Document Title	Specification of Diagnostic Communication Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	018
Document Classification	Standard

Document Version	4.1.0
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change Description
28.02.2014	4.1.0	AUTOSAR Release Management	• Rework interfaces with PduR • Remove mandatory jump in bootloader with LinkControl service (0x87) • Editorial changes • Removed chapter(s) on change documentation
17.05.2012	4.0.0	AUTOSAR Administration	• Introduced automatic response for ComM indication to Dcm • Improvement of bootloader interaction • Introduced C/S Interface for CommunicationControl • Removed OpStatus parameter from GetSeed function
28.04.2011	3.4.0	AUTOSAR Administration	• Add bootloader interaction (partially imported from R4.0) • Coordination with BswM for session management • Change RecordNumber range to 0x00..0xFE for UDS Service 0x019 with subfunction 0x04 in accordance with ISO 14229-1 • Add support for cancelation of reception TP I-PDUS (similar to AUTOSAR 4.0) • Legal disclaimer revised

Document Change History

Date	Version	Changed by	Change Description
16.09.2010	3.3.0	AUTOSAR Administration	• Add Dcm_MsgItemType, MsgContextType sub-types, add the returned value for interfaces "Dcm_GetSecurityLevel()", "Dcm_GetActiveProtocol" and "Dcm_GetSesCtrlType()", add reference to InputOutputControlByIdentifier (0x2F) in ReadData section., add Dcm687 • Remove doubloon definition for DsdInternal_ProcessingDone and DspInternal_<DiagnosticService>, remove Dcm_GetSesTimingValues, Dcm_PrepareSesTimingValues,Dcm_SetSesTimingValues • Change Dcm385, change Dcm333 • Legal disclaimer revised
29.01.2010	3.2.0	AUTOSAR Administration	• Correct DcmDslProtocolRxAddrType • Legal disclaimer revised
06.08.2008	3.1.0	AUTOSAR Administration	• Introduction of OBD support • generation of artefacts from the models according to the AUTOSAR process • Identification of requirements and correct formulation of specification items as requirements • General cleanup • Legal discaimer revised
28.11.2007	3.0.0	AUTOSAR Administration	• Rework of the interfaces with RTE (remove of Central Diagnostic SWC concept) • Correction of issues identified on R2.1 • Document meta information extended • Small layout adaptations made
24.01.2007	2.1.1	AUTOSAR Administration	• "Advice for users" revised • "Revision Information" added
05.12.2006	2.1.0	AUTOSAR Administration	• Corrections in configuration chapter • Rework on interface between DCM and DEM according to changes in DEM SWS • Corrections in Sequence diagram • Addition of header files inclusions • Legal disclaimer revised
29.06.2006	2.0.1	AUTOSAR Administration	Layout Adaptations

<table>
<tr><td colspan="4"><h1>Document Change History</h1></td></tr>
</table>

Date	Version	Changed by	Change Description
26.04.2006	2.0.0	AUTOSAR Administration	• Document structure adapted to common Release 2.0 SWS Template. • Major changes in chapter 10 • Structure of document changed partly • Other changes see chapter 11
10.07.2005	1.0.0	AUTOSAR Administration	Initial release

Disclaimer

This specification and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only.
For any other purpose, no part of the specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

AUTOSAR specifications 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 specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, 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 DCM SWS describes the functionality, the API, and the configuration of the AUTOSAR Basic Software module DCM (Diagnostic Communication Manager). The DCM module provides a common API for diagnostic services. The functionality of the DCM module is used by external diagnostic tools during the development, manufacturing or service.
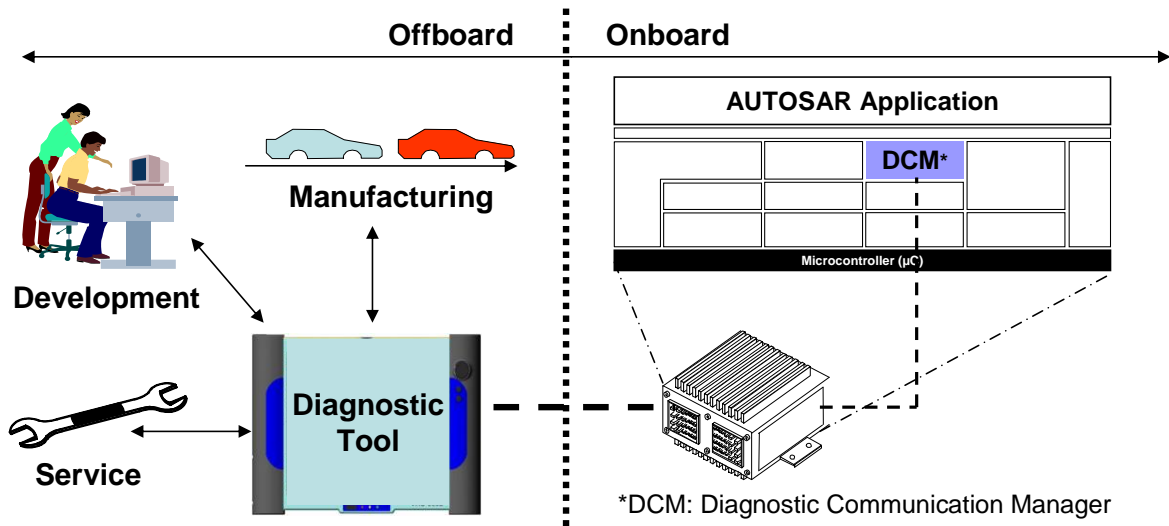
Figure 1 Overview of the communication between the external diagnostic tools and the onboard AUTOSAR Application

The DCM module ensures diagnostic data flow and manages the diagnostic states, especially diagnostic sessions and security states. Furthermore, the DCM module checks if the diagnostic service request is supported and if the service may be executed in the current session according to the diagnostic states.The DCM module provides the OSI-Layers 5 to 7 of Table 1 Diagnostic protocols and OSI-Layer.

OSI-Layer	Protocols				
7	UDS-Protocol – ISO14229-1				Legislated OBD – ISO15031-5
6	-	-	-	-	-
5	ISO15765-3	-	-	-	ISO 15765-4
4	ISO15765-2	-	-	-	-
3	ISO15765-2	-	-	-	ISO 15765-4
2	CAN-Protocol	LIN-Protocol	FlexRay	MOST	ISO 15765-4
1	CAN-Protocol	LIN-Protocol	FlexRay	MOST	ISO 15765-4

Table 1 Diagnostic protocols and OSI-Layers

At OSI-level 7, the DCM module provides an extensive set of ISO14229-1 [9] services. In addition, the DCM module provides mechanisms to support the OBD services $01 - $0A defined in documents [14] and [10]. With these services, Autosar OBD functionality is capable of meeting all light duty OBD regulations worldwide (California OBDII, EOBD, Japan OBD, and all others).

At OSI-level 5, the DCM module handles the network-independent sections of the following specifications:

- ISO15765-3 [12]: Implementation of unified diagnostic services (UDS on CAN)
- ISO15765-4 [13]: Requirements for emission-related systems

In the AUTOSAR Architecture the Diagnostic Communication Manager is located in the Communication Services (Service Layer).
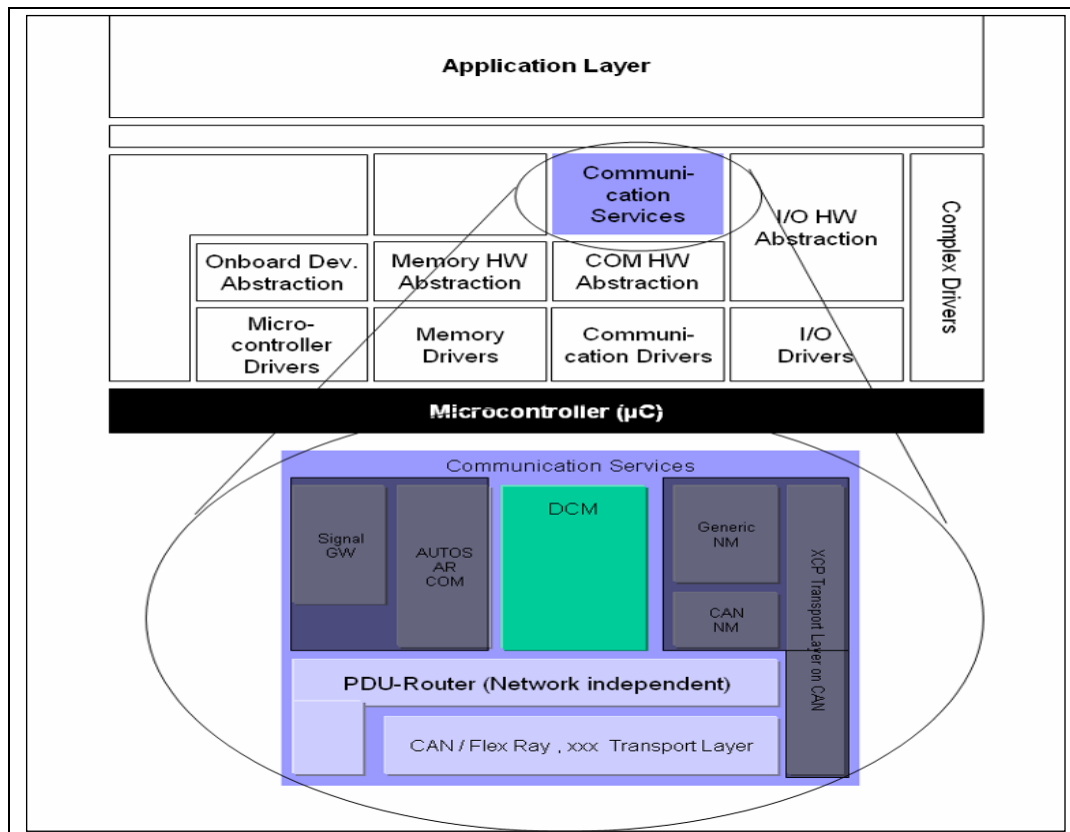
Figure 2 Position of the DCM module in AUTOSAR Architecture

The DCM module is network-independent. All network-specific functionality (the specifics of networks like CAN, LIN, FlexRay or MOST) is handled outside of the DCM module. The PDU Router (PduR) module provides a network-independent interface to the DCM module.

The DCM module receives a diagnostic message from the PduR module. The DCM module processes and checks internally the diagnostic message. As part of processing the requested diagnostic service, the DCM will interact with other BSW modules or with SW-Components (through the RTE) to obtain requested data or to execute requested commands. This processing is very service-specific. Typically, the DCM will assemble the gathered information and send a message back through the PduR module.

2 Acronyms and abbreviations

2.1 Terms

Term	Description
Application Layer	The Application Layer is placed above the RTE. Within the Application Layer the AUTOSAR Software-Components are placed.
Channel	A link at which a data transfer can take place. If there is more than one Channel, there is normally some kind of ID assigned to the Channel.
Diagnostic Channel	A link at which a data transfer between a diagnostic tool and an ECU can take place. Example: An ECU is connected via CAN and the diagnostic channel has an assigned CAN-ID. Diagnostic channels connected to other bus-systems such as MOST, FlexRay, LIN, etc. are also possible.
External Diagnostic Tool	A device which is NOT permanently connected to the vehicle communication network. This External Diagnostic Tool can be connected to the vehicle for various purposes, as e.g. for: • development, • manufacturing, and • service (in a garage). Example External Diagnostic Tools are: • a diagnostic tester, • an OBD scan tool. The External Diagnostic Tool is to be connected by a mechanic to gather information from "inside" the car.
Freeze Frame	A set of the vehicle/system operation conditions at a specific time.
Functional Addressing	The diagnostic communication model where a group or all nodes of a specific communication network receive a message from one sending node (1-n communication). This model is also referred to as 'broadcast' or 'multicast'. OBD communication will always be done in the Functional Addressing mode.
Internal Diagnostic Tool	A device/ECU which is connected to the vehicle communication network. The Internal Diagnostic Tool can be used for: advanced event tracking, advanced analysis, for service. The behavior of the Internal Diagnostic Tool can be the same as of an External Diagnostic Tool. The notion of "Internal Diagnostic Tool" does not imply that it is included in each ECU as an AUTOSAR Software-Component.
Physical Addressing	The diagnostic communication model where a node of a specific communication network receives a message from one sending node (1-1 communication). This model is also referred to as 'unicast'.
UDS Service	this refers to a UDS Service as defined in ISO14229-1 (see [9])
OBD Service	This refers to an OBD Service as defined in ISO15031-5 (see [10])

2.2 Abbreviations

Abbreviation/ Acronym:	Description:
API	Application Programming Interface
CAN	Controller Area Network
DCM	Diagnostic Communication Manager
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DID	Data IDentifier
DSD	Diagnostic Service Dispatcher (submodule of the DCM module)
DSL	Diagnostic Session Layer (submodule of the DCM module)
DSP	Diagnostic Service Processing (submodule of the DCM module)
DTC	Diagnostic Trouble Codes
ID	Identifier
LIN	Local Interconnect Network
MCU	Micro-Controller Unit
MOST	Media Orientated System Transport
NRC	Negative Response Code
OBD	On-Board Diagnosis
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PID	Parameter Identifier
ROE	ResponseOnEvent
RTE	Runtime Environment
SAP	Service Access Point
SDU	Service Data Unit
SID	Service Identifier
SW-C	Software-Component
TP	Transport Protocol
UDS	Unified Diagnostic Services
Xxx_	Placeholder for an API provider

2.3 Typographical Conventions

This document uses the following typographical conventions:
- See configuration parameter *myConfigurationParameter*: this is a reference to a configuration parameter which can be found in Chapter 10.
- `myFunction()`: this is a function provided or required by the module as defined in Chapter 8.

3 Related documentation

3.1 Input documents

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

[2] Specification of Module PDU Router,
AUTOSAR_SWS_PDU_Router.pdf

[3] AUTOSAR Requirements on Basic Software Module Diagnostic
AUTOSAR_SRS_Diagnostic.pdf

[4] Specification of Module Communication Manager,
AUTOSAR_SWS_ComManager.pdf

[5] AUTOSAR Layered Software Architecture,
AUTOSAR_LayeredSoftwareArchitecture.pdf

[6] AUTOSAR ECU Configuration Specification,
AUTOSAR_ECU_Configuration.pdf

[7] Specification of Diagnostics Event Manager,
AUTOSAR_SWS_DEM.pdf

[8] AUTOSAR Basic Software Module Description Template,
AUTOSAR_BSW_Module_Description.pdf

3.2 Related standards and norms

[9] ISO14229-1 Unified diagnostic services (UDS) - Part 1: Specification and
Requirements (Release 2006 12-01)

[10] ISO15031-5.4 Communication between vehicle and external equipment for
emissions-related diagnostics - Part 5: Emission-related diagnostic services
(2005-01-13)

[11] ISO15765-2: Road vehicles -- Diagnostics on Controller Area Networks (CAN)
-- Part 2: Network layer services

[12] ISO15765-3: Diagnostics on controller area network (CAN) - Part 3:
Implementation of unified diagnostic services (UDS on CAN) (Release 2004 10-
06)

[13] ISO15765-4 Diagnostics on controller area network (CAN) - Part 4:
Requirements for emission-related systems (Release 2005 01-04)

[14] SAE J1979 Rev May 2007

4   Constraints and assumptions

4.1 Limitations

The following limitations apply when using the DCM module:
- The DCM module does not provide any diagnostic multi-channel capabilities. This implies that the DCM module does not support BSW04080. This means that parallel requests of a tester addressed to different independent functionalities cannot be processed by a single DCM module. Furthermore, the concept currently implemented does not take more than one instance of a DCM module residing in one ECU into account. As the legislator requires that emission-related service requests according to ISO15031-5 [10] shall be processed prior to any enhanced diagnostic requests, the DCM module provides a protocol switching mechanism based on protocol prioritization.
- The DCM module expects that the Length parameter in `Dcm_ProvideTxBuffer()` (indicating the minimal size of the transmit-buffer) is always 0 (zero). As a result, the DCM will not support implementations of a TP that call `ProvideTxBuffer()` with a specific minimal length (such as the "Retry mechanism" of the FlexRay TP).
- On calling `Dcm_ProvideRxBuffer()`, certain TPs (like FlexRay TP) will put an initial value in the length of the parameter `PduInfoPtr`. This value is the minimal size of the receive-buffer that must be provided by the DCM module. The limitation is that the DCM module will not take into account this value (it will behave as if the value is always 0). The DCM module will return a buffer large enough to contain the entire diagnostic message. This size is part of the DCM configuration.
- The DCM module cannot use the paged-buffer mechanism for the following services:
  - UDS Service 0x19 (subfunctions 0x02, 0x0F, 0x13, 0x14, 0x15)
  - OBD Service 0x03
  - OBD Service 0x07
  - OBD Service 0x0A
- The reason is that the DEM module cannot guarantee that the number of matching DTCs returned by a function like `Dem_GetNumberOfFilteredDTC()` will exactly correspond to the number of DTCs that are consequently found by the DCM module when looping through the DTCs with `Dem_GetNextFilteredDTC()`.
- The AUTOSAR concepts do not allow diagnostic services which require direct memory access. This implies that the DCM module does not support BSW04033 because the following diagnostic services are not supported:
  - UDS Service ReadMemoryByAddress (0x23),
  - UDS Service WriteMemoryByAddress (0x3D),
  - UDS Service RequestDownload (0x34),
  - UDS Service RequestUpload (0x35),
  - UDS Service TransferData (0x36),
  - UDS Service RequestTransferExit (0x37), and
  - DynamicallyDefineDataIdentifier for subservice 02 (defined by memory address).

- UDS Service AccessTimingParameter (0x83) is not supported by the ISO standards in CAN and LIN. Also it is not planned to support this service with FlexRay. Therefore no support for this service is planned.
- Subfunction onDTCStatusChange of Service ResponseOnEvent is not supported in the current release.
- UDS Service SecuredDataTransmission (0x84) is not supported in the current release.
- UDS Service LinkControl (0x87) is not supported in the current release.
- The DCM SWS does not cover any SAE J1939 related diagnostic requirements.

4.2 Applicability to car domains

The DCM module can be used for all car domains.

4.3 Applicability to emission-related environments (OBD)

This DCM SWS is intended to fulfill the emission related requirements given by legislator. However, the supplier of the emission related system is responsible to fulfill the OBD requirements.
Certain requirements cannot be fulfilled by the DCM module by itself, but need to be considered at the level of the entire ECU or system. Example: During the integration of the DCM module within the system, the timing requirements (50ms response time) must be fulfilled.

5 Dependencies to other modules

The AUTOSAR Diagnostic Communication Manager (DCM) has interfaces and dependencies to the following Basic Software modules and SW-Cs:

Figure 3 Interaction of the DCM with other modules

● Diagnostic Event Manager (DEM): The DEM module provides function to retrieve all information related to fault memory such that the DCM module is able to respond to tester requests by reading data from the fault memory.

● Protocol Data Unit Router (PduR module): The PduR module provides functions to transmit and receive diagnostic data. Proper operation of the DCM module presumes that the PduR interface supports all service primitives defined for the Service Access Point (SAP) between diagnostic application layer and underlying transport layer (see ISO14229-1 [9] chapter 5. Application layer services).

● Communication Manager (ComM): The ComM module provides functions such that the DCM module can indicate the states "active" and "inactive" for diagnostic communication. The DCM module provides functionality to handle

the communication requirements "Full-/ Silent-/ No-Communication".
Additionally, the DCM module provides the functionality to enable and disable
Diagnostic Communication if requested by the ComM module.

● SW-C and RTE: The DCM module has the capability to analyze the received diagnostic request data stream and handles all functionalities related to diagnostic communication such as protocol handling and timing. Based on the analysis of the request data stream the DCM module assembles the response data stream and delegates routines or IO-Control executions to SW-Cs .If any of the data elements or functional states cannot be provided by the DCM module itself the DCM requests data or functional states from SW-Cs via port-interfaces or from other BSW modules through direct function-calls.

● BSW Scheduler Module (SchM): The SchM module provides functions to activate/cancel main processing functions.

● EcuM: The EcuM is able to initialize the DCM module.

● BswM: The Basic Software Mode Manager allows the DCM to perform a controller reset.

5.1 File structure

5.1.1 Code file structure

Dcm054: The code file structure shall not be defined within the DCM SWS completely. At this point it shall be pointed out that the code-file structure shall include the following files named:
Dcm_Lcfg.c – for link time configurable parameters and
Dcm_PBcfg.c – for post build time configurable parameters.
These files shall contain all link time and post-build time configurable parameters.
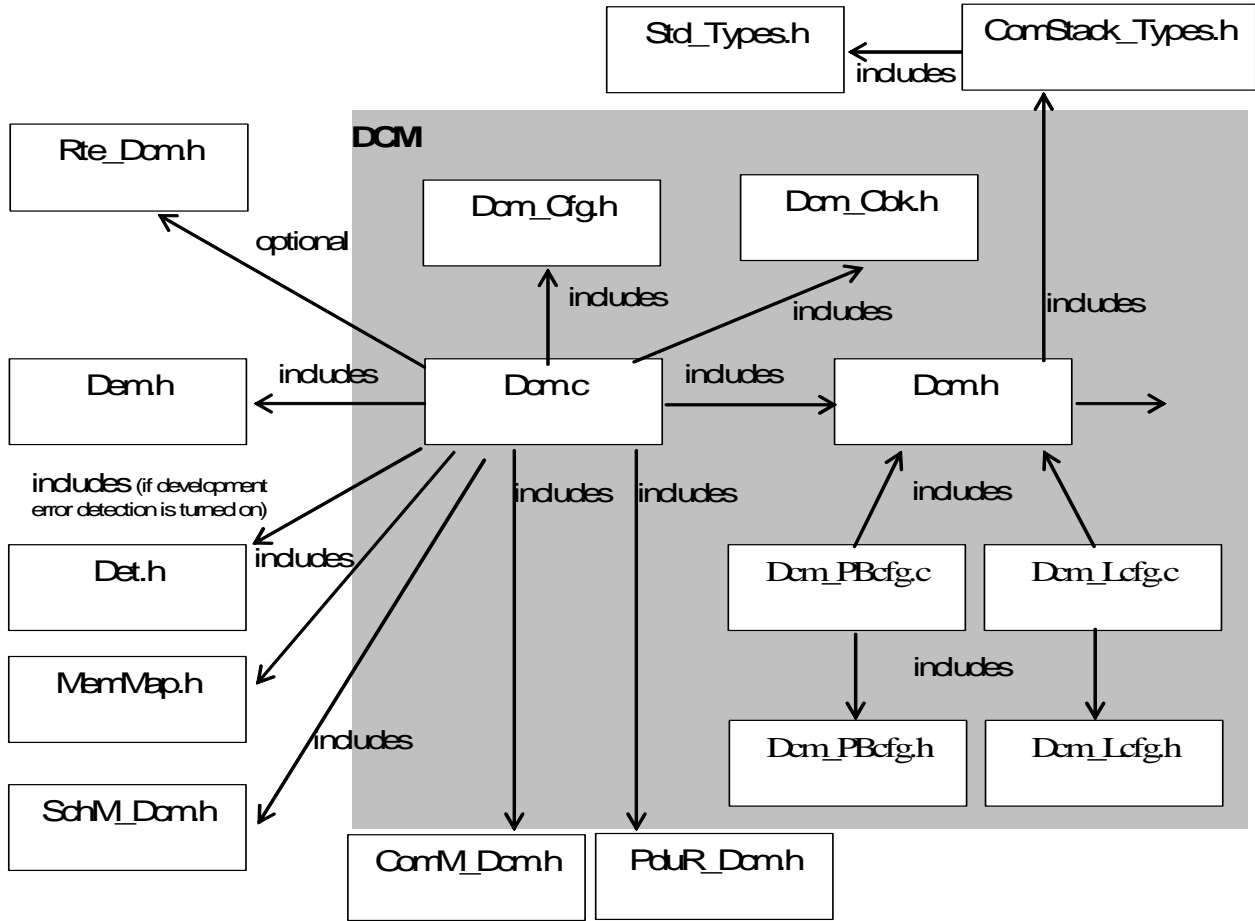
Figure 4: DCM module file structure

5.1.2 Header file structure

Dcm055: The DCM module shall use the header file structure shown in Figure 5
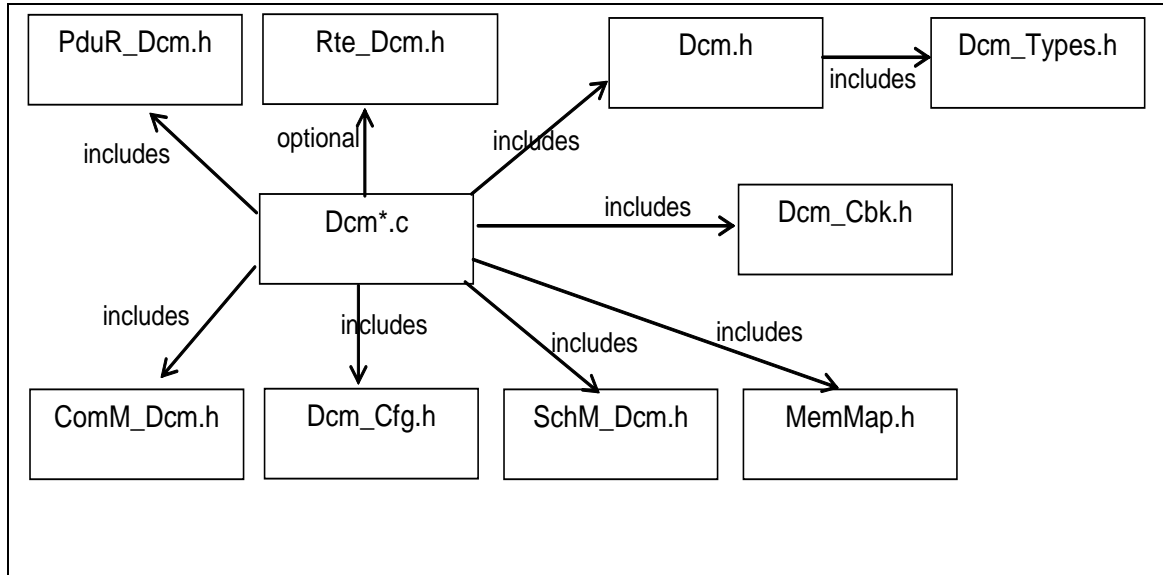
Figure 5: DCM module header file structure

Dcm110: The file Dcm.h shall include the public interface of the DCM module. This file contains all types, functions and parameters that are visible to any SW-C.

Dcm107: The implementation may be contained in one or multiple Dcm*.c files.

Dcm687: The file Dcm_Types.h include all DCM type definition

Dcm108: Any callbacks supplied by a SW-C shall be located in Rte_Dcm.h.

As shown in Figure 5 the DCM module header files are linked as followed:

Dcm109: Dcm.c shall include SchM_Dcm.h.

Dcm332: The DCM module shall include the file Dem.h.

By this way reporting production errors as well as the required Event Id symbols are included. The DCM SWS defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

Dcm367: The DCM module's implementer shall avoid the integration of incompatible files. Minimum implementation is the version check of the header file.
For included header files:
- DCM_AR_MAJOR_VERSION
- DCM_AR_MINOR_VERSION

shall be identical.
For the module internal c and h files:
- DCM_SW_MAJOR_VERSION
- DCM_SW_MINOR_VERSION
- DCM_AR_MAJOR_VERSION
- DCM_AR_MINOR_VERSION
- DCM_AR_PATCH_VERSION

shall be identical.

6 Requirements traceability

6.1 Document: General requirements on Basic Software modules:

Functional General Requirements	
Requirement	Satisfied by
Configuration	--
[BSW00344] Reference to link-time configuration	Fulfilled by chapter 10
[BSW00404] Reference to post build time configuration	Fulfilled by chapter 10
[BSW00405] Reference to multiple configuration sets	Fulfilled by chapter 10
[BSW00345] Pre-compile-time configuration	Fulfilled by chapter 10
[BSW159] Tool-based configuration	Not applicable
[BSW167] Static configuration checking	Requirement on configuration tool
[BSW171] Configurability of optional functionality	Fulfilled by chapter 10
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable
[BSW00380] Separate C-Files for configuration parameters	Dcm054
[BSW00419] Separate C-Files for pre-compile time configuration parameters	Dcm054
[BSW00381] Separate configuration header file for pre-compile time parameters	Dcm055
[BSW00412] Separate H-File for configuration parameters	Dcm055
[BSW00382] Not-used configuration elements need to be listed	Not applicable
[BSW00383] List dependencies of configuration files	Not applicable
[BSW00384] List dependencies to other modules	Fulfilled by chapter 5
[BSW00387] Specify the configuration class of callback function	Not applicable
[BSW00388] Introduce containers	Fulfilled by chapter 10
[BSW00389] Containers shall have names	Fulfilled by chapter 10
[BSW00390] Parameter content shall be unique within the module	Fulfilled by chapter 10
[BSW00391] Parameter shall have unique names	Fulfilled by chapter 10
[BSW00392] Parameters shall have a type	Fulfilled by chapter 10
[BSW00393] Parameters shall have a range	Fulfilled by chapter 10.2 to 10.5
[BSW00394] Specify the scope of the parameters	N/A
[BSW00395] List the required parameters (per parameter)	Fulfilled by chapter 10
[BSW00396] Configuration classes	Dcm171, Dcm172, Dcm173
[BSW00397] Pre-compile-time parameters	Fulfilled by chapter 10
[BSW00398] Link-time parameters	Fulfilled by chapter 10
[BSW00399] Loadable Post-build time parameters	Fulfilled by chapter 10
[BSW00400] Selectable Post-build time parameters	Fulfilled by chapter 10
[BSW00402] Published information	Fulfilled by chapter 10
Wake-Up	--
[BSW00375] Notification of wake-up reason	Not applicable
Initialization	--
[BSW101] Initialization interface	Dcm033, Dcm034; Dcm035, Dcm036, Dcm037; Dcm066

[BSW00416] Sequence of Initialization	Not applicable
[BSW00406] Check module initialization	Not applicable
Normal Operation	--
[BSW168] Diagnostic Interface of SW components	Not applicable
[BSW00407] Function to read out published parameters	Dcm065
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable
[BSW00424] BSW main processing function task allocation	Dcm053
[BSW00425] Trigger conditions for schedulable objects	Not applicable
[BSW00426] Exclusive areas in BSW modules	Not applicable
[BSW00427] ISR description for BSW modules	Not applicable
[BSW00428] Execution order dependencies of main processing functions	Not applicable
[BSW00429] Restricted BSW OS functionality access	Not applicable
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path [approved]	Not applicable
[BSW00433] Calling of main processing functions	Not applicable
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable
Shutdown Operation	--
[BSW00336] Shutdown interface	Not applicable
Fault Operation and Error Detection	--
[BSW00337] Classification of errors	Dcm012, Dcm041
[BSW00338] Detection and Reporting of development errors	Dcm040, Dcm042
[BSW00369] Do not return development error codes via API	Dcm044
[BSW00339] Reporting of production relevant error status	Not applicable
[BSW00421]Reporting of production relevant error events	Not applicable
[BSW00422] Debouncing of production relevant error status	Not applicable
[BSW00420] Production relevant error event rate detection	Not applicable
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable
[BSW00323] API parameter checking	Dcm043
[BSW004] Version check	Not applicable
[BSW00435] Header File Structure for the Basic Software Scheduler	Dcm055
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	Dcm055
[BSW00409] Header files for production code error IDs	Not applicable
[BSW00385] List possible error notifications	Not applicable
[BSW00386] Configuration for detecting an error	Not applicable
Non-functional Requirements	--
Software Architecture Requirements	--
[BSW161] Microcontroller abstraction	Not applicable
[BSW162] ECU layout abstraction	Not applicable
[BSW00324] Do not use HIS I/O Library	Not applicable
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable
[BSW00415] User dependent include files	Not applicable
Software Integration Requirements	--
[BSW164] Implementation of interrupt service routines	Not applicable
[BSW00325] Runtime of interrupt service routines	Implementation specific
[BSW00326] Transition from ISRs to OS tasks	Not applicable
[BSW00342] Usage of source code and object code	Not applicable

Document ID 018: AUTOSAR_SWS_DCM

[BSW00343] Specification and configuration of time	Fulfilled by chapter 10.2 to 10.5
[BSW160] Human-readable configuration data	Fulfilled by chapter DCM in the ECUC SWS
Software Module Design Requirements	--
Software quality	--
[BSW007] HIS MISRA C	Implementation specific
Naming conventions	--
[BSW00373] Main processing function naming convention [approved]	Dcm053
[BSW00350] Development error detection keyword [approved]	Not applicable

6.2 Document: AUTOSAR requirements on Basic Software, cluster Diagnostic

Requirements on Basic Software Module Diagnostic	
Requirement	Satisfied by
General	--
[BSW04010] Interface between Diagnostic service handling and Diagnostic event (error) management	Fulfilled by Chapter 8
[BSW04082] Support of ISO15031-5 and SAE J1979	Dcm243, Dcm244, Dcm245, Dcm410, Dcm411, Dcm246, Dcm414, Dcm417, Dcm421
Diagnostic communication management (DCM)	--
[BSW04007] Provide Diagnostic service handling	Fulfilled by Section 7.4
[BSW04021] Switch diagnostic communication access	Dcm015, Dcm003
[BSW04032] Support of different diagnostic addresses	Dcm682, Dcm683
[BSW04080] Support multi-channel capability for diagnostic communication	See "Limitations"
[BSW04065] Clearing of events or event groups	Dcm005
[BSW04067] Counting and evaluation of events according to ISO 14229-1 DTCStatusMask	Dcm293, Dcm378
[BSW04000] Support Diagnostic Standard UDS (ISO14229-1)	Fulfilled by the whole document
[BSW04001] Support Diagnostic Standard OBD (ISO15031-5)	Dcm243, Dcm244, Dcm245, Dcm410, Dcm411, Dcm246, Dcm414, Dcm417, Dcm421
[BSW04005] SecurityAccess level handling is managed by DCM	Dcm252
[BSW04006] Session handling is managed by DCM	Dcm022, Dcm250
[BSW04016] Provision of Busy Handling [Approved]	Dcm024
[BSW04019] Application callback after transmit confirmation	This is handled inside the DCM
[BSW04020] Suppression of Responses	Dcm001, Dcm200
[BSW04033] Upload/Download services for data handling	See "Limitations"
[BSW04036] Format checking of diagnostic services	Dcm272, Dcm273,

	Dcm071, Dcm074
Timing Requirements	--
[BSW04015] Provision of timing handling according to ISO15765-3	Dcm027, Dcm143, Dcm144, Dcm311, Dcm031
Resource Usage	--
[BSW04017] Provide optimized buffer handling	Dcm028, Dcm038, Dcm068
Interface and API	--
[BSW04078] Interface to fault memory, fault status	Chqpter 8
[BSW04011] Provide diagnostic state information	Dcm338, Dcm339, Dcm340
[BSW04003] Interface to PDU Router shall be network independent	Dcm030
[BSW04079] The size of a FreezeFrame shall be reported to the DCM by the DEM	Dcm441
Configuration	--
[BSW04059] Configuration of timing parameter	Dcm031
[BSW04024] Configurable size of transferred data	Dcm032

7 Functional specification

7.1 General design elements

7.1.1 Submodules within the DCM module

To define the functionality of the DCM module, The DCM SWS models the DCM module as consisting of the following submodules:

- Diagnostic Session Layer (DSL) submodule: The DSL submodule ensures data flow concerning diagnostic requests and responses, supervises and guarantees diagnostic protocol timing and manages diagnostic states (especially diagnostic session and security).
- Diagnostic Service Dispatcher (DSD) submodule: The DSD submodule processes a stream of diagnostic data. The submodule:
  - Receives a new diagnostic request over a network and forwards it to a data processor.
  - Transmits a diagnostic response over a network when triggered by the data processor (e.g. by the DSP submodule).
- Diagnostic Service Processing (DSP) submodule: The DSP submodule handles the actual diagnostic service (respectively subservice) requests.

The next graphic gives an overview of the interfaces between the submodules DSP, DSD, and DSL within the DCM module.
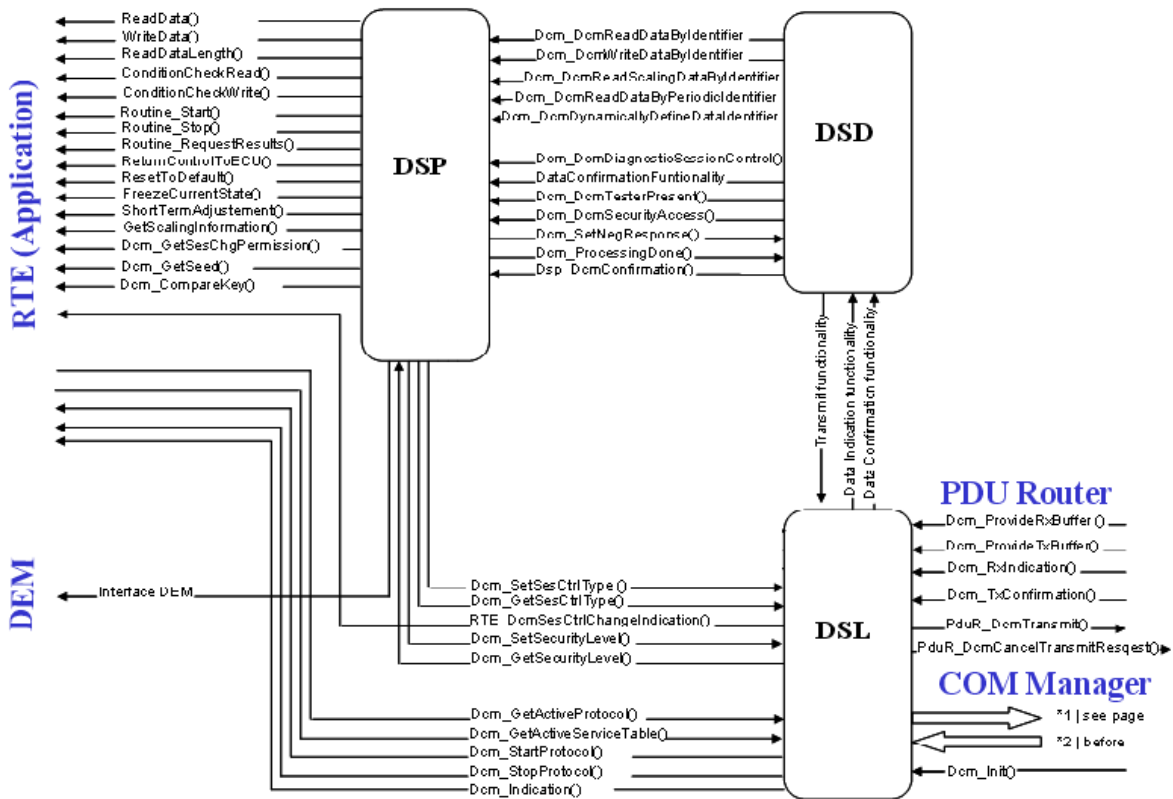
Figure 6 Possible interaction between the submodules in the DCM

Note: The implementation of these submodules and the interfaces between them is not mandatory. They are introduced only to improve the readability of the specification.

7.1.2 Negative Response Code (NRC)

The standards defining the UDS Services and OBD Services define the possible negative response codes (NRCs).
In addition, the DCM SWS uses a subset of these NRCs in the interfaces between the DCM and other BSW modules and the SW-Cs. This subset is defined in the data type `Dcm_NegativeResponseCodeType`.

7.2 Diagnostic Session Layer (DSL)

7.2.1 Introduction

Dcm030: All functional areas of the DSL submodule shall be in conformance with the specifications ISO14229-1 [9] and the network-independent part of ISO15765-3 [12].

There is no network-dependent functional area in the DSL submodule. Within the configuration, some parameters can be set dependent on the network.

7.2.2 Use cases

The DSL submodule provides the following functionalities:
- Session handling (as required by ISO14229-1 [9] and ISO 15765-3 [12]),
- Application layer timing handling (as required by ISO14229-1 [9] and ISO 15765-3 [12]),
- Specific response behavior (as required by ISO14229-1 [9] and ISO 15765-3 [12]).

7.2.3 Interaction with other modules

The DSL has the following interaction with other modules:
- PduR module
  - PduR module provides data of incoming diagnostic requests.
  - The DSL submodule triggers output of diagnostic responses.
- DSD submodule
  - The DSL submodule informs the DSD submodule about incoming requests and provides the data.
  - The DSD submodule triggers output of diagnostic responses.
- SW-Cs / DSP submodule. The DSL submodule provides access to security and session state.
- ComM module
  - The DSL submodule guarantees the communication behavior required by the ComM module

7.2.4 Functional description

7.2.4.1 Overview
The DSL submodule provides the following functionality:
Request Handling
- Forward requests from the PduR module to the DSD submodule.
- Concurrent "TesterPresent" ("keep alive logic").
Response Handling
- Forward responses from the DSD submodule to the PduR module.
- Guarantee response timing to tester.
- Support of periodic transmission.
- Support of ResponseOnEvent (ROE) transmission.
- Support of segmented response.
- Support of ResponsePending response triggered by the application.
Security Level Handling
- Manage security level.
Session State Handling
- Manage session state.
- Keep track of active non-default sessions.
- Informs depending modules on session change.

- Allows modifying timings.

Diagnostic Protocol Handling
- Handling of different diagnostic protocols.
- Manage resources.

Communication Mode Handling
- Handling of communication requirements (Full- / Silent- / No Communication).
- Indicating of active / inactive diagnostic.
- Enabling / disabling all kinds of diagnostic transmissions.

7.2.4.2 Forward requests from the PduR module to the DSD submodule

The PduR module indicates the DCM module whenever a reception of new diagnostic request content is started on a DcmRxPduId, which is assigned to the DCM module. This is done by calling `Dcm_ProvideRxBuffer()`, which requests the DCM module to provide the receive buffer.

In first call of Dcm_ProvideRxBuffer(), the TP's will ask for a buffer for First Frame or Single Frame.

If the configuration parameter ***DcmDslProtocolBlockSize*** is configured:
- The Dcm will return the configured size on the first call of `Dcm_ProvideRxBuffer()`
- The Dcm will provide the remaining size of entire buffer on the following calls.

If the configuration parameter ***DcmDslProtocolBlockSize*** is not configured:
- The Dcm will return the entire Rx Buffer size configured in DcmDslBufferSize on the first call of `Dcm_ProvideRxBuffer()`.
- `Dcm_ProvideRxBuffer()` will never return a buffer smaller than the total size of the diagnostic request that is to be received.

If the reception of a diagnostic request is finished (successful or with errors), the PduR module will call `Dcm_RxIndication()` to give a receive indication to the DCM module.

Dcm111: The DSL submodule shall forward received data to the DSD submodule only after a call of `Dcm_RxIndication()` with parameter Result = NTFRSLT_OK (see Dcm093).

Dcm241: As soon as a request message is received (after a call of `Dcm_RxIndication()` with parameter Result = NTFRSLT_OK (see Dcm093) and until a call to `Dcm_TxConfirmation()` (see Dcm351) for this DcmPduId), the DSL submodule shall block the corresponding DcmPduId. During the processing of this request, no other request of the same protocol type (e.g. an enhanced session can be ended by a OBD session) can be received, until the corresponding response message is sent and the DcmPduId is released again.

More descriptions of the APIs (prototype, input/output parameter) can be found in the interface description of PduR module (see [2]).

It is allowed to have different DcmPduIds for different diagnostic communication applications. For example:
- OBD DcmRxPduId: for reception of OBD requests,
- OBD DcmTxPduId: for transmission of OBD responses,
- UDS phys DcmRxPduId: for reception of UDS physically addressed requests,

- AUTOSAR confidential -

- UDS func DcmRxPduId: for reception of UDS functionally addressed requests,
- UDS DcmTxPduId: for transmission of UDS responses.

Address type (physical/functional addressing) is configured per DcmRxPduId (see configuration parameter **DcmDslProtocolRx**). A configuration per DcmRxPduId is possible because there will always be different DcmRxPduId values for functional and physical receptions, independent of the addressing format of the Transport Layer (extended addressing, normal addressing).

7.2.4.3 Concurrent "TesterPresent" ("keep alive logic")

It is possible, that functional "TesterPresent" commands are sent by the tester in parallel to physical requests/responses. This is called "keep alive logic" in ISO14229-1 [9]. This functional "TesterPresent" will be received on a separate DcmRxPduId (UDS func DcmRxPduId) with a Dcm-internal receive buffer which is not configured explicitly. Due to that reason, the functional TesterPresent (and only functional TesterPresent without response) is handled in the following way:

Dcm112: When the PduR module calls `Dcm_RxIndication()` with parameter Result=NTFRSLT_OK (see Dcm093) and if the request is a "TesterPresent" command with "suppressPosRspMsgIndicationBit" set to TRUE (SID equal to 0x3E, subfunction equal to 0x80), the DSL submodule shall reset the session timeout timer (S3Server).

Dcm113: When the PduR module calls `Dcm_RxIndication()` with parameter Result = NTFRSLT_OK (see Dcm093) and if the request is a "TesterPresent" command with "suppressPosRspMsgIndicationBit" set to TRUE (SID equal to 0x3E, subfunction equal to 0x80), the DSL submodule shall not forward this request to the DSD submodule for further interpretation.

