Define aging/unlearning of diagnostic events
Dem493	Define aging/unlearning of diagnostic events
Dem494	Define aging/unlearning of diagnostic events
Dem495	Define aging/unlearning of diagnostic events
Dem496	Define aging/unlearning of diagnostic events
Dem497	Define aging/unlearning of diagnostic events
Dem498	Define aging/unlearning of diagnostic events
Dem499	Support of warning indicator handling
Dem500	Support of warning indicator handling
Dem501	Support of warning indicator handling
Dem502	Support of warning indicator handling
Dem503	Support of warning indicator handling
Dem504	Support of warning indicator handling
Dem505	Support of warning indicator handling
Dem506	Support of warning indicator handling
Dem507	Support of warning indicator handling
Dem508	Support of warning indicator handling
Dem509	Support of warning indicator handling
Dem510	Support of warning indicator handling
Dem511	Support of warning indicator handling
Dem512	Requirements of AUTOSAR architecture
Dem513	Access to DTCs and status information
Dem514	Requirements of DTC storage
Dem515	Requirements of DTC storage
Dem516	Requirements of DTC storage
Dem517	Interaction with Diagnostic Log & Trace (Dlt)
Dem518	Requirements of Development Error Tracer (Det)
Dem519	Support for debugging
Dem520	Support for debugging
Dem521	Support for debugging
Dem522	Support for debugging
Dem523	Requirement for event occurrence
Dem524	Requirement for event occurrence
Dem525	Descripe status bit transition
Dem526	Requirement of counter based debouncing
Dem527	Requirement of time based debouncing
Dem528	Support fault confirmation mechanism
Dem529	Support fault confirmation mechanism
Dem530	Support fault confirmation mechanism
Dem531	Support fault confirmation mechanism
Dem532	Support fault confirmation mechanism
Dem533	Support fault confirmation mechanism
Dem534	Support fault confirmation mechanism
Dem535	Support fault confirmation mechanism
Dem536	Requirement for event combination
Dem537	Requirement for event combination
Dem538	Requirement for event combination
Dem539	Requirement for event combination
Dem540	Requirement for event combination
Dem541	Requirement for event combination
Dem542	Requirement for event combination
Dem543	Requirement of Enable/Storage condition
Dem544	Support of warning indicator handling

Dem545	Support of warning indicator handling
Dem546	Support of warning indicator handling
Dem547	Describe asynchronous Dcm operation
Dem548	Requirement for support of different event memories
Dem549	Requirement for configuration parameter DemMonitorGroupRef
Dem550	Support of immediate non-volatile storage
Dem551	Support of immediate non-volatile storage
Dem552	Support of immediate non-volatile storage
Dem553	Function Dem_SetOperationCycleCntValue
Dem554	Function Dem_SetAgingCycleState
Dem555	Function Dem_SetAgingCycleCounterValue
Dem556	Function Dem_SetStorageCondition
Dem557	Function Dem_GetEventExtendedDataRecord
Dem558	Function Dem_GetEventFreezeFrameData
Dem559	Function Dem_GetEventMemoryOverflow
Dem560	Function Dem_DcmCancelOperation
Dem561	Function Dem_DcmGetPID02
Dem562	Function EventDataChanged
Dem563	Function ClearEventAllowed
Dem564	Function ReadDataElement
Dem565	Interaction with Diagnostic Communication Manager (Dcm)
Dem566	Support of warning indicator handling (MS3 review)
Dem567	Support of warning indicator handling (MS3 review)
Dem568	Support of warning indicator handling (DemIndicatorFailureCycleSource)
Dem569	Clarify positive response after clear diagnostic information
Dem570	Clarify positive response after clear diagnostic information
Dem571	Clarify positive response after clear diagnostic information
Dem572	Clarify positive response after clear diagnostic information
Dem573	Clarify positive response after clear diagnostic information
Dem574	Clarify access of event related data
Dem575	Clarify access of event related data
Dem576	Clarify access of event related data
Dem577	Clarify storage of operation cycles (volatile or non-volatile)
Dem578	Define Dem behaviour in case of NvM-reading error
Dem579	Define Dem behaviour in case of NvM-writing error
Dem580	Requirement for event occurrence
Dem581	Requirement for FF record numbers
Dem582	Requirement for FF record numbers
Dem583	Requirement for FF record numbers
Dem584	OBD related timing constraints
DEM001_PI	Rework of Published Information