Rationale for Dcm113: Because of bypassing the functional "TesterPresent" in the DSL submodule, the DCM module is able to receive and process next physical requests without any delay.

7.2.4.4 Forward responses from the DSD submodule to the PduR module

Dcm114: The DSD submodule shall request the DSL submodule for transmission of responses.

Dcm115: After the minimum response time between reception of diagnostic request and transmission of diagnostic response is over (P2ServerMin, Definition of this parameter can be found in the document [12]), the DSL submodule shall trigger the transmission of the diagnostic response to the PduR module by calling `PduR_DcmTransmit()`.

Responses are sent with the DcmTxPduId, which is linked in the DCM module configuration to the DcmRxPduId, i.e. the ID the request was received with (see configuration parameter **DcmDslProtocolTx**)

Within `PduR_DcmTransmit()` only the length information is given to the PduR module. After the DCM module has called `PduR_DcmTransmit()`, the PduR module will call `Dcm_ProvideTxBuffer()` to request the DCM module to provide

the data to be transmitted and will call `Dcm_TxConfirmation()` after the complete PDU has successfully been transmitted or an error occurred.

Dcm117: If the DSL submodule receives a confirmation after the complete DCM PDU has successfully been transmitted or an error occurred by a call of `Dcm_TxConfirmation()`, then the DSL submodule shall forward this confirmation to the DSD submodule.

Dcm118: In case of a failed transmission (failed `PduR_DcmTransmit()` request) or error confirmation (`Dcm_TxConfirmation()` with error), the DSD submodule shall not repeat the diagnostic response transmission.

More descriptions of the APIs (prototype, input/output parameter) can be found in the interface description of the PduR module (see [2]).

7.2.4.5 Guarantee timing to tester by sending busy responses

Dcm024: If the Application (or the DSP submodule) is able to perform a requested diagnostic task, but needs additional time to finish the task and prepare the response, then the DSL submodule shall send a negative response with NRC 0x78 (Response pending) before reaching the response time (P2ServerMax respectively P2*ServerMax).

Rationale for Dcm024: The DSL submodule guarantees the response timing to tester.

Dcm119: The DSL submodule shall send negative responses as required in Dcm024 from a separate buffer.

Rationale for Dcm119: This is needed in order to avoid overwriting the ongoing processing of requests, e.g. the application already prepared response contents in the diagnostic buffer.
The number of negative responses with NRC 0x78 (Response pending) for one diagnostic request is limited by the configuration parameter **DcmDslDiagRespMaxNumRespPend**. This avoids deadlocks in the Application.

Dcm120: If the number of negative responses for a requested diagnostic tasks (see Dcm024) reaches the value defined in the configuration parameter **DcmDslDiagRespMaxNumRespPend**, the DCM module shall stop processing the active diagnostic request and shall send a negative response with NRC 0x10 (General reject).

7.2.4.6 Support of periodic transmission

The UDS service ReadDataByPeriodicIdentifier (0x2A) allows the tester to request the periodic transmission of data record values from the ECU identified by one or more periodicDataIdentifiers.
TYPE1 = messages on the DcmTxPduId already used for normal diagnostic responses.

Dcm121: If a pending message for normal diagnostic responses (higher priority) exists, then the DSL submodule shall wait for the transmission confirmation (Call to `Dcm_TxConfirmation()`) for this normal diagnostic response before sending the periodic transmission message.

TYPE2 = messages on a separate DcmTxPduId.
This type of information can be configured in ***DcmDslProtocolTransType***.
In case of TYPE2, the separate DcmPduId can be configured in ***DcmDslPeriodicTxPduRef***

Dcm122: The DCM module shall send responses for periodic transmissions using a separate protocol and a separate buffer of configurable size.

The ***DcmDslPeriodicTranmissionConRef*** configuration parameter allows linking the protocol used to receive the periodic transmission request to the protocol used for the response. Note that multiple DcmTxPduIds can be assigned to the periodic transmission protocol.
The DCM module respects several restrictions according to the communication mode:

Dcm123: Periodic transmission communication shall only take place in Full Communication Mode.

Periodic transmission events can occur when not in Full Communication Mode. So the following requirement exists:

Dcm125: The DCM module shall discard periodic transmission events beside Full Communication Mode and shall not queue it for transmission.

Dcm126: Periodic transmission events shall not activate the Full Communication Mode.

7.2.4.7 Support of ROE transmission
With the UDS Service ResponseOnEvent (0x86), a tester requests an ECU to start or stop transmission of responses initiated by a specified event. Upon registering an event for transmission, the tester also specifies the corresponding service to respond to (UDS Service ReadDataByIdentifier (0x22)).
7.2.4.7.1 TYPE1 = Responses are sent on the same DcmTxPduId that was used for the ROE service response.

Dcm127: If the UDS service ResponseOnEvent (0x86) is received, then the DSP submodule shall store the DcmRxPduId, which is provided in the pMsgContext parameter of the ROE service function.

Dcm128: The DSP submodule shall forward this stored DcmRxPduId as parameter in the `DslInternal_ResponseOnOneEvent()` function, where it is used to simulate a request.

Dcm129: If a pending message for normal diagnostic responses (higher priority) exists, then the DSL submodule shall wait for the transmission confirmation (Call to `Dcm_TxConfirmation()`) for this normal diagnostic response before sending the event-triggered responses.

7.2.4.7.2 TYPE2 = Responses are sent on a separate DcmTxPduId.
This type of information can be configured in ***DcmDslProtocolTransType***.
In case of TYPE2, the separate channel can be configured in ***DcmDslROETxPduRef***
Responses generated by UDS Service ResponseOnEvent (0x86) use a separate protocol and so a separate buffer of configurable size.

Dcm131: The configured protocol buffer shall be used for transmission of the ROE messages (as the reception shall use a separate protocol, a separate buffer needs to be used for reception).

Dcm132: The content of the pMsgContext pointer (ROE message) shall be copied into the buffer.

Configuration parameter ***DcmDslROEConnectionRef*** allows linking the protocol used to receive the ROE request to the protocol used for the response.

Dcm133: ROE communication shall only be performed in Full Communication Mode.

Dcm134: ROE events shall be disabled in any other Communication Mode except for the Full Communication Mode.

Dcm135: ROE events beside Full Communication Mode shall be discarded and not queued for later transmission.

Dcm136: ROE events requested by the Application shall not activate the Full Communication Mode.

7.2.4.7.3 Please note the following limitation:

Dcm137: While processing an ROE event, any additional requests (external and internal) shall be rejected with the same or lower priority of its Protocol Table.

This restriction is independent of using other diagnostic request/response CAN identifiers.
7.2.4.8 Support of segmented response (paged-buffer)

Dcm028: If enabled (***DcmPagedBufferEnabled***=TRUE), the DCM module shall provide a mechanism to send responses larger than the configured and allocated diagnostic buffer.

With paged-buffer handling the ECU is not forced to provide a buffer, which is as large as the maximum length of response.
Please note:

- paged-buffer handling is for transmit only – no support for reception.
- paged-buffer handling is not available for the Application (DCM-internal use only).

7.2.4.9 Support of ResponsePending response triggered by the Application

In some cases, e.g. in case of flash programming, the Application needs to request an immediate NRC 0x78 (Response pending), which shall be sent immediately and not just before reaching the response time (P2ServerMax respectively P2*ServerMax).

For example: To start the flash programming sequence, a diagnostic tester will use UDS Service DiagnosticSessionControl (0x10) with DiagnosticSessionType set to "ProgrammingSession". It is now good practice that the ECU first sends a negative response with NRC 0x78 (Response pending), before jumping to boot loader.

If enabled in the configuration (***DcmDslDiagRespForceRespPendEn***=TRUE), when the DCM module calls the operation `Xxx_GetSesChgPermission()` and gets an error status E_FORCE_RCRRP, the DSL submodule will trigger the transmission of a negative response with NRC 0x78 (Response pending).

This response needs to be sent from a separate buffer, in order to avoid overwriting the ongoing processing of the request.

7.2.4.10   Manage security level

Dcm020: The DSL submodule shall save the level of the current active security level.

For accessing this level, the DSL submodule provides interfaces to:
- get the current active security level: `Dcm_GetSecurityLevel()`
- set a new security level: `DslInternal_SetSecurityLevel()`

Dcm033: During DCM initialization the security level is set to the value 0x00 (DCM_SEC_LEV_LOCKED).

By [Dcm033](), the ECU is locked.

Dcm139: The DSL shall reset the security level to the value 0x00 (i.e. the security is enabled) under one of the following conditions:
- if a transition from any diagnostic session other than the defaultSession to another session other than the defaultSession (including the currently active diagnostic session) is performed or
- if a transition from any diagnostic session other than the defaultSession to the defaultSession (`DslInternal_SetSecurityLevel()`) (initiated by UDS Service DiagnosticSessionControl (0x10) or S3Server timeout) is performed.

Only one security level can be active at a time.

7.2.4.11   Manage session state

Dcm022: The DSL submodule shall save the state of the current active session.

For accessing this variable, the DSL submodule provides interfaces to:
- get the current active session: `Dcm_GetSesCtrlType()`
- set a new session: `DslInternal_SetSesCtrlType()`

Dcm034: During DCM initialization, the session state is set to the value 0x01 ("DefaultSession").

7.2.4.12 Keep track of active non-default sessions

Dcm140: Whenever a non-default session is active and when the session timeout (S3Server) is reached without receiving any diagnostic request, the DSL submodule shall reset to the default session state ("DefaultSession", 0x01).

According to following table, the start / stop of S3Server timeout timer is processed:

Dcm141:

Sub-sequent start	Completion of any final response message or an error indication (`Dcm_TxConfirmation()`: confirmation of complete PDU or indication of an error)
	Completion of the requested action in case no response message (positive and negative) is required / allowed.
	Indicates an error during the reception of a multi-frame request message. (`Dcm_RxIndication()`: indication of an error)
Sub-sequent stop	Start of a multi-frame request message (`Dcm_ProvideRxBuffer()`: indicates start of PDU reception)
	Reception of single-frame request message. (`Dcm_ProvideRxBuffer()`: indicates start of PDU reception)

"Start of S3Server" means reset the timer and start counting from the beginning.

7.2.4.13 Informs depending modules on session change

Dcm142: Whenever the value of the active session changes (including "self-transitions") (initiated by UDS Service DiagnosticSessionControl (0x10) or S3Server timeout), the DSL submodule shall inform the Application by calling the operation `Xxx_ChangeIndication()` on all configured SessionControl ports (see configuration parameter **DcmDslSessionControl**).

7.2.4.14 Allow to modify timings

Dcm027: The DCM module shall handle the following protocol timing parameters in compliance with [12]: P2ServerMin, P2ServerMax, P2*ServerMin, P2*ServerMax, S3Server

Dcm143: P2min / P2*min and S3Server shall be set to defined values: P2min = 0ms, P2*min = 0ms, S3Server = 5s.

All protocols (OBD, enhanced diagnosis) share the same protocol timing values (given by default session configuration). These timing values are set when the protocol is started.
These protocol timing parameters have influence on the session layer timing (no influence on Transport Layer timing). Some of these timing parameters can be modified while protocol is active with the following means:
- UDS Service DiagnosticSessionControl (0x10)
- UDS Service AccessTimingParameter (0x83)

The DSL submodule provides the following functionalities to modify the timing parameters:

- Provide the active timing parameters,
- Prepare setting of new timing parameters: examines, if new values are within the configured limits (see configuration parameter *DcmDslProtocolTimeLimit*)
- Set the new timing parameters. Activation of new timing values is only allowed after sending the response.

7.2.4.15 Handling of different diagnostic protocols

It is necessary to distinguish between different diagnostic protocols (e.g. OBD, enhanced diagnosis ...).

Dcm018: Different protocols shall be used for UDS Services versus OBD Services

Dcm018 is required because of
- different protocol settings
- different valid service tables
- prioritization of protocols
- preemption of protocols

7.2.4.15.1 Different service tables

For the different protocols a different set of allowed diagnostic services is valid (e.g. the UDS commands for the enhanced diagnosis, the OBD mode services for the OBD protocol). It is possible to create different service tables and link them to the diagnostic protocol.

Dcm035: With every protocol initialization, the DSL submodule sets a link to the corresponding service table (see configuration parameter *DcmDslProtocolSIDTable*).

The DSD submodule uses this link for further processing of diagnostic requests.

7.2.4.15.2 Prioritization of protocol

The configuration parameter *DcmDslProtocolPriority* makes it possible to give each protocol its own relative priority.

Possible use case: There are ECUs, communicating with a vehicle-internal diagnostic tester (running on enhanced diagnosis) and a vehicle-external OBD tester. The OBD communication must have a higher priority than the enhanced diagnosis.

Dcm015: A protocol with higher priority is allowed to preempt the already running protocol.

Differentiation of diagnostic protocols is possible, because of different DcmRxPduId values (configured per protocol, see configuration parameter *DcmDslProtocolRxPduRef*) referenced in the protocol configuration.

7.2.4.15.3 Preemption of protocol

Dcm459: If a running diagnostic request is preempted by a higher prior request (of another protocol), the DSL submodule shall call all configured `Xxx_StopProtocol()` functions (see configuration parameter *DcmDslCallbackDCMRequestService*).

Dcm079: In order to cancel pending transmission in lower-layer, related to the lower prior request, the DCM module shall call `PduR_DcmCancelTransmit()` with the following parameters:
DcmTxPduId: the id of the Pdu to be canceled

Dcm460: When `PduR_DcmCancelTransmit()` returns E_NOT_OK, the DCM module shall assume that the ongoing transmission cannot be cancelled and shall not retry to cancel the transmit request.

Dcm575: In order to cancel pending reception in lower-layer, related to the lower prior request, the DCM module shall call `PduR_DcmCancelReceive()` with the following parameters:
DcmRxPduId: the id of the Pdu to be canceled

Dcm576: When `PduR_DcmCancelReceive()` returns E_NOT_OK, the DCM module shall assume that the ongoing reception cannot be cancelled and shall not retry to cancel the receive request. The current protocol shall be stopped and the new one started.

7.2.4.15.4 Detection of protocol start

Dcm036: With first request of a diagnostic protocol, the DSL submodule shall call all configured `Xxx_StartProtocol()` functions (see configuration parameter ***DcmDslCallbackDCMRequestService***).

Inside this function, the Application can examine the environment conditions and enable/disable further processing of the protocol.

Dcm144: After all `Xxx_StartProtocol()` functions have returned E_OK (meaning all components have allowed the start of the protocol), the default timing parameters are loaded from the default session configuration (see configuration parameter ***DcmDspSessionRow***).

Dcm145: After all `Xxx_StartProtocol()` functions have returned E_OK (meaning all components have allowed the start of the protocol), the service table is set (see configuration parameter ***DcmDslProtocolSIDTable***).

Dcm146: After all `Xxx_StartProtocol()` functions have returned E_OK (meaning all components have allowed the start of the protocol), the security state is reset.

Dcm147: After all `Xxx_StartProtocol()` functions have returned E_OK (meaning all components have allowed the start of the protocol), the session state is reset.

After all `Xxx_StartProtocol()` functions have returned E_OK (meaning all components have allowed the start of the protocol), the DCM module will initialize all relevant service interpreters.

7.2.4.15.5 Parallel execution of diagnostic response

Dcm081: The DCM module shall allow the parallel execution of protocol response in the case of ResponseOnEvent and PeriodicTransmission services.

The DCM module provides a configuration parameter *DcmDslProtocolIsParallelExecutab* to allow a protocol to be executed in parallel. Only protocols used for ResponseOnEvent and PeriodicTransmission services can have the configuration parameter *DcmDslProtocolIsParallelExecutab* set to TRUE.

7.2.4.16 Manage resources

Due to limited resources, the following points should be considered as hints for the design:

- It is allowed to use and allocate only one diagnostic buffer in the DCM module. This buffer is then used for processing the diagnostic requests and responses.
- Output of NRC 0x78 (Response pending) responses is done with a separate buffer.
- paged-buffer handling (see [Dcm137](#)).

7.2.4.17 ComM interaction

7.2.4.17.1 Communication state initialization

Dcm066: During DCM initialization, the communication state is set to FULL_COMMUNICATION.

7.2.4.17.2 No Communication

The ComM module will indicate the No Communication Mode to the DCM module by calling `Dcm_ComM_NoComModeEntered()`. In response, the DCM won't take into account this notification.

7.2.4.17.3 Silent Communication

The ComM module will indicate the Silent Communication Mode to the DCM module by calling `Dcm_ComM_SilentComModeEntered()`. In response, the DCM won't take into account this notification.

7.2.4.17.4 Full Communication

The ComM module will indicate the Full Communication Mode to the DCM module by calling `Dcm_ComM_FullComModeEntered()`. In response, the DCM won't take into account this notification.

7.2.4.17.5 Default Session:

Dcm163: With the first request of a diagnostic protocol, the DCM shall call `ComM_DCM_ActiveDiagnostic()` to inform the ComM module about the need to stay in Full Communication Mode.

Dcm164: With the reception of `Dcm_TxConfirmation()` connected to the response given by the DSL submodule, the DCM shall call `ComM_DCM_InactiveDiagnostic()` to inform the ComM module that Full Communication is not longer needed.

Dcm165: The DCM shall not call `ComM_DCM_InactiveDiagnostic()` for NRC 0x78 (Response pending). The DCM shall only call `ComM_DCM_InactiveDiagnostic()` with the very last response (positive or negative) connected to the request.

Dcm166: If a "suppressPosRspMsgIndicationBit" is indicated and the positive response will be suppressed, the DCM shall call `ComM_DCM_InactiveDiagnostic()`.

7.2.4.17.6 Session Transitions:

Dcm167: If the actual diagnostic session is changed into a session different than the default session (initiated by UDS Service DiagnosticSessionControl), the DCM shall call `ComM_DCM_ActiveDiagnostic()` to inform the ComM module about the need to stay in Full Communication Mode.

Dcm168: If the actual diagnostic session is changed from a session different than the default into the default session (initiated by UDS Service DiagnosticSessionControl or S3Server timeout), then the DCM shall call `ComM_DCM_InactiveDiagnostic()` to inform the ComM module that Full Communication is not longer needed.

7.2.4.17.7 Non Default Session:

Dcm169: As long as the server is in a session other than the default session, the DCM shall not call `ComM_DCM_ActiveDiagnostic()` when receiving a request from a client provided by the PduR module.

Dcm170: As long as the server is in a session other than the default session, the DCM shall not call `ComM_DCM_InactiveDiagnostic()` with the reception of `Dcm_TxConfirmation()` connected to the response given by the DSL submodule.

7.3 DSD (Diagnostic Service Dispatcher)

7.3.1 Introduction

The DSD submodule is responsible to check the validity of an incoming diagnostic request (Verification of Diagnostic Session/Security Access levels/Application permission) and keeps track of the progress of a service request execution.

Dcm178: The DSD submodule shall only process valid requests and shall reject invalid ones.

7.3.2 Use cases

The following use cases are relevant and are described in detail in the following:
- Receive a request message and transmit a positive response message
- Receive a request message and suppress a positive response
- Receive a request message and transmit a negative response message
- Send a positive response message without corresponding request
- Segmented Responses

7.3.2.1 Receive a request message and transmit a positive response message

This is the standard use case of normal communication („ping-pong"). The server receives a diagnostic request message. The DSD submodule ensures the validity of the request message. In this use case, the request is valid and the response will be positive. The request will be forwarded to the appropriate data processor in the DSP submodule.

When the data processor has finished all actions of data processing, it triggers the transmission of the response message by the DSD submodule.

If the data processor processes the service immediately as part of the request indication function, the data processor can trigger the transmission inside this indication function ("synchronous").

If the processing takes a longer time (e. g. waiting on EEPROM driver), the data processor defers some processing ("asynchronous"). The response pending mechanism is covered by the DSL submodule. The data processor triggers the transmission explicitly, but from within the data processor's context.

As soon as a request message is received, the corresponding DcmPduId is blocked by the DSL submodule (see Dcm241). During the processing of this request, no other request of the same protocol type (e.g. an enhanced session can be ended by a OBD session) can be received, until the corresponding response message is sent and the DcmPduId is released again.

7.3.2.2 Receive a request message and suppress a positive response

This is a sub-use-case of the previous one.

Within the UDS protocol it is possible to suppress the positive response by setting a special bit in the request message (see Dcm200). This special suppression handling is completely performed within the DSD submodule.

7.3.2.3 Receive a request message and transmit a negative response message

There are a many different reasons why a request message is rejected and a negative response is to be sent.

If a diagnostic request is not valid or if a request may not be executed in the current session, the DSD submodule will reject the processing and a negative response will be returned.

But there are even many reasons to reject the execution of a well-formed request message, e.g. if the ECU or system state does not allow the execution. In this case, the DSP submodule will trigger a negative response including an NRC supplying additional information why this request was rejected.

In case of a request composed of several parameters (e.g. a UDS Service ReadDataByIdentifier (0x22) request with more than one identifier to read), each parameter is treated separately. And each of these parameters can return an error.

This kind of request returns a positive response only if all the parameters were processed successfully. If at least one of these parameters reports an error, the DSD submodule shall transmit a negative response with the first NRC reported and no positive response shall be send for this request.

7.3.2.4 Send a positive response message without corresponding request

There are two services within the UDS protocol, where multiple responses are sent for only one request. In general, one service is used to enable (and disable) an event- or time-triggered transmission of another service, which again is sent by the ECU without a corresponding request (see ISO14229-1 [9]).
These services are:
- UDS Service ReadDataByPeriodicIdentifier (0x2A). This service allows the client to request the periodic transmission of data record values from the server identified by one or more periodicDataIdentifiers.
  Type 1 = USDT messages on the DcmTxPduId already used for normal diagnostic responses (single frames only);
  Type 2 = UUDT message on a separate DcmTxPduId.
  For Type 1, the outgoing messages must be synchronized with "normal outgoing messages", which have a higher priority.
- ResponseOnEvent (0x86). This service requests a server to start or stop transmission of responses on a specified event.
  Way 1 = USDT messages on the DcmTxPduId already used for normal diagnostic responses,
  Way 2 = USDT messages on separate DcmTxPduId.
  For Way 1, the outgoing messages must be synchronized with "normal outgoing messages", which have a higher priority.

This handling is especially controlled by the DSL submodule. However, the DSD submodule also provides the possibility to generate a response without a corresponding request.

7.3.2.5 Segmented Responses (paged-buffer)

Within the diagnostic protocol, some services allow to exchange a significant amount of data, e.g. UDS Service ReadDTCInformation (0x19) and UDS Service TransferData (0x36).
In the conventional approach, the ECU internal buffer must be large enough to keep the longest data message which is to be exchanged (worst-case) and the complete buffer is filled before the transmission is started.
RAM memory in an ECU often is a critical resource, especially in smaller micros. In a more memory-saving approach, the buffer is filled only partly, transmitted partly and then refilled partly – and so on. This paging mechanism requires only a significantly reduced amount of memory, but demands a well-defined reaction time for buffer refilling.
The user can decide whether to use the "linear buffer" or paged-buffer for diagnostics.

7.3.3 Interaction of the DSD with other modules

The DSD submodule is called by the DSL submodule when receiving a diagnostic message and performs the following operations:
- delegates processing of request to the DSP submodule
- keeps track of request processing (provide `DsdInternal_ProcessingDone()` API)
- transmits the response of the Application to the DSL submodule (Transmit functionality)

7.3.3.1 Interaction of the DSD with the DSL main functionality

Direction	Explanation
Bidirectional	Exchange of the Diagnostic Messages (receive/transmit).
DSD submodule to DSL submodule	Obtain latest diagnostic session and latest security level.
DSL submodule to DSD submodule	Confirmation of transmission of Diagnostic Message.

Table 2 Interaction between the DSD submodule and the DSL submodule

7.3.3.2 Interaction of the DSD with the DSP

Direction	Explanation
DSD submodule to DSP submodule	-Delegate processing of request. -Confirmation of transmission of Diagnostic Message.
DSP submodule to DSD submodule	-Signal that processing is finished.

Table 3 Interaction between the DSD submodule and the DSP submodule

7.3.4 Functional Description of the DSD

7.3.4.1 Support checking the diagnostic service identifier and adapting the diagnostic message

The DSD submodule shall be triggered by the DSL submodule if a new diagnostic message is recognized. The DSD submodule will start processing by analyzing the diagnostic service identifier contained in the received diagnostic message.

Dcm084: If configured (configuration parameter **DcmRespondAllRequest**=FALSE), if the DCM module receives a diagnostic request that contains a service ID that is in the range from 0x40 to 0x7F or in the range from 0xC0 to 0xFF, the DCM shall not respond to such a request.

This range corresponds to the diagnostic response identifier.

Dcm192: The DSD submodule shall analyze the (incoming) diagnostic message for the diagnostic service identifier (based on first byte of the diagnostic message) and shall check the supported services with the newly received diagnostic service identifier.

Dcm193: During this check, the DSD submodule shall search the newly received diagnostic service identifier in the "Service Identifier Table".

For performance reasons it might be necessary that the support check is done with a "lookup table" functionality. In this "Service Identifier Table" all supported Service IDs of the ECU are predefined.

Dcm195: The DSL submodule shall provide the current "Service Identifier Table"

Rationale for Dcm195: The "Service Identifier Table" and the information about the supported services will be generated out of the configuration. More than one Service Identifier Table can be configured for selection. At one time only one Service Identifier Table can be active.

Dcm196: For the check, the DSD submodule shall scan the active "Service Identifier Table" for a newly received diagnostic service identifier. If the newly received diagnostic service identifier is supported, the DSD submodule shall call the associated interface.

The diagnostic service identifier is not supported when it is not included in the "Service Identifier Table".

Dcm197: If the newly received diagnostic service identifier is not supported, the DSD submodule shall transmit a negative response with NRC 0x11 (Service not supported) to the DSL submodule.

Dcm198: The DSD submodule shall store the newly received diagnostic service identifier for later use.

For example:
WriteDataByIdentifier (for writing VIN number):
1. A new diagnostic message is received by the DSL submodule. (Diagnostic Message WriteDataByIdentifier =
0x2E,0xF1,0x90,0x57,0x30,0x4C,0x30,0x30,0x30,0x30,0x34,0x33,0x4D,0x42,0x35,0x34,0x31,0x33,0x32,0x36)
2. The DSL submodule indicates a new diagnostic message with the "Data Indication" functionality to the DSD submodule. In the diagnostic message buffer the diagnostic message is stored (buffer = 0x2E,0xF1,0x90,..).
3. The DSD submodule executes a check of the supported services with the determined service identifier (first byte of buffer 0x2E) on the incoming diagnostic message.
4. The incoming diagnostic message is stored in the DCM variable Dcm_MsgContextType.

7.3.4.2 Handling of „suppressPosRspMsgIndicationBit"
The "suppressPosRspMsgIndicationBit" is part of the subfunction parameter structure (Bit 7 based on second byte of the diagnostic message, see ISO14229-1 [9] Section 6.5: Server response implementation rules).

Dcm200: If the "suppressPosRspMsgIndicationBit" is TRUE, the DSD submodule shall NOT send a positive response message.

Dcm201: The DSD submodule shall remove the "suppressPosRspMsgIndicationBit" (by masking the Bit) from the diagnostic message.

Dcm202: The DCM module shall transport the information on a suppression of a positive response being active (between the layers) via the parameter Dcm_MsgContextType.

Dcm203: In case of responsePending the DCM module shall clear the "suppressPosRspMsgIndicationBit."

Rationale for Dcm203: In the described case the final response (negative/positive) is required.

Dcm204: The DCM module shall only perform the "suppressPosRspMsgIndicationBit" handling when the configuration parameter ***DcmDsdSidTabSubfuncAvail*** is set for the newly received service identifier

Rationale for Dcm204: The "suppressPosRspMsgIndicationBit" is only available if a service has a subfunction.

7.3.4.3 Verification functionality

The DSD submodule will only accept a service, if the following three verifications are passed:
1. Verification of the Diagnostic Session
2. Verification of the Service Security Access levels
3. Verification of the Application environment/permission

7.3.4.3.1 Verification of the Diagnostic Session

The UDS Service DiagnosticSessionControl (0x10) is used to enable different diagnostic sessions in the ECU (e.g. Default session, Extended session). A diagnostic session enables a specific set of diagnostic services and/or functionality in the ECU. It furthermore enables a protocol-depending data set of timing parameters applicable to the started session.

On receiving a service request, the DSD module will obtain the current Diagnostic Session with `Dcm_GetSesCtrlType()` and will verify whether the execution of the requested service (NOT the UDS Service DiagnosticSessionControl (0x10)) is allowed in the current diagnostic session or not.

Note that the handling of the UDS Service DiagnosticSessionControl (0x10) itself is not part of the DSD submodule.

Dcm211: If the newly received diagnostic service is not allowed in the current Diagnostic Session (according to the configuration parameter ***DcmDsdSidTabSessionLevelRef***), the DSD submodule shall transmit a negative response with NRC 0x7F (Service not supported in active session) to the DSL submodule.

7.3.4.3.2 Verification of the Service Security Access levels

The purpose of the Security Access level handling is to provide a possibility to access data and/or diagnostic services, which have restricted access for security, emissions, or safety reasons. The DSD submodule shall perform this handling with the UDS Service SecurityAccess (0x27). The DSD submodule will perform a verification whether the execution of the requested service (NOT the UDS Service SecurityAccess (0x27)) is allowed in the current Security level by asking for the current security level, using the DSL function `Dcm_GetSecurityLevel()`.
The management of the security level is not part of the DSD submodule.

Note: For some use cases (e.g. UDS Service ReadDataByIdentifier (0x22), where some DataIdentifier can be secure) it will be necessary for the Application to call also the function `Dcm_GetSecurityLevel()`.

Dcm217: If the newly received diagnostic service is not allowed in the current Security level (according to the configuration parameter **DcmDsdSidTabSecLevelRef**), the DSD submodule shall transmit a negative response with NRC 0x33 (Security access denied) to the DSL submodule.

7.3.4.3.3 Verification of the Application environment/permission

The purpose of this functionality is that, right before processing the diagnostic message, the Application is requested to check permission/environment.
E.g. in after-run ECU state, it might be not allowed to process OBD requests.

Dcm218: If configured (configuration parameter **DcmRequestIndicationEnabled**=TRUE), the DSD submodule shall call the operation `Xxx_Indication()` on all configured ServiceRequestIndication ports (see configuration parameter **DcmDslServiceRequestIndication**).

Dcm462: If at least a single `Xxx_Indication()` function called according to Dcm218 returns E_REQUEST_NOT_ACCEPTED, the DSD submodule shall give no response.

Dcm463: If at least a single `Xxx_Indication()` function called according to Dcm218 has returned E_REQUEST_ENV_NOK and no function has returned E_REQUEST_NOT_ACCEPTED, the DSD submodule shall trigger a negative response with NRC 0x22 (Conditions not correct).

7.3.4.4 Distribution of diagnostic message to DSP submodule

Dcm221: The DSD submodule shall search for the executable functionality of the DSP submodule for newly received diagnostic service identifier and shall call the corresponding DSP service interpreter.

7.3.4.5 Assemble positive or negative response

Dcm222: When the DSP submodule has finished the execution of the requested Diagnostic Service the DSD submodule shall assemble the response.

The execution of the DSP service interpreter can have the results:
-   positive Result or
-   negative Result.

Following possible Responses can be assembled:
-   positive Response,
-   negative Response, or
-   no Response (in the case of suppression of responses).

7.3.4.5.1 Positive Response

The DSP submodule indicates a positive result by calling `DsdInternal_ProcessingDone()`. The parameter "Dcm_MsgContextType" comprises the diagnostic (response) message.

Dcm223: The DSD submodule shall add the response service identifier and the response data stream (returned by the Application) in the parameter "Dcm_MsgContextType".

Dcm224: The DSD submodule shall therefore transfer the Dcm_MsgContextType into a (response) buffer and shall add the service identifier at the first byte of the buffer.

Dcm225: The DSD submodule shall execute the "Initiate transmission" functionality in the next execution step

7.3.4.5.2 Negative Response

The DSP submodule can trigger the transmission of a negative response with a specific NRC to the DSD submodule.
For the allowed NRC of the executed Service ID please refer to the specification of the service in ISO14229-1 [9] (see Section 4.2.4 Response code parameter definition Table 12) and ISO15031-5 [10]. The DSP and the Application have to take care of the correct use of NRC of the executed Service ID.

Dcm228: The DSD submodule shall handle all NRCs supported from the Application and defined in `Dcm_NegativeResponseCodeType`.

7.3.4.5.3 Suppression of response

Dcm231: In the case that the "suppressPosRspMsgIndicationBit" is indicated in the functionality "Handling of suppressPosRspMsgIndicationBit" (stored in the Variable Dcm_MsgContextType (Element: Dcm_MsgAddInfo)), the DSD submodule shall activate the suppression of Positive Responses.

Dcm001: In the case of a Negative Result of the execution and active Functional Addressing the DSD submodule shall activate the suppression of the following Negative Responses:
NRC 0x11 (Service not supported),
NRC 0x12 (SubFunction not supported),
NRC 0x31 (Request out of range).

7.3.4.6 Initiate transmission

Dcm232: The DSD submodule shall forward the diagnostic (response) message (positive or negative response) to the DSL submodule.

Dcm237: The DSL submodule shall forward the diagnostic (response) message (positive or negative response) further to the PduR module by executing a DSL transmit functionality.

The DSL submodule will receive a confirmation by the PduR module upon forwarding the data.

Dcm235: The DSL submodule shall forward the received confirmation from the PduR module to the DSD submodule.

Dcm236: The DSD submodule shall forward the confirmation via the internal function `DspInternal_DcmConfirmation()` to the DSP submodule.

Dcm238: In the case that no diagnostic (response) message shall be sent (Suppression of Responses) the DSL submodule shall not transmit any response.

In this case no Data Confirmation is sent from the DSL submodule to the DSD submodule but the DSD submodule will still call internal function `DspInternal_DcmConfirmation()`.

Dcm240: The DSD submodule shall always finish the processing of one Diagnostic Message of the Diagnostic Service Dispatcher by calling `DspInternal_DcmConfirmation()`.

Rationale for Dcm240: The DSP submodule is waiting for the execution of the `DspInternal_DcmConfirmation()` functionality. So it has to be sent, also when no Data Confirmation is provided.
Altogether this means that in any of the following cases:
- Positive Response,
- Negative Response,
- Suppressed Positive Response, and
- Suppressed Negative Response
the DSD submodule will finish by calling `DspInternal_DcmConfirmation()`.

7.4 Diagnostic Service Processing – DSP

7.4.1 General

When receiving a function call from the DSD submodule requiring the DSP submodule to process a diagnostic service request, the DSP always carries out following basic process steps:
- analyze the received request message,
- check format and whether the addressed subfunction is supported,
- acquire data or execute the required function call on the DEM, SW-Cs or other BSW modules
- assemble the response
The following sections are some general clarifications.

7.4.1.1 Check format and subfunction support
The DSP submodule checks whether a specific subfunction is supported before executing the requested command.

Dcm273: The DSP submodule shall trigger a negative response with NRC 0x12 (SubFunction not supported), when the analysis of the request message results in subfunction not supported.

The DSP submodule will check for appropriate message length and structure before executing the requested command.

Dcm272: The DSP submodule shall trigger a negative response with NRC 0x13 (Incorrect message length or invalid format), when the analysis of the request message results in formatting or length failure.

Note: It is up to the implementation in which detail the format check might be executed and depends on the level of detail the diagnostic data description provides at compile time.

7.4.1.2 Assemble response

Dcm039: The DSP submodule shall assemble the response message excluding response service identifier and determine the response message length.

Dcm038: If the paged-buffer mechanism is used, the DSP submodule shall determine the overall response length before any data is passed to the DSD submodule or the DSL submodule respectively.

Requirement Dcm038 is needed because of segmented diagnostic data transmission on CAN using ISO15765-2 [11], which requires the provision of the overall length of the complete data stream in the very first CAN frame of the respective data transmission (please refer to Section 7.2.4.8 for details about the paged-buffer mechanism).

Dcm269: The DSP submodule shall confirm the completion of the request processing with the function call `DsdInternal_ProcessingDone()`.

7.4.1.3 Additional Negative Response Codes (NRCs)

Dcm271: The DSP submodule shall trigger a negative response with NRC 0x22 (Conditions not correct), when the API calls made to execute the service do not return OK.

Dcm274: The DSP submodule shall trigger a negative response with NRC 0x7F (Service not supported in active session), when the analysis of the request message results in subfunction not supported in active diagnostic session.

Dcm275: The DSP submodule shall trigger a negative response with NRC 0x31 (Request out of range), when the analysis of the request message results in other unsupported message parameters.

Dcm424: The DSP submodule shall trigger a negative response with NRC 0x33 (Security access denied), when the analysis of the request message results in subfunction not supported in active security level.

7.4.2 UDS Services

Dcm442: The DCM module shall implement the services of UDS according to the following table:

SID	Service	Subfunction	Supported
0x10	DiagnosticSessionControl		supported
0x11	ECUReset		supported
0x14	ClearDiagnosticInformation		supported
0x19	ReadDTCInformation		supported
0x22	ReadDataByIdentifier		supported
0x23	ReadMemoryByAddress		NRC "ServiceNotSupported"
0x24	ReadScalingDataByIdentifier		supported
0x27	SecurityAccess		supported
0x28	CommunicationControl		supported
0x2A	ReadDataByPeriodicIdentifier		supported
0x2C	DynamicallyDefineDataIdentifier	0x02, defineByMemoryAddress	NRC "SubFunctionNotSupported"
0x2C	DynamicallyDefineDataIdentifier	No 0x02	supported
0x2E	WriteDataByIdentifier		supported
0x2F	InputOutputControlByIdentifier		supported
0x31	RoutineControl		supported
0x34	RequestDownload		NRC "ServiceNotSupported"
0x35	RequestUpload		NRC "ServiceNotSupported"
0x36	TransferData		NRC "ServiceNotSupported"
0x37	RequestTransferExit		NRC "ServiceNotSupported"
0x3D	WriteMemoryByAddress		NRC "ServiceNotSupported"
0x3E	TesterPresent	0x00, 0x80	supported
0x3E	TesterPresent	Not 0x00 and not 0x80	NRC "SubFunctionNotSupported"
0x83	AccessTimingParameter		NRC "ServiceNotSupported"
0x84	SecuredDataTransmission		NRC "ServiceNotSupported"
0x85	ControlDTCSetting	On, off	supported
0x86	ResponseOnEvent		NRC "ServiceNotSupported"
0x87	LinkControl		NRC "ServiceNotSupported"

Table 4: Support of UDS Services

7.4.2.1 General behavior using DEM interfaces

Dcm007: The DCM requirements use the term DTCStatusAvailabilityMask to indicate the value returned by the function `Dem_GetDTCStatusAvailabilityMask()`.

The mask DTCStatusAvailabilityMask reflects the status bits supported by the ECU.

Dcm371: To avoid updating data values while reading them out, the DCM module shall call the following API sequence: first lock updates by calling `Dem_DisableDTCRecordUpdate()` (which is an exclusive area function concerning to BSW00434) then call `Dem_GetSizeOfExtendedDataRecordByDTC()`, then `Dem_GetExtendedDataRecordByDTC(RecNum)` and finally shall re-enable updates by calling `Dem_EnableDTCRecordUpdate()`.

Dcm372: To avoid updating data values while reading them out, the DCM module shall call the following API sequence: first lock updates by calling `Dem_DisableDTCRecordUpdate()` (which is an exclusive area function concerning to BSW00434) then call `Dem_GetSizeOfFreezeFrameByDTC(RecNum=0xFF)`, then `Dem_GetFreezeFrameDataByDTC(RecNum)` and finally shall re-enable updates by calling Dem_EnableDTCRecordUpdate().

Dcm379: If function Dem_DisableDTCRecordUpdate() returns DEM_DISABLE_DTCRECUP_PENDING, the DCM shall retry to get the lock in the next Dcm_MainFunction().

Note: A timeout or maximum counter is not necessary, because the DEM guarantees not to lock the DTC for longer time.

Dcm700: When the DCM module receives a request with the DTCStatusMask set to 0x00, it shall handle this value on its own and shall not use the DEM interface `Dem_SetDTCFilter()`.

7.4.2.2 UDS Service 0x10 - Diagnostic Session Control
UDS Service 0x10 allows an external tester to enable different diagnostic sessions in the server. A diagnostic session enables a specific set of diagnostic services and/or functionality in the server. The service request contains the parameter:
- diagnosticSessionType

Dcm250: The DCM module shall implement the UDS Service 0x10

Dcm307: When responding to UDS Service 0x10, if the requested subfunction value is not configured in the ECU (configuration parameter *DcmDspSessionLevel*), the DSP submodule shall trigger a negative response with NRC 0x12 (SubFunction not supported).

If the requested subfunction value is configured, the following steps are processed even if the requested session type is equal to the already running session type (see ISO14229-1 [9] Section 8.2).

When responding to UDS Service 0x10, after verification of the validity of the request (see Dcm307), the DSP submodule will start calling the operations `Xxx_GetSesChgPermission()` in all configured ports (see configuration parameter **DcmDslSessionControl**) to examine environment settings and get the permission of the Application to change the session.
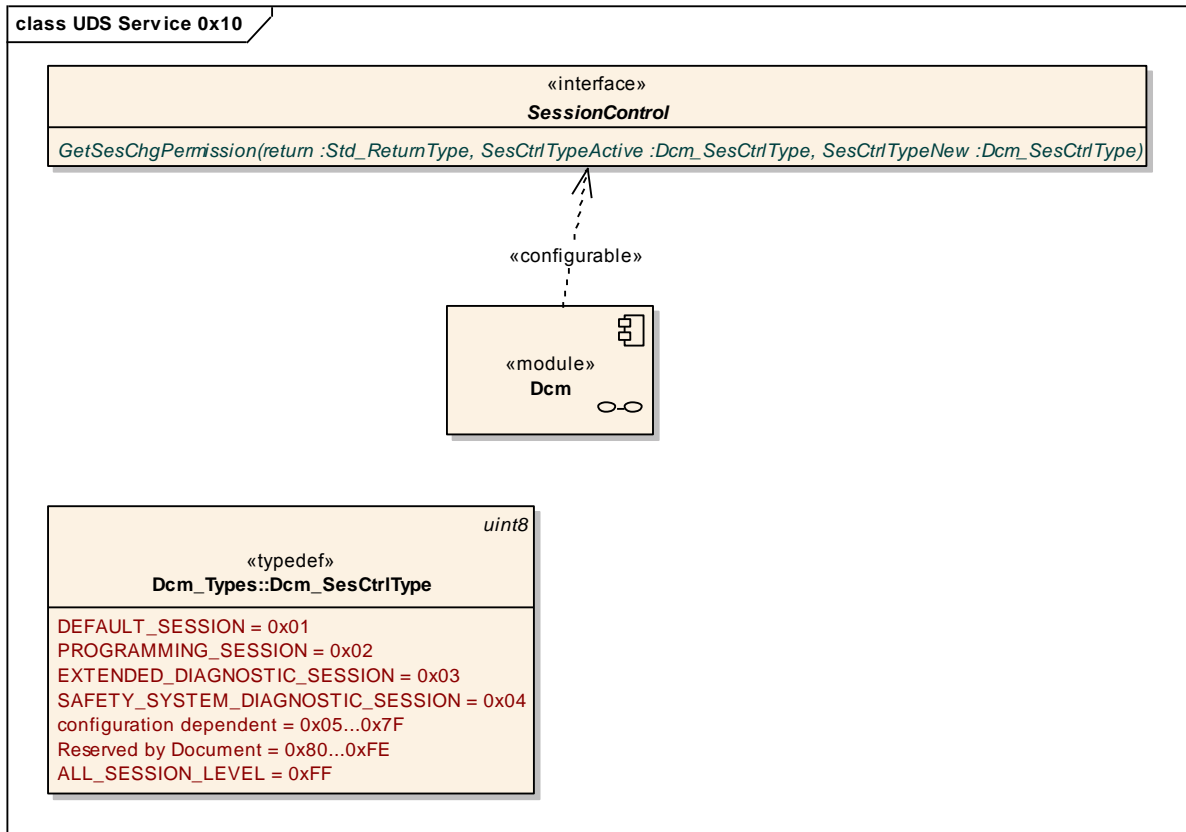
Figure 7: Interaction between DCM and Application for the realization of UDS Service 0x10

Dcm308: When responding to UDS Service 0x10, after verification of the validity of the request (see Dcm307): if at least one of the `Xxx_GetSesChgPermission()` functions returns E_SESSION_NOT_ALLOWED, the DSP submodule shall trigger a negative response with NRC 0x22 (Conditions not correct).

Dcm138: When responding to UDS Service 0x10, after verification of the validity of the request (see Dcm307) and if enabled (through configuration parameter **DcmDslDiagRespForceRespPendEn**): if at least one of the `Xxx_GetSesChgPermission()` functions return E_FORCE_RCRRP and none returns E_SESSION_NOT_ALLOWED, the DSP submodule shall trigger a negative response with NRC 0x78 (Response pending).

Dcm311: When responding to UDS Service 0x10, after the verification of the validity of the request (see Dcm307) and if all `Xxx_GetSesChgPermission()` return E_OK, the sent confirmation function shall set the diagnostic session type with `DslInternal_SetSesCtrlType()` and shall set the new timing parameters (P2ServerMax, P2ServerMax*) (see configuration parameters *DcmDspSessionP2ServerMax* and *DcmDspSessionP2StarServerMax*).

Dcm085: The DSP submodule shall manage internally a read access for the dataIdentifier 0xF186 (ActiveDiagnosticSessionDataIdentifier) defined in ISO14229-1 [9].

7.4.2.3 UDS Service 0x11 - ECUReset

UDS Service ECUReset (0x11) allows an external tester to request a server reset.

Dcm260: The DCM module shall implement the UDS Service ECUReset (0x11).
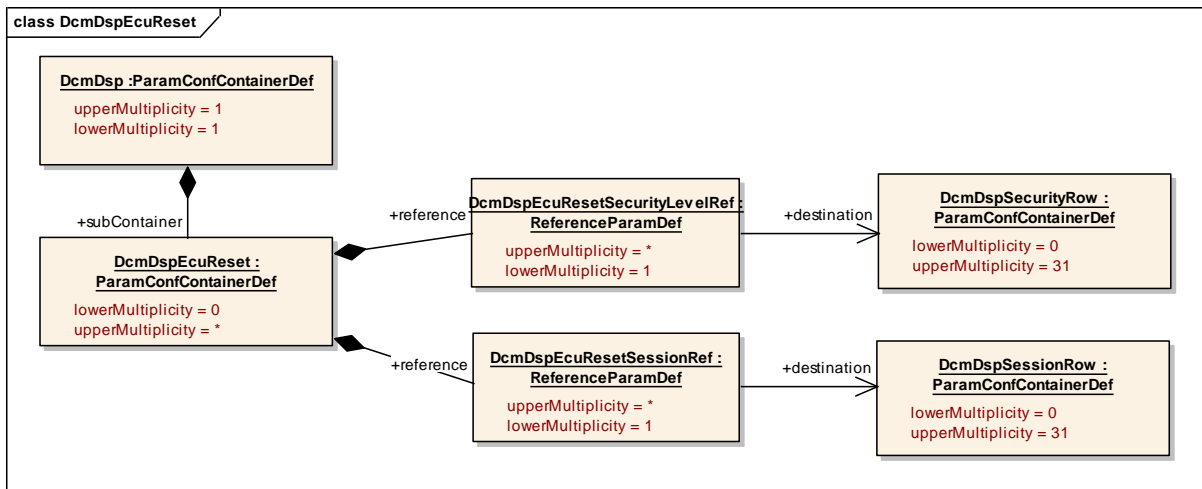
Figure 8: Extract from the DCM configuration related to UDS Service ECUReset (0x11).
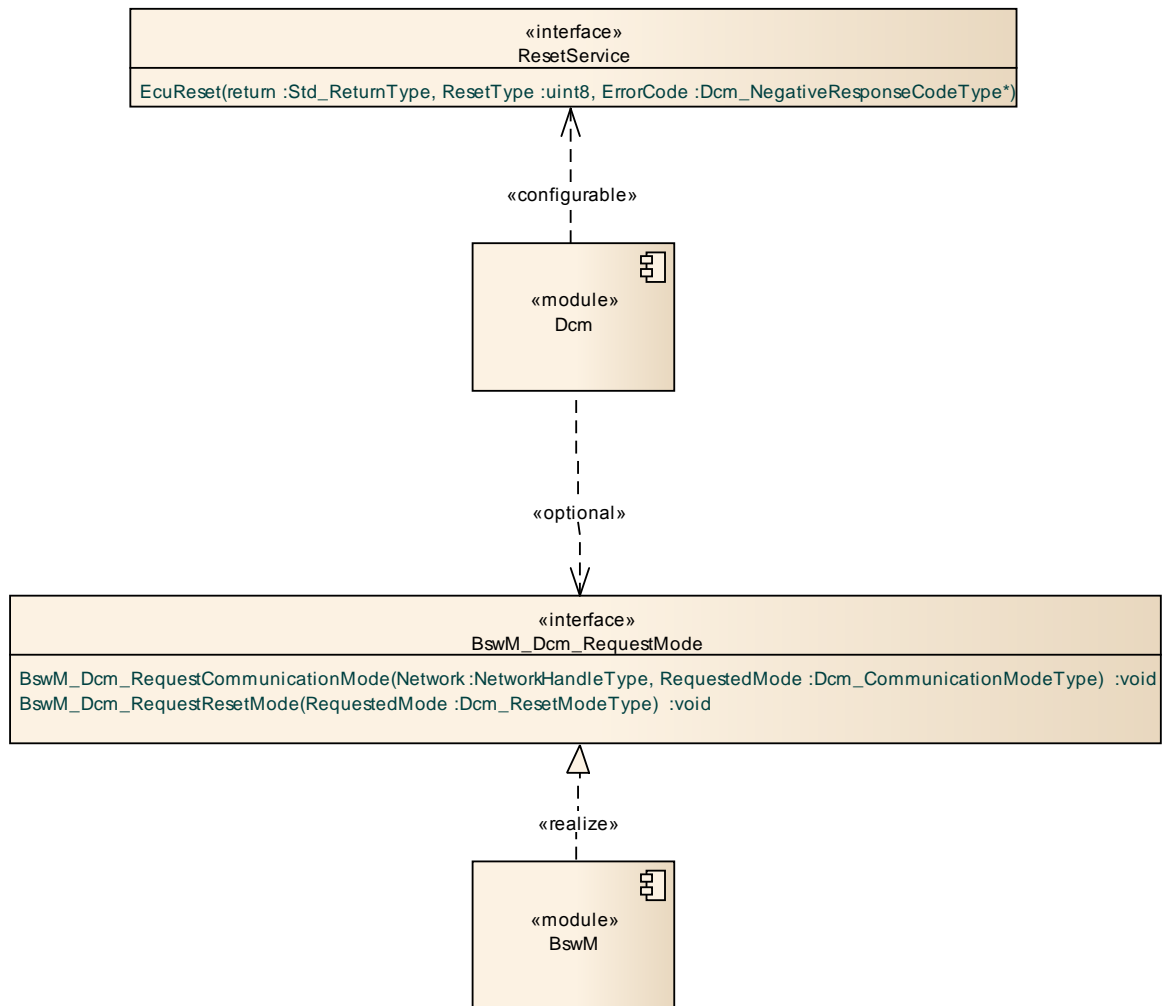
Figure 9: Interaction between DCM and other modules during the execution of UDS Service 0x11

Dcm373: On reception of a request for UDS Service 0x11, the DCM module shall invoke the configured `Xxx_EcuReset()` functions (see configuration parameter **DcmDspEcuReset**) with the following parameter values:
ResetType: the resetType received in the UDS Service 0x11 request.

Dcm374: After having executed Dcm373, when all the `Xxx_EcuReset()` functions return a value E_OK, the DCM module shall respond positively to the request and after the response transmission confirmation the DCM shall call `BswM_Dcm_RequestResetMode()` with the following parameter values:
ResetModeType: the resetType received in the UDS Service 0x11 request.

Dcm375: After having executed [Dcm373](#), when at least one `Xxx_EcuReset()` function returns E_NOT_OK, the DCM module shall not call `BswM_Dcm_RequestResetMode()` but shall trigger a negative response with the NRC value returned in ErrorCode

Dcm477: After having executed [Dcm373](#), when one or more `Xxx_EcuReset()` functions return E_PENDING and no function returns E_NOT_OK, the DCM module shall invoke the ones returning E_PENDING again in a next cycle.

7.4.2.4 UDS Service 0x14 - Clear Diagnostic Information

UDS Service ClearDiagnosticInformation (0x14) requests an ECU to clear the error memory. The service request contains the parameter:

- groupOfDTC.

Dcm247: The DCM module shall implement UDS Service 0x14.
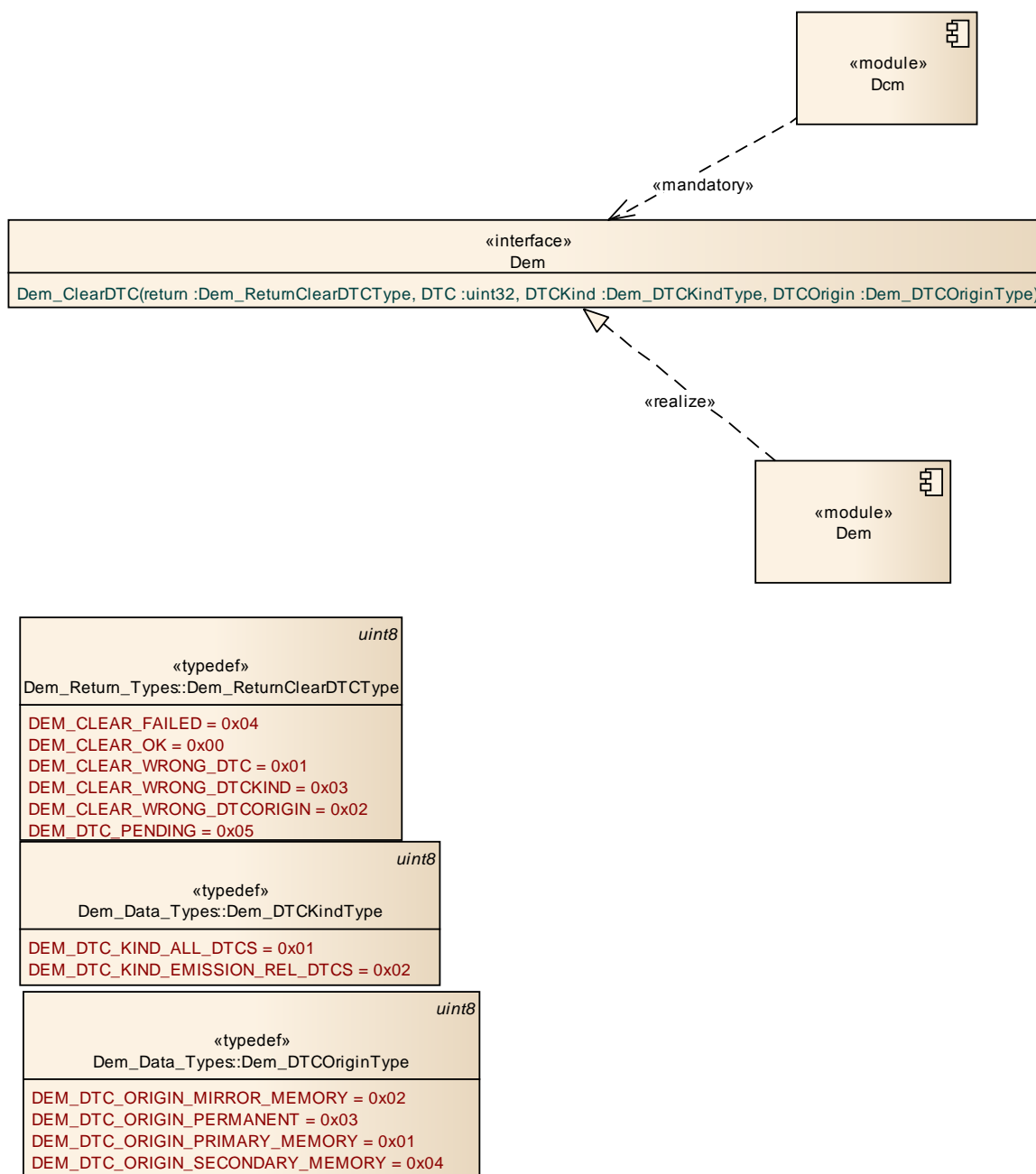
Figure 10: Interaction between DEM and DCM for the realization of UDS Service 0x14

Dcm005: Upon reception of a UDS Service ClearDiagnosticInformation (0x14) request with parameter groupOfDTC, the DCM module shall execute the function `Dem_ClearDTC()` with the following parameter values:
DTC: groupOfDTC from the service request
DTCKind: DEM_DTC_KIND_ALL_DTCS
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY

7.4.2.5 UDS Service 0x19 - Read DTC Information

Dcm248: The DCM module shall implement the UDS Service 0x19.
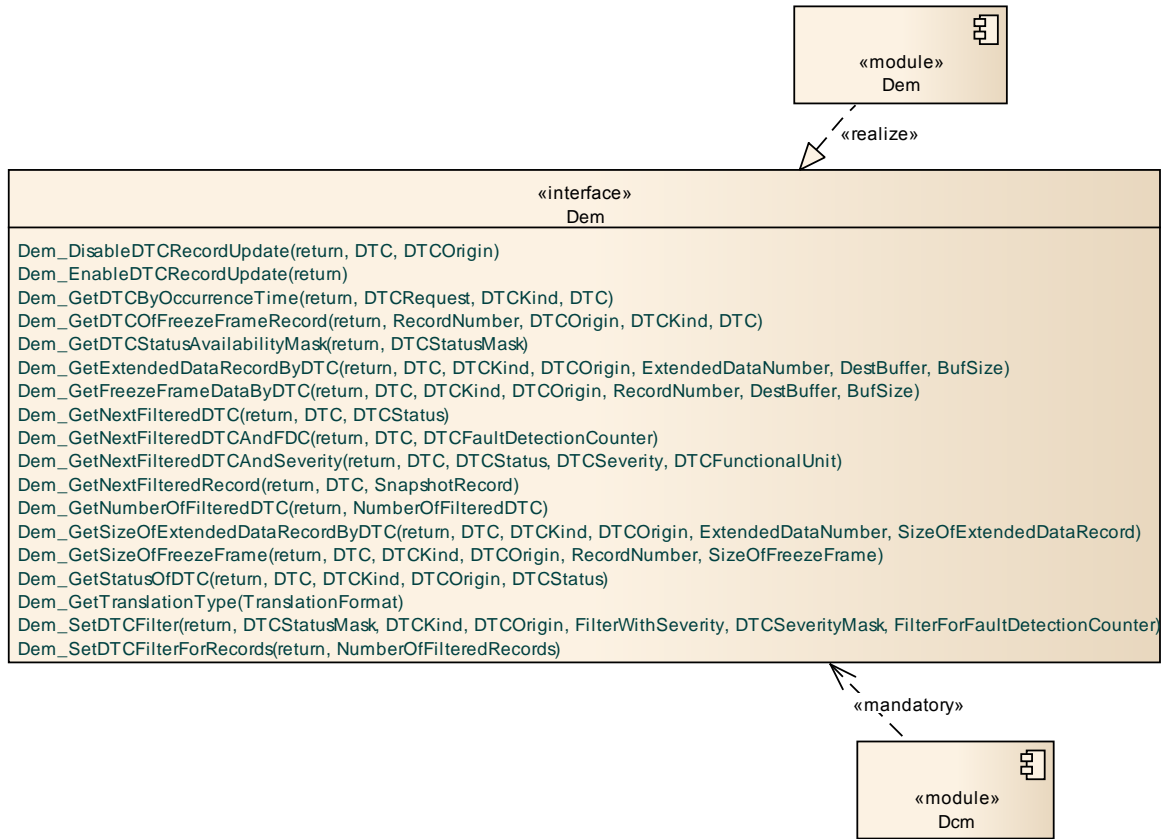
Figure 11: Interaction between DEM and DCM for the realization of UDS Service 0x19

Dcm713: If DEM_WRONG_FILTER value is returned from Dem_ReturnSetDTCFilterType, the Dcm module shall send a negative response with 0x31.

7.4.2.5.1 UDS Service 0x19 with subfunctions
0x01(reportNumberOfDTCByStatusMask),
0x07(reportNumberOfDTCBySeverityMaskRecord) ,
0x11(reportNumberOfMirrorMemoryDTCByStatusMask),
0x12(reportNumberOfEmissionsRelatedOBDDTCByStatusMask)

UDS Service 0x19 with subfunctions 0x01, 0x11 or 0x12 requests the ECU to report the number of DTCs matching tester-defined criteria. The service request contains the parameter:

- DTCStatusMask

UDS Service 0x19 with subfunction 0x07 requests the ECU to report the number of DTCs matching tester-defined criteria. The service request contains the parameters:

- AUTOSAR confidential -

- DTCSeverityMask
- DTCStatusMask

Dcm376: When sending a positive response to UDS Service 0x19 with subfunction 0x01, 0x07, 0x11 or 0x12, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see Dcm007)
DTCFormatIdentifier	value returned by `Dem GetTranslationType()`
DTCCount	value calculated according to Dcm293

Dcm293: When responding to UDS Service 0x19 with subfunction 0x01, 0x07, 0x11 or 0x12, the DCM module shall calculate the number of DTCs using `Dem_GetNumberOfFilteredDTC()` after having set the DEM-filter with `Dem_SetDTCFilter()` using the parameter values of the following table:

	reportNumberOfDTCByStatusMask	reportNumberOfDTCBySeverityMaskRecord	reportMirrorMemoryDTCByStatusMask	reportNumberOfEmissionsRelatedOBDDTCByStatusMask
	0x01	0x07	0x11	0x12
DTCStatusMask	DTCStatusMask from request (see Dcm700)	DTCStatusMask from request (see Dcm700)	DTCStatusMask from request (see Dcm700)	DTCStatusMask from request (see Dcm700)
DTCKind	ALL_DTCS	ALL_DTCS	ALL_DTCS	EMISSION_REL_DTCS
DTCOrigin	PRIMARY_MEMORY	PRIMARY_MEMORY	MIRROR_MEMORY	PRIMARY_MEMORY
FilterWithSeverity	NO	YES	NO	NO
DTCSeverity	Not relevant	DTCSeverityMask from request	Not relevant	Not relevant
FilterForFaultDetection	NO	NO	NO	NO

7.4.2.5.2 UDS Service 0x19 with subfunctions 0x02(reportDTCByStatusMask), 0x0A(reportSupportedDTCs), 0x0F(reportMirrorMemoryDTCByStatusMask), 0x13(reportEmissionsRelatedOBDDTCByStatusMask), 0x15(reportDTCWithPermantStatus)

UDS Service 0x19 with subfunctions 0x02, 0x0F or 0x13 requests the DTCs (and their associated status) that match certain conditions. The service request contains the parameter:

- DTCStatusMask

UDS Service 0x19 with subfunction 0x0A requests all supported DTCs and their associated status. UDS Service 0x19 with subfunction 0x15 requests all DTCs with permanent status.

Dcm377: When sending a positive response to UDS Service 0x19 with subfunction 0x02, 0x0A, 0x0F, 0x13 or 0x15, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see Dcm007)

DTCAndStatusRecord	As defined in Dcm008 and Dcm378

Dcm008: On reception of a UDS Service 0x19 request with subfunction 0x02, 0x0A, 0x0F, 0x13 or 0x15 and if DTCStatusAvailabilityMask (see Dcm007) is equal to 0, the DCM module shall answer positively with 0 DTC.

Dcm378: When responding to UDS Service 0x19 with subfunctions 0x02, 0x0A, 0x0F, 0x13 or 0x15, the DCM module shall obtain the records with DTCs (and their associated status) by repeatedly calling `Dem_GetNextFilteredDTC()` after having configured the filter with `Dem_SetDTCFilter()` using the parameter values of the following table:

	reportDTC ByStatusM ask	reportSupp ortedDTCs	reportMirror MemoryDTC ByStatusMa sk	reportEmiss ionsRelated OBDDTCBy StatusMask	reportDTC WithPerma nentStatus
	0x02	0x0A	0x0F	0x13	0x15
DTCStatusMask	DTCStatus Mask from request (see Dcm700)	0x00	DTCStatusM ask from request (see Dcm700)	DTCStatusM ask from request (see Dcm700)	0x00
DTCKind	ALL_DTCS	ALL_DTCS	ALL_DTCS	EMISSION_ REL_DTCS	ALL_DTCS
DTCOrigin	PRIMARY_ MEMORY	PRIMARY_ MEMORY	MIRROR_ME MORY	PRIMARY_M EMORY	PERMANE NT
FilterWithSeverity	NO	NO	NO	NO	NO
DTCSeverity	Not relevant	Not relevant	Not relevant	Not relevant	Not relevant
FilterForFaultDete ction	NO	NO	NO	NO	NO

Note:
- The DCM module can obtain the number of records that will be found using `Dem_GetNextFilteredDTC()` by using `Dem_GetNumberOfFilteredDTC()`. This allows the implementation to calculate the total size of the response before cycling through the DTCs.
- The value 0x00 used as DTCStatusMask for the subfunctions 0x0A and 0x15 disables the status byte filtering in `Dem_SetDTCFilter()`.

7.4.2.5.3 UDS Service 0x19 with subfunction 0x08 (report DTC by Severity Mask Record)

UDS Service 0x19 with subfunction 0x08 requests the DTCs and the associated status that match a tester-defined severity mask record. The service request contains the following parameters:
- DTCSeverityMask
- DTCStatusMask

Dcm379: When sending a positive response to UDS Service 0x19 with subfunction 0x08, the DCM module shall use the following data in the response message:

Parameter name	Value

DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see Dcm007)
DTCAndSeverityRecord	As defined in Dcm380

	reportDTCBySeverityMaskRecord
DTCStatusMask	DTCStatusMask from request (see Dcm700)
DTCKind	ALL_DTCS
DTCOrigin	PRIMARY_MEMORY
FilterWithSeverity	YES
DTCSeverity	DTCSeverityMask from request
FilterForFaultDetection	NO

Dcm380: When responding to UDS Service 0x19 with subfunction 0x08, the DCM module shall obtain the DTCAndSeverityRecords by repeatedly calling `Dem_GetNextFilteredDTCAndSeverity()` after having configured the filter with `Dem_SetDTCFilter()` using the parameter values of the following table:

Note: The DCM module can obtain the number of records that will be found using `Dem_GetNextFilteredDTCAndSeverity()` by using `Dem_GetNumberOfFilteredDTC()`.

7.4.2.5.4 UDS Service 0x19 with subfunction 0x09 (report Severity Information of DTC)

UDS Service 0x19 with subfunction 0x09 requests the severity information of a DTC. The service request contains the parameter:

- DTCMaskRecord

Dcm381: When sending a positive response to UDS Service 0x19 with subfunction 0x09, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see Dcm007)

DTCAndSeverityRecord	DTCSeverity : use `Dem_GetSeverityOfDTC()`
	 DTCFunctionalUnit : use
	 `Dem_DcmGetFunctionalUnitOfDTC()`
	 DTCxxx : the given DTC of the request
	 statusOfDTC : use `Dem_GetStatusOfDTC()` with
	 parameters value:
	
	 `DTC: DTC from the request`
	 `DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY`

7.4.2.5.5 Service 0x19 subfunctions 0x06/0x10 + DTC + 0xFF - Report all Extended Data Records for a particular DTC

UDS Service 0x19 with subfunction 0x06 or 0x10 and DTCExtendedDataRecordNumber=0xFF requests all Extended Data Records for a specific DTC. The request contains:

- DTCMaskRecord
- DTCExtendedDataRecordNumber (equal to 0xFF)

Dcm297: When sending a positive response to UDS Service 0x19 with subfunction 0x06 or 0x10 and DTCExtendedDataRecordNumber=0xFF, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCAndStatusRecord	see Dcm295
DTCExtendedDataRecordNumber	see Dcm296
DTCExtendedDataRecord	see Dcm296

Dcm295: When responding to UDS Service 0x19 with subfunction 0x06 or 0x10, the DCM module shall calculate the statusOfDTC by calling `Dem_GetStatusOfDTC()` with the following parameters:

	reportDTCExtendedDataRecordByDTCNumber	reportMirrorMemoryDTCExtendedDataRecordByDTCNumber
	0x06	0x10
DTC	DTCMaskRecord from request	DTCMaskRecord from request
DTCKind	ALL_DTCS	ALL_DTCS
DTCOrigin	PRIMARY_MEMORY	MIRROR_MEMORY

Dcm296: When responding to UDS Service 0x19 with subfunction 0x06 or 0x10, the DCM module shall assemble the ExtendedDataRecords by calling `Dem_GetExtendedDataRecordByDTC()` repeatedly (where ExtendedDataNumber goes from 0x01 to 0xEF) with the following parameter values:

	reportDTCExtendedDataRecordByDTCNumber	reportMirrorMemoryDTCExtendedDataRecordByDTCNumber
	0x06	0x10
DTC	DTCMaskRecord from	DTCMaskRecord from request

	request	
DTCKind	ALL_DTCS	ALL_DTCS
DTCOrigin	PRIMARY_MEMORY	MIRROR_MEMORY
ExtendedDataNumber	0x01 to 0xEF	0x01 to 0xEF

Dcm478: The DCM module shall obtain the size of the extended data record by using `Dem_GetSizeOfExtendedDataRecordByDTC()`.

7.4.2.5.6 Service 0x19 subfunctions 0x06/0x10 + DTC + 0xFE - Report all OBD Extended Data Records for a particular DTC

UDS Service 0x19 with subfunction 0x06 or 0x10 and DTCExtendedDataRecordNumber=0xFE requests all OBD Extended Data Records for a specific DTC. The request contains:

- DTCMaskRecord
- DTCExtendedDataRecordNumber (equal to 0xFE)

Dcm474: When sending a positive response to UDS Service 0x19 with subfunction 0x06 or 0x10 and DTCExtendedDataRecordNumber=0xFE, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCAndStatusRecord	see Dcm475
DTCExtendedDataRecordNumber	see Dcm476
DTCExtendedDataRecord	see Dcm476

Dcm475: When responding to UDS Service 0x19 with subfunction 0x06 or 0x10, the DCM module shall calculate the statusOfDTC by calling `Dem_GetStatusOfDTC()` with the following parameters:

	reportDTCExtendedDataRecordByDTCNumber	reportMirrorMemoryDTCExtendedDataRecordByDTCNumber
	0x06	0x10
DTC	DTCMaskRecord from request	DTCMaskRecord from request
DTCKind	ALL_DTCS	ALL_DTCS
DTCOrigin	PRIMARY_MEMORY	MIRROR_MEMORY

Dcm476: When responding to UDS Service 0x19 with subfunction 0x06 or 0x10, the DCM module shall assemble the ExtendedDataRecords by calling `Dem_GetExtendedDataRecordByDTC()` repeatedly (where ExtendedDataNumber goes from 0x90 to 0xEF) with the following parameter values:

	reportDTCExtendedDataRecordByDTCNumber	reportMirrorMemoryDTCExtendedDataRecordByDTCNumber
	0x06	0x10
DTC	DTCMaskRecord from	DTCMaskRecord from request

	request	
DTCKind	ALL_DTCS	ALL_DTCS
DTCOrigin	PRIMARY_MEMORY	MIRROR_MEMORY
ExtendedDataNumber	0x90 to 0xEF	0x90 to 0xEF

Dcm479: The DCM module shall obtain the size of the extended data record by using `Dem_GetSizeOfExtendedDataRecordByDTC()`.

7.4.2.5.7 Service 0x19 subfunctions 0x06/0x10 + DTC + (not 0xFF nor 0xFE) - Report one specific Extended Data Record for a particular DTC

The UDS Service 0x19 with subfunction 0x06 or 0x10 and DTCExtendedDataRecordNumber different from 0xFF or 0xFE requests a specific Extended Data Records for a specific DTC. The service request contains the parameters:
- DTCMaskRecord
- DTCExtendedDataRecordNumber (different from 0xFF or 0xFE)

Dcm386: When sending a positive response to UDS Service 0x19 with subfunction 0x06 or 0x10 and DTCExtendedDataRecordNumber different from 0xFF or 0xFE, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCAndStatusRecord	see Dcm295
DTCExtendedDataRecordNumber	DTCExtendedDataRecordNumber from request
DTCExtendedDataRecord	see Dcm382

Dcm382: When responding to UDS Service 0x19 with subfunction 0x06 or 0x10 and DTCExtendedDataRecordNumber different from 0xFF or 0xFE, the DCM module shall calculate the DTCExtendedDataRecord from `Dem_GetExtendedDataRecordByDTC()` with the following parameter values.

	reportDTCExtendedDataRecordByDTCNumber	reportMirrorMemoryDTCExtendedDataRecordByDTCNumber
	0x06	0x10
DTC	DTCMaskRecord from request	DTCMaskRecord from request
DTCKind	ALL_DTCS	ALL_DTCS
DTCOrigin	PRIMARY_MEMORY	MIRROR_MEMORY
ExtendedDataNumber	DTCExtendedDataRecordNumber from request	DTCExtendedDataRecordNumber from request

Dcm480: The DCM module shall obtain the size of the extended data record by using `Dem_GetSizeOfExtendedDataRecordByDTC()`.

7.4.2.5.8 UDS Service 0x19 subfunctions 0x03 - Report DTC Snapshot Record Identification

UDS Service 0x19 with subfunction 0x03 allows an external tester to request the corresponding DTCs for all FreezeFrame records present in an ECU.

Dcm300: When sending a positive response to UDS Service 0x19 with subfunction 0x03, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCRecord/ DTCSnapshotRecordNumber	As defined in Dcm299
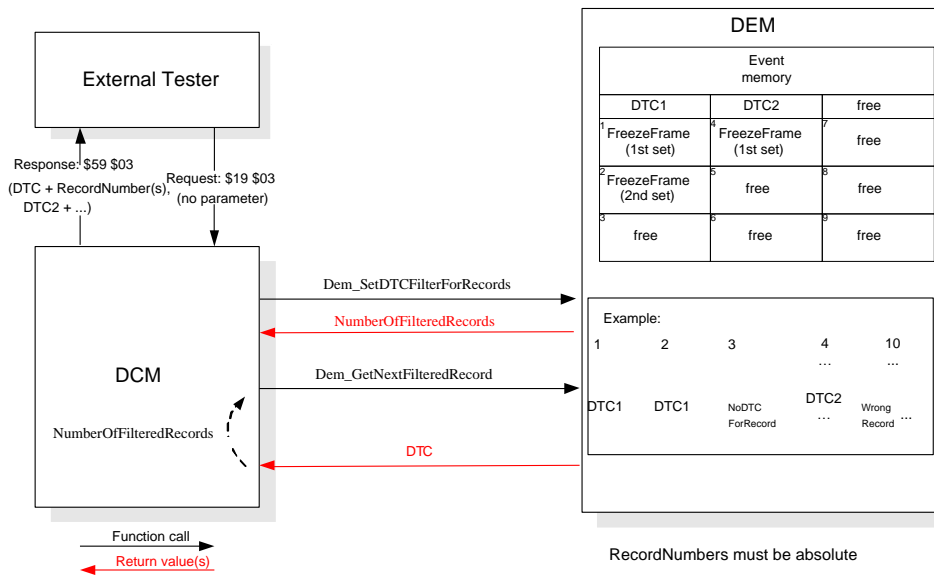
Figure 12: Request DTC Snapshot Record Identification

Dcm298: The DSP submodule shall call `Dem_SetDTCFilterForRecords()` that returns the NumberOfFilteredRecords value.

Dcm299: When responding to UDS Service 0x19 with subfunction 0x03, the DCM module shall obtain the consecutive DTCs and DTCSnapshotRecordNumbers by repeatedly calling `Dem_GetNextFilteredRecord()`.

7.4.2.5.9 UDS Service 0x19 subfunctions 0x04 – Report DTC Snapshot Record by DTC Number

Using UDS Service 0x19 with subfunction 0x04 an external tester can request FreezeFrame information for one or all FreezeFrames of a specific DTC. The service request contains parameters:

- DTCMaskRecord

- DTCSnapshotRecordNumber

Dcm302: When sending a positive response to UDS Service 0x19 with subfunction 0x04 and DTCSnapshotRecordNumber not 0xFF, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCAndStatusRecord	DTC from the request, statusOfDTC according to Dcm383
DTCSnapshotRecordNumber	The DTCSnapshotRecordNumber from the request
DTCSnapshotRecordNumberOfIdentifiers/ DTCSnapshotRecord	As defined in Dcm384

Dcm387: When sending a positive response to UDS Service 0x19 with subfunction 0x04 and DTCSnapshotRecordNumber=0xFF, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCAndStatusRecord	DTC from the request, statusOfDTC according to Dcm383
DTCSnapshotRecordNumber	As defined in Dcm385
DTCSnapshotRecordNumberOfIdentifiers	As defined in Dcm427
DTCSnapshotRecord	As defined in Dcm385

Dcm383: When responding to UDS Service 0x19 with subfunction 0x04, the DCM module shall obtain the status of the DTC by calling `Dem_GetStatusOfDTC()` with the following parameters:
DTC: DTC from the request
DTCKind: DEM_DTC_KIND_ALL_DTCS
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY

Dcm384: Upon reception of UDS Service 0x019 with subfunction 0x04 and DTCSnapshotRecordNumber not 0xff, DCM module shall obtain the FreezeFrame by calling `Dem_GetFreezeFrameDataByDTC()` with the following parameter values:
DTC: DTCMaskRecord from the request
DTCKind: DEM_DTC_KIND_ALL_DTCS
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY
RecordNumber: DTCSnapshotRecordNumber from the request

Dcm385: Upon reception of UDS Service 0x019 with subfunction 0x04 and DTCSnapshotRecordNumber 0xff, the DCM module shall cycle through all FreezeFrame numbers from 0x00 to 0xfe and obtain the corresponding FreezeFrame by calling `Dem_GetFreezeFrameDataByDTC()` with the following parameter values:
DTC: DTCMaskRecord from the request
DTCKind: DEM_DTC_KIND_ALL_DTCS
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY
RecordNumber: value from 0x00 -> 0xFE

Dcm441:The DCM module shall calculate the size of data returned in `Dem_GetFreezeFrameDataByDTC()` ahead of time using `Dem_GetSizeOfFreezeFrame()`.

Dcm427: To determine the "DTCSnapshotRecordNumberOfIdentifiers" UDS parameter, the DCM module shall call `Dem_GetFreezeFrameDataIdentifierByDTC()`.

7.4.2.5.10 UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber is 0xFF (Report all DTC Snapshot Record)

UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber parameter set to 0xFF allows an external tester to request FreezeFrame information for all FreezeFrame record number.

Dcm391: When sending a positive response to UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber set to 0xFF, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCSnapshotRecordNumber	DTCSnapshotRecordNumber requested by DCM
DTCAndStatusRecord	DTC according to Dcm429, statusOfDTC according to Dcm430
DTCSnapshotRecordNumberOfIdentifiers	As defined in Dcm432
DTCSnapshotRecord	As defined in Dcm431

Dcm429: When responding to UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber set to 0xFF, the DCM module shall obtain the DTC by calling `Dem_GetDTCOfFreezeFrameRecord()` repeatedly (where RecordNumber goes from 0x01 to 0xEF) with the following parameter values:
RecordNumber: 0x01 to 0xEF
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY
DTCKind: DEM_DTC_KIND_ALL_DTCS

Dcm430: When responding to UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber set to 0xFF, the DCM module shall obtain the status of the DTC by calling `Dem_GetStatusOfDTC()` repeatedly (after every call of `Dem_GetDTCOfFreezeFrameRecord()`, see Dcm429) with the following parameters:
DTC: DTC as defined in
DTCKind: DEM_DTC_KIND_ALL_DTCS
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY

Dcm431: When responding to UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber set to 0xFF, the DCM module shall obtain the DTCSnapshotRecordNumberOfIdentifiers and the DTCSnapshotRecord by calling `Dem_GetFreezeFrameDataByDTC()` repeatedly (after every call of `Dem_GetDTCOfFreezeFrameRecord()`, see Dcm430) with the following parameter values:
DTC: DTC as defined in
DTCKind: DEM_DTC_KIND_ALL_DTCS
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY
RecordNumber: DTCSnapshotRecordNumber from the request

The function `Dem_GetFreezeFrameDataByDTC()` returns the DTCSnapshotRecord data as defined by UDS for the response message to Service 0x19 subfunction 0x05. This allows the DCM module to use the buffer directly in responding to the service.
The DCM module can calculate the size of data returned in `Dem_GetFreezeFrameDataByDTC()` using `Dem_GetSizeOfFreezeFrame()`.

Dcm432: To determine the "DTCSnapshotRecordNumberOfIdentifiers" UDS parameter, the DCM module shall call `Dem_GetFreezeFrameDataIdentifierByDTC()`.

7.4.2.5.11 UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber is not 0xFF (Report DTC Snapshot Record by Record Number)

UDS Service 0x19 with subfunction 0x05 allows an external tester to request FreezeFrame information for a specific FreezeFrame record number. The service request contains parameter:

- DTCSnapshotRecordNumber

Dcm391: When sending a positive response to UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber is not 0xFF, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCSnapshotRecordNumber	DTCSnapshotRecordNumber from request
DTCAndStatusRecord	DTC according to Dcm388, statusOfDTC according to Dcm389
DTCSnapshotRecordNumberOfIdentifiers	As defined in Dcm428
DTCSnapshotRecord	As defined in Dcm390
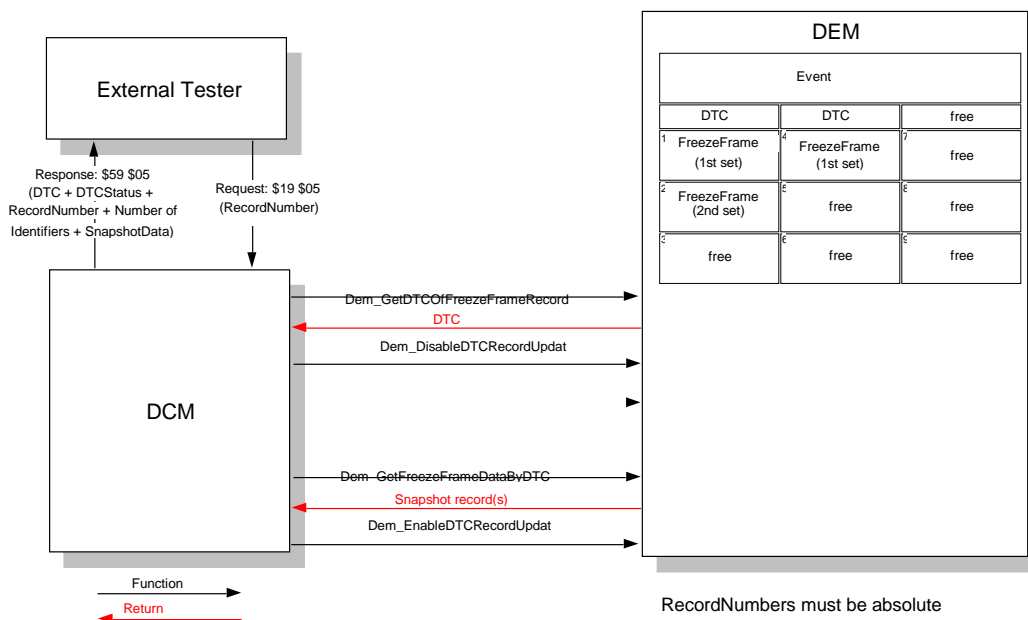
Figure 13: Request DTC Snapshot Record by Snapshot Record Number

Dcm388: When responding to UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber not 0xFF, the DCM module shall obtain the DTC by calling `Dem_GetDTCOfFreezeFrameRecord()` with the following parameter values:
RecordNumber: DTCSnapshotRecordNumber from the request
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY
DTCKind: DEM_DTC_KIND_ALL_DTCS

Dcm389: When responding to UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber not 0xFF, the DCM module shall obtain the status of the DTC by calling Dem_GetStatusOfDTC() with the following parameters:
DTC: DTC as defined in Dcm388
DTCKind: DEM_DTC_KIND_ALL_DTCS
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY

Dcm390: When responding to UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber not 0xFF, the DCM module shall obtain the DTCSnapshotRecordNumberOfIdentifiers and the DTCSnapshotRecord by calling Dem_GetFreezeFrameDataByDTC() with the following parameter values:
DTC: DTC as defined in Dcm388
DTCKind: DEM_DTC_KIND_ALL_DTCS
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY
RecordNumber: DTCSnapshotRecordNumber from the request

The function `Dem_GetFreezeFrameDataByDTC()` returns the DTCSnapshotRecord data as defined by UDS for the response message to Service 0x19 subfunction 0x05. This allows the DCM module to use the buffer directly in responding to the service.

The DCM module can calculate the size of data returned in `Dem_GetFreezeFrameDataByDTC()` using `Dem_GetSizeOfFreezeFrame()`.

Dcm428: To determine the "DTCSnapshotRecordNumberOfIdentifiers" UDS parameter, the DCM module shall call `Dem_GetFreezeFrameDataIdentifierByDTC()`.

7.4.2.5.12 UDS Service 0x19 with subfunctions 0x0B(reportFirstTestFailedDTC), 0x0C(reportFirstConfirmedDTC), 0x0D(reportMostRecentTestFailedDTC), 0x0E(reportMostRecentConfirmedDTC)

An external test tool may request an ECU to report DTCs and the associated status, matching a by the tester defined occurrence criterion, by sending a UDS Service request 0x19 including one of the following subfunctions 0x0B, 0x0C, 0x0D, 0x0E.

Dcm392: When sending a positive response to UDS Service 0x19 with subfunction 0x0B, 0x0C, 0x0D or 0x0E, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see Dcm007)
DTCAndStatusRecord	The DTC is obtained according to Dcm466, the StatusOfDtc is obtained according to Dcm393

Dcm393: For the purpose of responding to UDS Service 0x19 with subfunctions 0x0B, 0x0C, 0x0D or 0x0E, the DCM module shall obtain the StatusOfDtc by calling `Dem_GetStatusOfDTC()` with the following parameter values:
DTC: the DTC value as defined in Dcm466
DTCKind: DEM_DTC_KIND_ALL_DTCS
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY

Dcm466: For the purpose of responding to UDS Service 0x19 with subfunctions 0x0B, 0x0C, 0x0D or 0x0E, the DCM shall obtain the DTC with `Dem_GetDTCByOccurrenceTime()` using the parameter values of the following table:

	reportFirstTestFailedDTC	reportFirstConfirmedDTC	reportMostRecentTestFailedDTC	reportMostRecentConfirmedDTC
	0x0B	0x0C	0x0D	0x0E
DTCRequest	FIRST_ FAILED_DTC	FIRST_ DET_CONFIRME D_DTC	MOST_RECEN T_ FAILED_DTC	MOST_REC_ DET_CONFIRME D_DTC
DTCKind	ALL_DTCS	ALL_DTCS	ALL_DTCS	ALL_DTCS

7.4.2.5.13 UDS Service 0x19 with subfunction 0x14(reportDTCFaultDetectionCounter)

An external test tool may request an ECU to report the FaultDetectionCounter for all DTCs with a "Prefailed" status, by sending a UDS Service request 0x19 with subfunction 0x14.

Dcm464: When sending a positive response to UDS Service 0x19 with subfunction 0x14, the DCM module shall use the following data in the response message:

Parameter name	Value
DTC	The DTC is obtained according from the call to `Dem_GetNextFilteredDTCAndFDC()`
DTCFaultDetectionCounter	The DTCFaultDetectionCounter is obtained according from the call to `Dem_GetNextFilteredDTCAndFDC()`

Dcm465: When responding to UDS Service 0x19 with subfunctions 0x14, the DCM module shall obtain the DTCFaultCounter of every DTCs with status "prefailed" by repeatedly calling `Dem_GetNextFilteredDTCAndFDC()` after having configured the filter with `Dem_SetDTCFilter()` using the parameter values of the following table:

DTCStatusMask	0x00
DTCKind	ALL_DTCS
DTCOrigin	PRIMARY_MEMORY
FilterWithSeverity	NO
DTCSeverity	Not relevant
FilterForFaultDetection	YES

Dcm681: The DCM module shall obtain the number of records that will be found using `Dem_GetNextFilteredDTCAndFDC()` by using `Dem_GetNumberOfFilteredDTC()`.

This allows the implementation to calculate the total size of the response before cycling through the DTCs.

7.4.2.6 UDS Service 0x22 - ReadDataByIdentifier

Dcm253: The DCM module shall implement the UDS Service ReadDataByIdentifier (0x22)
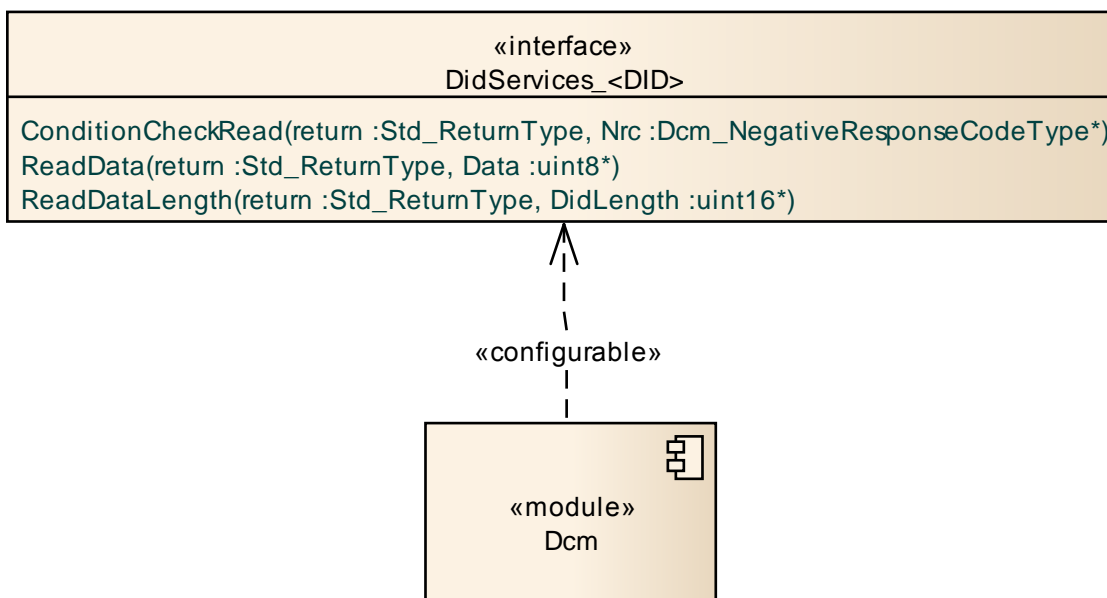
Figure 14: DCM module interfaces used for UDS Service 0x22

With UDS Service 0x22, the tester can request the value of one or more DIDs.