During Revision 2:

SWS Item	***Rationale***
Dem585	Freeze frame displacement
Dem586	DTC suppression
Dem587	DTC suppression
Dem588	DTC suppression
Dem589	DTC suppression
Dem590	Permanent DTC
Dem591	Storage Conditions
Dem592	Event Significance
Dem593	DTC functional unit
Dem594	DTC functional unit
Dem595	Requirement on Dem_SetFreezeFrameRecordFilter
Dem596	Requirement on Dem_ReadDataOfOBDFreezeFrame

Dem597	Requirement on Dem_ReadDataOfOBDFreezeFrame
Dem598	Requirement on Dem Service Interface via Rte
Dem599	Requirement on Dem Service Interface via Rte
Dem600	Requirement on Dem Service Interface via Rte
Dem601	Requirement on Dem Service Interface via Rte
Dem602	Requirement on Dem Service Interface via Rte
Dem603	Requirement on Dem Service Interface via Rte
Dem604	Requirement on Dem Service Interface via Rte
Dem605	Requirement on Dem Service Interface via Rte
Dem606	Requirement on Dem Service Interface via Rte
Dem607	Requirement on Dem Service Interface via Rte
Dem608	Requirement on Dem Service Interface via Rte
Dem609	Requirement on Dem Service Interface via Rte
Dem610	Requirement on Dem Service Interface via Rte
Dem611	Requirement on Dem Service Interface via Rte
Dem612	Requirement on Dem Service Interface via Rte
Dem613	Requirement on Dem Service Interface via Rte
Dem614	Requirement on Dem Service Interface via Rte
Dem615	Requirement on Dem Service Interface via Rte
Dem616	Requirement on Dem Service Interface via Rte
Dem617	Requirement on Dem Service Interface via Rte
Dem618	Requirement on Dem Service Interface via Rte
Dem619	Requirement on Dem Service Interface via Rte
Dem620	Requirement on Dem Service Interface via Rte
Dem621	Requirement on Dem Service Interface via Rte
Dem622	Requirement on Dem Service Interface via Rte
Dem623	Function Dem_GetDTCOfOBDFreezeFrameData
Dem624	Function Dem_GetDTCOfOBDFreezeFrameData
Dem625	Requirement on occurrence counter
Dem626	Requirement on control DTC setting

During Revision 3:

SWS Item	***Rationale***
Dem627	Requirement Id on Dem_SetPtoStatus API
Dem628	Requirement on Published Information
Dem629	Requirement on cancellation of Dem_ClearDTC
Dem630	Requirement on empty response behavior (related to Dcm)
Dem631	Requirement on empty response behavior (related to Dcm)
Dem632	Requirement on new Dem-Dlt APIs
Dem633	Requirement on new Dem-Dlt APIs
Dem634	Requirement on new Dem-Dlt APIs
Dem635	Requirement on new Dem-Dlt APIs
Dem636	Requirement on new Dem-Dlt APIs
Dem637	Requirement on new Dem-Dlt APIs
Dem638	Requirement on E_NOT_OK-handling of Dem_ResetEventStatus
Dem639	Requirement on external aging
Dem640	Requirement on external aging
Dem641	Requirement on external aging
Dem642	Requirement on external aging
Dem643	Requirement on aging counter reporting
Dem644	Requirement on aging counter reporting
Dem645	Requirement on DTC format
Dem646	Requirement on aging counter reporting
Dem647	Requirement on aging counter reporting
Dem648	Requirement on Dem_DisableDTCRecordUpdate
Dem999	Requirement on non-applicable SRS BSW General requirements

Dem781_Conf	New parameter DemObdDTC
Dem782_Conf	New parameter DemSecondaryFunctionIdRef
Dem783_Conf	New parameter DemOperationCycleProcessing
Dem784_Conf	New parameter DemStatusBitHandlingTestFailedSinceLastClear
Dem785_Conf	New container DemAgingCycle

12 Not applicable requirements

[Dem999]「These requirements are not applicable to this specification.」(BSW5, BSW161, BSW162, BSW164, BSW168, BSW170, BSW171, BSW324, BSW326, BSW327, BSW331, BSW337, BSW338, BSW339, BSW341, BSW347, BSW348, BSW350, BSW353, BSW357, BSW359, BSW360, BSW361, BSW369, BSW374, BSW375, BSW379, BSW00382, BSW00387, BSW00433, BSW00434)