Dcm707: If a DID is set as unused (***DcmDspDidUsed*** set to FALSE), the DCM shall consider the DID as not supported (according to **Dcm438**).**Dcm438**: On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID the DCM module shall check if the DID is supported (see configuration parameter ***DcmDspDid***) If not, the DCM module shall send NRC 0x31 (Request out of range).

Dcm433: On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID the DCM module shall check if the DID has a Read access configured (see configuration parameter ***DcmDspDidRead*** in ***DcmDspDidAccess***). If not, the DCM module shall send NRC 0x31 (Request out of range).

Dcm434: On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID the DCM module shall check if the DID can be read in the current session (see configuration parameter ***DcmDspDidReadSessionRef***). If not, the DCM module shall send a NRC 0x7F (Service not supported in active session).

Dcm435:On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID the DCM module shall check if the DID can be read in the current security level (see configuration parameter ***DcmDspDidReadSecurityLevelRef***). If not, the DCM module shall send NRC 0x33 (Security access denied).

Dcm439: On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID the DCM module shall request the application if the DID can be read by calling the configured function (see configuration parameter *DcmDspDidConditionCheckReadFnc* and *DcmDspDidUsePort*). If not, the DCM module shall send a negative response with NRC 0x22 (Conditions not correct).

Dcm436: On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID the DCM module shall check if the DID has a fixed data length (see configuration parameter *DcmDspDidFixedLength*): if not (Parameter set to False) the DCM module shall call the configured function (see configuration parameter *DcmDspDidReadDataLengthFnc* and *DcmDspDidUsePort*) to get the DID length.

Dcm437:After all verification (see Dcm433, Dcm434, Dcm435 and Dcm436) the DCM module shall get for every requested DID the DID value by calling the configured function (see configuration parameter *DcmDspDidReadFnc* and *DcmDspDidUsePort*)

Dcm440: If the requested DID references several other DID (see configuration parameter *DcmDspDidRef*), the DCM module shall process the verification and the reading of every referenced DID.

7.4.2.7 UDS Service 0x24 - ReadScalingDataByIdentifier

Dcm258: The DCM module shall implement the UDS Service ReadScalingDataByIdentifier (0x24)
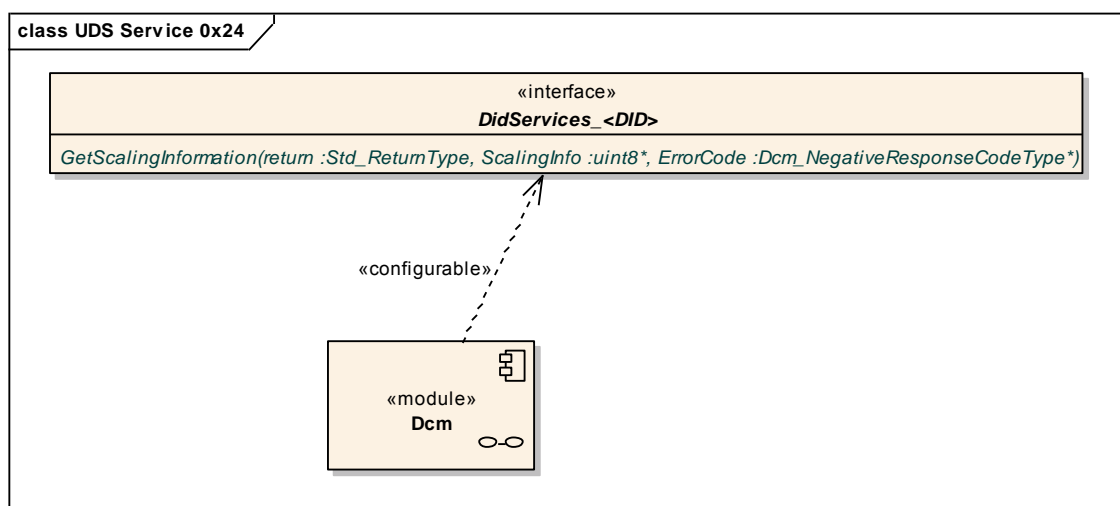
Figure 15: DCM module interfaces used for UDS Service 0x24

To obtain scaling information, the tester can invoke UDS Service 0x24 with the 2-byte DID as parameter.
The configuration of the DCM contains for each configured DID:

- The 2-byte DID (see configuration parameter ***DcmDspDidIdentifier***)
- The function GetScalingInformation (see configuration parameters ***DcmDspDidGetScalingInfoFnc*** and ***DcmDspDidUsePort***)
- The length of the ScalingInfo returned by the GetScalingInformation function (see configuration parameter ***DcmDspDidScalingInfoSize***)

Dcm394: On reception of a request for UDS Service ReadScalingByIdentifier, the DCM module shall call the function `Xxx_GetScalingInformation()` configured for the DID received in the request and return the data received in the response

7.4.2.8 UDS Service 0x27 - SecurityAccess

Dcm252: The DCM module shall implement the UDS Service SecurityAccess (0x27)

The purpose of this service is to provide a means to access data and/or diagnostic services, which have restricted access for security, emissions, or safety reasons.
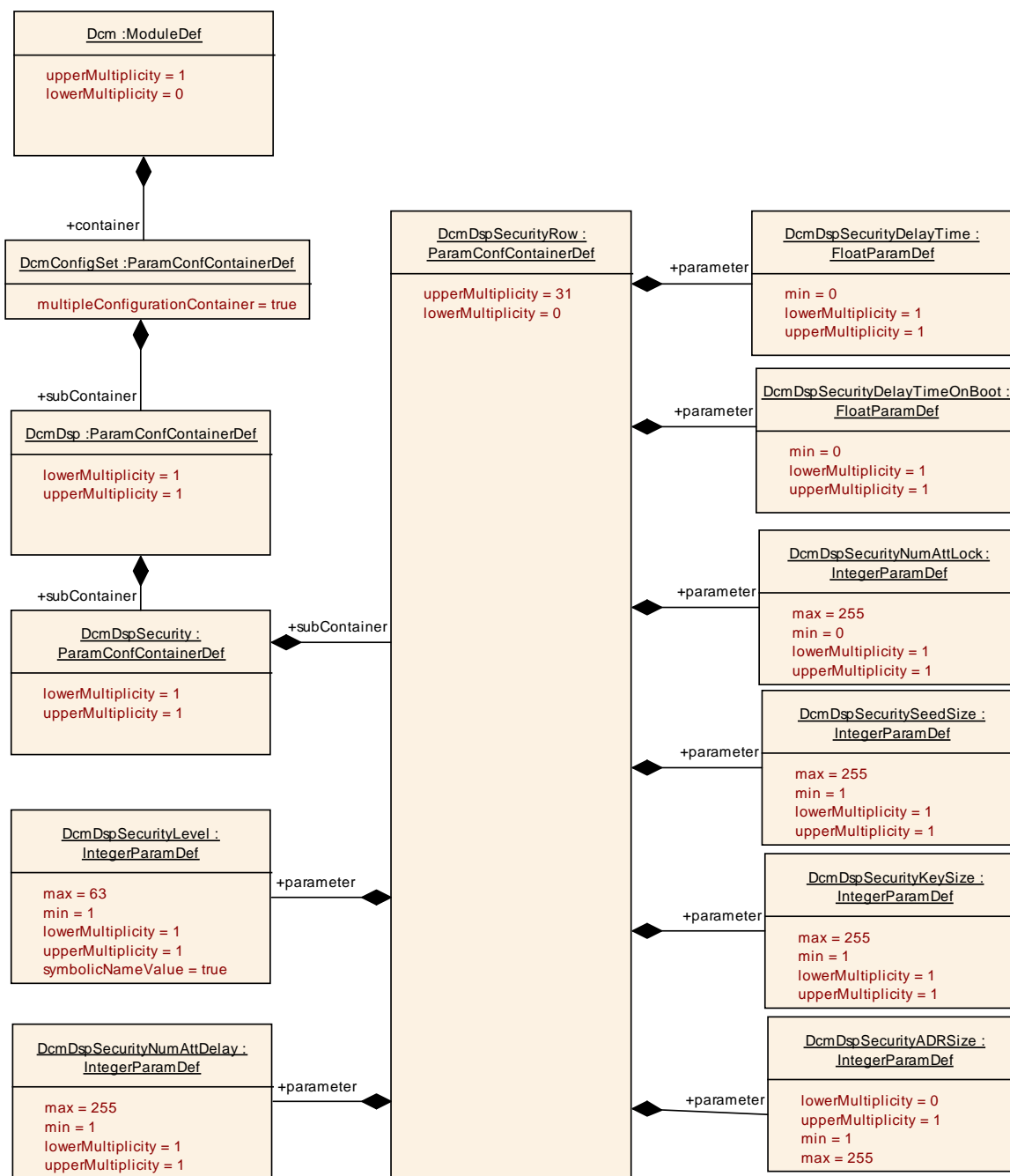
Figure 16: Configuration related to UDS Service 0x27

Dcm321: If the request length is correct, the DSP submodule shall check if the requested subfunction value (access type) is configured in the ECU (see configuration parameter *DcmDspSecurityLevel*). If the requested subfunction value is not configured, the DSP submodule shall trigger a negative response with NRC 0x12 (SubFunction not supported).

Dcm323: If the requested subfunction value is configured and a service with subfunction type request seed (= odd value) has been received and if the requested access type is already active (see `Dcm_GetSecurityLevel()`), the DSP submodule shall set the seed content to 0x00.

Dcm324: In the other case than the one described in Dcm323 (access type is not active or "send key" request), the DSP submodule shall call the configured operation `Xxx_GetSeed()` (in case "request seed" is received) or `Xxx_CompareKey()` (in case "send key" is received).

```
«interface»
SecurityAccess_<LEVEL>
CompareKey(return :Std_ReturnType, Key :uint8*)
GetSeed(return :Std_ReturnType, SecurityAccessRecord :uint8*, Seed :uint8*, ErrorCode :Dcm_NegativeResponseCodeType*)

              ▲
              ¦
        «configurable»

        «module»
          Dcm
```
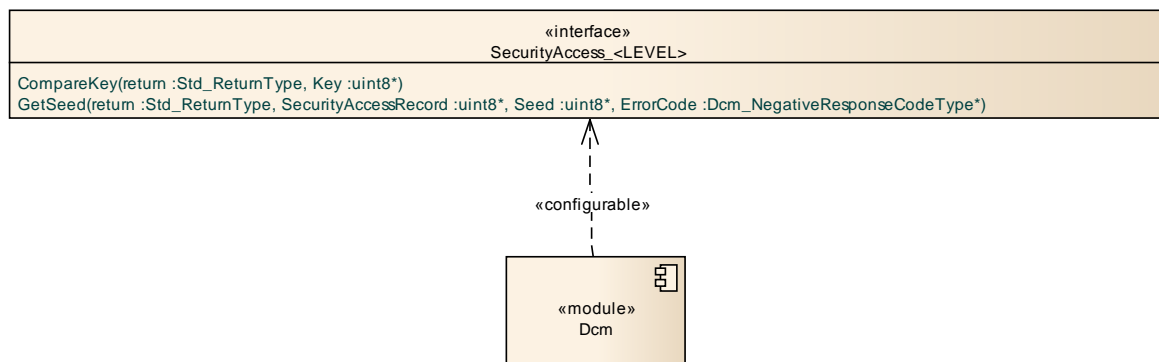
Figure 17: Dcm interfaces used for UDS Service 0x27

The following list gives as an example, which errors can be detected by the security access service and stored in the error code information:

- RequestSequenceError (NRC 0x24), when invalid access type is send at "send key",
- RequiredTimeDelayNotExpired (NRC 0x37), when time delay is active (see configuration parameter ***DcmDspSecurityDelayTime***),
- ExeedNumberOfAttempts (NRC 0x36), when number of attempts to get security access exceeds (see configuration parameter ***DcmDspSecurityNumAttLock***), and
- InvalidKey (NRC 0x35), when invalid key is send at "send key".

Dcm325: If no errors are detected, the DSP submodule shall set the new access type with `DslInternal_SetSecurityLevel()`.

7.4.2.9 UDS Service 0x28 – CommunicationControl

In order to implement the CommunicationControl service, the DCM use the interfaces provided by the BswM module, for normal communication messages control and for network management communication messages control.

Dcm511: The DCM module shall implement the CommunicationControl (service 0x28) of the Unified Diagnostic Services.

Dcm512: On reception of the UDS Service CommunicationControl (0x28), the DCM shall call the Api BswM_Dcm_RequestCommunicationMode(NetworkHandleType Network, Dcm_CommunicationModeType RequestedCommunicationMode)
 with:
- **Network**:
    o if the subnet parameter is equal to 0x00 (CommunicationControl on all connected networks) the DCM shall call the BswM in a row for all configured **Networks** (see *DcmDspComControlAllChannel*).
    o if the subnet parameter is between 0x01 and 0x0E: depending on the subnetNumber parameter from the CommunicationControl request the mapped **Network** shall be used (see *DcmDspComControlSpecificChannel*).
    o If the subnet parameter is equal to 0x0F (CommunicationControl on the network which request is received on) the **Network** is set with the value associated to the received PduId (see *DcmDslProtocolRxComMChannelRef*)
- **RequestedMode:** the communicationTypes (bit 0-1 in communicationType byte) and the controlTypes (SubFunction 0x00 to 0x03) from the CommunicationControl request shall be combined to the Dcm_CommunicationModeType definition.

Dcm701: When the DCM requests a communication mode to BswM by calling BswM_Dcm_RequestCommunicationMode, the BswM shall not reject this request because the communication control state will be managed by the DCM.

Dcm591: After calling of BswM_Dcm_RequestCommunicationMode() (see Dcm512) the DCM shall send a positive response.

Note: The BswM will not reject this mode request by definition.

7.4.2.10   UDS Service 0x2A - ReadDataByPeriodicIdentifier

Dcm254: The DSP submodule shall implement the UDS Service ReadDataByPeriodicIdentifier (0x2A)

7.4.2.11   UDS Service 0x2C - DynamicallyDefineDataIdentifier

Dcm259: The DSP submodule shall implement the DynamicallyDefineDataIdentifier (service 0x2C, diagnostic data access) of the Unified Diagnostic Services.

7.4.2.12   UDS Service 0x2E - WriteDataByIdentifier

Dcm255: The DCM module shall implement the UDS Service WriteDataByIdentifier (0x2E) of the Unified Diagnostic Services.

When using Service 0x2E, the request of the tester contains a 2-byte DID and a dataRecord with the data to be written.

«interface»
DidServices_<DID>

«function»
ConditionCheckWrite(return :Std_ReturnType, Nrc :Dcm_NegativeResponseCodeType*)
WriteData(return :Std_ReturnType, Data :uint8*, DataLength :uint16, ErrorCode :Dcm_NegativeResponseCodeType*)

«configurable»

«module»
Dcm

---

uint8

«typedef»
Dcm_Types::Dcm_NegativeResponseCodeType

DCM_E_BRAKESWITCH_NOTCLOSED :uint8 = 0x8F
DCM_E_BUSYREPEATREQUEST :uint8 = 0x21
DCM_E_CONDITIONSNOTCORRECT :uint8 = 0x22
DCM_E_ENGINEISNOTRUNNING :uint8 = 0x84
DCM_E_ENGINEISRUNNING :uint8 = 0x83
DCM_E_ENGINERUNTIMETOOLOW :uint8 = 0x85
DCM_E_GENERALPROGRAMMINGFAILURE :uint8 = 0x72
DCM_E_GENERALREJECT :uint8 = 0x10
DCM_E_REQUESTOUTOFRANGE :uint8 = 0x31
DCM_E_REQUESTSEQUENCEERROR :uint8 = 0x24
DCM_E_RPMTOOHIGH :uint8 = 0x81
DCM_E_RPMTOOLOW :uint8 = 0x82
DCM_E_SECURITYACCESSDENIED :uint8 = 0x33
DCM_E_SHIFTERLEVERNOTINPARK :uint8 = 0x90
DCM_E_SUBFUNCTIONNOTSUPPORTEDINACTIVESESSION :uint8 = 0x7E
DCM_E_TEMPERATURETOOHIGH :uint8 = 0x86
DCM_E_TEMPERATURETOOLOW :uint8 = 0x87
DCM_E_THROTTLE_PEDALTOOHIGH :uint8 = 0x8A
DCM_E_THROTTLE_PEDALTOOLOW :uint8 = 0x8B
DCM_E_TORQUECONVERTERCLUTCHLOCKED :uint8 = 0x91
DCM_E_TRANSMISSIONRANGENOTINGEAR :uint8 = 0x8D
DCM_E_TRANSMISSIONRANGENOTINNEUTRAL :uint8 = 0x8C
DCM_E_VEHICLESPEEDTOOHIGH :uint8 = 0x88
DCM_E_VEHICLESPEEDTOOLOW :uint8 = 0x89
DCM_E_VOLTAGETOOHIGH :uint8 = 0x92
DCM_E_VOLTAGETOOLOW :uint8 = 0x93
range of values reserved by ISO 15764 = 0x38..0x4F

Figure 18: Dcm interfaces used for UDS Service 0x2E

The configuration of the DCM contains a list of supported DIDs and defines for each configured DID:

- The 2-byte DID (see configuration parameter ***DcmDspDidIdentifier***)
- The function WriteData to be used for this DID (see configuration parameters ***DcmDspDidWriteFnc*** and ***DcmDspDidUsePort***)
- The function ConditionCheckWrite to be used for this DID (see configuration parameters ***DcmDspDidConditionCheckWriteFnc*** and ***DcmDspDidUsePort***)

Dcm708: If a DID is set as unused (***DcmDspDidUsed*** set to FALSE), the DCM shall consider the DID as not supported (according to **Dcm467**).

Dcm467 On reception of the UDS Service WriteDataByIdentifier (0x2E), the DCM module shall check if the DID is supported (see configuration parameter ***DcmDspDid***) If not, the DCM module shall send NRC 0x31 (Request out of range).

Dcm468: On reception of the UDS Service WriteDataByIdentifier (0x2E), the DCM module shall check if the DID has a Write access configured (see configuration parameter ***DcmDspDidWrite*** in ***DcmDspDidAccess***). If not, the DCM module shall send NRC 0x31 (Request out of range).

Dcm469: On reception of the UDS Service WriteDataByIdentifier (0x2E), the DCM module shall check if the DID can be write in the current session (see configuration parameter ***DcmDspDidWriteSessionRef***). If not, the DCM module shall send a NRC 0x7F (Service not supported in active session).

Dcm470:On reception of the UDS Service WriteDataByIdentifier (0x2E), the DCM module shall check if the DID can be write in the current security level (see configuration parameter ***DcmDspDidWriteSecurityLevelRef***). If not, the DCM module shall send NRC 0x33 (Security access denied).

Dcm471: On reception of the UDS Service WriteDataByIdentifier (0x2E), the DCM module shall request the application if the DID can be write by calling the configured function (see configuration parameter ***DcmDspDidConditionCheckWriteFnc*** and ***DcmDspDidUsePort***). If not, the DCM module shall send a negative response with NRC 0x22 (Conditions not correct).

Dcm472:On reception of the UDS Service WriteDataByIdentifier (0x2E), the DCM module shall check if the DID has a fixed data length (see configuration parameter ***DcmDspDidFixedLength***): if not (Parameter set to False) the DCM module shall call the configured function (see configuration parameter ***DcmDspDidReadDataLengthFnc*** and ***DcmDspDidUsePort***) to get the DID length.

Dcm473: On reception of the UDS Service WriteDataByIdentifier (0x2E), the DCM module shall check if the received data length corresponds to the DID data length (get from Dcm472 or DcmDspDidSize)

Dcm395: After all verification (see Dcm467, Dcm468, Dcm469, Dcm470, Dcm473 and Dcm471) the DCM module shall write the DID value by calling the configured function (see configuration parameter **DcmDspDidWriteFnc** and **DcmDspDidUsePort**) with the following parameter values:
Data: the dataRecord form the request
DataLength: the number of bytes in the dataRecord

7.4.2.13 UDS Service 0x2F - InputOutputControlByIdentifier

Dcm256: The DCM module shall implement the UDS Service InputOutputControlByIdentifier (0x2F).
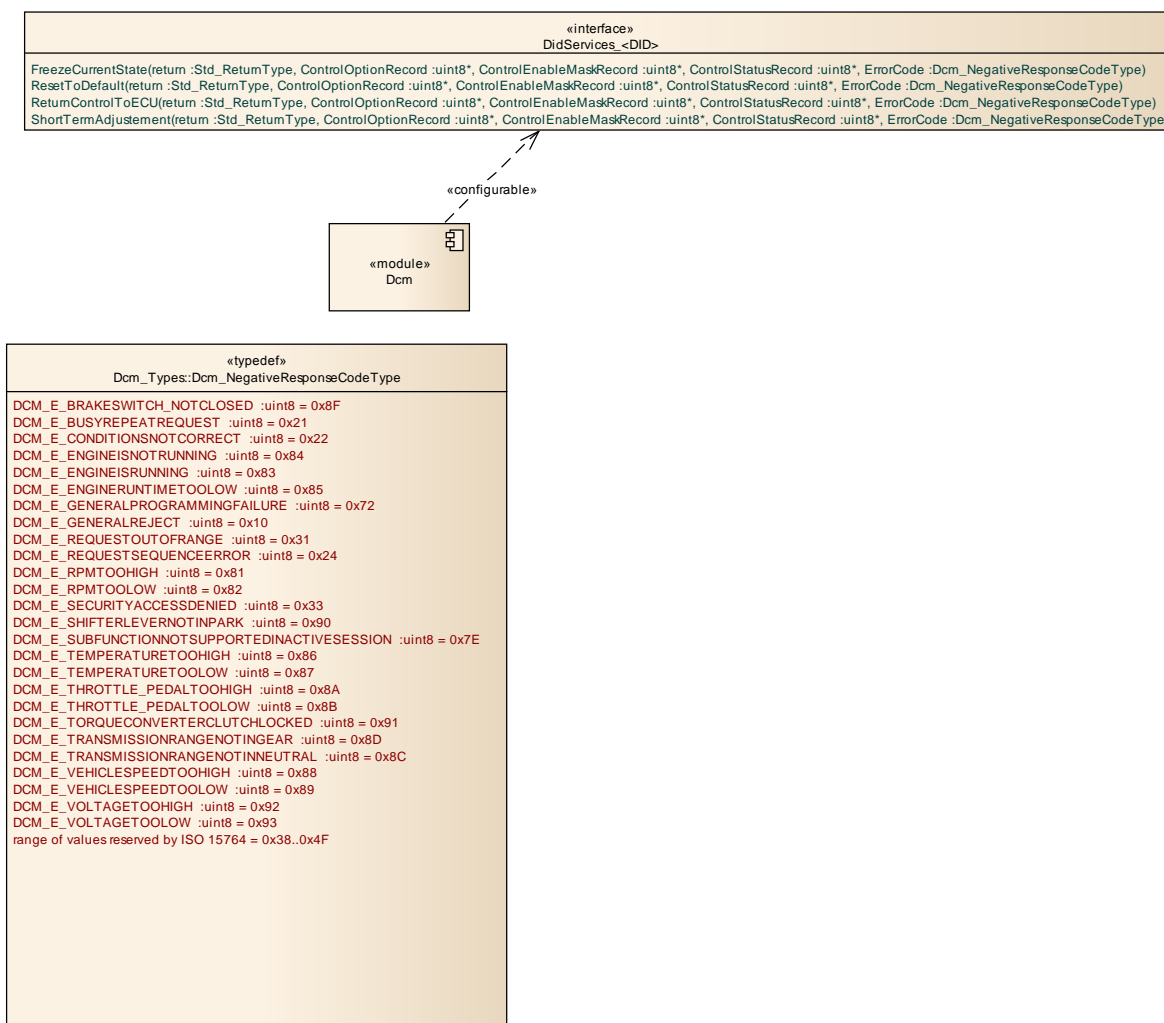
Figure 19: Dcm interfaces used for UDS Service 0x2F

When using Service 0x2F, the request of the tester contains a 2-byte DID.
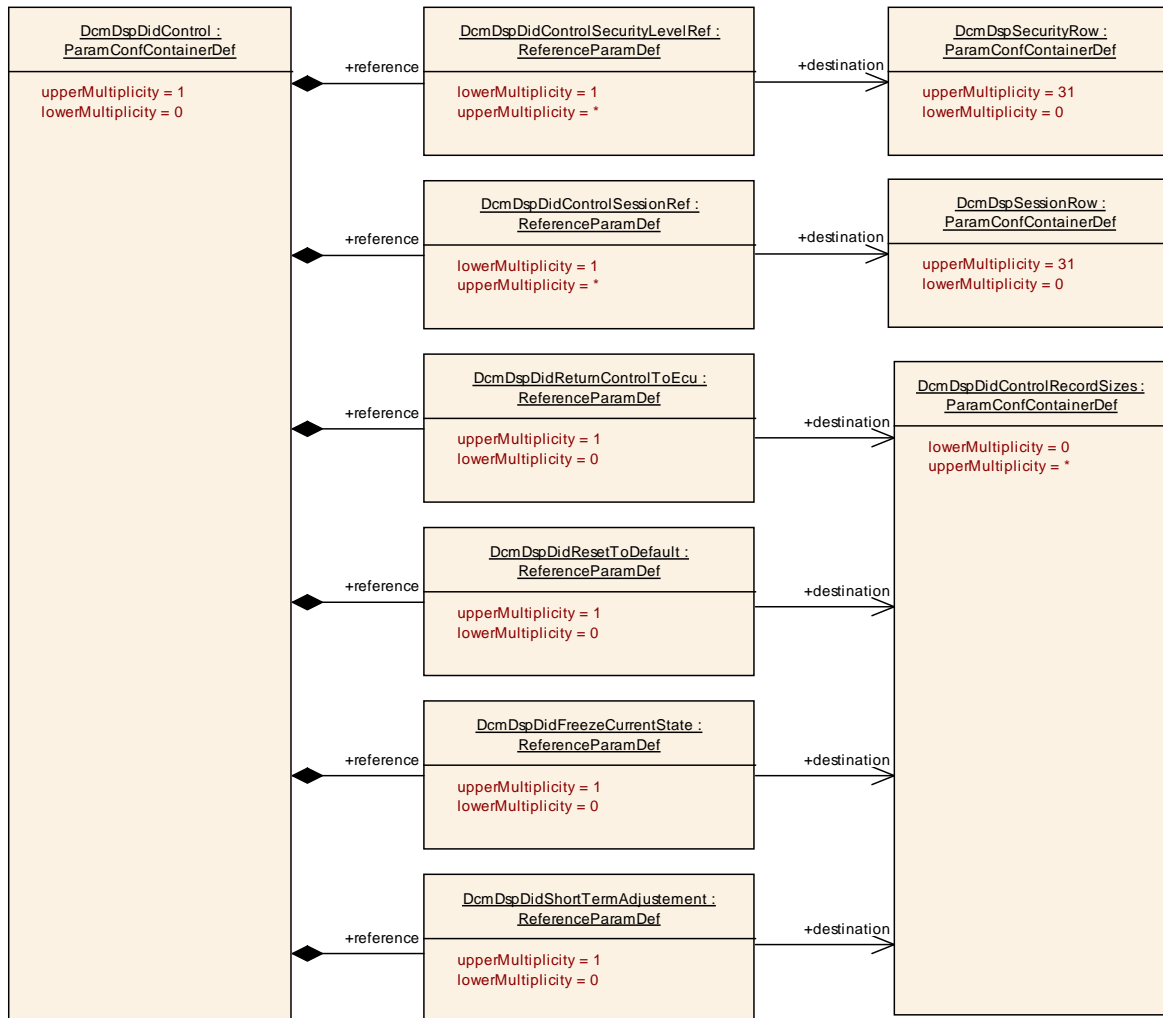
Figure 20: Dcm configuration related to UDS Service 0x2F

The configuration of the DCM contains a list of supported DID's. For each DID, the DCM configuration specifies:

- The 2-byte DID (see configuration parameter ***DcmDspDidIdentifier***)
- The function `Xxx_ReturnControlToECU()` for this DID (see configuration parameters ***DcmDspDidReturnControlToEcuFnc*** and ***DcmDspDidUsePort***)
- The sizes of the data-blocks used in the function `Xxx_ReturnControlToECU()` (see configuration parameter ***DcmDspDidReturnControlToEcu*** and ***DcmDspDidControlRecordSizes***)
- The function `Xxx_ResetToDefault()` for this DID (see configuration parameters ***DcmDspDidResetToDefaultFnc*** and ***DcmDspDidUsePort***)
- The sizes of the data-blocks used in the function `Xxx_ResetToDefault()` (see configuration parameter ***DcmDspDidResetToDefault*** and ***DcmDspDidControlRecordSizes***)

- The function `Xxx_FreezeCurrentState()` for this DID (see configuration parameters **DcmDspDidFreezeCurrentStateFnc** and **DcmDspDidUsePort**)
- The sizes of the data-blocks used in the function `Xxx_FreezeCurrentState()` (see configuration parameter **DcmDspDidFreezeCurrentState** and **DcmDspDidControlRecordSizes**)
- The function `Xxx_ShortTermAdjustment()` for this DID (see configuration parameters **DcmDspDidShortTermAdjustmentFnc** and **DcmDspDidUsePort**)
- The sizes of the data-blocks used in the function `Xxx_ShortTermAdjustment()` (see configuration parameter **DcmDspDidShortTermAdjustment** and **DcmDspDidControlRecordSizes**)

Dcm709: If a DID is set as unused (**DcmDspDidUsed** set to FALSE), the DCM shall consider the DID as not supported (according to **Dcm710**).

Dcm710: On reception of the UDS Service InputOutputControlByIdentifier (0x2F), the DCM module shall check if the DID is supported (see configuration parameter **DcmDspDid**) If not, the DCM module shall send NRC 0x31 (Request out of range).

Dcm396: On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to returnControlToEcu, the DCM module shall invoke the function `Xxx_ReturnControlToECU()` configured with the DID received in the request.

Dcm397: On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to resetToDefault, the DCM module shall invoke the function `Xxx_ResetToDefault()` configured with the DID received in the request.

Dcm398: On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to freezeCurrentState, the DCM module shall invoke the function `Xxx_FreezeCurrentState()` configured with the DID received in the request.

Dcm399: On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to shortTermAdjustment, the DCM module shall invoke the function `Xxx_ShortTermAdjustment()` configured with the DID received in the request.

7.4.2.14 UDS Service 0x31 - RoutineControl

Dcm257: The DCM module shall implement the UDS Service RoutineControl (0x31) for subFunctions startRoutine, stopRoutine and requestsRoutineResults.
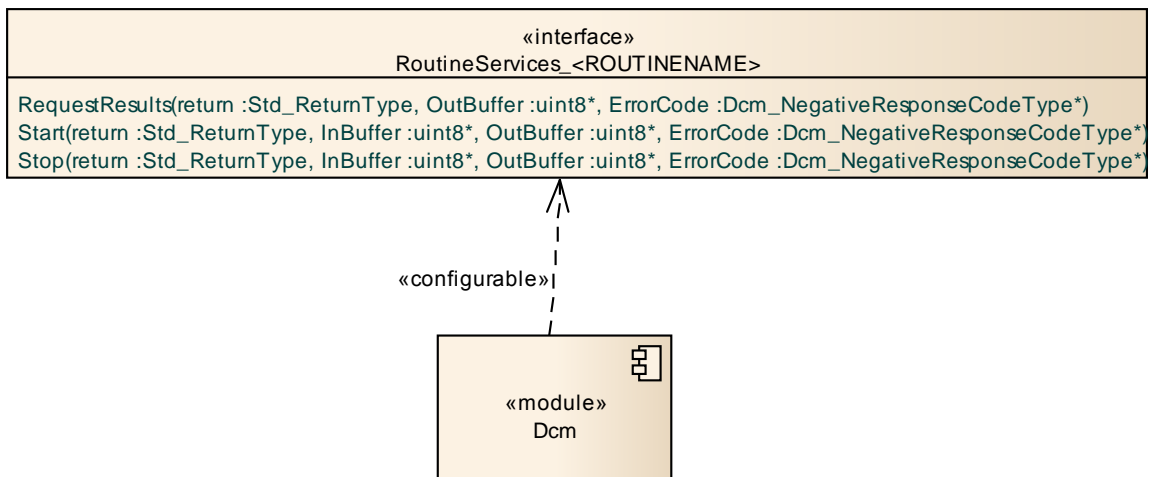
Figure 21: Dcm interfaces used for UDS Service 0x31

A tester can use UDS Service 0x31 to start, stop or obtain the results of a routine identified by a 2-byte routineIdentifier.

The DCM module configuration contains a list of the routineIdentifiers (see configuration parameter ***DcmDspRoutineIdentifier***) supported by the DCM. For each routineIdentifier, the DCM configuration specifies:

- The function `Xxx_Start()` associated with this routineIdentifier (see configuration parameters ***DcmDspStartRoutineFnc*** and ***DcmDspRoutineUsePort***)
- Number of bytes expected in the routineControlOptionRecord as part of the service request to start this routine (see configuration parameter ***DcmDspStartRoutine***)
- Number of bytes returned in the routineStatusRecord when responding to the request to start this routine
- The function `Xxx_Stop()` associated with this routineIdentifier (see configuration parameters ***DcmDspStopRoutineFnc*** and ***DcmDspRoutineUsePort***)
- Number of bytes expected in the routineControlOptionRecord as part of the service request to stop this routine (see configuration parameter ***DcmDspRoutineStop***)
- Number of bytes returned in the routineStatusRecord when responding to the request to stop this routine
- The function `Xxx_RequestResults()` associated with this routineIdentifier (see configuration parameters ***DcmDspRequestResultsRoutineFnc*** and ***DcmDspRoutineUsePort***)
- Number of bytes returned in the routineStatusRecord when responding to the requestResults subfunction (see configuration paramter ***DcmDspRoutineRequestRes***)

Dcm711: On reception of the UDS Service RoutineControl (0x31), the DCM module shall check if the Routine is supported (see configuration parameter *DcmDspRoutine*) If not, the DCM module shall send NRC 0x31 (Request out of range).

Dcm712: If a Routine is set as unused (*DcmDspRoutineUsed* set to FALSE), the DCM shall consider the Routine as not supported (according to **Dcm711**)

Dcm400: When receiving a request for UDS Service RoutineControl (0x31) with subfunction startRoutine, the DCM module shall call the configured `Xxx_Start()` function passing in an InBuffer and an OutBuffer of the configured size, where the InBuffer contains the routineControlOptionRecord received with the request.

Dcm401: Upon completing [Dcm400](#), when `Xxx_Start()` returns no ErrorCode, the DCM module shall reply with a positive response with the data returned by `Xxx_Start()` in the OutBuffer as routineStatusRecord.

Dcm402: When receiving a request for UDS Service RoutineControl (0x31) with subfunction stopRoutine, the DCM module shall call the configured `Xxx_Stop()` function passing in an InBuffer and an OutBuffer of the configured size, where the InBuffer contains the routineControlOptionRecord received with the request.

Dcm403: Upon completing [Dcm402](#), when `Xxx_Stop()` returns no ErrorCode, the DCM module shall reply with a positive response with the data returned by `Xxx_Stop()` in the OutBuffer as routineStatusRecord.

Dcm404: When receiving a request for UDS Service RoutineControl (0x31) with subfunction requestRoutineResults, the DCM module shall call the configured `Xxx_RequestResults()` function and provide a buffer of the configured size in OutBuffer.

Dcm405: Upon completing [Dcm404](#), when `Xxx_RequestResults()` returns no ErrorCode, the DCM module shall reply with a positive response with the data returned by `Xxx_RequestResults()` in OutBuffer as routineStatusRecord.

7.4.2.15   UDS Service 0x3E - Tester Present

Dcm251: The DCM module shall implement the Tester Present (service 0x3E, diagnostic communication and security) of the Unified Diagnostic Services for the subfunction values 0x00 and 0x80.

This service is used to keep one or multiple servers in a diagnostic session being different than the defaultSession.

7.4.2.16 UDS Service 0x85 - ControlDTCSetting

Dcm249: The DCM module shall implement UDS Service ControlDTCSetting (0x85) for DTCSettingType on or off.

An external test tool can request an ECU to either disable or enable DTC storage in the ECUs error memory by sending a UDS Service 0x85 request with subfunction on or off.
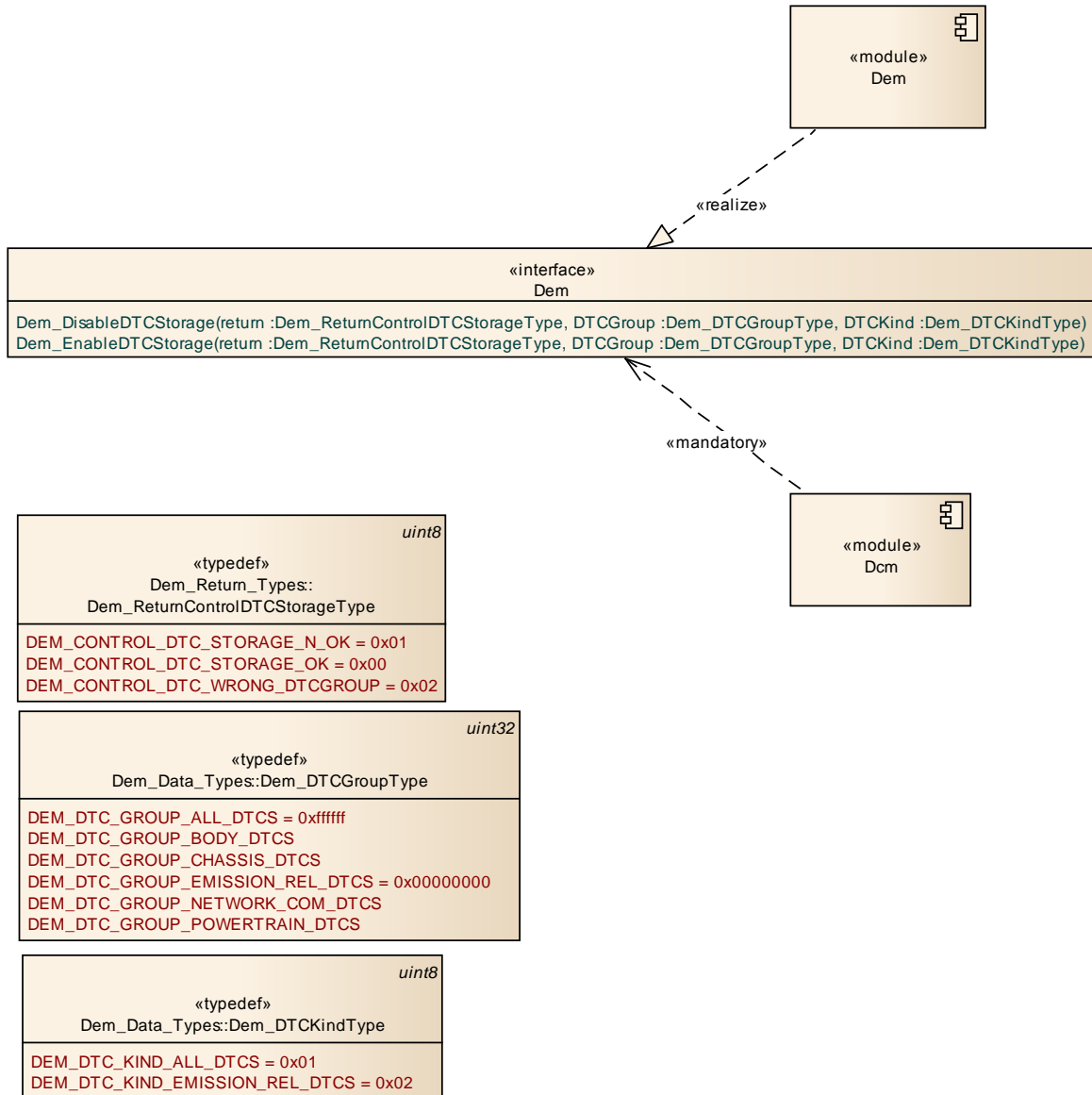
Figure 22: Dcm interfaces used for UDS Service 0x85

Dcm304: Upon receiving a request for UDS Service 0x85 with DTCSettingType=on, the DCM module shall call `Dem_EnableDTCStorage()`.

Dcm406: Upon receiving a request for UDS Service 0x85 with DTCSettingType=off, the DCM module shall call `Dem_DisableDTCStorage()`.

7.4.3 OBD Services

7.4.3.1 Overview
The following table defines the OBD Services supported by the DCM.

Relevant OBD Service Identifier	Support in the DCM
$01	Supported
$02	Supported
$03	Supported
$04	Supported
$06	Supported
$07	Supported
$08	Supported
$09	Supported
$0A	Supported

Table 5: Support for OBD services in the DCM

The following diagram shows the interfaces that the DCM module uses to provide the OBD services.
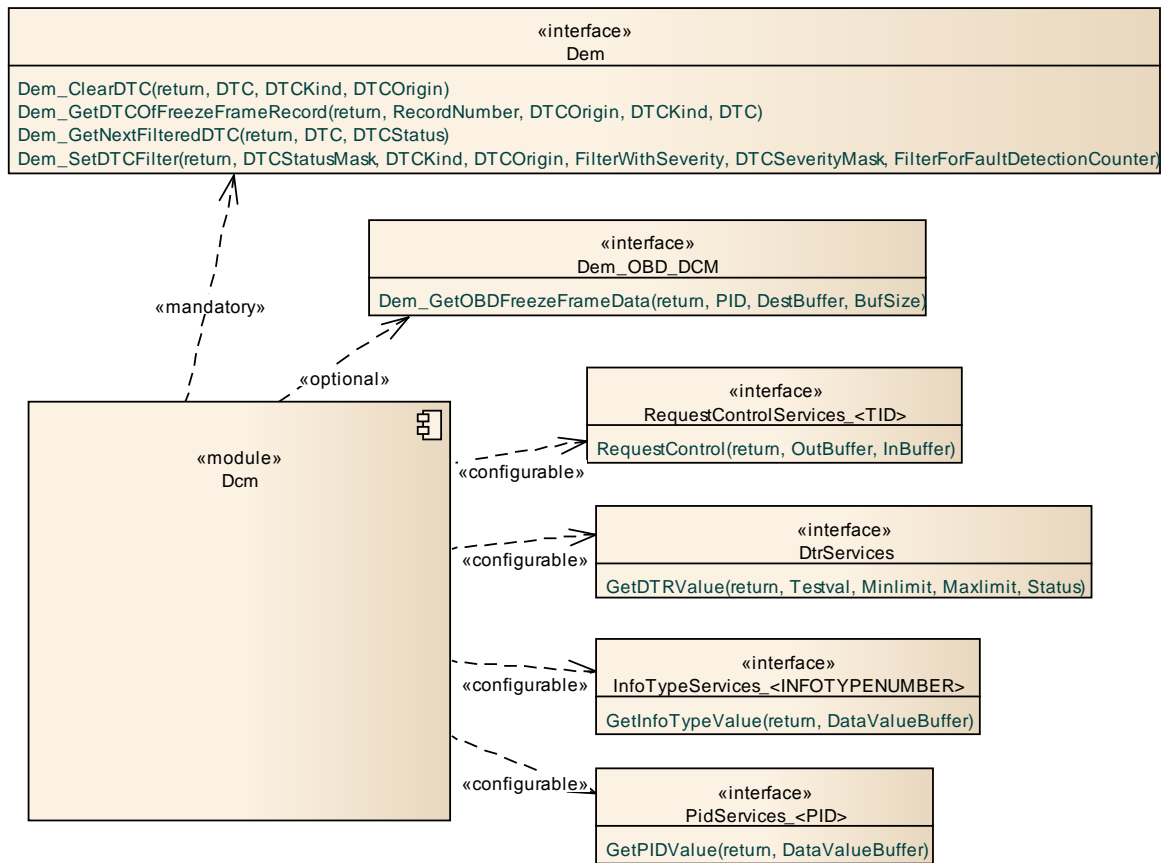
Figure 23: Overview of the functions used by the DCM to provide OBD Services.

7.4.3.2 General behavior

In many cases, the DCM protocol allows the bundling of several requests (for example several "PIDs") and the corresponding bundling of the responses. The descriptions of the behavior for the individual services do not explicitly consider this. As the DCM needs to comply with OBD standard (as is defined through various requirements below), the DCM might need to repeat the steps defined below to parse a request and assemble a valid response.

Dcm077: When calling the DEM module for OBD services, the DCM module shall use the following values for the parameter DTCOrigin:
Service $0A uses DEM_DTC_ORIGIN_PERMANENT_MEMORY
All other services use DEM_DTC_ORIGIN_PRIMARY_MEMORY

7.4.3.3 OBD Service $01 – Request Current Powertrain diagnostic Data

Dcm243: The DCM module shall implement the OBD service $01 (Request Current Powertrain diagnostic Data) in compliance to all provisions of the OBD standard.

Using Service $01, an external test tool can request an emission-related ECU to return PID-values or to return the supported PIDs. OBD reserves certain PIDs for the special purpose of obtaining the list of available PIDs in a certain range. These

PIDs are called "availability PIDs" and are $00, $20, $40, $60, $80, $A0, $C0 an $E0.

Figure 24: Extract of the configuration for OBD Service $01

The DCM configuration defines the PIDs that are available on the ECU. The DCM configuration defines for each such PID:

- Indication of the PID is used or not (for postbuild configuration) (see configuration parameter **DcmDspPidUsed**)
- The length of the data associated with the PID (see configuration parameter **DcmDspPidSize**)
- The `Xxx_GetPIDValue()` function that the DCM must call to obtain the current value of the PID or the name of the port that the DCM uses to obtain

the current value through the RTE from a SW-C (see configuration parameters **DcmDspGetPidValFnc** and **DcmDspUsePort**)

Dcm407: On reception of an OBD Service $01 request with one or more "availability PIDs" as parameter, the DCM shall respond with the corresponding supported (=configured) PIDs encoded according to the OBD standard.

To obtain the value for a PID, the DCM uses the configured `Xxx_GetPIDValue()` function.

```
«interface»
PidServices_<PID>
─────────────────────────────────────────────
GetPIDValue(return :Std_ReturnType, DataValueBuffer :uint8*)
```

```
            △
            ┆
            ┆
      «configurable»
            ┆
```

```
«module»
Dcm
```

Figure 25: To provide OBD Service $01, the DCM relies on external functions that allow it to obtain the value of the PIDs. There is one such function per PID that is needed by the DCM.

When using a `Xxx_GetPIDValue()` function, the DCM provides a buffer of the correct length, which is filled by the function with the PID value.

Dcm408: On reception of an OBD Service $01 request with one or more PIDs that are not "availability PIDs", the DCM shall obtain the current value of these PIDs by invoking the configured `Xxx_GetPIDValue()` functions and shall return these values as response to Service $01.

The entity providing the actual implementation of the `Xxx_GetPIDValue()` function for a specific PID might be a SW-C or another basic software module. The origin of the function is not known to the DCM but is part of the DCM configuration. Some PIDs are provided by the DEM. These PIDs are also explicitly configured in the DCM configuration and it is the responsibility of a correct DCM configuration to make the `Xxx_GetPIDValue()` function point to the correct function provided by the DEM.
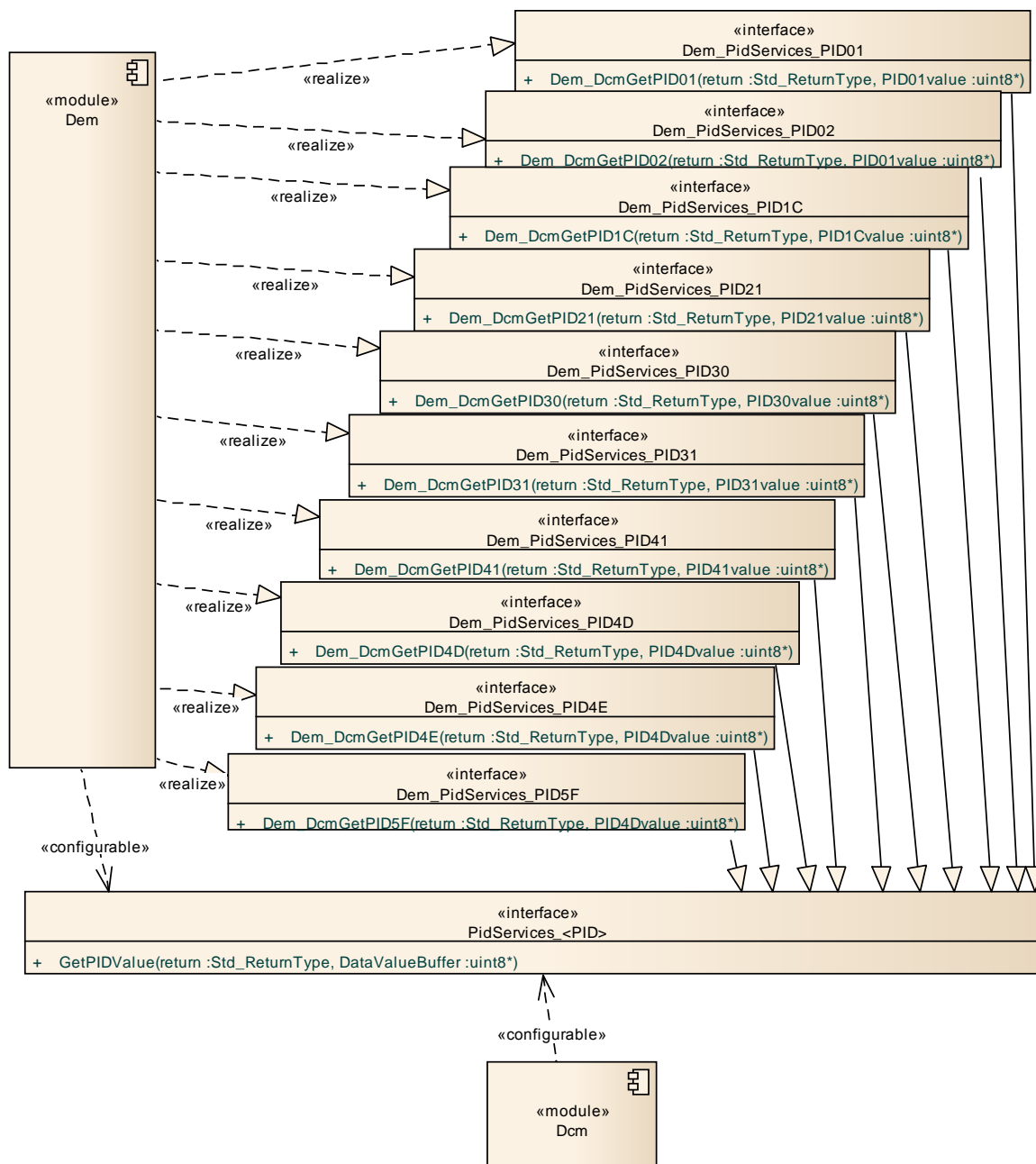
Figure 26: For certain PIDs, the DEM provides the function to obtain the PID value. Which PIDs come from the DEM are part of the DCM configuration.

7.4.3.4 OBD Service $02 – Request Power Train FreezeFrame Data

Dcm244: The DCM shall implement OBD Service $02 (Request Power Train FreezeFrame Data) in compliance to all provisions of the OBD standard.

For OBD-relevant FreezeFrames AUTOSAR only supports frame 0, which is the minimum required by legislation.

Dcm409: The DCM shall ignore all requests regarding record-numbers that are not 0
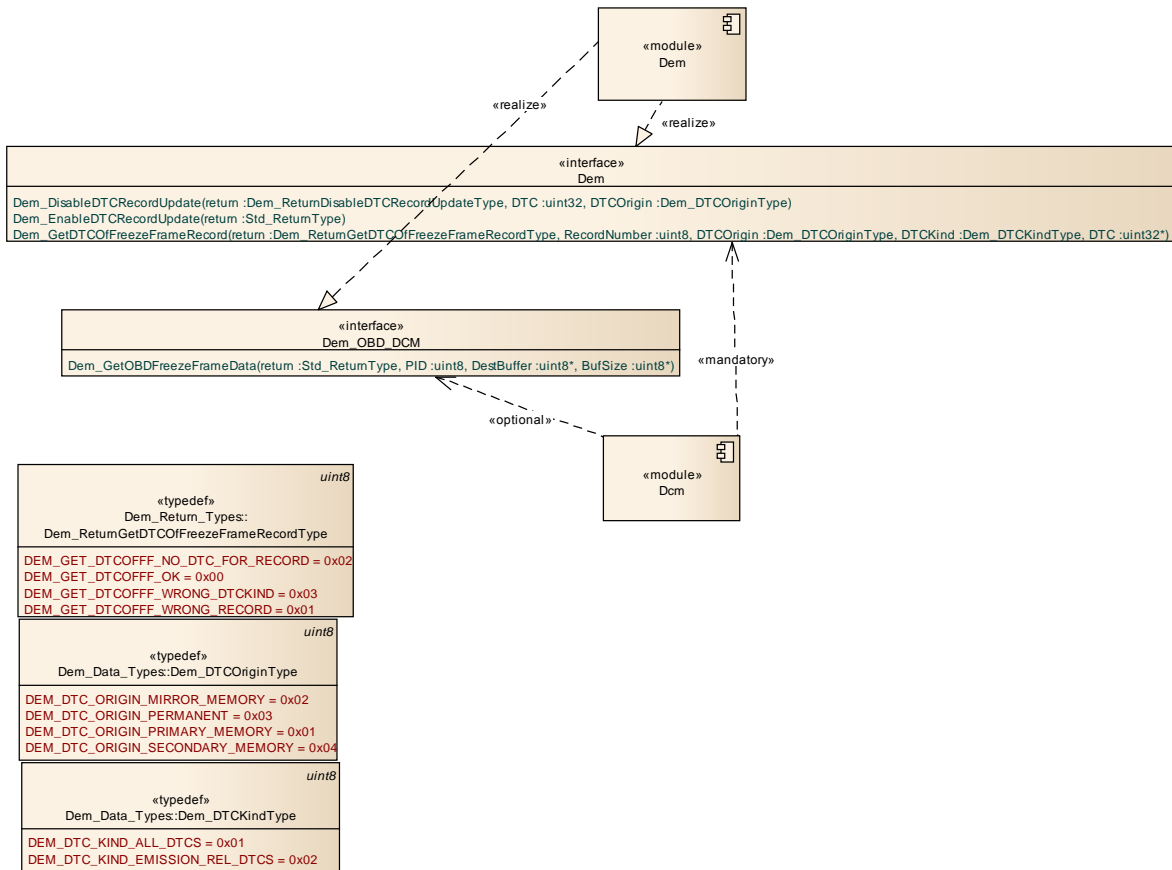
Figure 27: Interaction between DCM and DEM for OBD Service $02

The following sections define how specific PIDs are handled by the DCM.

7.4.3.4.1 OBD Service $02 - PID$02 - Request for the DTC of a specific FreezeFrame

An external tester can request the DTC that caused a FreezeFrame to be stored by using the Service $02 with the PID value $02.

Dcm279: On reception of a request for Service $02 with PID $02, the DCM shall call `Dem_GetDTCOfFreezeFrameRecord()` with the following parameter values:
RecordNumber = frame-number received in the OBD request
DTCOrigin = DEM_DTC_ORIGIN_PRIMARY_MEMORY
DTCKind = DEM_DTC_KIND_EMISSION_REL_DTCS

The DEM module returns the corresponding DTC. Note that this 2-byte DTC is packed into the 4-byte data returned by the `Dem_GetDTCOfFreezeFrameRecord()`. See DEM specification on how this is done.

Dcm280: The DCM shall unpack the DTC from the return-value of the `Dem_GetDTCOfFreezeFrameRecord()` and return the response to the service request.

7.4.3.4.2 OBD Service $02 – availability PID - Request supported PIDs of a particular FreezeFrame

Using Service $02, an external tester may request the supported PIDs for a specific freeze-frame by using the "availability PIDs".

Dcm284: On reception of a service $02 request with an "availability PID", the DCM shall call `Dem_GetOBDFreezeFrameData()` with the following parameter values:
PID = the "availability PID" received in the OBD request
DestBuffer = a buffer in which the callee can write 4 bytes
BufSize = 4

The DEM module will fill the provided buffer with the 4 bytes that encode which PIDs are available, according to the conventions of OBD.

Dcm278: After execution of [Dcm284](#), the DCM shall respond to the service using the buffer filled by `Dem_GetOBDFreezeFrameData()` as the 4 data-bytes containing the supported PIDs.

7.4.3.4.3 OBD Service $02 – other PIDs - Request the data records assigned to a specific PID included in a specific FreezeFrame

Using Service $02, an external tester may request the values of specific PIDs in specific FreezeFrames.

Dcm286: On reception of a service $02 request with a PID that is not an "availability PID" and is not $02, the DCM shall call `Dem_GetOBDFreezeFrameData()` with the following parameter values:
PID = the PID received in the OBD request
DestBuffer = a buffer in which the callee can write the value of the PID
BufSize = the size of the DestBuffer, this must be at least equal to the size needed to store the value of the PID as configured in the DCM

Note that is not necessary for the DCM module to lock or unlock the record updates of the DEM module.

Dcm287: Upon the completion of [Dcm286](#), the DCM shall generate a response message including the respective PID, FreezeFrame Number and the associated data record for the requested FreezeFrame number.

7.4.3.5 OBD Service $03 / $07 / $0A – Obtaining DTCs

Dcm245: The DCM module shall implement OBD Service $03 (Request emission-related diagnostic trouble codes) in compliance to all provisions of the OBD standard.

Dcm410: The DCM module shall implement OBD Service $07 (Request Emission-Related Diagnostic Trouble Codes Detected during Current or Last Completed Driving Cycle) in compliance to all provisions of the OBD standard.

Dcm411: The DCM module shall implement OBD Service $0A (Request Emission-Related Diagnostic Trouble Codes with Permanent Status) in compliance to all provisions of the OBD standard.

An external test tool can request an emission-related ECU to report all stored, pending or permanent emission-related DTCs by sending the request $03, $07, $0A respectively.
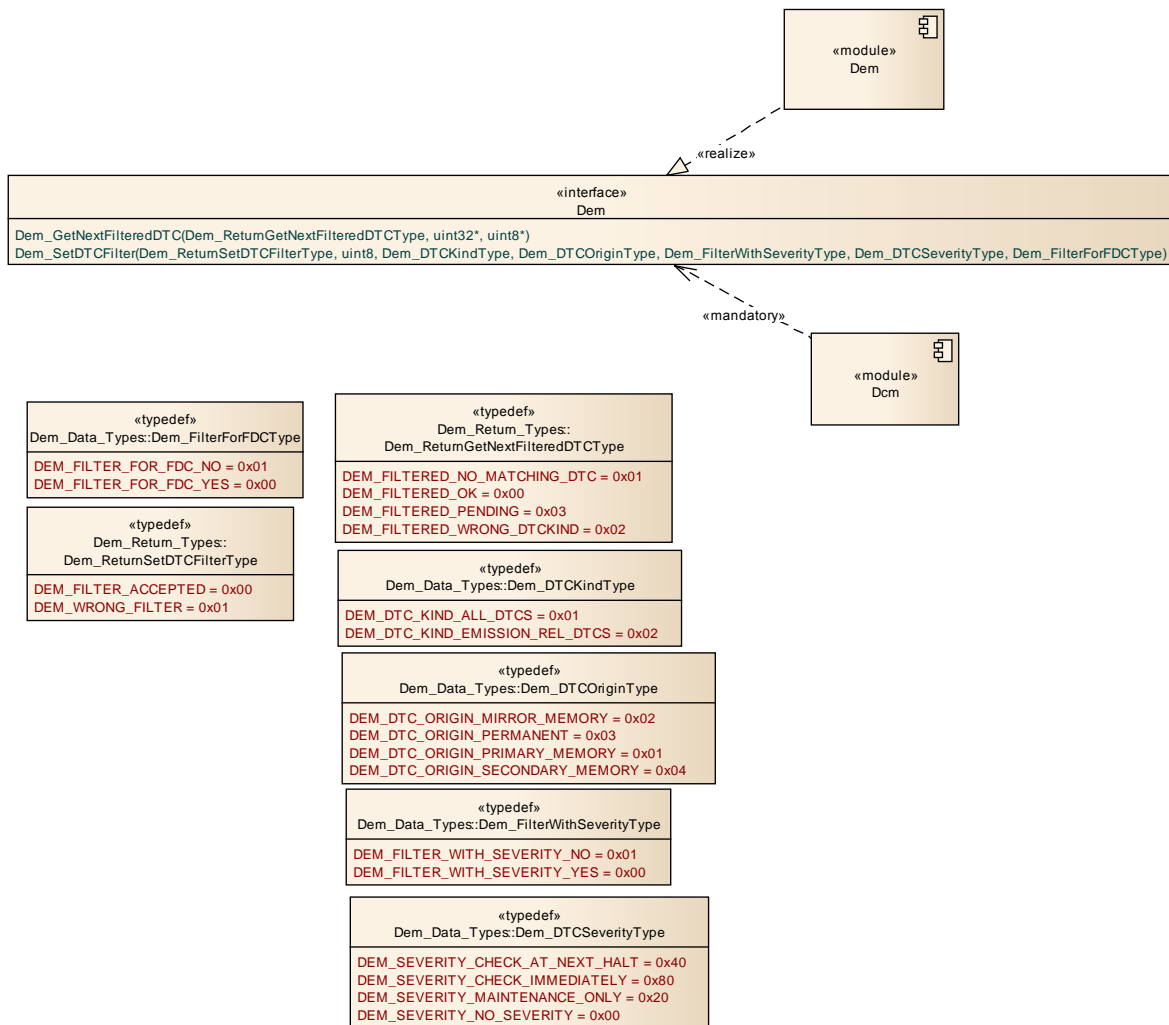
Figure 28: Interaction between DCM and DEM for OBD Services $03, $07 and $0A

Dcm289: When receiving a request for OBD Service $03, the DCM module shall obtain from the DEM all DTCs in primary memory and with a "confirmed" status using the functions `Dem_SetDTCFilter()` and `Dem_GetNextFilteredDTC()`.

Dcm412: When receiving a request for OBD Service $07, the DCM module shall obtain from the DEM module all DTCs in primary memory with a "pending" status using the functions `Dem_SetDTCFilter()` and `Dem_GetNextFilteredDTC()`.

Dcm330: When receiving a request for OBD Service $0A, the DCM module shall obtain from the DEM all DTCs stored in permanent memory using the functions `Dem_SetDTCFilter()` and `Dem_GetNextFilteredDTC()`.

The following table illustrates the parameters the DCM module must use when calling `Dem_SetDTCFilter()` in response to a request for OBD Service $03, $07 or $0A.

Parameters to Dem_SetDTCFilter			
OBD Service	$03	$07	$0A
DTCStatusMask	0x08 (confirmed bit set)	0x04 (pending bit set)	0x00
DTCKind	DEM_DTC_KIND_ EMISSION_REL_DTCS	DEM_DTC_KIND_ EMISSION_REL_DTCS	DEM_DTC_KIND_ EMISSION_REL_DTCS
DTCOrigin	DEM_DTC_ORIGIN_ MEMORY	DEM_DTC_ORIGIN_ PRIMARY_MEMORY	DEM_DTC_ORIGIN_ PERMANENT
FilterWithSeverity	DEM_FILTER_WITH_ SEVERITY_NO	DEM_FILTER_WITH_ SEVERITY_NO	DEM_FILTER_WITH_ SEVERITY_NO
DTCSeverity	Not relevant	Not relevant	Not relevant
FilterForFault DetectionCounter	DEM_FILTER_ FOR_FDC_NO	DEM_FILTER_ FOR_FDC_NO	DEM_FILTER_ FOR_FDC_NO

7.4.3.6 OBD Service $04 – Clear/reset emission-related diagnostic information

Dcm246: The DCM module shall implement OBD Service $04 (Clear/reset emission-related diagnostic information) in compliance to all provisions of the OBD standard.

An external test tool can request an emission–related ECU to clear the error memory by sending the request $04.
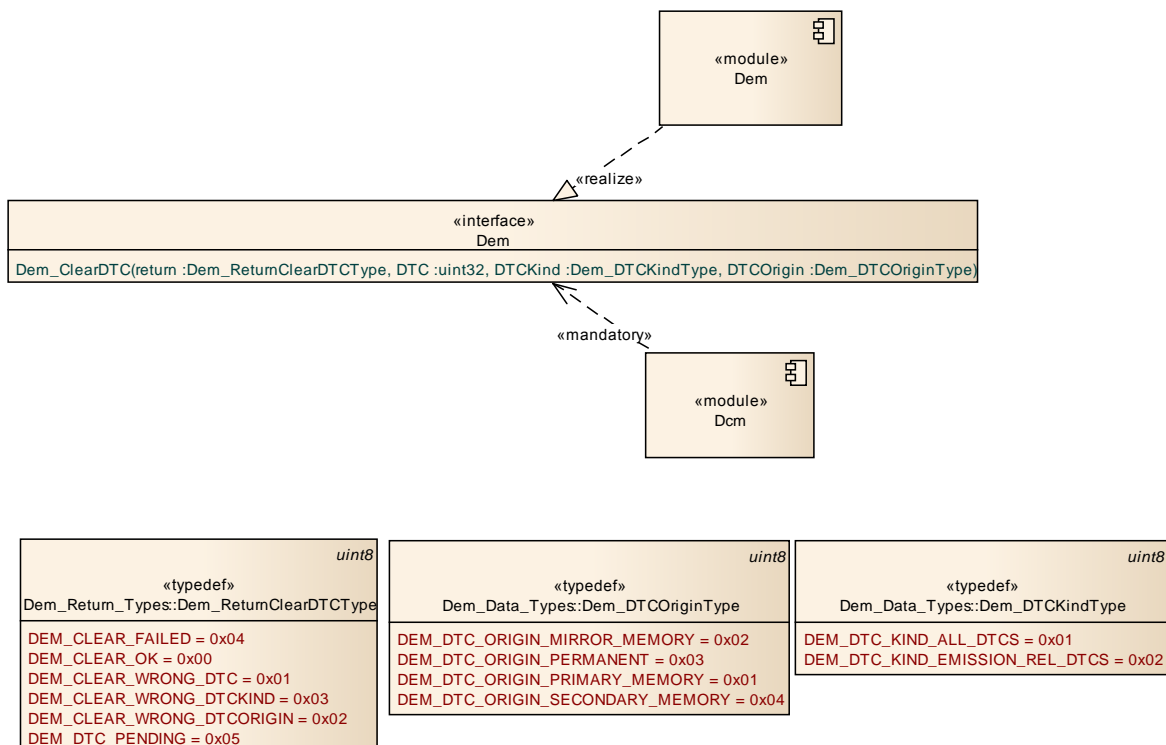
Figure 29: OBD Service $04

Dcm004: When receiving a request for OBD Service $04, the DCM module shall call `Dem_ClearDTC()` with the following parameter values:
DTC = ALL_DTCS (0xffffff)
DTCKind = DCM_OBD_SERVICE04_KIND
DTCOrigin = DEM_DTC_ORIGIN_PRIMARY_MEMORY

The configuration parameter **DcmObdService04Kind** makes it possible to configure the DCM module so that the OBD Service $04 clears either all DTCs (**DcmObdService04Kind**=DEM_DTC_KIND_ALL_DTCS) or only the OBD-relevant DTCs (**DcmObdService04Kind**=DEM_DTC_KIND_EMISSION_REL_DTCS).

Dcm413: The Dcm module shall only respond to the OBD Service $04 after `Dem_ClearDTC()` has been effectively executed

7.4.3.7 OBD Service $06 – Request On-Board Monitoring Test-results for Specific Monitored Systems

Dcm414: The DCM module shall implement OBD Service $06 (Request On-Board Monitoring Test-results for Specific Monitored Systems) in compliance to all provisions of the OBD standard.

Using Service $06, an external test tool can request an emission-related ECU to return the DTR's associated with the OBDMID or to return the supported OBDMIDs.

OBD reserves certain OBDMIDs for the special purpose of obtaining the list of supported OBDMIDs in a certain range. These OBDMIDs are called "availability OBDMIDs" and are $00, $20, $40, $60, $80, $A0, $C0 an $E0.
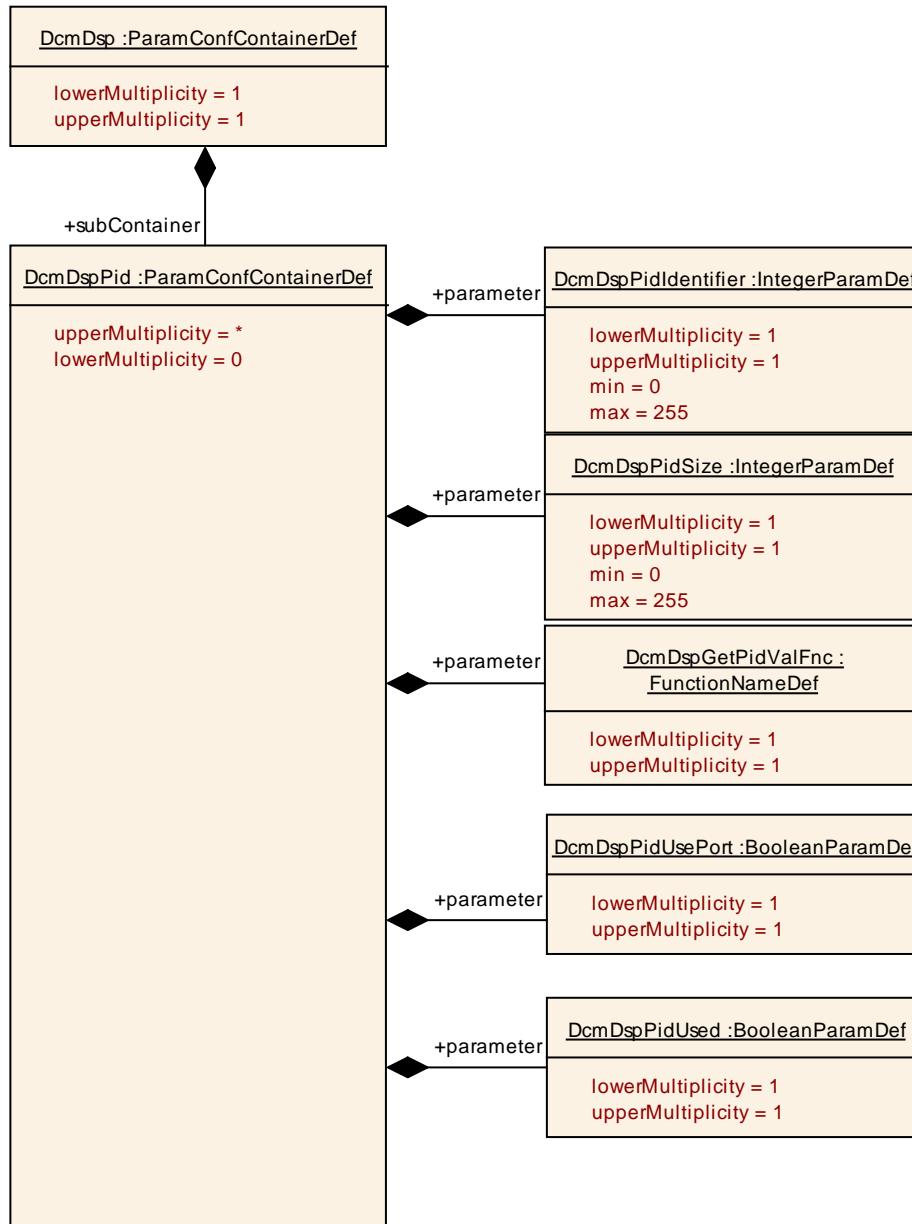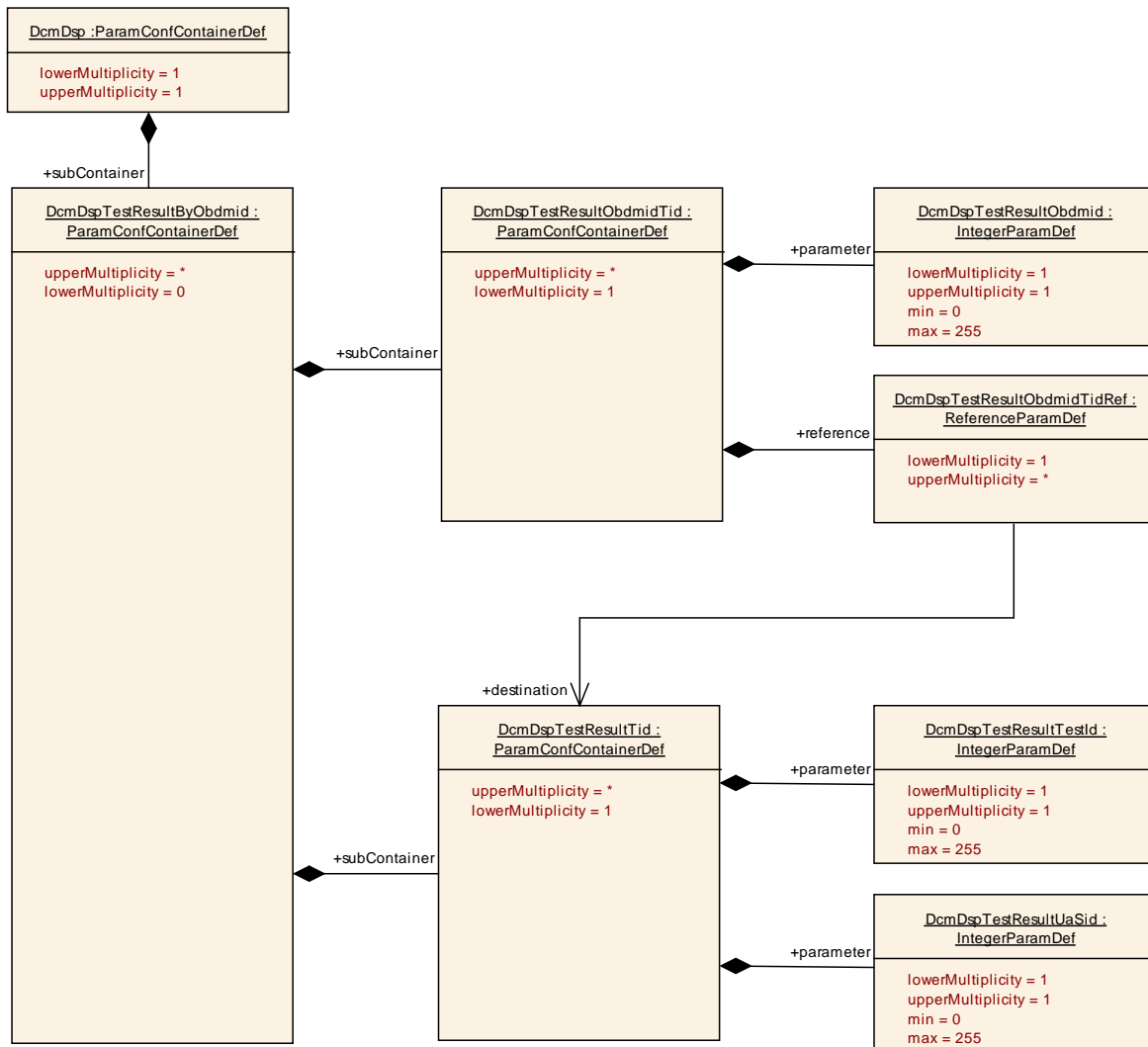

Figure 30: Extract from the DCM configuration related to OBD Service $06.

The DCM module's configuration defines the OBDMIDs that are available to the DCM. For each OBDMID, the configuration defines (see configuration parameter ***DcmDspTestResultObdmid***):

- A list of TIDs (see configuration parameter ***DcmDspTestResultObdmidTidRef***)
- And for each TID:
  - The name of the port through which the DCM module has access to a DTRService interface (see configuration parameter ***DcmDspTestResultTid***)

      o   The Unit and Scaling ID to be returned with the service for the given OBDMID and TID (see configuration parameter ***DcmDspTestResultUaSid***)

Dcm415: On reception of an OBD Service $06 request with "availability OBDMIDs" as parameter, the DCM module shall respond with the corresponding supported (=configured) OBDMIDs.
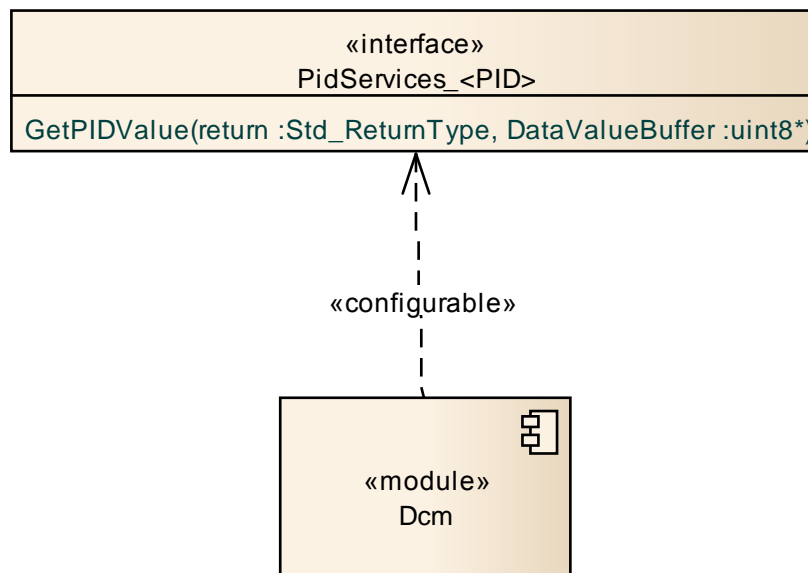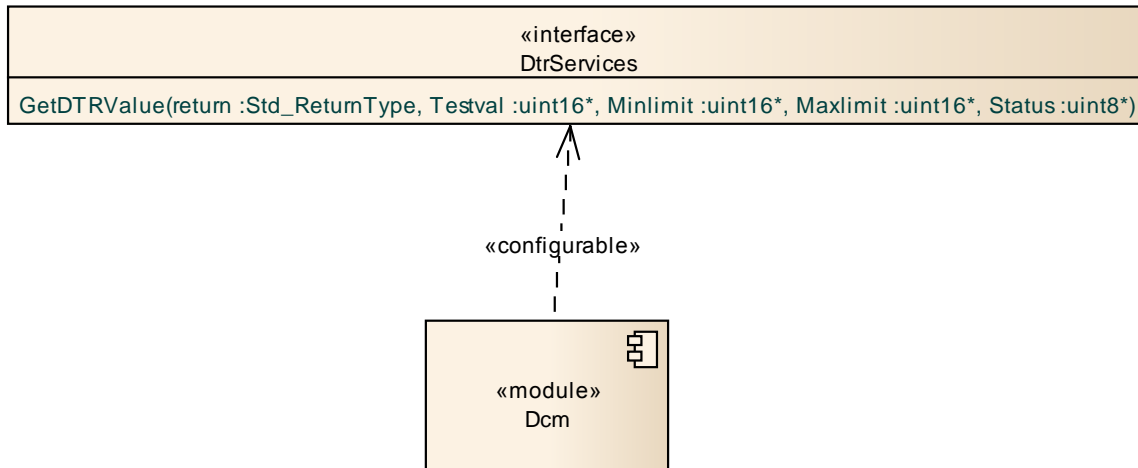

Figure 31: To provide OBD Service $06, the DCM relies on external functions that allow it to obtain the DTR of an OBDMID. There is one such function per OBDMID that is needed by the DCM.

Dcm416: On reception of an OBD Service $06 request with an OBDMID that is not an "availability OBDMID", the DCM module shall run through the list of TIDs configured for the OBDMID, shall call the configured `Xxx_GetDTRValue()` function. In case the returned Status==DCM_DTRSTATUS_VISIBLE, the DCM module shall assemble the response to the service call as follows:
Std./Manuf. Defined TID: TID from the configuration
Unit And Scaling ID: this is configured for the OBDMID/TID combination
Test Value (High/Low Byte): as returned from the `Xxx_GetDTRValue()` function called for the OBDMID/TID combination
Min. Test Limit (High/Low Byte): as returned from the `Xxx_GetDTRValue()` function called for the OBDMID/TID combination
Max. Test Limit (High/Low Byte): as returned from the `Xxx_GetDTRValue()` function called for the OBDMID/TID combination

[Dcm416](#) implies that DTRs for which the function `Xxx_GetDTRValue()` returns a status of DCM_DTRSTATUS_INVISIBLE are ignored.
When using the `Xxx_GetDTRValue()` functions, the DCM module provides buffers in which the DTR can be stored. The entity providing the actual implementation of the `Xxx_GetDTRValue()` function for a specific OBDMID might be a SW-C or

another basic software module. The origin of the function is not known to the DCM but is part of the DCM configuration.

7.4.3.8 OBD Service $08 - Request Control of On-Board System, Test or Component

Dcm417: The DCM module shall implement OBD Service $08 (Request Control of On-Board System, Test or Component) in compliance to all provisions of the OBD standard.

Using Service $08, an external test tool can control an on-board system, test or component using a TID. OBD reserves certain TIDs for the special purpose of obtaining the list of supported TIDs in a certain range. These TIDs are called "availability TIDs" and are $00, $20, $40, $60, $80, $A0, $C0 an $E0.
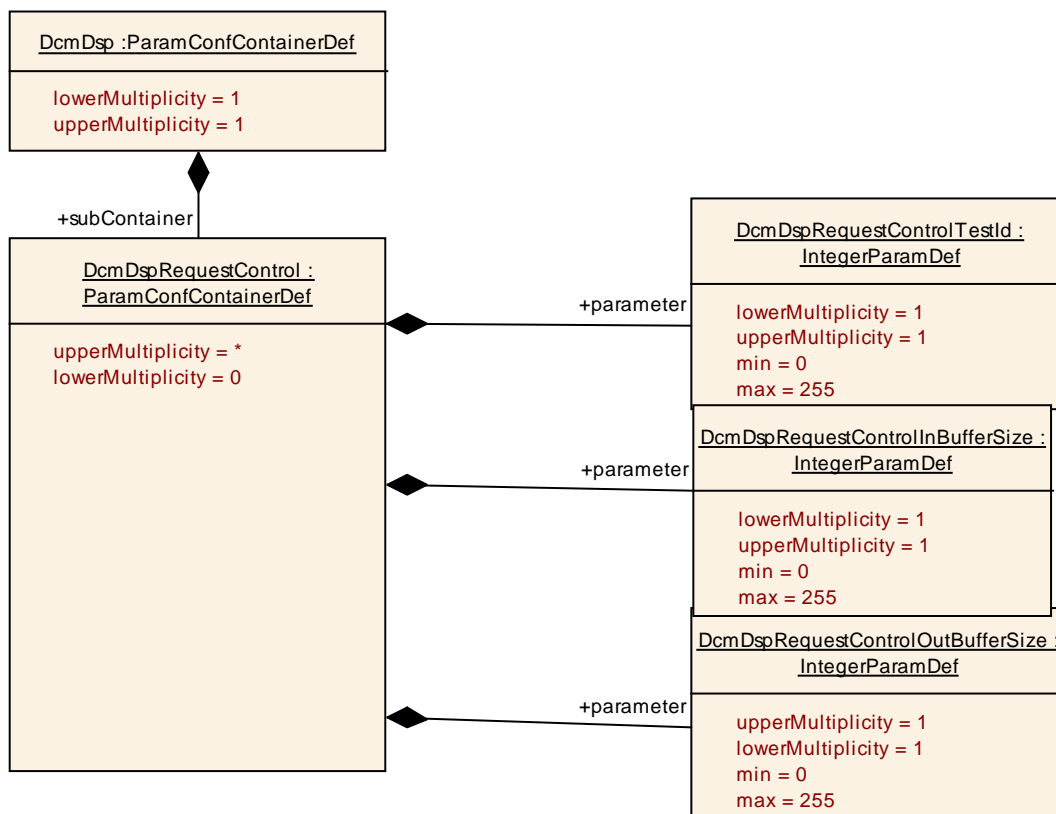
Figure 32: Extract from the DCM configuration related to OBD Service $08.

The DCM module's configuration defines the TIDs that are available on the ECU for the purpose of OBD Service $08. The configuration defines for each such TID (see configuration parameter *DcmDspRequestControlTestId*):
- the name of the port the DCM uses to access the RequestControlServices interface (see configuration parameter *DcmDspRequestControl*)
- the number of bytes this function takes as input (see configuration parameter *DcmDspRequestControlInBufferSize*)

- the number of bytes this function writes as output (see configuration parameter **DcmDspRequestControlOutBufferSize**)
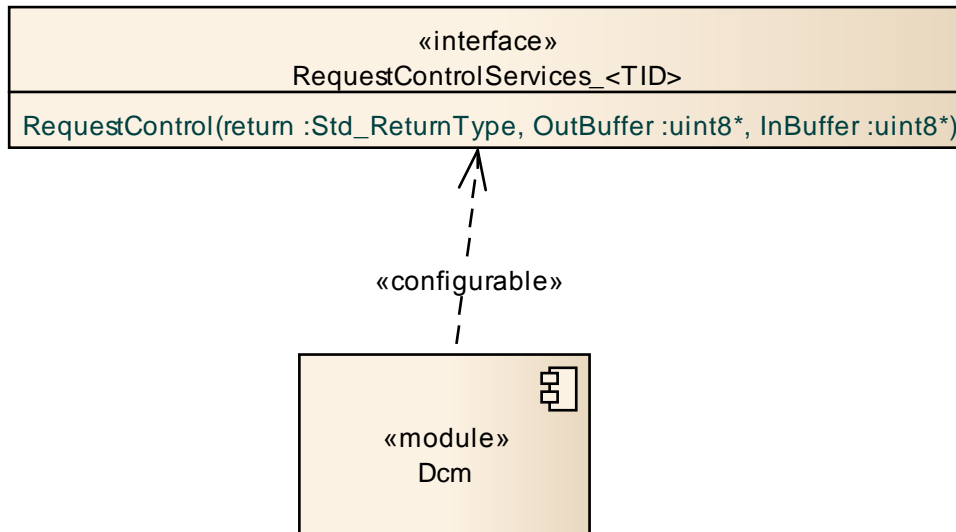
Figure 33: To provide OBD Service $08, the DCM relies on external functions configured per TID

Dcm418: On reception of an OBD Service $08 request with one or more "availability TIDs" as parameter, the DCM module shall respond with the corresponding supported (=configured) TIDs.

Dcm419: On reception of an OBD Service $08 request with a TID that is not an "availability TID", the DCM module shall invoke the configured `Xxx_RequestControl()` function with the following parameters values:
InBuffer: data contained in the OBD request (the size of which must correspond to the size configured in the DCM module's configuration)
OutBuffer: space in which the RequestControl function can store its result (the size of the buffer is taken from the DCM module's configuration)

Dcm420: After the execution of Dcm419, the Dcm module shall respond to the service request using the data stored by the RequestControl function in the OutBuffer.

7.4.3.9 OBD Service $09 - Request Vehicle Information

Dcm421: The DCM module shall implement OBD Service $09 (Request Vehicle Information) in compliance to all provisions of the OBD standard.

Using Service $09, an external test tool can request vehicle information or can obtain lists of supported vehicle information. OBD reserves certain InfoTypes for the special purpose of obtaining the list of supported InfoTypes in a certain range. These Infotypes are called "availability InfoTypes" and are $00, $20, $40, $60, $80, $A0, $C0 an $E0.
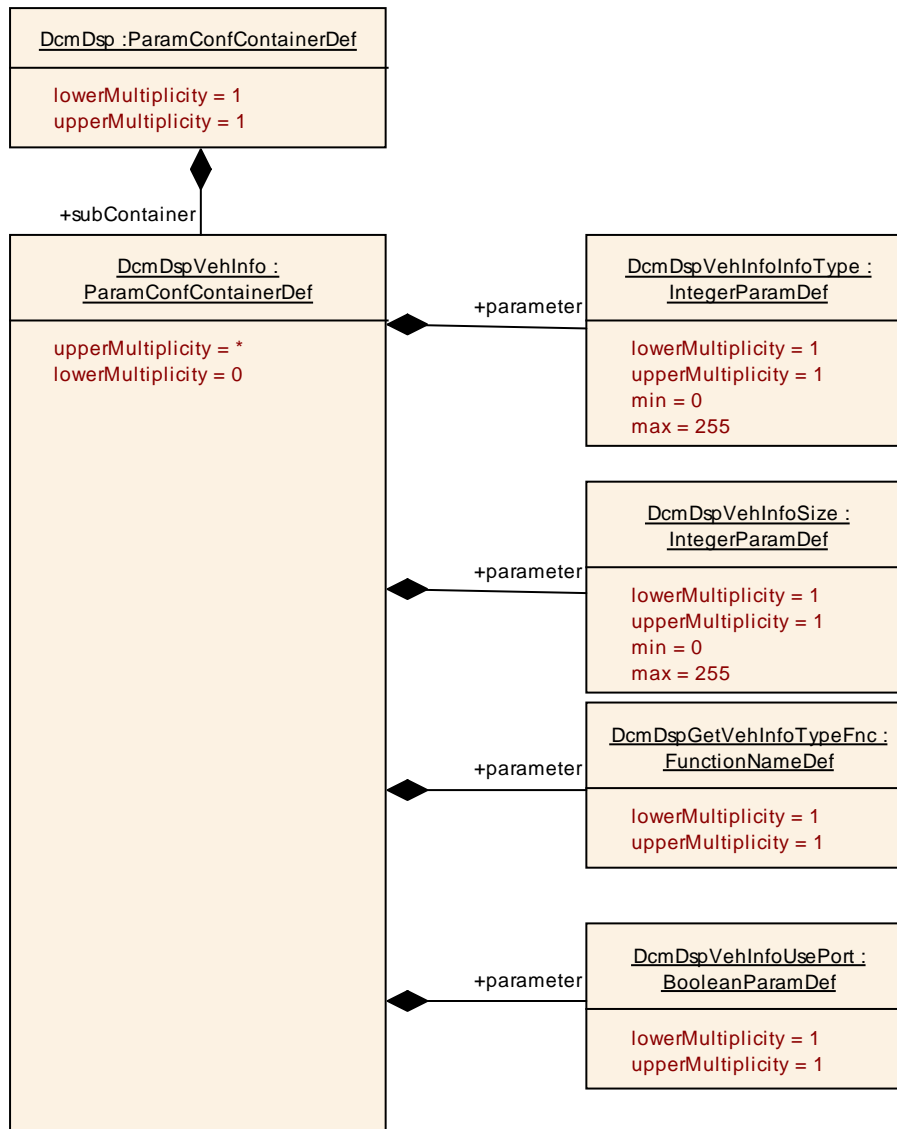
Figure 34: Configuration related to OBD Service $09

The DCM module's configuration defines the InfoTypes that are available on the ECU. The configuration defines for each such InfoType:
- the size of the value of the InfoType (see configuration parameter **_DcmDspVehInfoSize_**)
- the function that the DCM module must call to obtain the value for this InfoType OR the port-name through which the DCM module can obtain the

value for this InfoType (see configuration parameter ***DcmDspGetVehInfoTypeFnc*** and ***DcmDspVehInfoUsePort***).
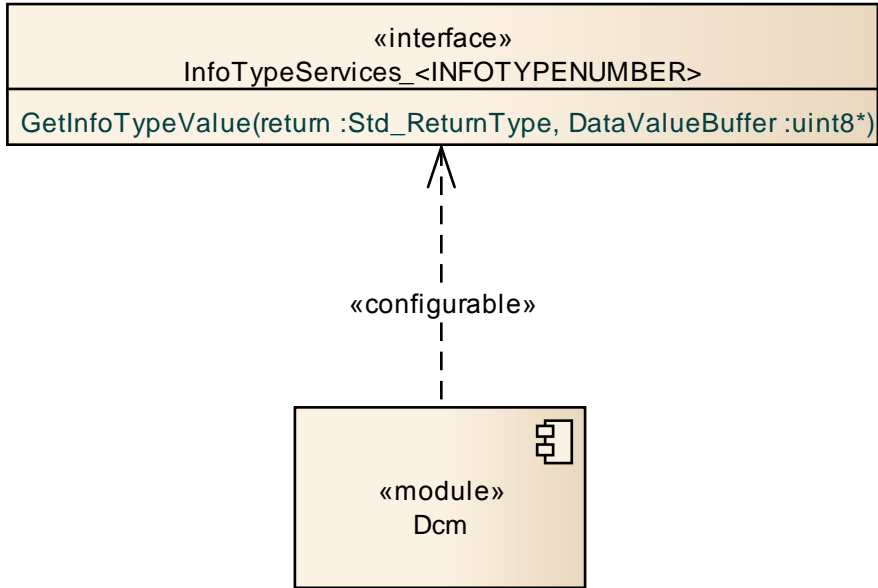
Figure 35: To provide OBD Service $09, the DCM relies on external functions that allow it to obtain the value of an InfoType. There is one such function per InfoType that is needed by the DCM.

When invoking a `Xxx_GetInfoTypeValue()` function, the DCM module provides a buffer of the correct size in which the value of the InfoType can be stored. The entity providing the actual implementation of the `Xxx_GetInfoTypeValue()` function for a specific InfoType might be a SW-C or another basic software module. The origin of the function is part of the DCM module's configuration. Certain InfoTypes are provided by the DEM.
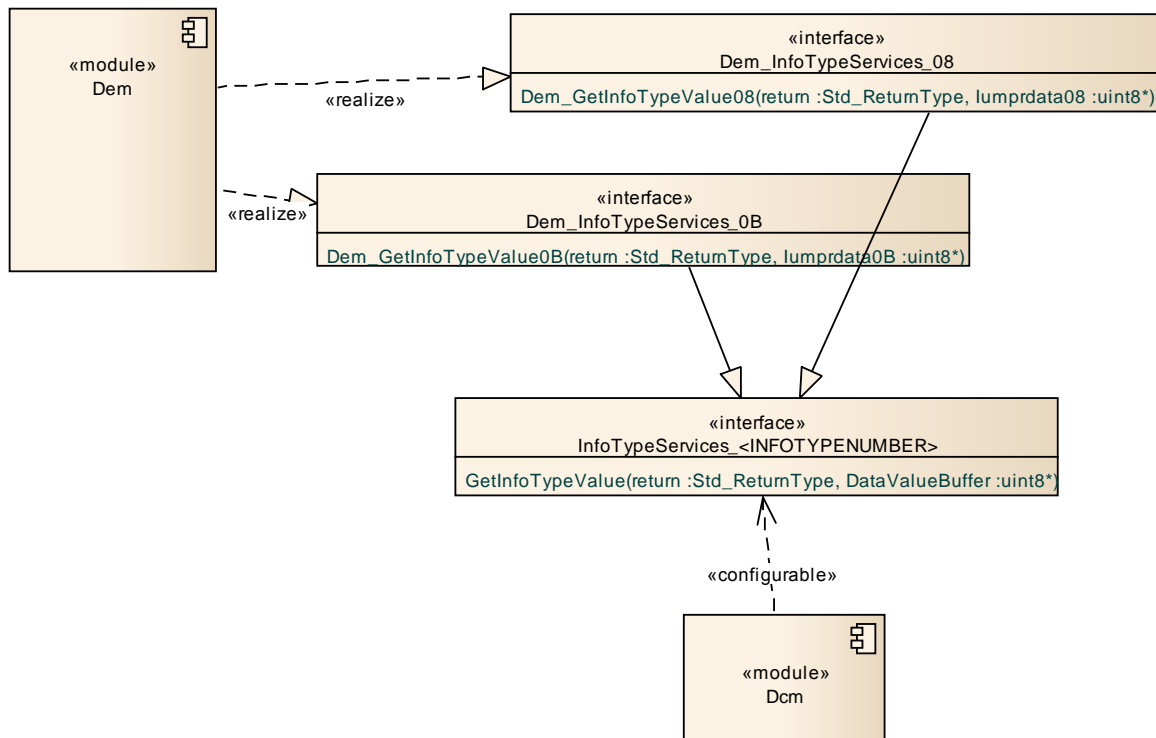
Figure 36: Certain InfoTypes needed by the DCM to provide Service $09 are provided by the DEM. This is handled in the DCM configuration.

Dcm422: On reception of an OBD Service $09 request with one or more "availability InfoTypes" as parameter, the DCM module shall respond with the corresponding supported (=configured) InfoTypes.

Dcm423: On reception of an OBD Service $09 request for an InfoType that is not an "availability InfoType", the DCM module shall obtain the value of this InfoType by invoking the configured `Xxx_GetInfoTypeValue()` function and shall return the value as response to Service $09.

7.4.4 General

The DCM shall be able to manage a jump to the bootloader. Due to the diversity of possibility to realize this jump, this will be done using callout call.

7.4.4.1 Jump to bootloader

Dcm531: A jump to bootloader is possible only with services DiagnosticSessionControl and LinkControl services

Dcm532: On reception of service DiagnosticSessionControl if the provided session is used to jump to OEM bootloader (parameter DcmDspSessionForBoot set to DCM_OEM_BOOT) the DCM shall jump to the OEM bootloader (see Dcm535 and Dcm538)

Dcm592: On reception of service DiagnosticSessionControl if the provided session is used to jump to System Supplier bootloader (parameter DcmDspSessionForBoot set to DCM_SYS_BOOT) the DCM shall jump to the System Supplier bootloader (see Dcm535 and Dcm538)

Dcm660: If the jump to bootloader is requested (see Dcm532, Dcm592 and Dcm533), the DCM module shall invoke the configured Xxx_EcuReset() functions (see configuration parameter DcmDspEcuReset) with the following parameter values: DCM_BOOTLOADER_RESET (OEM bootloader) or DCM_SS_BOOTLOADER_RESET (System Supplier bootloader) to request the BswM if the jump to bootloader is allowed.

Dcm659: After having executed Dcm660, if the jump to bootloader mode is accepted (when all the Xxx_EcuReset() functions return a value E_OK), and the configuration parameter DcmSendRespPendOnTransToBoot is set to FALSE, the DCM module shall respond positively to the request.

Dcm706: After having executed Dcm660, if the jump to bootloader mode is accepted (when all the Xxx_EcuReset() functions return a value E_OK), and the configuration parameter DcmSendRespPendOnTransToBoot is set to TRUE, the DCM module shall trigger transmission of NRC 0x78 (Response pending).

Dcm658: After requested jump to bootloader (see Dcm660), if the jump to bootloader mode is not accepted (when at least one Xxx_EcuReset() function returns E_NOT_OK), the DCM module shall not call BswM_Dcm_RequestResetMode() but shall trigger a negative response with the NRC value returned in ErrorCode NRC 0x22 (Conditions not correct).

Dcm540: After having executed Dcm660, when one or more Xxx_EcuReset() functions return E_PENDING and no function returns E_NOT_OK, the DCM module shall invoke the ones returning E_PENDING again in a next cycle.

Dcm535: If the jump to bootloader is requested (see Dcm532, Dcm592 and Dcm533), and the configuration parameter DcmSendRespPendOnTransToBoot is set to TRUE , the DCM shall call Dcm_SetProgConditions() after confirmation of the transmission of NRC 0x78 (Response pending).

This will allow to store all relevant information prior to jumping to the bootloader.

Dcm705: If the jump to bootloader is requested (see Dcm532, Dcm592 and Dcm533), and the configuration parameter DcmSendRespPendOnTransToBoot is set to FALSE, the DCM shall call Dcm_SetProgConditions() immediately.

Note: It is up to the software integrator to decide where to store that data. Usually it will be stored in non-volatile memory like e.g. data flash. It is also acceptable to "store" this data in a RAM section which is not initialized out of reset.

Dcm538: If the jump to bootloader is requested (see Dcm532, Dcm592 and Dcm533) and if the call to Dcm_SetProgConditions() has been done (see **Dcm535**) the DCM shall call BswM_Dcm_RequestResetMode() with Reset type DCM_BOOTLOADER_RESET (OEM bootloader) or DCM_SS_BOOTLOADER_RESET (System Supplier bootloader) to perform the jump to bootloader.

7.4.4.2 Jump from bootloader

Dcm536: At DCM initialization (First call to Dcm_MainFunction), the DCM shall call Dcm_GetProgConditions() to know if the initialization is the consequence of a jump from the bootloader.

Note: It is the responsibility of the software integrator to ensure that the data contained in Dcm_ProgRequestType is valid when Dcm_Init is called. E.g. if this data is stored in non-volatile memory, it may take some time to make it available after an ECU reset. This has to be taken into account when designing the ECU's startup process.

Dcm537: If the initialization of the DCM is the consequence of a jump from the bootloader (see Dcm536) and a response is required (ResponseRequired == true), the DCM shall call ComM_DCM_ActiveDiagnostic(NetworkId) to request the ComManager for the full communication mode.

Dcm539: When the ComM reports full communication to the Dcm, the Dcm shall send the Response to the Service Id passed in the Dcm_ProgConditionsType.

Dcm541: If the initialization of the DCM is the consequence of a jump from the bootloader (see Dcm536) and a the application is updated by an FLASH download (Dcm_ProgConditionsType.ApplUpdated == True). The DCM shall call BswM_Dcm_ApplicationUpdated() to notify the BswM that the application was updated.

Dcm702 : If the configuration parameter DcmStartupTimerDuration exists, a startup timer shall be started with the first Dcm_MainFunction call after coming back from the bootloader (when the Dcm needs to send a response).

Dcm703: When the startup timer has expired, the Dcm shall try to send its response message (if this was not done already due to **Dcm539**).

Dcm704: If the transmission fails, the Dcm shall retry for a limited number of times (see configuration parameter DcmStartupNumberOfRetries).

7.5 Error classification

This section describes how the DCM module has to treat the several error classes that may happen during the life cycle of the DCM module.

The general requirements document of AUTOSAR [1] specifies that all Basic Software modules shall distinguish (according to the product life cycle) two error types:

- Development errors: those errors shall be detected and fixed during the development phase. In most cases, those errors are software errors. The detection of errors that shall only occur during development can be switched off for production code (by static configuration namely preprocessor switches).

- Production errors: those errors are hardware errors and software exceptions that cannot be avoided and are expected to occur in production (i.e. series) code.

Dcm012: The DCM module shall provide no Production-Errors

Diagnostic-Communication-Errors are handled directly in the ISO-Protocols by NRCs.

Dcm044: The used return values shall be the same for development and production. Only the values given by the DCM SWS shall be used.

Dcm364: Development error values are of type uint8.

Type or error	Relevance	Related error code	Value
Interface: Timeout occurred during interaction with another module (e.g. maximum number of response pending is reached, refer to Dcm120)	Development	DCM_E_INTERFACE_TIMEOUT	0x01
Interface return-value is out of range	Development	DCM_E_INTERFACE_RETURN_VALUE	0x02
Interface: Boundary check of buffers provided by the Dcm failed during interaction with another module (application, Dem, PduR, etc.)	Development	DCM_E_INTERFACE_BUFFER_OVERFLOW	0x03
Internal: DCM not initialized	Development	DCM_E_UNINIT	0x05
DCM API function with invalid input parameter	Development	DCM_E_PARAM	0x06

Dcm040: The following errors and exceptions shall be detectable by the DCM module depending on its build version (development/production mode).

Dcm041: Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the DCM module implementation specification. The classification and enumeration shall be compatible to the errors listed above

7.6 Error detection

Dcm042: The detection of development errors is configurable (ON/OFF) at pre-compile time. The configuration parameter *DcmDevErrorDetect* shall activate or deactivate the detection of all development errors.

Dcm043: If the development error detection is enabled for this module, for every request except Dcm_Init, it shall be ensured that the DCM module is already initialized. Detected errors shall be reported to the Debug Error Tracer.

Dcm048: If the *DcmDevErrorDetect* configuration parameter is enabled, API parameter checking shall be enabled except for `Dcm_Init()`.

7.7 Error notification

Dcm049: Detected development errors shall be reported to the `Det_ReportError()` service of the Development Error Tracer (DET) if the pre-processor switch *DcmDevErrorDetect* is set.

The Development Error Tracer module is just help for BSW development and integration. It must not be contained inside the production code. The API is defined, but the functionality can be chosen and implemented according to the development needs (e.g. errors count, send error information via a serial interface to an external logger, and so on).

Dcm052: The header file of the DCM, DCM.h, shall provide a module ID called DCM_MODULE_ID set to the value 0x35.

8 API specification

This section defines:
- The syntax and semantics of the functions that are provided and required from other BSW modules. These take the form of "C"-APIs.
- The syntax and semantics of a subset of those functions which are used by software-components through the RTE. These take the form of descriptions using the concepts of the Software-Component Template.

8.1 Imported types

This section lists all types included from other modules.

8.2 Dcm333:

Module	Imported Type
ComStack_Types	BufReq_ReturnType
	NetworkHandleType
	NotifResultType
	PduIdType
	PduInfoType
	PduLengthType
	TPParameterType
Dem	Dem_DTCGroupType
	Dem_DTCKindType
	Dem_DTCOriginType
	Dem_DTCRequestType
	Dem_DTCSeverityType
	Dem_FilterForFDCType
	Dem_FilterWithSeverityType
	Dem_ReturnClearDTCType
	Dem_ReturnControlDTCStorageType
	Dem_ReturnControlEventUpdateType
	Dem_ReturnDisableDTCRecordUpdateType
	Dem_ReturnGetDTCByOccurrenceTimeType
	Dem_ReturnGetDTCOfFreezeFrameRecordType
	Dem_ReturnGetExtendedDataRecordByDTCType
	Dem_ReturnGetFreezeFrameDataByDTCType
	Dem_ReturnGetFreezeFrameDataIdentifierByDTCType
	Dem_ReturnGetFunctionalUnitOfDTCType
	Dem_ReturnGetNextFilteredDTCType
	Dem_ReturnGetNumberOfFilteredDTCType
	Dem_ReturnGetSeverityOfDTCType
	Dem_ReturnGetSizeOfExtendedDataRecordByDTCType
	Dem_ReturnGetSizeOfFreezeFrameType
	Dem_ReturnGetStatusOfDTCType
	Dem_ReturnSetDTCFilterType
SchM	SchM_ReturnType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.3 Type Definitions

This section lists the types which are defined by the DCM SWS.
Dcm_SecLevelType

Name:	Dcm_SecLevelType		
Type:	uint8		
Range:	DCM_SEC_LEV_LOCKED	0x00	--
	configuration dependent	0x01...0x3F	--
	Reserved by Document	0x40...0xFF	--
Description:	Security Level type definition		

Corresponding to the definition given above, the same data-type is defined in the following way when used in an AUTOSAR Interface:

```
IntegerType SecLevelType {
     LOWER-LIMIT = 0
     UPPER-LIMIT = 255
};
0 -> DCM_SEC_LEV_LOCKED
1 -> Configuration dependent (up to 127)
128 -> Reserved by document (up to 254)
```

Dcm_SesCtrlType

Name:	Dcm_SesCtrlType		
Type:	uint8		
Range:	DEFAULT_SESSION	0x01	--
	PROGRAMMING_SESSION	0x02	--
	EXTENDED_DIAGNOSTIC_SESSION	0x03	--
	SAFETY_SYSTEM_DIAGNOSTIC_SESSION	0x04	--
	configuration dependent	0x05...0x7F	(according to "diagnosticSessionType" parameter of DiagnosticSessionControl request)
	Reserved by Document	0x80...0xFE	--
	ALL_SESSION_LEVEL	0xFF	--
Description:	Session type definition		

Corresponding to the definition given above, the same data-type is defined in the following way when used in an AUTOSAR Interface:

```
IntegerType SesCtrlType {
     LOWER-LIMIT = 1
     UPPER-LIMIT = 255
};
1 -> DEFAULT_SESSION
2 -> PROGRAMMING_SESSION
3 -> EXTENDED_DIAGNOSTIC_SESSION
4 -> SAFETY_SYSTEM_DIAGNOSTIC_SESSION
5 -> Configuration dependent (up to 127)
128 -> Reserved by document (up to 254)
255 -> ALL_SESSION_LEVEL
```

8.3.1 Dcm_ProtocolType

Name:	Dcm_ProtocolType		
Type:	uint8		
Range:	DCM_OBD_ON_CAN	0x00	OBD on CAN (ISO15765-4; ISO15031-5)
	DCM_UDS_ON_CAN	0x01	UDS on CAN (ISO15765-3; ISO14229-1)
	DCM_UDS_ON_FLEXRAY	0x02	UDS on FlexRay (Manufacturer specific; ISO14229-1)
	DCM_ROE_ON_CAN	0x03	Response On Event on CAN
	DCM_ROE_ON_FLEXRAY	0x04	Response On Event on FlexRay
	DCM_PERIODICTRANS_ON_CAN	0x05	Periodic Transmission on CAN
	DCM_PERIODICTRANS_ON_FLEXRAY	0x06	Periodic Transmission on FlexRay
	Reserved for further AUTOSAR implementation	0x07..0xEF	--
	Reserved for Supplier specifics	0xF0..0xFF	--
Description:	Protocol type definition		

Corresponding to the definition given above, the same data-type is defined in the following way when used in an AUTOSAR Interface:

```
IntegerType ProtocolType {
     LOWER-LIMIT = 0
     UPPER-LIMIT = 255
};
0 -> DCM_OBD_ON_CAN
1 -> DCM_UDS_ON_CAN
2 -> DCM_UDS_ON_FLEXRAY
3 -> DCM_ROE_ON_CAN
4 -> DCM_ROE_ON_FLEXRAY
5 -> DCM_PERIODICTRANS_ON_CAN
6 -> DCM_PERIODICTRANS_ON_FLEXRAY
7 -> Reserved for further AUTOSAR implementation (up to 239)
240 -> Reserved for software supplier specific implementation (up to
255)
```

8.3.2 Dcm_NegativeResponseCodeType

Name:	Dcm_NegativeResponseCodeType		
Type:	uint8		
Range:	DCM_E_GENERALREJECT	0x10	This return value indicates that the requested action has been rejected by the application. The generalReject return value shall only be implemented in the application if none of the negative return values defined in this document meet the needs of the

			implementation. At no means shall this return value be a general replacement for the return values defined in this document.
	`DCM_E_BUSYREPEATREQUEST`	`0x21`	This return value indicates that the application is temporarily too busy to perform the requested operation. In this circumstance the client shall perform repetition of the "identical request message" or "another request message". The repetition of the request shall be delayed by a time specified in the respective implementation documents. Example: In a multi-client environment the diagnostic request of one client might be blocked temporarily by a NRC 0x21 while a different client finishes a diagnostic task. Note: If the application is able to perform the diagnostic task but needs additional time to finish the task and prepare the response, the NRC 0x78 shall be used instead of NRC 0x21. This return value is in general supported by each diagnostic service, as not otherwise stated in the data link specific implementation document, therefore it is not listed in the list of applicable return values of the diagnostic services.
	`DCM_E_CONDITIONSNOTCORRECT`	`0x22`	This return value indicates that the requested action will not be taken because the application prerequisite conditions are not met.
	`DCM_E_REQUESTSEQUENCEERROR`	`0x24`	This return value indicates that the requested action will not be taken because the application expects a different sequence of request messages or message as sent by the client. This may occur when sequence sensitive requests are issued in the wrong order. EXAMPLE: A successful SecurityAccess service

			specifies a sequence of requestSeed and sendKey as sub-functions in the request messages. If the sequence is sent different by the client the application shall send a negative response message with the negative return value 0x24- . requestSequenceError.
	DCM_E_REQUESTOUTOFRANGE	0x31	This return value indicates that the requested action will not be taken because the application has detected that the request message contains a parameter which attempts to substitute a value beyond its range of authority (e.g. attempting to substitute a data byte of 111 when the data is only defined to 100), or which attempts to access a dataIdentifier_routineIdentifer that is not supported or not supported in active session. This return value shall be implemented for all services, which allow the client to read data, write data or adjust functions by data in the application.
	DCM_E_SECURITYACCESSDENIED	0x33	This return value indicates that the requested action will not be taken because the application's security strategy has not been satisfied by the client. The application shall send this return value if one of the following cases occur: - the test conditions of the application are not met, - the required message sequence e.g. DiagnosticSessionControl, securityAccess is not met, - the client has sent a request message which requires an unlocked application. Beside the mandatory use of this negative return value as specified in the applicable services within this standard, this negative return value can also be used for any case where security is required and is not yet granted to perform the required service.

Document ID 018: AUTOSAR_SWS_DCM

	range of values reserved by ISO 15764	0x38..0x4F	--
	DCM_E_GENERALPROGRAMMINGFAILURE	0x72	This return value indicates that the application detected an error when erasing or programming a memory location in the permanent memory device (e.g. Flash Memory).
	DCM_E_SUBFUNCTIONNOTSUPPORTEDINACTIVESESSION	0x7E	The requested subfunction is not supported in the currently active session
	DCM_E_RPMTOOHIGH	0x81	This return value indicates that the requested action will not be taken because the application prerequisite condition for RPM is not met (current RPM is above a pre-programmed maximum threshold).
	DCM_E_RPMTOOLOW	0x82	This return value indicates that the requested action will not be taken because the application prerequisite condition for RPM is not met (current RPM is below a pre-programmed minimum threshold).
	DCM_E_ENGINEISRUNNING	0x83	This is required for those actuator tests which cannot be actuated while the Engine is running.This is different from RPM too high negative response, and needs to be allowed.
	DCM_E_ENGINEISNOTRUNNING	0x84	This is required for those actuator tests which cannot be actuated unless the Engine is running. This is different from RPM too low negative response, and needs to be allowed.
	DCM_E_ENGINERUNTIMETOOLOW	0x85	This return value indicates that the requested action will not be taken because the application prerequisite condition for engine run time is not met (current engine run time is below a pre-programmed limit).
	DCM_E_TEMPERATURETOOHIGH	0x86	This return value indicates that the requested action will not be taken because the application prerequisite condition for temperature is not met (current temperature is above a pre-programmed maximum threshold).
	DCM_E_TEMPERATURETOOLOW	0x87	This return value indicates that the requested action will

			not be taken because the application prerequisite condition for temperature is not met (current temperature is below a pre-programmed minimum threshold).
	DCM_E_VEHICLESPEEDTOOHIGH	0x88	This return value indicates that the requested action will not be taken because the application prerequisite condition for vehicle speed is not met (current VS is above a pre-programmed maximum threshold).
	DCM_E_VEHICLESPEEDTOOLOW	0x89	This return value indicates that the requested action will not be taken because the application prerequisite condition for vehicle speed is not met (current VS is below a pre-programmed minimum threshold).
	DCM_E_THROTTLE_PEDALTOOHIGH	0x8A	This return value indicates that the requested action will not be taken because the application prerequisite condition for throttle/pedal position is not met (current TP/APP is above a pre-programmed maximum threshold).
	DCM_E_THROTTLE_PEDALTOOLOW	0x8B	This return value indicates that the requested action will not be taken because the application prerequisite condition for throttle/pedal position is not met (current TP/APP is below a pre-programmed minimum threshold).
	DCM_E_TRANSMISSIONRANGENOTINNEUTRAL	0x8C	This return value indicates that the requested action will not be taken because the application prerequisite condition for being in neutral is not met (current transmission range is not in neutral).
	DCM_E_TRANSMISSIONRANGENOTINGEAR	0x8D	This return value indicates that the requested action will not be taken because the application prerequisite condition for being in gear is not met (current transmission range is not in gear).
	DCM_E_BRAKESWITCH_NOTCLOSED	0x8F	For safety reasons, this is required for certain tests before it begins, and must be maintained for the entire duration of the test.

DCM_E_SHIFTERLEVERNOTINPARK	0x90	For safety reasons, this is required for certain tests before it begins, and must be maintained for the entire duration of the test.
DCM_E_TORQUECONVERTERCLUTCHLOCKED	0x91	This return value indicates that the requested action will not be taken because the application prerequisite condition for torque converter clutch is not met (current TCC status above a pre-programmed limit or locked).
DCM_E_VOLTAGETOOHIGH	0x92	This return value indicates that the requested action will not be taken because the application prerequisite condition for voltage at the primary pin of the application (ECU) is not met (current voltage is above a pre-programmed maximum threshold).
DCM_E_VOLTAGETOOLOW	0x93	This return value indicates that the requested action will not be taken because the application prerequisite condition for voltage at the primary pin of the application (ECU) is not met (current voltage is below a pre-programmed maximum threshold).
Description: This Table of available Negative Response Codes represents the allowed Response Codes an AUTOSAR SW Component shall return after a function call. For the allowed NRC of the executed Service ID please refer to the specification of the service in ISO14229-1 (UDS) and ISO15031-5 (OBD/CARB) (see chapter 4.2.4 Response code parameter definition Table 12).		

For the implementation of this table a comment or a special concept is needed in the c-code because the table is not structured conform to the MISRA rules.
Corresponding to the definition given above, the same data-type is defined in the following way when used in an AUTOSAR Interface:

```
IntegerType NegativeResponseCodeType {
      LOWER-LIMIT = 0
      UPPER-LIMIT = 255
0x10 -> DCM_E_GENERALREJECT
0x21 -> DCM_E_BUSYREPEATREQUEST
0x22 -> DCM_E_CONDITIONSNOTCORRECT
0x24 -> DCM_E_REQUESTSEQUENCEERROR
0x31 -> DCM_E_REQUESTOUTOFRANGE
0x33 -> DCM_E_SECURITYACCESSDENIED
0x72 -> DCM_E_GENERALPROGRAMMINGFAILURE
0x7E -> DCM_E_SUBFUNCTIONNOTSUPPORTEDINACTIVESESSION
0x81 -> DCM_E_RPMTOOHIGH
0x82 -> DCM_E_RPMTOOLOW
0x83 -> DCM_E_ENGINEISRUNNING
0x84 -> DCM_E_ENGINEISNOTRUNNING
```

```
0x85 -> DCM_E_ENGINERUNTIMETOOLOW
0x86 -> DCM_E_TEMPERATURETOOHIGH
0x87 -> DCM_E_TEMPERATURETOOLOW
0x88 -> DCM_E_VEHICLESPEEDTOOHIGH
0x89 -> DCM_E_VEHICLESPEEDTOOLOW
0x8A -> DCM_E_THROTTLE_PEDALTOOHIGH
0x8B -> DCM_E_THROTTLE_PEDALTOOLOW
0x8C -> DCM_E_TRANSMISSIONRANGENOTINNEUTRAL
0x8D -> DCM_E_TRANSMISSIONRANGENOTINGEAR
0x8F -> DCM_E_BRAKESWITCH_NOTCLOSED
0x90 -> DCM_E_SHIFTERLEVERNOTINPARK
0x91 -> DCM_E_TORQUECONVERTERCLUTCHLOCKED
0x92 -> DCM_E_VOLTAGETOOHIGH
0x93 -> DCM_E_VOLTAGETOOLOW
};
```

8.3.3 Dcm_MsgContextType

Please note that the following table describes a struct type definition - including its struct items "elements".

Name:	Dcm_MsgContextType		
Type:	Structure		
Element:	PduIdType	dcmRxPduId	Pdu identifier on which the request was received. The PduId of the request can have consequences for message processing. E. g. an OBD request will be received on the OBD PduId and will be processed slightly different than an enhanced diagnostic request received on the physical PduId.
	Dcm_IdContextType	idContext	This message context identifier can be used to determine the relation between request and response confirmation. This identifier can be stored within the application at request time, so that the response can be assigned to the original request. Background: Within the confirmation, the message context is no more valid, all message data is lost. You need an additional information to determine the request to which this confirmation belongs.
	Dcm_MsgLenType	resMaxDataLen	The maximal length of a response is restricted by the size of the buffer. The buffer size can depend on the diagnostic protocol identifier which is assigned to this message, e. g. an OBD protocol id can obtain other properties than the enhanced diagnostic protocol id. The resMaxDataLen is a property of the diagnostic protocol assigned by the DSL. The value does not change during communication. It cannot be

			implemented as a constant, because it can differ between different diagnostic protocols.
	`Dcm_MsgAddInfoType`	`msgAddInfo`	Additional information about service request and response (see: Dcm_MsgAddInfo)
	`Dcm_MsgLenType`	`resDataLen`	Positive response data length (excluding service identifier)
	`Dcm_MsgType`	`resData`	Positive response data, starting directly after service identifier (which is not part of this data).
	`Dcm_MsgLenType`	`reqDataLen`	Request data length (excluding service identifier)
	`Dcm_MsgType`	`reqData`	Request data, starting directly after service identifier (which is not part of this data)
Description:	This data structure contains all information which is necessary to process a diagnostic message from request to response and response confirmation.		

8.3.4 Dcm_MsgType

Name:	`Dcm_MsgType`
Type:	`Dcm_MsgItemType*`
Description:	Base type for diagnostic message (request, positive or negative response)

8.3.5 Dcm_IdContextType

Name:	`Dcm_IdContextType`
Type:	`uint8`
Description:	This message context identifier can be used to determine the relation between request and response confirmation.

8.3.6 Dcm_MsgLenType

Name:	`Dcm_MsgLenType`
Type:	`uint32`
Description:	Length of diagnostic message (request, positive or negative response). The maximum length is dependent of the underlying transport protocol/media. E. g. the maximum message length for CAN Transport Layer is 4095bytes.

8.3.7 Dcm_MsgAddInfoType

Name:	Dcm_MsgAddInfoType		
Type:	Structure		
Element:	bit	cancelOperation	Position LSB+2 0 = No cancel 1 = Cancel pending operation
	bit	suppressPosResponse	Position LSB+1 0 = no (do not suppress) 1 = yes (no positive response will be sent)
	bit	reqType	(Pos LSB+0) 0 = physical request 1 = functional request
Description:	Additional information on message request. Datastructure: Bitfield		

8.3.8 Dcm_MsgItemType

Name:	Dcm_MsgItemType
Type:	uint8
Description:	Base type for diagnostic message item

8.3.9 Dcm_ProgConditionsType

Name:	Dcm_ProgConditionsType		
Type:	Structure		
Element:	uint8	ProtocolId	Id of the protocol on which the request has been received
	uint8	TesterSourceAddr	Tester source address configured per protocol
	uint8	Sid	Service identifier of the received request
	uint8	SubFncId	Identifier of the received subfunction
	boolean	ReprogramingRequest	Set to true in order to request reprogramming of the ECU. HIS representation of FL_ExtProgRequestType.
	boolean	ApplUpdated	Indicate whether the application has been updated or not. HIS representation of FL_ApplicationUpdateType.
	boolean	ResponseRequired	Set to true in case the flashloader or application shall send a response. HIS representation of FL_ResponseRequiredType.
Description:	Used in Dcm_SetProgConditions() to allow the integrator to store relevant information prior to jumping to bootloader.		

8.3.10 Dcm_CommunicationModeType

Name:	Dcm_CommunicationModeType		
Type:	uint8		
Range:	DCM_ENABLE_RX_TX_NORM	0x00	Enable the Rx and Tx for normal

			communication
DCM_ENABLE_RX_DISABLE_TX_NORM		0x01	Enable the Rx and disable the Tx for normal communication
DCM_DISABLE_RX_ENABLE_TX_NORM		0x02	Disable the Rx and enable the Tx for normal communication
DCM_DISABLE_RX_TX_NORMAL		0x03	Disable Rx and Tx for normal communication
DCM_ENABLE_RX_TX_NM		0x04	Enable the Rx and Tx for network management communication
DCM_ENABLE_RX_DISABLE_TX_NM		0x05	Enable Rx and disable the Tx for network management communication
DCM_DISABLE_RX_ENABLE_TX_NM		0x06	Disable the Rx and enable the Tx for network management communication
DCM_DISABLE_RX_TX_NM		0x07	Diable Rx and Tx for network management communication
DCM_ENABLE_RX_TX_NORM_NM		0x08	Enable Rx and Tx for normal and network management communication
DCM_ENABLE_RX_DISABLE_TX_NORM_NM		0x09	Enable the Rx and disable the Tx for normal and network management communication
DCM_DISABLE_RX_ENABLE_TX_NORM_NM		0x0A	Disable the Rx and enable the Tx for normal and network management communication
DCM_DISABLE_RX_TX_NORM_NM		0x0B	Disable Rx and Tx for normal and network management communication
Description:		--	

8.3.11 Dcm_ResetModeType

Name:	Dcm_ResetModeType		
Type:	uint8		
Range:	DCM_NO_RESET	0x00	No reset
	DCM_HARD_RESET	0x01	Hardware reset
	DCM_KEY_ON_OFF_RESET	0x02	Key ON/OFF reset
	DCM_SOFT_RESET	0x03	Software reset
	DCM_ENABLE_RAPID_POWER_SHUTDOWN_RESET	0x04	Enable the Rapid power shutdown
	DCM_DISABLE_RAPID_POWER_SHUTDOWN_RESET	0x05	Disable the Rapid power shutdown
	DCM_BOOTLOADER_RESET	0x06	Jump to Bootloader
	DCM_SS_BOOTLOADER_RESET	0x07	Jump to System Supplier Bootloader
Description:	Used in BswM_Dcm_RequestResetMode() to indicate which kind of reset shall be realized.		
Name:	Dcm_MsgItemType		
Type:	uint8		
Description:	Base type for diagnostic message item		

8.4 Function definitions

This section defines the functions provided for other modules.
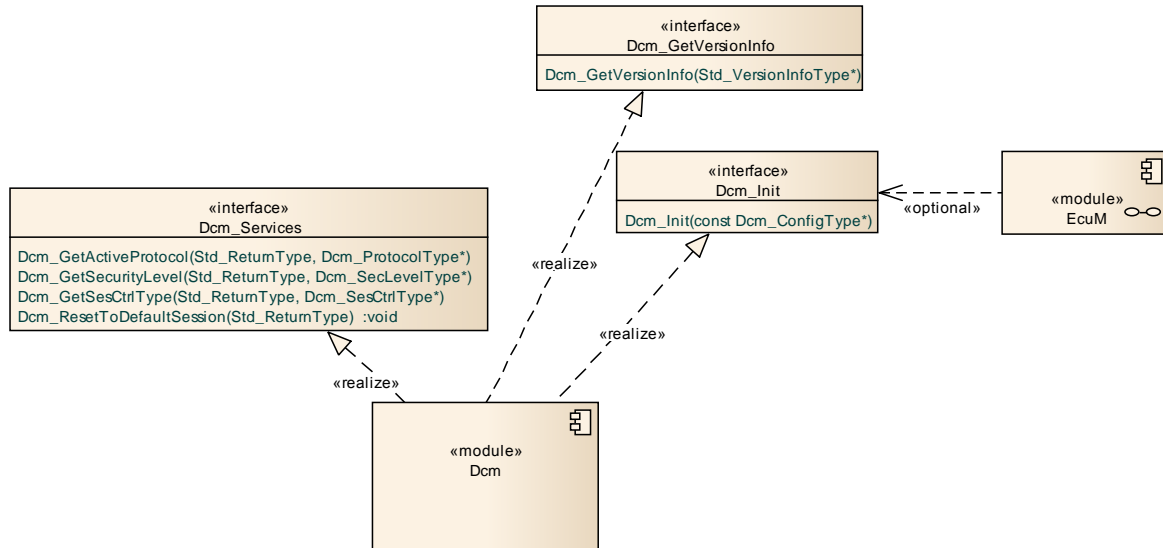
Figure 37: Overview of the functions provided by the DCM.

8.4.1 Functions provided for other BSW components

8.4.1.1 Dcm_Init

Dcm037:

Service name:	Dcm_Init	

Syntax:	`void Dcm_Init(`
	`    const Dcm_ConfigType* ConfigPtr`
	`)`	

Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfigPtr	Pointer to configuration parameter set, used e.g. for post build parameters.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Service for basic initialization of DCM module. If a NULL pointer is passed for ConfigPtr to this function the default configuration shall be used. In case only one configuration setup is used, a NULL pointer is sufficient to choose the one static existing configuration setup.	

Dcm334: `Dcm_Init()` shall initialize all DCM global variables with the values of the configuration
The call of this service is mandatory before using the DCM module for further processing.

8.4.1.2 Dcm_GetVersionInfo

Dcm065:

Service name:	Dcm_GetVersionInfo	

Syntax:	`void Dcm_GetVersionInfo(`
	`    Std_VersionInfoType* versionInfo`
	`)`	

Service ID[hex]:	0x24	
Sync/Async:	Synchronous	
Reentrancy:	Non 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	

Dcm335: `Dcm_GetVersionInfo()` shall return the version information of this module. The version information includes: Module Id, Vendor Id, Vendor specific version numbers (BSW00407).

Dcm336: If source code for caller and callee of `Dcm_GetVersionInfo()` is available, the DCM module shall realize `Dcm_GetVersionInfo()` as a macro, defined in the module's header file.

Dcm337: `Dcm_GetVersionInfo()` shall be pre compile time configurable On/Off by the configuration parameter **_DcmVersionInfoApi_**.

8.4.2 Functions provided to BSW modules and to SW-Cs

The functions defined in this section can also be used by SW-Cs through the RTE.
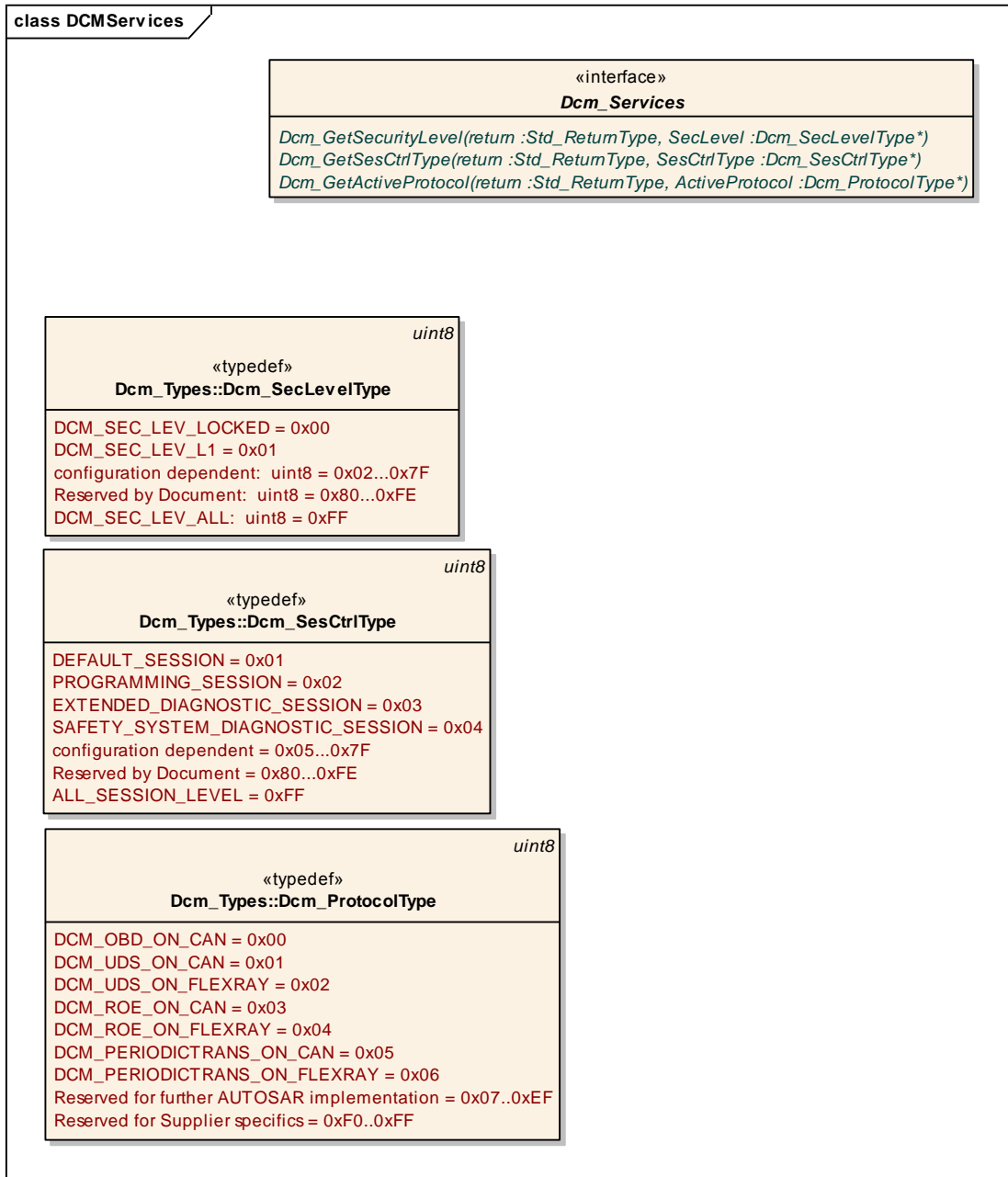
Figure 38: Definition of the interface DCMServices

```
ClientServerInterface DCMServices
{
PossibleErrors {E_OK = 0,
               E_NOT_OK = 1};

GetActiveProtocol(OUT ProtocolType ActiveProtocol, ERR{E_NOT_OK});
GetSesCtrlType(OUT SesCtrlType SesCtrlType, ERR{E_NOT_OK});
GetSecurityLevel(OUT SecLevelType SecLevel, ERR{E_NOT_OK});
}
```

- AUTOSAR confidential -

8.4.2.1 Dcm_GetSecurityLevel

Dcm338:

Service name:	Dcm_GetSecurityLevel	

Syntax:	`Std_ReturnType Dcm_GetSecurityLevel(`
	    `Dcm_SecLevelType* SecLevel`
	`)`	

Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	SecLevel	Active Security Level value Conversion formula to calculate SecurityLevel out of tester requested SecurityAccessType parameter: SecurityLevel = (SecurityAccessType + 1) / 2 Content of SecurityAccessType is according to "securityAccessType" parameter of SecurityAccess request (see [11])
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	This function provides the active security level value.	

8.4.2.2 Dcm_GetSesCtrlType

Dcm339:

Service name:	Dcm_GetSesCtrlType	

Syntax:	`Std_ReturnType Dcm_GetSesCtrlType(`
	    `Dcm_SesCtrlType* SesCtrlType`
	`)`	

Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	SesCtrlType	Active Session Control Type value Content is according to "diagnosticSessionType" parameter of DiagnosticSessionControl request (see [11])
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	This function provides the active session control type value.	

8.4.2.3 Dcm_GetActiveProtocol

Dcm340:

Service name:	Dcm_GetActiveProtocol	

Syntax:	`Std_ReturnType Dcm_GetActiveProtocol(`
	    `Dcm_ProtocolType* ActiveProtocol`
	`)`	

Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	

Parameters (out):	ActiveProtocol	Active protocol type value
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	This function returns the active protocol name.	

8.4.2.4 Dcm_ResetToDefaultSession

Dcm341:

Service name:	Dcm_ResetToDefaultSession	

Syntax:	`Std_ReturnType Dcm_ResetToDefaultSession(`
	
	`)`	

Service ID[hex]:	0x2d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	--
Description:	--	

8.5 Callback Notifications

This section defines the functions provided for lower layer BSW modules. The function prototypes of the callback functions will be provided in the file Dcm_Cbk.h
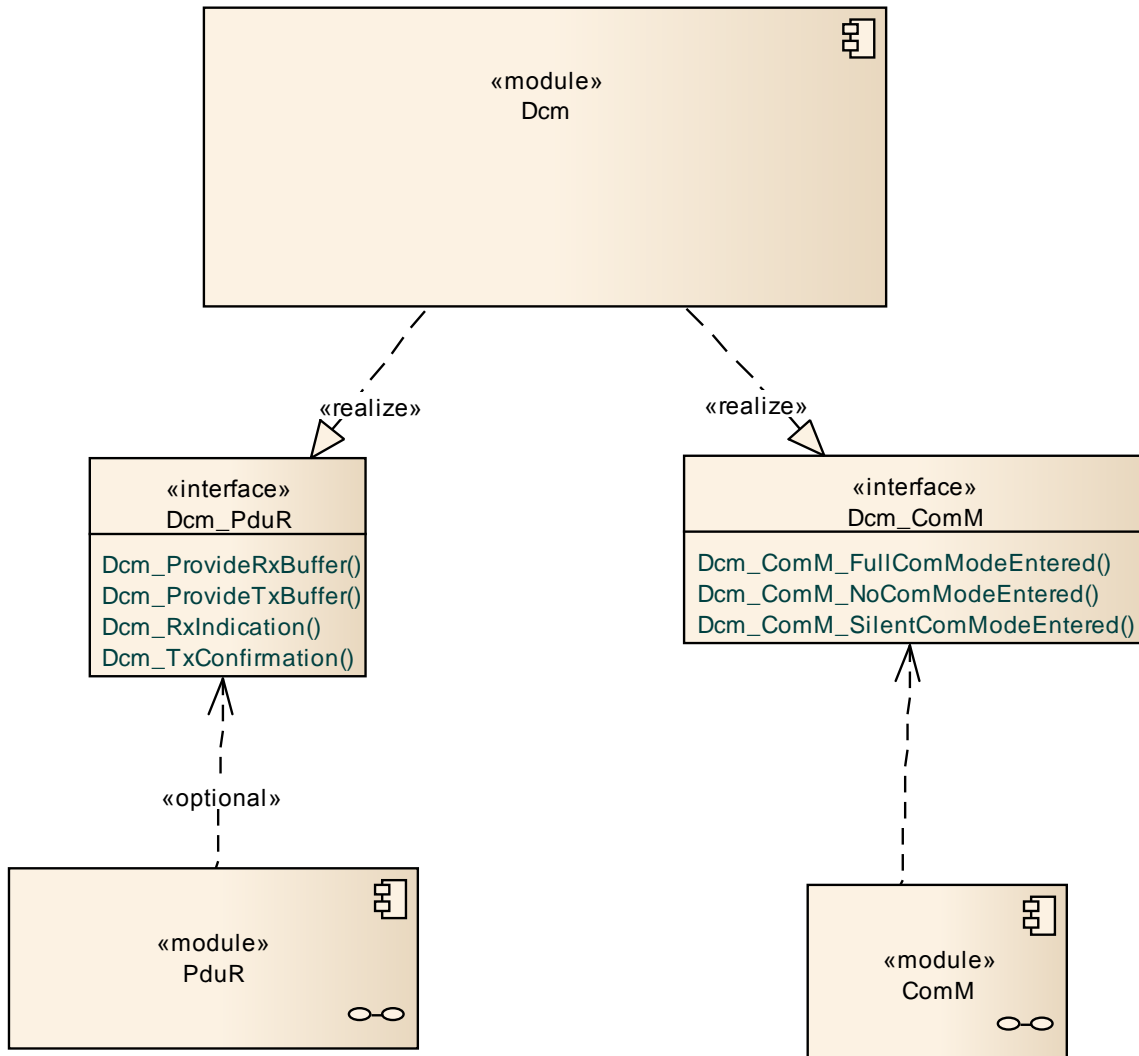
Figure 39: Overview of the callbacks provided by the DCM

8.5.1 Dcm_ProvideRxBuffer

Dcm094:

Service name:	Dcm_ProvideRxBuffer	

Syntax:	`BufReq_ReturnType Dcm_ProvideRxBuffer(`
	`    PduIdType DcmRxPduId,`
	`    PduLengthType TpSduLength,`
	`    PduInfoType** PduInfoPtr`
	`)`	

Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	DcmRxPduId	Identifies the DCM data to be received. This information is

		used within the DCM to distinguish two or more receptions at the same time.
	TpSduLength	This length identifies the overall number of bytes to be received.
Parameters (inout):	PduInfoPtr	Pointer to pointer to PduInfoType containing data pointer and length of a receive buffer. IN: In the second and further calls of Dcm_ProvideRxBuffer(), the TP's may return part of the provided buffer space unused, indicated by a smaller value for SduLength than provided by DCM in the previous call. OUT: In first call of Dcm_ProvideRxBuffer(), the Dcm will provide via PduInfoType->SduLength the BS (DcmDslProtocolBlockSize) and in the following calls the remaining size on the entire buffer.
Parameters (out):	None	
Return value:	BufReq_ReturnType	--
Description:	By this service the DCM module is requested to provide a buffer for a transport layer on first frame or single frame reception.	

By the function `Dcm_ProvideRxBuffer()` the receiver (e.g. DCM) is also informed implicitly about a first frame reception or a single frame reception. If the function `Dcm_ProvideRxBuffer()` returns a return value not equal to BUFREQ_E_OK, the values of the out parameters are not specified and should not be evaluated by the caller.

Dcm443: If `Dcm_ProvideRxBuffer()` returns BUFREQ_OK, it shall return a buffer in the parameter PduInfoPtr that is large enough for the entire diagnostic request (see Dcm094).

Dcm342: After returning a valid buffer, the DCM module shall not access this buffer until it is notified by the service `Dcm_RxIndication()` about the successful completion or unsuccessful termination of the reception.

Dcm444: If the requested size is larger than the buffer availble in the DCM, the function `Dcm_ProvideRxBuffer()` shall return BUFREQ_E_OVFL (see Dcm094).

Dcm445: If a buffer is temporarily not available (previous request not finished) the function `Dcm_ProvideRxBuffer()` shall return E_NOT_OK (see Dcm094).

8.5.2 Dcm_RxIndication

Dcm093:

Service name:	Dcm_RxIndication

Syntax:	`void Dcm_RxIndication(`
	`    PduIdType DcmRxPduId,`
	`    NotifResultType Result`
	`)`

Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	DcmRxPduId	ID of DCM I-PDU that has been received. Identifies the data that has been received. Range: 0..(maximum number of I-PDU IDs received by DCM) – 1
	Result	- NTFRSLT_OK: the complete N-PDU has been received and is stored in the receive buffer - any other value: the N_PDU has not been received; the receive buffer can be unlocked by the DCM
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This is called by the PduR to indicate the competion of a reception	

Dcm344: If `Dcm_RxIndication()` is called with parameter Result = NTFRSLT_E_NOT_OK, then the DCM module shall not evaluate the buffer assigned to the I-PDU, which is referenced in parameter DcmTxPduId.

Rationale for Dcm344: It is undefined which part of the buffer contains valid data in this case

Dcm345: `Dcm_RxIndication()` shall be callable in interrupt context.

8.5.3 Dcm_ProvideTxBuffer

Dcm092:

Service name:	Dcm_ProvideTxBuffer	

Syntax:	`BufReq_ReturnType Dcm_ProvideTxBuffer(`
	`    PduIdType DcmTxPduId,`
	`    PduInfoType** PduInfoPtr,`
	`    PduLengthType Length`
	`)`	

Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	DcmTxPduId	Identifies the DCM data to be sent. This information is used to derive the PCI information within the transport protocol. The value has to be same as in the according service call PduR_DcmTransmit().
	Length	This is the minimum length given in bytes of the buffer requested from the DCM. The minimum length is needed by the transport protocol to perform error recovery mechanisms. The value of this parameter shall not exceed the number of Bytes still to be sent. A length of zero indicates that the length of the buffer can be of arbitrary size (larger than zero). The Length parameter is expected by DCM to be always 0 (zero).
Parameters (inout):	PduInfoPtr	Pointer to pointer to PduInfoStructure containing data pointer and length of a transmit buffer.

		This length must not be smaller than the length given by parameter Length
		IN: In the second and further calls of Dcm_ProvideTxBuffer(), the TP's may return part of the provided data, indicated by a smaller value for SduLength than provided by DCM in the previous call.
		OUT: The Dcm returns the remaining size on the entire buffer.
Parameters (out):	None	
Return value:	BufReq_ReturnType	--
Description:	By this service the DCM module is requested to provide a buffer containing data to be transmitted via a transport protocol.	

The length of the buffer need not be in the length of the DCM SDU to be transmitted. It only needs to be as large as required by the caller of that service (Length). If the returned buffer is smaller than the length requested to transmit within the service `PduR_DcmTransmit()` the DCM module will be requested by the service `Dcm_ProvideTxBuffer()` to provide another buffer after the data of the current buffer have been transmitted.

Dcm346: If the function `Dcm_ProvideTxBuffer()` is called and the DCM module provides a buffer of the size Length or larger, then this function shall return BUFREQ_OK.

Dcm347: If the function `Dcm_ProvideTxBuffer()` is called and the DCM module cannot provide a buffer of the size Length or larger (except Length is equal zero), then the function shall return BUFREQ_E_NOT_OK.

Dcm348: If `Dcm_ProvideTxBuffer()` is called and the DCM module temporarily cannot provide a buffer of the requested length, the function shall return BUFREQ_E_BUSY.

If the function `Dcm_ProvideTxBuffer()` returns BUFREQ_E_BUSY, the transport protocol will call this function again at a later point in time.

Dcm349: After returning a valid buffer, the DCM module shall not access this buffer unless it is requested to provide a new buffer with service `Dcm_ProvideTxBuffer()` for the same DcmTxPduId or it is notified about the successful transmission (Call of service `Dcm_TxConfirmation()`) or it is notified by an error indicating that the transmission was aborted (Call of service `Dcm_TxConfirmation()`).

Dcm350: Caveats of `Dcm_ProvideTxBuffer()`:

- The value of parameter Length of function `Dcm_ProvideTxBuffer()` shall not exceed the number of Bytes still to be sent.

- If this service returns BUFREG_E_NOT_OK the transmit requests issued by calling the service `PduR_DcmTransmit()` is still not finished. A final confirmation (indicating an error with call of service `Dcm_TxConfirmation()`) is required to finish this service and to start a subsequent request by calling `PduR_DcmTransmit()`. So it is up to the transport protocol if the transmission is aborted on BUFREQ_E_NOT_OK or not.

- The value of parameter DcmTxPduId in a call of `Dcm_ProvideTxBuffer()` has to be same as in the according service call `PduR_DcmTransmit()`.

8.5.4 Dcm_TxConfirmation

Dcm351:

Service name:	Dcm_TxConfirmation	
Syntax:	```void Dcm_TxConfirmation(PduIdType DcmTxPduId, NotifResultType Result)```	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	DcmTxPduId	ID of DCM I-PDU that has been transmitted. Range: 0..(maximum number of I-PDU IDs transmitted by DCM) – 1
	Result	- NTFRSLT_OK if the complete N-PDU has been transmitted. - any other value: an error occurred during transmission, the DCM can unlock the transmit buffer
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This is called by the PduR to confirm a Transmit	

Dcm352: If the function `Dcm_TxConfirmation()` is called, then the DCM module shall unlock the transmit buffer.

Dcm353: If the function `Dcm_TxConfirmation()` is called, then the DCM module shall stop error handling (time-out management).

Dcm354: `Dcm_TxConfirmation()` shall be callable in interrupt context (e.g. from a transmit interrupt)

For transmission via FlexRay the following restriction has to be considered: Since the FlexRay Specification does not mandate the existence of a transmit interrupt, the exact meaning of this confirmation (i.e. "transfer into the FlexRay controller's send

buffer" OR "transmission onto the FlexRay network") depends on the capabilities of the FlexRay communication controller and the configuration of the FlexRay Interface.

8.5.5 Dcm_ComM_NoComModeEntered

Dcm356:

Service name:	Dcm_ComM_NoComModeEntered

Syntax:	`void Dcm_ComM_NoComModeEntered(`
	
	`)`

Service ID[hex]:	0x21
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This call informs the Dcm module about a ComM mode change to COMM_NO_COMMUNICATION.

Dcm148: `Dcm_ComM_NoComModeEntered()` shall disable all kinds of transmissions (receive and transmit) of communication. This means that the message reception and also the message transmission shall be off.

Dcm149: `Dcm_ComM_NoComModeEntered()` shall disable the ResponseOnEvent transmissions.

Dcm150: `Dcm_ComM_NoComModeEntered()` shall disable the periodicId transmissions (ReadDataByPeriodicIdentifier).

Dcm151: `Dcm_ComM_NoComModeEntered()` shall disable normal transmissions.

Dcm152: After `Dcm_ComM_NoComModeEntered()` has been called, the DCM module shall not call the function `PduR_DcmTransmit()`.

8.5.6 Dcm_ComM_SilentComModeEntered

Dcm358:

Service name:	Dcm_ComM_SilentComModeEntered

Syntax:	`void Dcm_ComM_SilentComModeEntered(`
	
	`)`

Service ID[hex]:	0x22

Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This call informs the Dcm module about a ComM mode change to COMM_SILENT_COMMUNICATION.

Dcm153: `Dcm_ComM_SilentComModeEntered()` shall disable all kind (receive and transmit) of communication. This means that the message reception and also the message transmission shall be off.

Dcm154: `Dcm_ComM_SilentComModeEntered()` shall disable the ResponseOnEvent transmissions.

Dcm155: `Dcm_ComM_SilentComModeEntered()` shall disable the periodicId transmissions (ReadDataByPeriodicIdentifier) shall be disabled.

Dcm156: `Dcm_ComM_SilentComModeEntered()` shall disable the normal transmissions.

8.5.7 Dcm_ComM_FullComModeEntered

Dcm360:

Service name:	Dcm_ComM_FullComModeEntered

Syntax:	`void Dcm_ComM_FullComModeEntered(`
	
	 `)`

Service ID[hex]:	0x23
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This call informs the Dcm module about a ComM mode change to COMM_FULL_COMMUNICATION.

Dcm157: `Dcm_ComM_FullComModeEntered()` shall enable all kind of communication. This means that the message reception and also the message transmission shall be on.

Dcm159: `Dcm_ComM_FullComModeEntered()` shall enable the ResponseOnEvent transmissions.

Dcm160: `Dcm_ComM_FullComModeEntered()` shall enable the periodicId transmissions (ReadDataByPeriodicIdentifier).

Dcm161: `Dcm_ComM_FullComModeEntered()` shall enable the normal transmissions.

Dcm162: After `Dcm_ComM_FullComModeEntered()` has been called, the DCM shall handle the functions `DslInternal_ResponseOnOneDataByPeriodicId()` or `DslInternal_ResponseOnOneEvent()` without restrictions.

8.6 Dcm_SetProgConditions

Dcm543

Service name:	Dcm_SetProgConditions	

Syntax:	`Std_ReturnType Dcm_SetProgConditions(`
	`    Dcm_ProgConditionsType* ProgConditions`
	`)`	

Service ID[hex]:	-	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ProgConditions	Conditions on which the jump to bootloader has been requested
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Conditions have correctly been set E_NOT_OK: Conditions cannot be set E_PENDING: Conditions set is in progress, a further call to this API is needed to end the setting
Description:	The Dcm_SetProgConditions callout allows the integrator to store relevant information prior to jumping to bootloader. The context parameter are defined in Dcm_ProgConditionsType.	

8.7 Dcm_GetProgConditions

Dcm544

Service name:	Dcm_GetProgConditions	

Syntax:	`void Dcm_GetProgConditions(`
	`    Dcm_ProgConditionsType * ProgConditions`
	`)`	

Service ID[hex]:	-	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	ProgConditions	Conditions on which the jump from the bootloader has been

	requested
Return value:	None
Description:	The Dcm_GetProgConditions callout is called upon DCM initialization and allows to determine if a response ($50 or $51) has to be sent depending on request within the bootloader. The context parameter are defined in Dcm_ProgConditionsType.

8.8 Scheduled Functions

8.8.1 Dcm_MainFunction

Dcm053:

Service name:	Dcm_MainFunction

Syntax:	`void Dcm_MainFunction(`
	
	`)`

Service ID[hex]:	0x25
Timing:	FIXED_CYCLIC
Description:	This service is used for processing the tasks of the main loop.

Dcm362: `Dcm_MainFunction` shall integrate all scheduled functions (e.g. Diagnostic timer handling)

`Dcm_MainFunction()` must be called periodically.

8.9 Expected Interfaces

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

8.9.1 Mandatory Interfaces

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

API function	*Description*
ComM_DCM_ActiveDiagnostic	Indication of active diagnostic by the DCM.
ComM_DCM_InactiveDiagnostic	Indication of inactive diagnostic by the DCM.
Dem_ClearDTC	Clears a DTC
Dem_DisableDTCRecordUpdate	Disables the event memory update of a specific DTC (only one at one time).
Dem_DisableDTCStorage	Disables the storage of a DTC group
Dem_DisableEventStatusUpdate	Disables the event status update of a DTC group
Dem_EnableDTCRecordUpdate	Enables the event memory update of the DTC disabled by Dem_DisableDTCRecordUpdate() before.
Dem_EnableDTCStorage	Enables the storage of a DTC group
Dem_EnableEventStatusUpdate	Enables the event status update of a DTC group
Dem_GetDTCByOccurrenceTime	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.
Dem_GetDTCOfFreezeFrameRecord	Gets a DTC associated with a FreezeFrame.

Dem_GetDTCStatusAvailabilityMask	Gets the DTC Status availability mask
Dem_GetExtendedDataRecordByDTC	Returns the DTCExtendedDataRecord as defined in UDS Service 0x19 subfunction 0x06, 0x10.
Dem_GetFreezeFrameDataByDTC	Gets a FreezeFrame Data by DTC. The function stores the data in the provided DestBuffer. The data returned includes the DTCSnapshotNumberOfIdentifiers and DTCSnapshotRecord as defined by UDS for the response message to Service 0x19 subfunction 0x05 or subfunction 0x04.
Dem_GetFreezeFrameDataIdentifierByDTC	Gets a FreezeFrame Data identifier by DTC
Dem_GetNextFilteredDTC	Gets the next filtered DTC.
Dem_GetNextFilteredDTCAndFDC	Gets the current DTC and its associated Fault Detection Counter (FDC) from the DEM.
Dem_GetNextFilteredDTCAndSeverity	Gets the current DTC and its Severity from the DEM.
Dem_GetNextFilteredRecord	Gets the current DTC and its associated snapshot record numbers from the DEM.
Dem_GetNumberOfFilteredDTC	Gets the number of a filtered DTC
Dem_GetSeverityOfDTC	Gets the severity of the requested DTC.
Dem_GetSizeOfExtendedDataRecordByDTC	Gets the size of an extended data record by DTC
Dem_GetSizeOfFreezeFrame	Gets the size of a FreezeFrame
Dem_GetStatusOfDTC	Gets the status of a DTC
Dem_GetTranslationType	Gets the supported DTC formats of the ECU. The supported formats are configured via DemTypeOfDTCSupported.
Dem_SetDTCFilter	Sets the filter mask over all DTCs. The server shall perform a bit-wise logical AND-ing operation between the mask specified in the client's request and the actual status associated with each DTC supported by the server. In addition to the DTCStatusAvailabilityMask, the server shall return all DTCs for which the result of the AND-ing operation is non-zero [i.e. (statusOfDTC & DTCStatusMask) != 0]. If the client specifies a status mask that contains bits that the server does not support, then the server shall process the DTC information using only the bits that it does support. If no DTCs within the server match the masking criteria specified in the client's request, no DTC or status information shall be provided following the DTCStatusAvailabilityMask byte in the positive response message. ((statusOfDTC & DTCStatusMask) & (severity & DTCSeverityMask)) != 0
Dem_SetDTCFilterForRecords	Sets DTC Filter for records.
PduR_DcmCancelReceive	By calling this API with the corresponding DcmRxPduId the currently ongoing data reception is terminated immediately. When the function returns, no reception is in progress with the given DcmRxPduId.
PduR_DcmCancelTransmit	By calling this API with the corresponding DcmTxPduId the currently ongoing TP transmission is terminated immediately. When the function returns, no transmission is in progress with the given DcmTxPduId.

PduR_DcmChangeParameter	This service is used to request the change of reception parameters BS, STmin or BandwidthControl (Fr only) for a specified PDU. The API PduR_DcmChangeParameter is not called by any standard DCM services. This API can be called by project specific implementation.
PduR_DcmTransmit	Requests a transmission for the DCM module.
SchM_ActMainFunction_<ModulePrefix>	Invokes the SchM_ActMainFunction function to trigger the activation of a corresponding main processing function.
SchM_CancelMainFunction_<ModulePrefix>	Invokes the SchM_CancelMainFunction function to trigger the cancellation of the requested activation of a corresponding main processing function.

8.9.2 Optional Interfaces

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

API function	Description
BswM_Dcm_RequestCommunicationMode	Function called by DCM to request communication modes.
BswM_Dcm_RequestResetMode	Function called by DCM to request a reset of the ECU.
Dem_DcmGetFunctionalUnitOfDTC	Gets the functional unit of the requested DTC.
Dem_GetOBDFreezeFrameData	--
Det_ReportError	Service to report development errors.

Dem_GetOBDFreezeFrameData is only required when OBD Service $02 is configured (see configuration parameter DcmDsdSidTabServiceId).

8.9.3 Configurable Interfaces

This section defines the interfaces where the DCM configuration defines the actual functions that the DCM will use. Depending on the configuration, an implementation of these functions could be provided by other BSW-modules (typically the DEM) or by software-components (through the RTE).
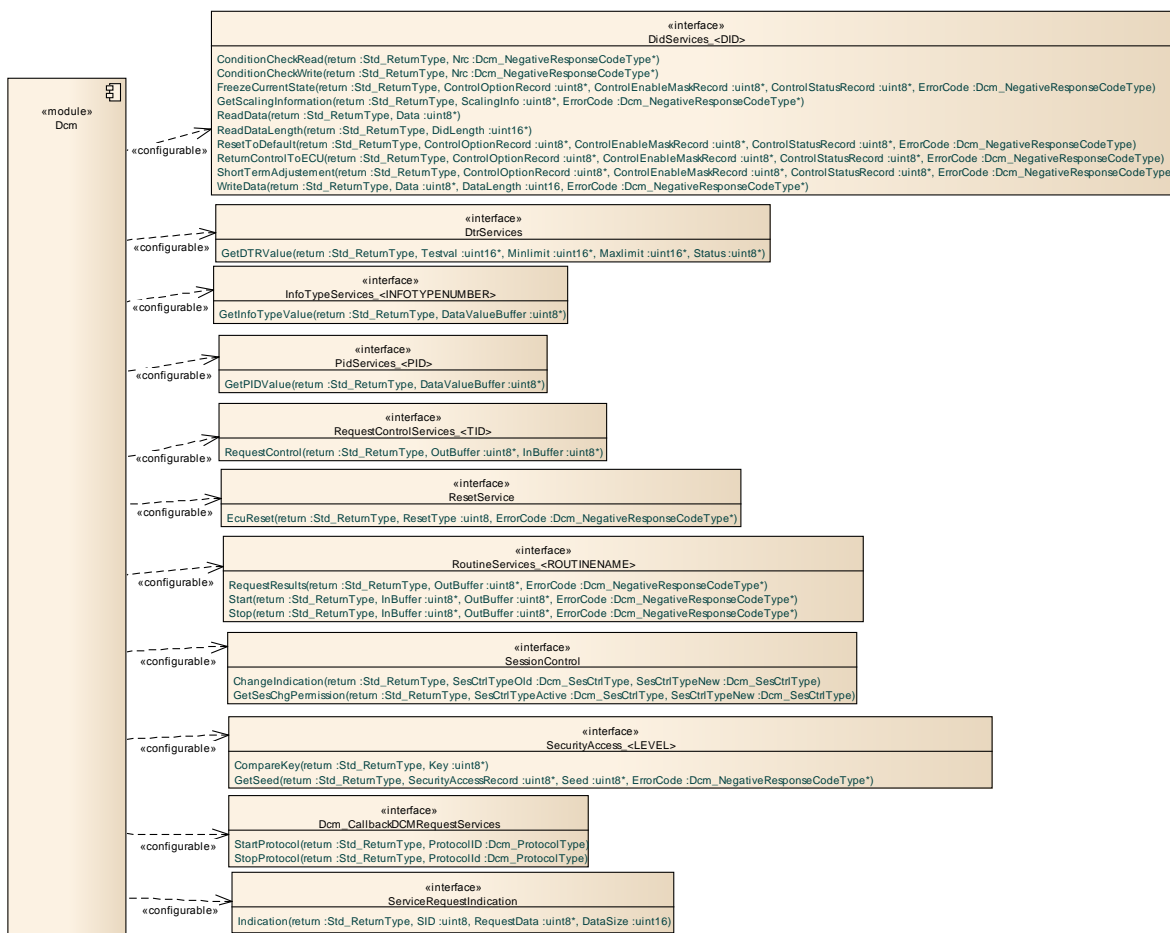
Figure 40: Configurable Interfaces of the DCM

8.9.3.1 SessionControl

«interface»
SessionControl
ChangeIndication(return :Std_ReturnType, SesCtrlTypeOld :Dcm_SesCtrlType, SesCtrlTypeNew :Dcm_SesCtrlType) GetSesChgPermission(return :Std_ReturnType, SesCtrlTypeActive :Dcm_SesCtrlType, SesCtrlTypeNew :Dcm_SesCtrlType)

uint8
«typedef» Dcm_Types::Dcm_SesCtrlType
ALL_SESSION_LEVEL = 0xFF configuration dependent = 0x05...0x7F DEFAULT_SESSION = 0x01 EXTENDED_DIAGNOSTIC_SESSION = 0x03 PROGRAMMING_SESSION = 0x02 Reserved by Document = 0x80...0xFE SAFETY_SYSTEM_DIAGNOSTIC_SESSION = 0x04

Figure 41: Definition of the interface SessionControl

Using the concepts of the SW-C template, the interface is defined as follows:
```
ClientServerInterface SessionControl {
PossibleErrors {
  E_OK = 0,
  E_NOT_OK = 1,
  E_SESSION_NOT_ALLOWED = 4,
  E_PENDING = 10, //application process not yet completed,
                  //another  call is required
  E_FORCE_RCRRP = 12, //application request the transmission of a
                       //response Response Pending (NRC 0x78)
  };

//Request to application to check if the conditions
//allow starting requested session control
//SesCtrlTypeActive: Active session control type value
//SesCtrlTypeNew: Requested session control type value
GetSesChgPermission(IN SesCtrlType SesCtrlTypeActive,
                    IN SesCtrlType SesCtrlTypeNew,
                    ERR{E_PENDING, E_SESSION_NOT_ALLOWED,
E_FORCE_RCRRP} );

//Indication to application, that value of active session
//control is changed (including "self-transitions").
//This change was either initiated by
//"DiagnosticSessionControl" command or S3Server timeout.
//SesCtrlTypeOld: Former active session control type value
//SesCtrlTypeNew: Newly set session control type value
ChangeIndication(IN SesCtrlType SesCtrlTypeOld,
                 IN SesCtrlType SesCtrlTypeNew,
                 ERR{E_NOT_OK});

//Confirms the successful transmission or a transmission error
//of a negative response with code 0x78, which was triggered
//directly by application (return value DCM_E_FORCE_RCRRP  on
//GetSesChgPermission call )
//This interface exists only if DcmDslDiagRespForceRespPendEn is
//set to TRUE
//Status: indication about confirmation (differentiate failure
//indication and normal confirmation) / The parameter "Result"
//of "Dcm_TxConfirmation" shall be forwarded to status
//depending if a positive or negative responses was sent
//before.
ConfirmationRespPend(IN ConfirmationStatusType status,
                     ERR{E_NOT_OK});
}

IntegerType ConfirmationStatusType {
     LOWER-LIMIT = 0
     UPPER-LIMIT = 255
  0x00 -> DCM_RES_POS_OK
  0x01 -> DCM_RES_POS_NOT_OK
  0x02 -> DCM_RES_NEG_OK
  0x03 -> DCM_RES_NEG_NOT_OK
};
```

From the point of view of the DCM, these operations have the following signatures:
```
Std_ReturnType GetSesChgPermission(
  Dcm_SesCtrlType SesCtrlTypeActive,
  Dcm_SesCtrlType SesCtrlTypeNew)

Std_ReturnType ChangeIndication(
  Dcm_SesCtrlType SesCtrlTypeOld,
  Dcm_SesCtrlType SesCtrlTypeNew)

Std_ReturnType ConfirmationRespPend(
  Dcm_ConfirmationStatusType status)
```

8.9.3.2 SecurityAccess_<LEVEL>

Provides functions required for the UDS Service SecurityAccess (see Dcm323, Dcm324).


```
«interface»
SecurityAccess_<LEVEL>

CompareKey(return :Std_ReturnType, Key :uint8*)
GetSeed(return :Std_ReturnType, SecurityAccessRecord :uint8*, Seed :uint8*, ErrorCode :Dcm_NegativeResponseCodeType*)
```

Figure 42: Definition of the interface SecurityAccess_<LEVEL>

Using the concepts of the SW-C template, the interface is defined as follows:
```
ClientServerInterface SecurityAccess_<LEVEL>
{
PossibleErrors {
  E_OK = 0,
  E_NOT_OK = 1,
  E_PENDING = 10, // application process not yet completed
  E_COMPARE_KEY_FAILED = 11 // compare failed
};

//Request to application for provision of seed value
//SecurityAccessDataRecord: This data record contains additional
//data to calculate the seed value and the size is given by
DcmDspSecurityADRSize.
//Seed: Pointer for provided seed
//If the parameter DcmDspSecurityADRSize is present, the following
interface
is used:
GetSeed(
  IN uint8 SecurityAccessDataRecord[<DcmDspSecurityADRSize>],
  OUT uint8 Seed[<DcmDspSecuritySeedSize>],
  OUT NegativeResponseCodeType ErrorCode,
  ERR{E_NOT_OK, E_PENDING});

//If the parameter DcmDspSecurityADRSize is not present, the
following
interface is used:
GetSeed(
  OUT uint8 Seed[<DcmDspSecuritySeedSize>],
  OUT NegativeResponseCodeType ErrorCode,
  ERR{E_NOT_OK, E_PENDING});
```

From the point of view of the DCM, the operation has the following signature (when DcmDspSecurityADRSize is present):

```
Std_ReturnType Xxx_GetSeed(uint8* SecurityAccessDataRecord,
                           uint8* Seed,
                           Dcm_NegativeResponseCodeType* ErrorCode)
Std_ReturnType Xxx_CompareKey(uint8* Key)
```

From the point of view of the DCM, the operation has the following signature (when DcmDspSecurityADRSize is not present):

```
Std_ReturnType Xxx_GetSeed(uint8* Seed,
                           Dcm_NegativeResponseCodeType* ErrorCode)
Std_ReturnType Xxx_CompareKey(uint8* Key)
```

8.9.3.3 PidServices_<PID>

The following interface defines an operation needed to supply OBD Service $01 (see Dcm408).

Figure 43: Definition of the interface PidServices_<PID>

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

```
ClientServerInterface PidServices_<PID>
{
PossibleErrors {
  E_OK = 0,
  E_NOT_OK = 1,
  E_PENDING = 10
};

//used to get the PID values from SW-C upon request
GetPIDValue(
  OUT uint8 DataValueBuffer[<DcmDspPidSize>],
  ERR{E_NOT_OK, E_PENDING})
}
```

From the point of view of the DCM, the operation has the following signature:

```
Std_ReturnType Xxx_GetPIDValue(uint8* DataValueBuffer)
```

8.9.3.4 DidServices_<DID>


```
                                        «interface»
                                      DidServices_<DID>
ConditionCheckRead(return :Std_ReturnType, Nrc :Dcm_NegativeResponseCodeType*)
ConditionCheckWrite(return :Std_ReturnType, Nrc :Dcm_NegativeResponseCodeType*)
FreezeCurrentState(return :Std_ReturnType, ControlOptionRecord :uint8*, ControlEnableMaskRecord :uint8*, ControlStatusRecord :uint8*, ErrorCode :Dcm_NegativeResponseCodeType)
GetScalingInformation(return :Std_ReturnType, ScalingInfo :uint8*, ErrorCode :Dcm_NegativeResponseCodeType*)
ReadData(return :Std_ReturnType, Data :uint8*)
ReadDataLength(return :Std_ReturnType, DidLength :uint16*)
ResetToDefault(return :Std_ReturnType, ControlOptionRecord :uint8*, ControlEnableMaskRecord :uint8*, ControlStatusRecord :uint8*, ErrorCode :Dcm_NegativeResponseCodeType)
ReturnControlToECU(return :Std_ReturnType, ControlOptionRecord :uint8*, ControlEnableMaskRecord :uint8*, ControlStatusRecord :uint8*, ErrorCode :Dcm_NegativeResponseCodeType)
ShortTermAdjustement(return :Std_ReturnType, ControlOptionRecord :uint8*, ControlEnableMaskRecord :uint8*, ControlStatusRecord :uint8*, ErrorCode :Dcm_NegativeResponseCodeType)
WriteData(return :Std_ReturnType, Data :uint8*, DataLength :uint16, ErrorCode :Dcm_NegativeResponseCodeType*)
```

Figure 44: Definition of the interface DidServices_<DID>

One DidServices interface will be generated for each DID, with following possible operations:

8.9.3.4.1 ReadData

ReadData allows requesting to the application the data value of a DID. A ReadData interface is defined for each DID with read access. The DID specific type is an array of uint8 which represents either the fix length of this DID or the maximum possible length of this DID. If the length is variable, the operation ReadDataLength has to provide the current valid data length of this DID.

This interface is used for UDS Service ReadDataByIdentifier, for UDS Service ReadDataByPeriodicIdentifier (0x2A).

8.9.3.4.2 WriteData

WriteData requests the application to write the data value of a DID. The DID specific type is an array of uint8 which represent either the fix length of this DID or the maximum possible length of this DID. A WriteData interface is defined for each DID with write access

This interface is used for the UDS Service WriteDataByIdentifier (0x2E).

8.9.3.4.3 ReadDataLength

ReadDataLength requests the application to return the data length of a DID. A ReadDataLength interface is defined for each DID with variable data length.

This interface is used for UDS Service ReadDataByIdentifier and for UDS Service ReadDataByPeriodicIdentifier (0x2A)

8.9.3.4.4 ConditionCheckRead

ConditionCheckRead requests to the application if the conditions (System state,…) to read the DID are correct. This operation is called for all requested DIDs before requesting the data of each DID.

A ConditionCheckRead interface is defined for each DID with read access

8.9.3.4.5 ConditionCheckWrite

ConditionCheckWrite: Request to the application if the conditions (System state,…) to read the DID are correct. A ConditionCheckRead interface is defined for each DID with write access

8.9.3.4.6 ReturnControlToEcu

Request to the application to return control to ECU of an IOControl (see Dcm396)

8.9.3.4.7 ResetToDefault

Request to the application to reset an IOControl to default value (see Dcm397)

8.9.3.4.8 FreezeCurrentState

Request to the application to freeze the current state of an IOControl (see Dcm398)

8.9.3.4.9 ShortTermAdjustement

Request to the application to adjust the IO signal (see Dcm399).

8.9.3.4.10 GetScalingInformation

Request to the application for the scaling information of the DID (see Dcm394)

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

```
ClientServerInterface DidServices_<DID>
{
PossibleErrors {
  E_OK = 0,
  E_NOT_OK = 1,
  E_PENDING = 10,
};

//the next operation is provided if the DID can be read
ReadData(OUT uint8 data[<DcmDspDidSize>],
         ERR{E_NOT_OK, E_PENDING})

//the next operation is provided if the DID can be written
//data : Did value
//dataLength: optional. Available, if the data length of
//the DID is variable. It defines the size of valid data in
//the parameter data
WriteData(IN uint8 data[<DcmDspDidSize>],
          IN uint16 dataLength,
          OUT NegativeResponseCodeType ErrorCode,
          ERR{E_NOT_OK, E_PENDING})

//the next operation is provided if the DID's length is variable
//DidLength: Data length of the DID
ReadDataLength(OUT uint16 DidLength,
               ERR{E_NOT_OK, E_PENDING})

//the next operation is provided if the DID can be read
ConditionCheckRead(OUT NegativeResponseCodeType ErrorCode,
                   ERR{E_NOT_OK, E_PENDING})

//the next operation is provided if the DID can be written
ConditionCheckWrite(OUT NegativeResponseCodeType ErrorCode,
                    ERR{E_NOT_OK, E_PENDING})

//the next operation is provided optionally
ReturnControlToECU(IN uint8 ControlOptionRecord[<size>],
                   IN uint8 ControlEnableMaskRecord[<size>],
                   OUT uint8 ControlStatusRecord[<size>],
                   OUT NegativeResponseCodeType ErrorCode,
                   ERR{E_NOT_OK, E_PENDING})

//the next operation is provided optionally
ResetToDefault(IN uint8 ControlOptionRecord[<size>],
               IN uint8 ControlEnableMaskRecord[<size>],
               OUT uint8 ControlStatusRecord[<size>],
               OUT NegativeResponseCodeType ErrorCode,
               ERR{E_NOT_OK, E_PENDING})

//the next operation is provided optionally
FreezeCurrentState(IN uint8 ControlOptionRecord[<size>],
                   IN uint8 ControlEnableMaskRecord[<size>],
```

```
                    OUT uint8 ControlStatusRecord[<size>],
                    OUT NegativeResponseCodeType ErrorCode,
                    ERR{E_NOT_OK, E_PENDING})


//the next operation is provided optionally
ShortTermAdjustement(IN uint8 ControlOptionRecord[<size>],
                    IN uint8 ControlEnableMaskRecord[<size>],
                    OUT uint8 ControlStatusRecord[<size>],
                    OUT NegativeResponseCodeType ErrorCode,
                    ERR{E_NOT_OK, E_PENDING})


//the next operation is provided optionally
GetScalingInformation(
  OUT uint8 ScalingInfo[<DcmDspDidScalingInfoSize>],
  OUT NegativeResponseCodeType ErrorCode,
  ERR{E_NOT_OK, E_PENDING} )
}
```

From the point of view of the DCM, the operations have the following signatures:
```
Std_ReturnType Xxx_ReadData(uint8* data)

Std_ReturnType Xxx_WriteData(
    uint8* data, uint16 dataLength,
    Dcm_NegativeResponseCodeType* ErrorCode)

Std_ReturnType Xxx_ReadDataLength(uint16* DidLength)

Std_ReturnType Xxx_ConditionCheckRead(
    Dcm_NegativeResponseCodeType* ErrorCode)

Std_ReturnType Xxx_ConditionCheckWrite(
    Dcm_NegativeResponseCodeType* ErrorCode)

Std_ReturnType Xxx_ReturnControlToECU(
    uint8* ControlOptionRecord,
    uint8* ControlEnableMaskRecord,
    uint8* ControlStatusRecord,
    Dcm_NegativeResponseCodeType* ErrorCode)

Std_ReturnType Xxx_ResetToDefault(
    uint8* ControlOptionRecord,
    uint8* ControlEnableMaskRecord,
    uint8* ControlStatusRecord,
    Dcm_NegativeResponseCodeType* ErrorCode)

Std_ReturnType Xxx_FreezeCurrentState(
    uint8* ControlOptionRecord,
    uint8* ControlEnableMaskRecord,
    uint8* ControlStatusRecord,
    Dcm_NegativeResponseCodeType* ErrorCode)

Std_ReturnType Xxx_ShortTermAdjustement(
    uint8* ControlOptionRecord,
    uint8* ControlEnableMaskRecord,
    uint8* ControlStatusRecord,
    Dcm_NegativeResponseCodeType* ErrorCode)

Std_ReturnType Xxx_GetScalingInformation(
```

```
uint8* ScalingInfo,
Dcm_NegativeResponseCodeType* ErrorCode)
```

8.9.3.5 InfotypeServices_<INFOTYPENUMBER>

The following interface defines an operation needed to supply OBD Service $09 (see Dcm423).

«interface»
InfoTypeServices_<INFOTYPENUMBER>
GetInfoTypeValue(return :Std_ReturnType, DataValueBuffer :uint8*)
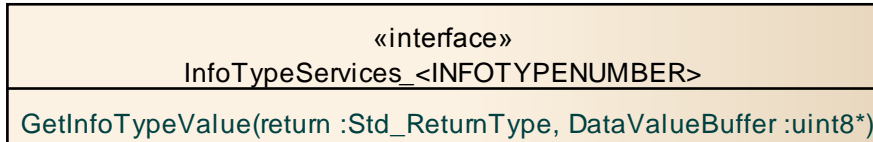
Figure 45: Definition of the interface InfoTypeServices_<INFOTYPENUMBER>

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

```
ClientServerInterface InfotypeServices_<INFOTYPENUMBER>
{
PossibleErrors {
  E_OK = 0,
  E_NOT_OK = 1,
  E_PENDING = 10
};

// used to get Infotype data from SW-C
GetInfotypeValue(
  OUT uint8 DataValueBuffer[<size Infotype>],
  ERR{E_NOT_OK, E_PENDING})
}
```

From the point of view of the DCM, the operation has the following signature:

```
Std_ReturnType Xxx_GetInfoTypeValue(uint8* DataValueBuffer)
```

8.9.3.6 DTRServices

The following interface defines an operation needed to supply OBD Service $06 (see Dcm416).

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

```
ClientServerInterface DTRServices
{
PossibleErrors {
  E_OK = 0,
  E_NOT_OK = 1,
  E_PENDING = 10,
};

// used to get DTR data from SW-C for service 6
GetDTRValue(
  OUT uint16 Testval,
  OUT uint16 Minlimit,
  OUT uint16 Maxlimit,
  OUT DTRStatusType Status,
  ERR{E_NOT_OK, E_PENDING});
}
```

```
IntegerType DTRStatusType {
      LOWER-LIMIT = 0
      UPPER-LIMIT = 255
  0x00 -> DCM_DTRSTATUS_VISIBLE
  0x01 -> DCM_DTRSTATUS_INVISIBLE
}
```

From the point of view of the DCM, the operation has the following signature:
```
Std_ReturnType Xxx_GetDTRValue(uint16* Testval,
                               uint16* Minlimit,
                               uint16* Maxlimit,
                               uint8* Status)
```

8.9.3.7 RoutineServices_<ROUTINENAME>

The following interface defines operations needed for the UDS Service RoutineControl (0x31) (see Dcm400, Dcm401, Dcm402, Dcm403, Dcm404, Dcm405).

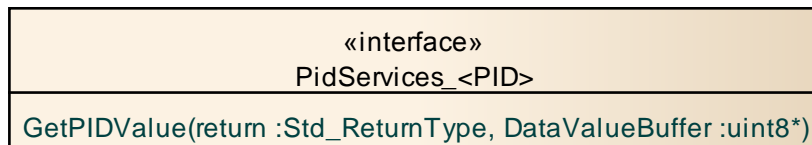| «interface» |
| :-- |
| RoutineServices_<ROUTINENAME> |
| RequestResults(return :Std_ReturnType, OutBuffer :uint8*, ErrorCode :Dcm_NegativeResponseCodeType*)
Start(return :Std_ReturnType, InBuffer :uint8*, OutBuffer :uint8*, ErrorCode :Dcm_NegativeResponseCodeType*)
Stop(return :Std_ReturnType, InBuffer :uint8*, OutBuffer :uint8*, ErrorCode :Dcm_NegativeResponseCodeType*) |

Figure 46: Definition of the interface RoutineServices_<ROUTINENAME>

Using the concepts of the SW-C template, the interface is defined as follows:
```
ClientServerInterface RoutineServices_<RoutineName>
{
PossibleErrors {
      E_OK = 0,
      E_NOT_OK = 1,
      E_PENDING = 10,
      };

//general note: if the specific-sizes are configured to be 0,
//the parameter is still present, but the caller can pass in a
//"NULL"-pointer

//the next operation is always present
Start(IN Inbuffer uint8[<Interface specific size>],
      OUT OutBuffer uint8[<Interface specific size>],
      OUT NegativeResponseCodeType ErrorCode,
      ERR{E_NOT_OK, E_PENDING})

//the next operation is present if the routine can be stopped
Stop(IN Inbuffer uint8[<Interface specific size>],
     OUT OutBuffer uint8[<Interface specific size>],
     OUT NegativeResponseCodeType ErrorCode,
     ERR{ E_NOT_OK, E_PENDING})
```

```
//the next operation is present if the status of
//the routine can be requested
RequestResults(OUT OutBuffer uint8[<Interface specific size>],
               OUT NegativeResponseCodeType ErrorCode,
               ERR{ E_NOT_OK, E_PENDING})
}
```

From the point of view of the DCM, the operations have the following signatures:
```
Std_ReturnType Xxx_Start(uint8* InBuffer,
                         uint8* OutBuffer,
                         Dcm_NegativeResponseCodeType* ErrorCode)
Std_ReturnType Xxx_Stop(uint8* InBuffer,
                        uint8* OutBuffer,
                        Dcm_NegativeResponseCodeType* ErrorCode)
Std_ReturnType Xxx_RequestResults(uint8* OutBuffer,
                        Dcm_NegativeResponseCodeType* ErrorCode)
```

8.9.3.8 RequestControlServices_<TID>

The interface RequestControlServices_<TID> allows the DCM to provide OBD Service $08 (see Dcm419).

Using the concepts of the SW-C template, the interface is defined as follows:
```
ClientServerInterface RequestControlServices_<TID> {
PossibleErrors {
  E_OK = 0,
  E_NOT_OK = 1,
  E_PENDING = 10,
};

RequestControl(
  OUT OutBuffer uint8[<DcmDspRequestControlOutBufferSize>],
  IN InBuffer uint8[<DcmDspRequestControlInBufferSize>],
  ERR{E_NOT_OK, E_PENDING})
}
```

From the point of view of the DCM, the operation has the following signature:
```
Std_ReturnType Xxx_RequestControl(uint8* OutBuffer, uint8* InBuffer)
```

8.9.3.9 CallBackDCMRequestServices

The interface CallbackDCMRequestServices provides information on the status of the protocol communication and allows the Application to disallow a protocol (see Dcm036, Dcm144, Dcm145, Dcm146; Dcm147, Dcm459).

```
                           «interface»
                   Dcm_CallbackDCMRequestServices

 «function»
 StartProtocol(return :Std_ReturnType, ProtocolID :Dcm_ProtocolType)
 StopProtocol(return :Std_ReturnType, ProtocolId :Dcm_ProtocolType)
```

```
                                                           uint8
                           «typedef»
                   Dcm_Types::Dcm_ProtocolType

 DCM_OBD_ON_CAN = 0x00
 DCM_PERIODICTRANS_ON_CAN = 0x05
 DCM_PERIODICTRANS_ON_FLEXRAY = 0x06
 DCM_ROE_ON_CAN = 0x03
 DCM_ROE_ON_FLEXRAY = 0x04
 DCM_UDS_ON_CAN = 0x01
 DCM_UDS_ON_FLEXRAY = 0x02
 Reserved for further AUTOSAR implementation = 0x07..0xEF
 Reserved for Supplier specifics = 0xF0..0xFF
```

Figure 47: Definition of the interface CallbackDCMRequestServices

Using the concepts of the SW-C template, the interface is defined as follows:
```
SenderReceiverInterface CallbackDCMRequestServices {
PossibleErrors {
  E_OK = 0,
  E_NOT_OK = 1,
  E_PROTOCOL_NOT_ALLOWED = 5, //conditions in application allows no
                             // further procession of protocol
  E_PENDING = 10 // application process not yet completed,
                 // another call is required
};

//Indication of protocol start. Application is able to reject
//further processing of requested protocol due to failed
//conditions.
StartProtocol(IN ProtocolType ProtocolID,
              ERR{E_PENDING, E_PROTOCOL_NOT_ALLOWED});

//Indication of protocol stop. If a running diagnostic
//requested is preempted by a higher prior request (of an
//other protocol, e.g. OBD), application is requested to abort
//further processing of running request
//ProtocolID: Name of  the protocol(IDs configured within
//DCM_PROTOCOL_ID)
StopProtocol(IN ProtocolType ProtocolID,
             ERR{E_NOT_OK});
```

```
}
```

From the point of view of the DCM, the operations have the following signatures:
```
Std_ReturnType Xxx_StartProtocol(Dcm_ProtocolType ProtocolID)
Std_ReturnType Xxx_StopProtocol(Dcm_ProtocolType ProtocolID)
```

8.9.3.10   ServiceRequestIndication

The interface ServiceRequestIndication indicates to the Application that a service is about to be executed and allows the Application to reject the execution of the service request (see Dcm218, Dcm462, Dcm463).

«interface»
ServiceRequestIndication
Indication(return :Std_ReturnType, SID :uint8, RequestData :uint8*, DataSize :uint16)
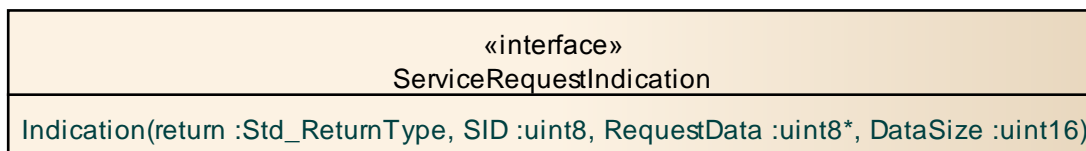
Figure 48: Definition of the interface ServiceRequestIndication

Using the concepts of the SW-C template, the interface is defined as follows:
```
ClientServerInterface ServiceRequestIndication {
PossibleErrors {
  E_OK = 0,
  E_REQUEST_NOT_ACCEPTED = 8,
  E_REQUEST_ENV_NOK = 9,
  E_PENDING = 10
};

//Indication of the successful reception of a new request to
//application and it is called right before processing the
//diagnostic service is processed. Within this function
//application can examine the permission of the diagnostic
//service / environment (e.g. ECU state afterrun). Return of
//application will lead to an acceptance of diagnostic
//request, reject of diagnostic request (-> no response will
//be send) or to an NRC 0x22 (ConditionsNotCorrect)
//SID: Value of service identifier
//RequestData: This parameter contains the complete request
//data (diagnostic buffer), except the service ID.
//DataSize: This parameter defines how many bytes in the
//RequestData parameter are valid
Indication(IN uint8 SID,
          IN uint8 RequestData[<Data size>],
          IN uint16 DataSize,
          ERR{E_PENDING, E_REQUEST_NOT_ACCEPTED,
E_REQUEST_ENV_NOK});
}
```

From the point of view of the DCM, the operation has the following signature:
```
Std_ReturnType Xxx_Indication(uint8 SID, uint8* RequestData, uint16
DataSize)
```

8.9.3.11 ResetService

The interface ResetService indicates to the Application that a request for a reset has been received and allows the Application to decide whether the reset can be executed or not (see Dcm373, Dcm374, Dcm375, Dcm477).

```
                        «interface»
                        ResetService

EcuReset(return :Std_ReturnType, ResetType :uint8, ErrorCode :Dcm_NegativeResponseCodeType*)
```
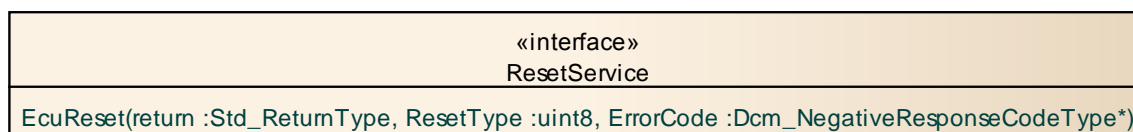
Figure 49: Definition of the interface ResetService

Using the concepts of the SW-C template, the interface is defined as follows:
```
ClientServerInterface ResetService {
PossibleErrors {
  E_OK = 0,
  E_NOT_OK = 1,
  E_PENDING = 10
  };

EcuReset(IN uint8 resetType,
        OUT NegativeResponseCodeType ErrorCode,
        ERR{E_NOT_OK, E_PENDING});
}
```

From the point of view of the DCM, the operation has the following signature:
```
Std_ReturnType Xxx_EcuReset(
  uint8 ResetType,
  Dcm_NegativeResponseCodeType* ErrorCode)
```

8.9.3.12 ComControl

The interface ComControl indicates to the Application that a CommunicationControl request has been received and allows the Application to decide whether the execution of the request is allowed or rejected (see **Dcm511**, **Dcm512**, **Dcm701** and **Dcm591**).

```
                        «interface»
                        ComControl

ChangeComMode(return :Std_ReturnType, communicationMode :Dcm_CommunicationModeType, ErrorCode :Dcm_NegativeResponseCodeType*)
```

Figure 50: Definition of the interface ComControl

Using the concepts of the SW-C template, the interface is defined as follows:
```
ClientServerInterface ComControl {
PossibleErrors {
  E_OK = 0,
```

```
  E_NOT_OK = 1,
  E_PENDING = 10
  };

ChangeComMode(IN communicationModeType communicationMode,
        OUT NegativeResponseCodeType ErrorCode,
        ERR{E_NOT_OK, E_PENDING});
}
```

From the point of view of the DCM, the operation has the following signature:

```
Std_ReturnType Xxx_ChangeComMode (
  Dcm_CommunicationModeType communicationMode,
  Dcm_NegativeResponseCodeType* ErrorCode)
```

8.10 DCM as Service-Component

This section formally specifies the corresponding AUTOSAR Service using the concepts of the Software-Component-Template.

Dcm_Services

«realize»

«module»
Dcm

«configurable» — SessionControl

«configurable» — SecurityAccess_<LEVEL>

«configurable» — PidServices_<PID>

«configurable» — DidServices_<DID>

«configurable» — InfoTypeServices_<INFOTYPENUMBER>

«configurable» — DtrServices

«configurable» — RoutineServices_<ROUTINENAME>

«configurable» — RequestControlServices_<TID>

«configurable» — Dcm_CallbackDCMRequestServices

«configurable» — ServiceRequestIndication

«configurable» — ResetService

«configurable» — ComControl

Figure 51: DCM as a Service-Component

The following definition can be generated completely out of the configuration of the DCM, which defines the exact ports that are present and their names.

```
ServiceComponent Dcm {

  //the presence and name of this port is configuration-independent
  ProvidePort DCMServices DCMServices;

  //see configuration parameter DcmDslSessionControl
  RequirePort SessionControl SessionControl_<SWC>;
  …

  //see configuration parameter DcmDspSecurityRow
  RequirePort SecurityAccess_<LEVEL> SecurityAccess_<LEVEL>;
  …

  //see configuration parameter DcmDspPid
  RequirePort PidServices_<PID> PidServices_<PID>;
  …

  //see configuration parameter DcmDspDid
  RequirePort DidServices_<DID> DidServices_<DID>;
  …

  //see configuration parameter DcmDspVehInfo
  RequirePort InfotypeServices_<INFOTYPENUMBER>
      InfotypeServices_<INFOTYPENUMBER>;
  …

  //see configuration parameter DcmDspTestResultTid
  RequirePort DTRServices DtrServices_<TID>;
  …

  //see configuration parameter DcmDspRoutine
  RequirePort RoutineServices_<ROUTINENAME>
    RoutineServices_<ROUTINENAME>;
  …

  //see configuration parameter DcmDspRequestControl
  RequirePort RequestControlServices_<TID>
    RequestControlServices_<TID>;
  …

  //see configuration parameter DcmDslCallbackDCMRequestService
  RequirePort CallbackDCMRequestServices
    CallBackDCMRequestServices_<SWC>;
  …

  //see configuration parameter DcmDslServiceRequestIndication
  RequirePort ServiceRequestIndication
    ServiceRequestIndication_<SWC>;
  …

  //see configuration parameter DcmDspEcuReset
  RequirePort ResetService ResetService_<SWC>;
```

- AUTOSAR confidential -

…

```
    //see configuration parameter DcmDspComControl
    RequirePort ComControl ComControl_<SWC>
}
```

8.11 Internal interfaces (mandatory)

In order to allow ECU Supplier diagnosis, some internal Apis shall be provided mandatory.

8.11.1   DsdInternal_SetNegResponse

Dcm490 : void DsdInternal_SetNegResponse(Dcm_MsgContextType* pMsgContext, uint8 ErrorCode)

Used by service interpreter outside of DCM to indicate that a the final response shall be a negative one.

8.11.2   DsdInternal_ProcessingDone

Dcm491 : `void DsdInternal_ProcessingDone(Dcm_MsgContextType * pMsgContext)`

Used by service interpreter outside of DCM to indicate that a final response can be sent.

8.11.3   DspInternal_<DiagnosticService>

Dcm492 : void DspInternal_<DiagnosticService>(const Dcm_MsgContextType* pMsgContext)

The DSD submodule calls `DspInternal_<DiagnosticService>()` as soon as a valid message is received on relevant DcmRxPduId. The usecase of multiple diagnostic protocols in parallel will be possible by using different arguments and the function should be programmed in a way that it is reentrant. This feature is not supported for current version 3.1.

Caveats of `DspInternal_<DiagnosticService>`: The caller (lower layer) of this function is responsible for the lifetime of the argument pMsgContext. The DSP submodule does not have to care about any timing requirement to process a request

Configuration of `DspInternal_<DiagnosticService>`: This function is only available if the corresponding diagnostic protocol service is supported.

8.12 Internal interfaces (not normative)

The following interfaces are used in the DCM SWS in order to improve the understanding of the DCM module behaviour. An implementation is not required to use these interfaces.

8.12.1 DslInternal_SetSecurityLevel

```
void
DslInternal_SetSecurityLevel(Dcm_SecLevelType SecurityLevel)
```
This function sets a new security level value in the DCM module. NOTE: for the definition of the parameter, refer to `Dcm_GetSecurityLevel()`

8.12.2 DslInternal_SetSesCtrlType

```
void
DslInternal_SetSesCtrlType(Dcm_SesCtrlType SesCtrlType)
```
This function sets a new session control type value in the DCM module. NOTE: for the definition of the parameter, refer to the `Dcm_GetSecCtrlType()`

8.12.3 DspInternal_DcmConfirmation

```
void
DspInternal_DcmConfirmation(Dcm_IdContextType idContext,
                            PduIdType dcmRxPduId,
                            Dcm_ConfirmationStatusType status)
```
This function confirms the successful transmission or a transmission error of a diagnostic service. This is the right time to perform any application state transitions.

8.12.4 DslInternal_ResponseOnOneEvent

```
Dcm_StatusType
DslInternal_ResponseOnOneEvent(const Dcm_MsgType MsgPtr,
                               Dcm_MsgLenType MsgLen,
                               PduIdType DcmRxPduId)
```
This API executes the processing of one event, requested internally in the DCM.

8.12.5 DslInternal_ResponseOnOneDataByPeriodicId

```
Dcm_StatusType
DslInternal_ResponseOnOneDataByPeriodicId(uint8 PeriodicId)
```

This API provides the processing of one periodic ID event, requested internally in the DCM. The frequency of calling this function depends on the rate given in the original ReadDataByPeriodicID request (parameter transmissionMode).

8.12.6 DsdInternal_StartPagedProcessing

```
void
DsdInternal_StartPagedProcessing(const Dcm_MsgContextType*
pMsgContext)
```
With this API, the DSP submodule gives the complete response length to the DCM module and starts paged-buffer handling. This API starts no transmission!

8.12.7 DspInternal_CancelPagedBufferProcessing

```
void
DspInternal_CancelPagedBufferProcessing()
```
DCM informs DSP, that processing of paged-buffer was cancelled due to errors.
Upon this call, DSP is not allowed to process further on paged-buffer handling.

8.12.8 DsdInternal_ProcessPage

```
void
DsdInternal_ProcessPage(Dcm_MsgLenType FilledPageLen)
```
DSP requests transmission of filled page

9 Sequence diagrams

9.1 Overview

For clarification, the following sequence diagrams don't represent the full communication mechanism between the DCM module and the PduR module. This is to keep the sequence diagrams simple.

Before the `Dcm_RxIndication()` call, the PduR module will ask the DCM module for a buffer by calling `Dcm_ProvideRxBuffer()`. This exchange is not shown on the next sequence diagrams.

After a `PduR_DcmTransmit()` request from the DCM module to the PduR module, data exchanges with `Dcm_ProvideTxBuffer()` service, are not shown in the sequence diagrams.

The function `Xxx_StartProtocol()` shall be called with the very first diagnostic request.

9.2 DSL (Diagnostic Session Layer)

9.2.1 Start Protocol
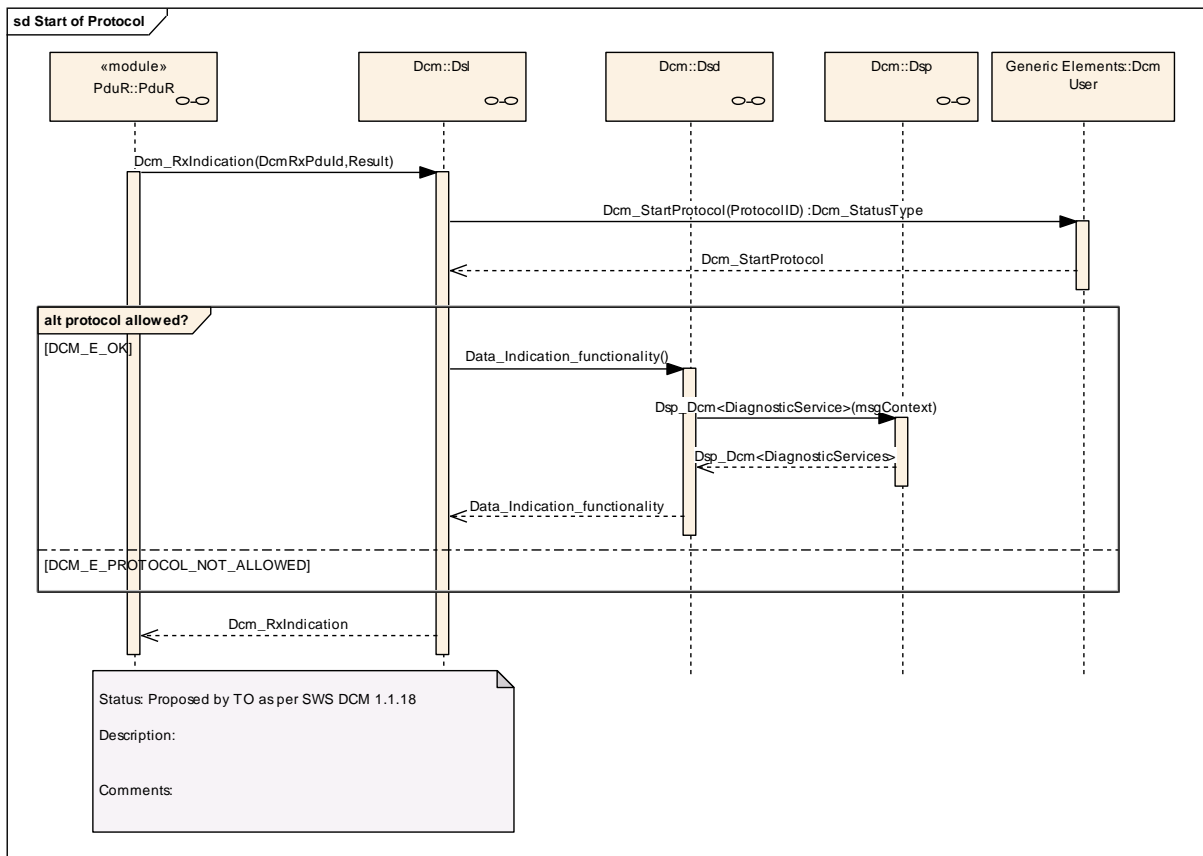
9.2.2 Process Busy behavior
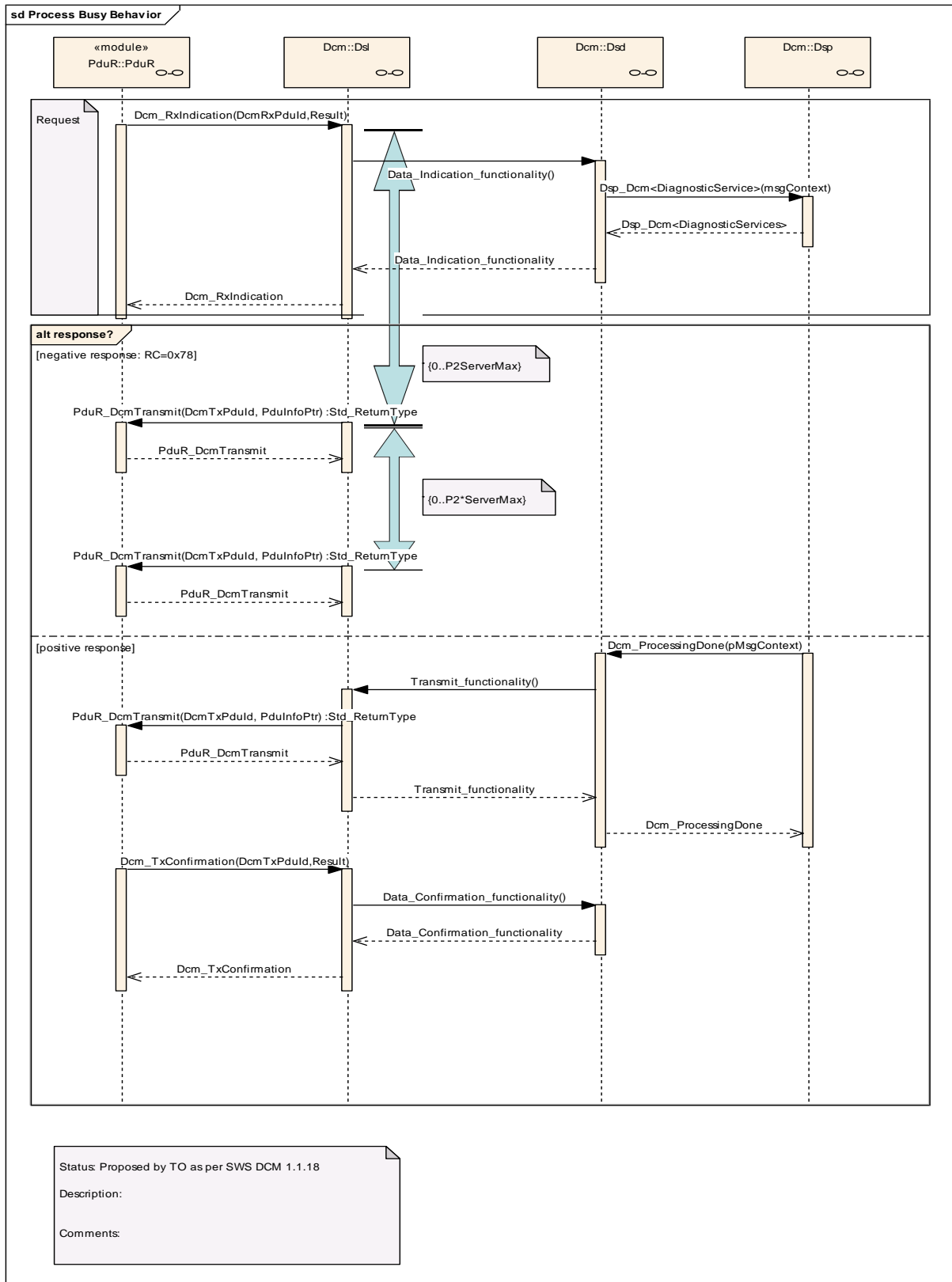
Internally, the DSL submodule calculates the time to response the tester. In the case that the DSP submodule doesn't close the request with
`DsdInternal_ProcessingDone()` (in case of normal response handling) or
`DsdInternal_ProcessPage()` (in case of paged-buffer handling) during the

P2ServerMax and/or P2*ServerMax, the DSL submodule sends a negative response (requestCorectlyReceived-ResponsePending) independently.

9.2.3 Update Diagnostic Session Control when timeout occurs

The DSL submodule resets session control value to default, if in a non-default session S3server timeout occurs. S3server timeout timer will be started with every data confirmation from the PduR module.
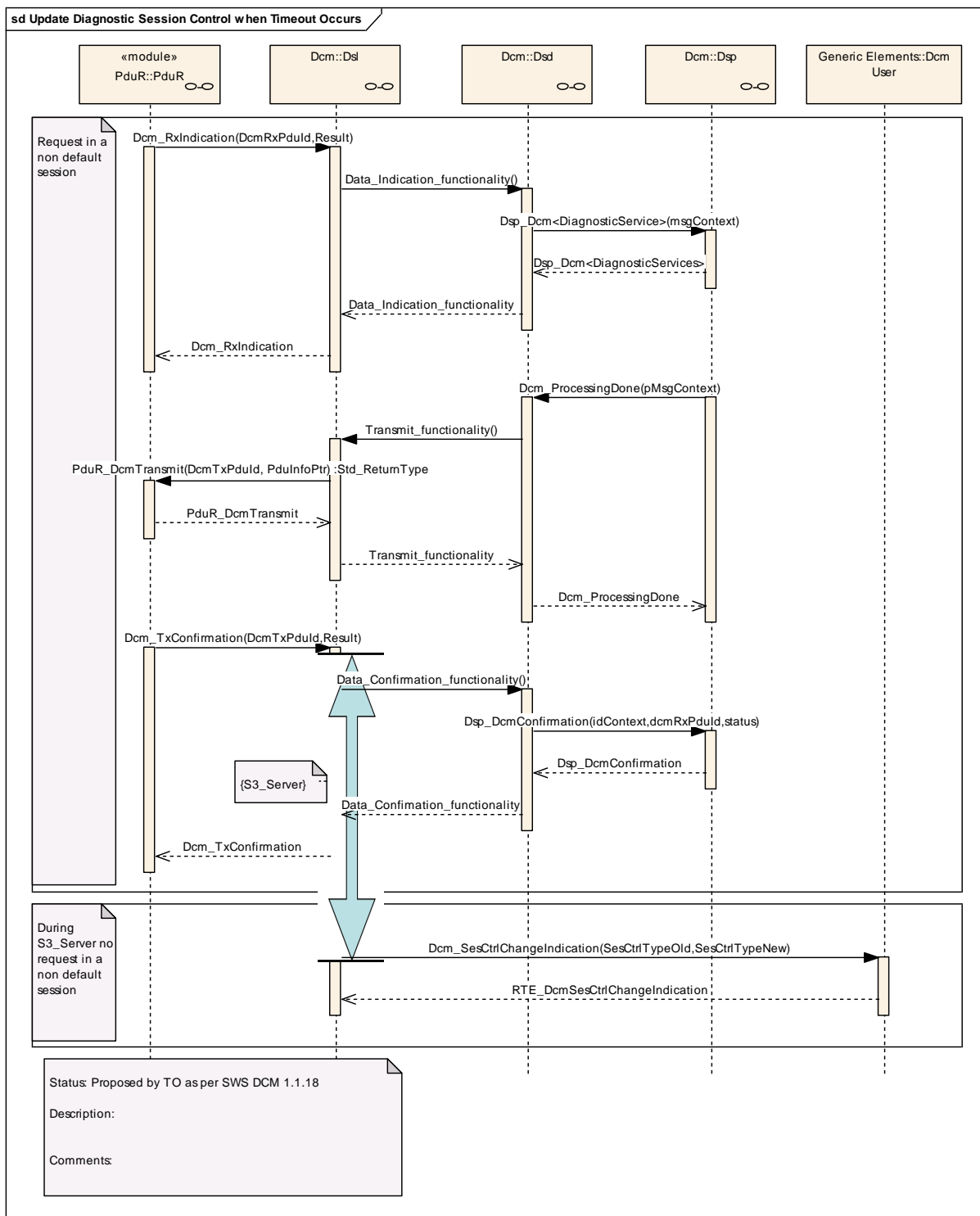
9.2.4 Process single response of ReadDataByPeriodicIdentifier
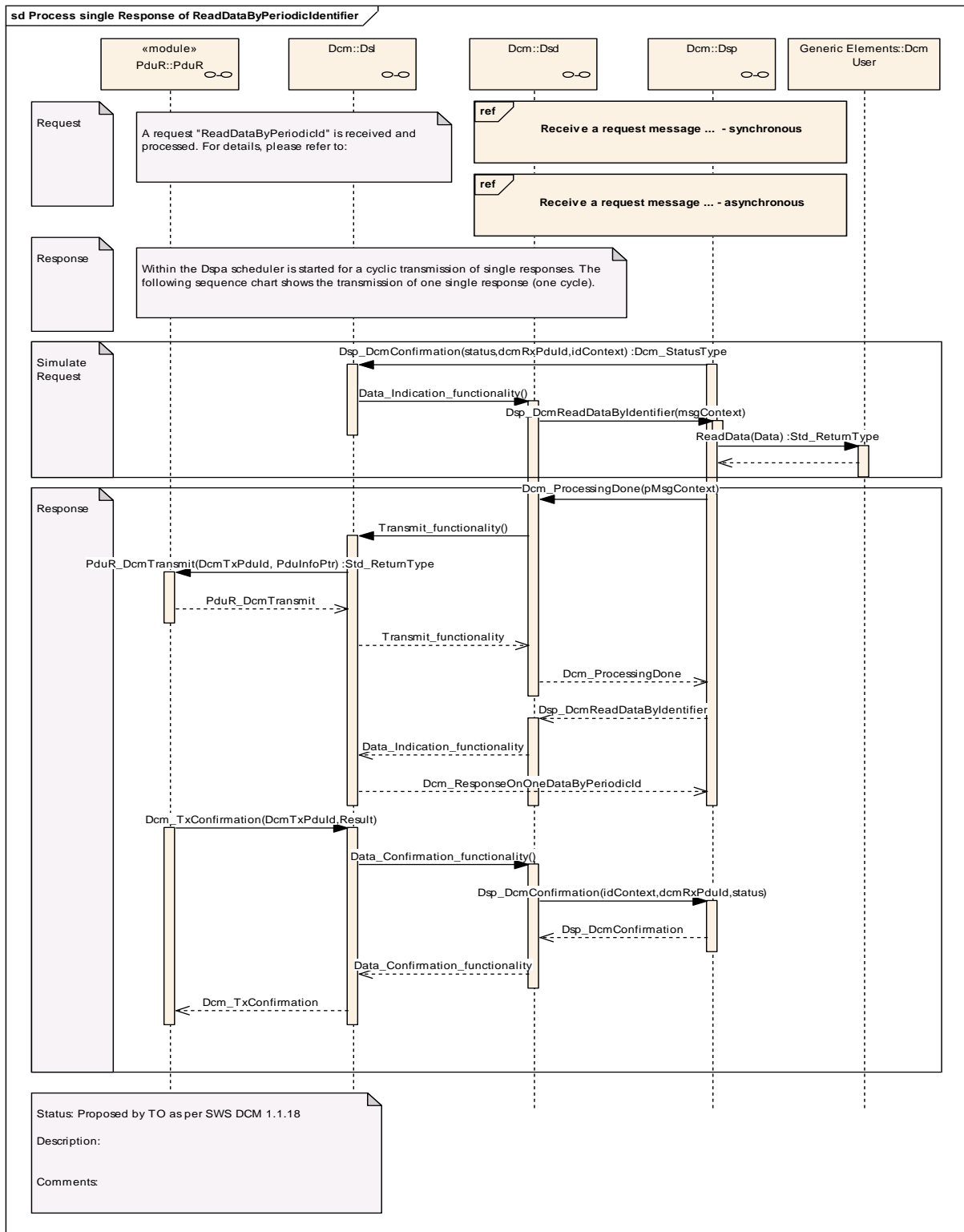
The DSP submodule requests sampling and transmission of Periodic Identifier data, when an event to Periodic Identifier occurs (i. e. a given time period is over). The DSP submodule initiates the sending of one periodic identifier calling the function ResponseOneDataByPeriodicId() provided by the DSL submodule.

Within this function the DSL submodule simulates a "ReadDataByIdentifier" request for the given PeriodicId. The High byte of the DataIdentifier shall be set to 0xF2 as specified in [11]) and the low byte is set to value of the PeriodicId.
The ReadData interface of the corresponding DID is called to get the DID value.
The DCM module is not able to receive another periodic identifier event request from the DSP submodule, unless the last periodic identifier event request is finished and the confirmation is received.

9.2.5 Process single event-triggered response of ResponseOnEvent
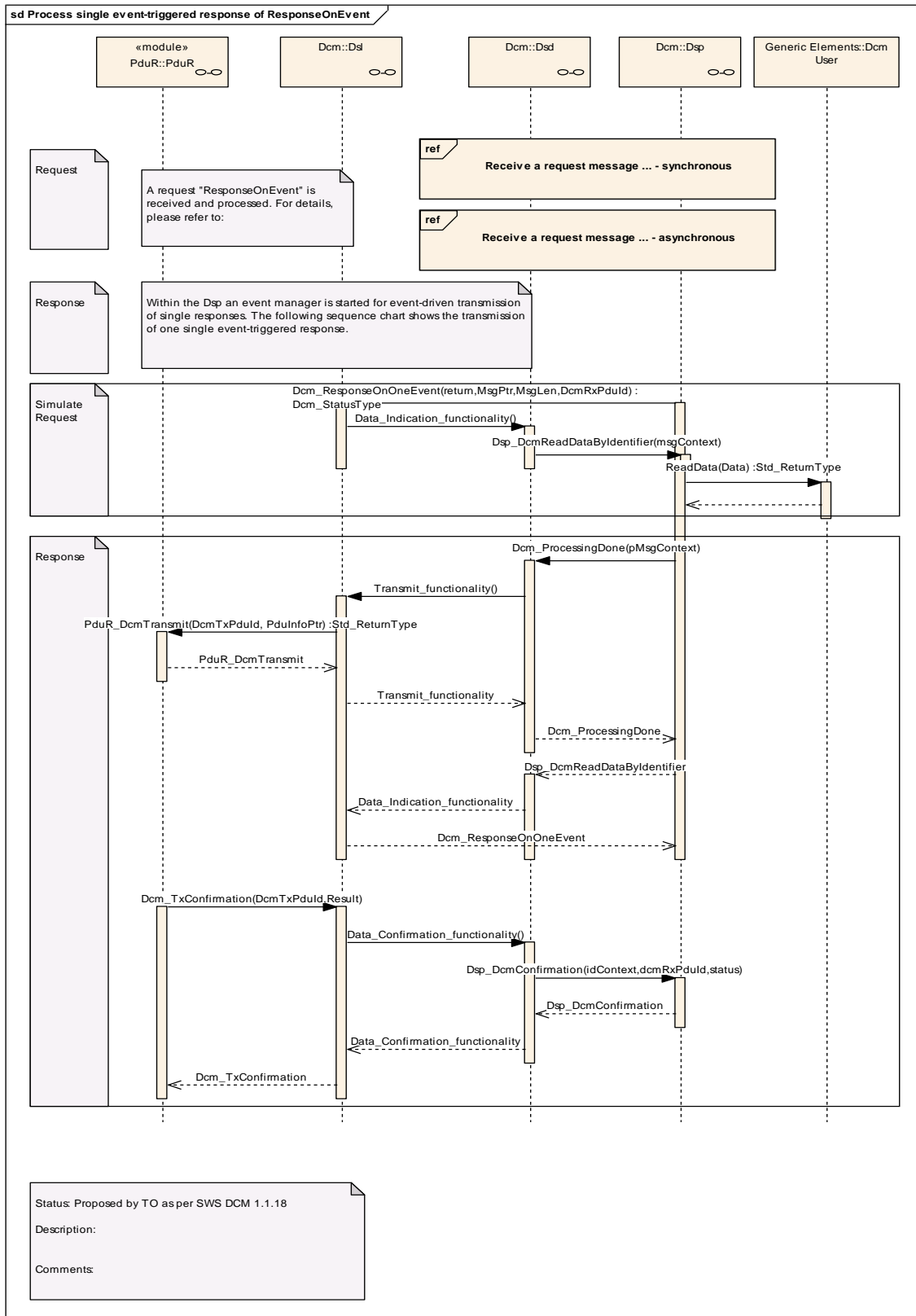
The DSP submodule requests transmission of ResponseOnEvent data, when ROE event occurs (`DslInternal_ResponseOnOneEvent()`).

Within this function the DSL submodule simulates a corresponding diagnostic request that has already been registered. This request will be processed by the DSP submodule.
The ReadData interface of the corresponding DID is called to get the DID value.
The DCM module will not be able to process multiple event-triggered responses at one time.

9.2.6 Process concurrent requests
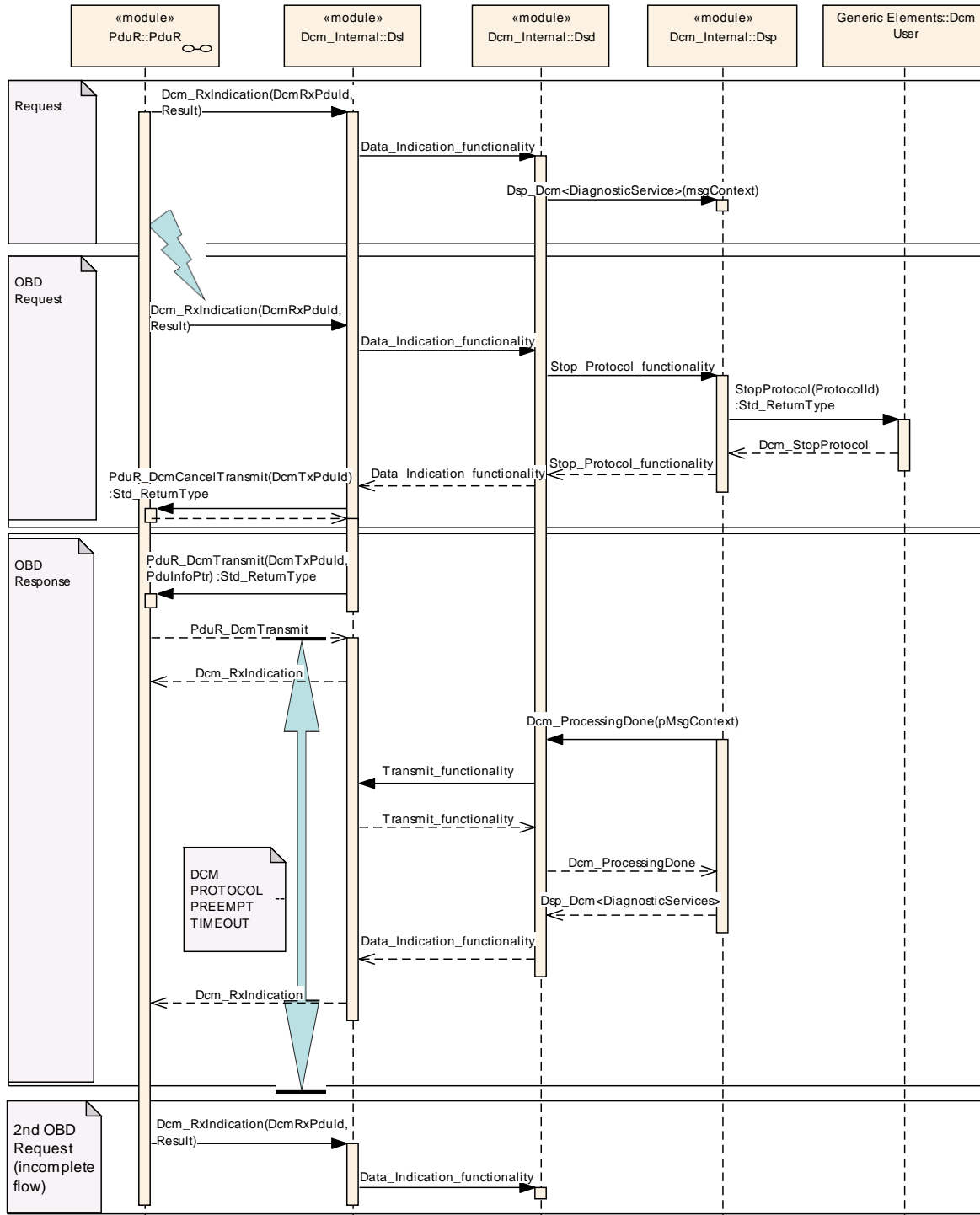
On reception of OBD request in parallel to processing of a normal diagnostic request (e.g enhanced diagnostic protocol, customer diagnostic protocol), running diagnostic request will be preempted. This is due to the configured higher priority of OBD protocol (see configuration parameter ***DcmDslProtocolPriority***).
The following is processed on reception of 1st OBD request:

- The Application is informed of the protocol stop (done with `Xxx_StopProtocol()`) and resets to a stable state (e.g. switch of digital IOs,..).
- Lower Layer is requested to cancel ongoing transmission on the same N-PDU (done with `PduR_DcmCancelTransmit()`).
- The DSL submodule responses with a negative response "BusyRepeatRequest" (NRC 0x21) to OBD tester..
- Timeout tracking of the Application finishes is started (timeout value configured in parameter ***DcmDslProtocolPreemptTimeout*** of the preempting protocol (here OBD protocol)).

As long as the DSP submodule is not finished (finish is indicated with `DsdInternal_ProcessingDone()`) or no timeout occurs, the DSL submodule responses with negative response "BusyRepeatRequest".

With receiving `DsdInternal_ProcessingDone()`, the DSL submodule will not transmit a response to old request. There will also not given any negative response to inform first tester about preemption of diagnostic request.

If the DSP submodule triggers no `DsdInternal_ProcessingDone()`, the DSL submodule runs into timeout and switches directly to further processing of preempting protocol.

9.2.7 Interface to ComManager

9.2.7.1 Handling in Default Session

9.2.7.2 Handling in Non-Default Session

sd Handling in Non-Default Session

Request
Dcm_RxIndication(DcmRxPduId,Result)
Dsp_Dcm<DiagnosticService>(msgContext)
Dsp_Dcm<DiagnosticServices>
Dcm_RxIndication

Response
Dcm_TxConfirmation(DcmTxPduId,Result)
Dsp_DcmConfirmation(idContext,dcmRxPduId,status)
Dsp_DcmConfirmation
Dcm_TxConfirmation

Status: Proposed by TO as per SWS DCM 1.1.18

Description:

Comments:
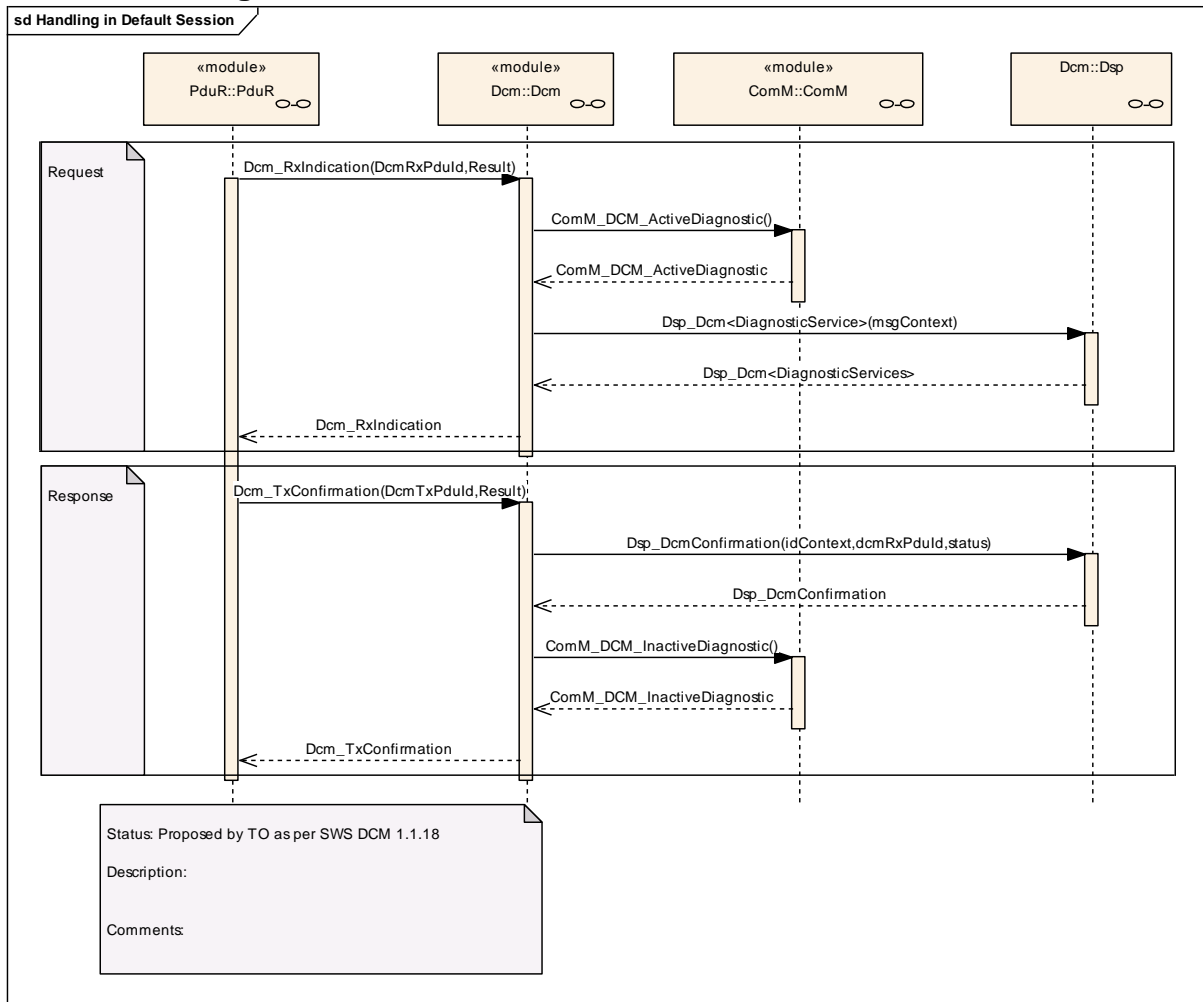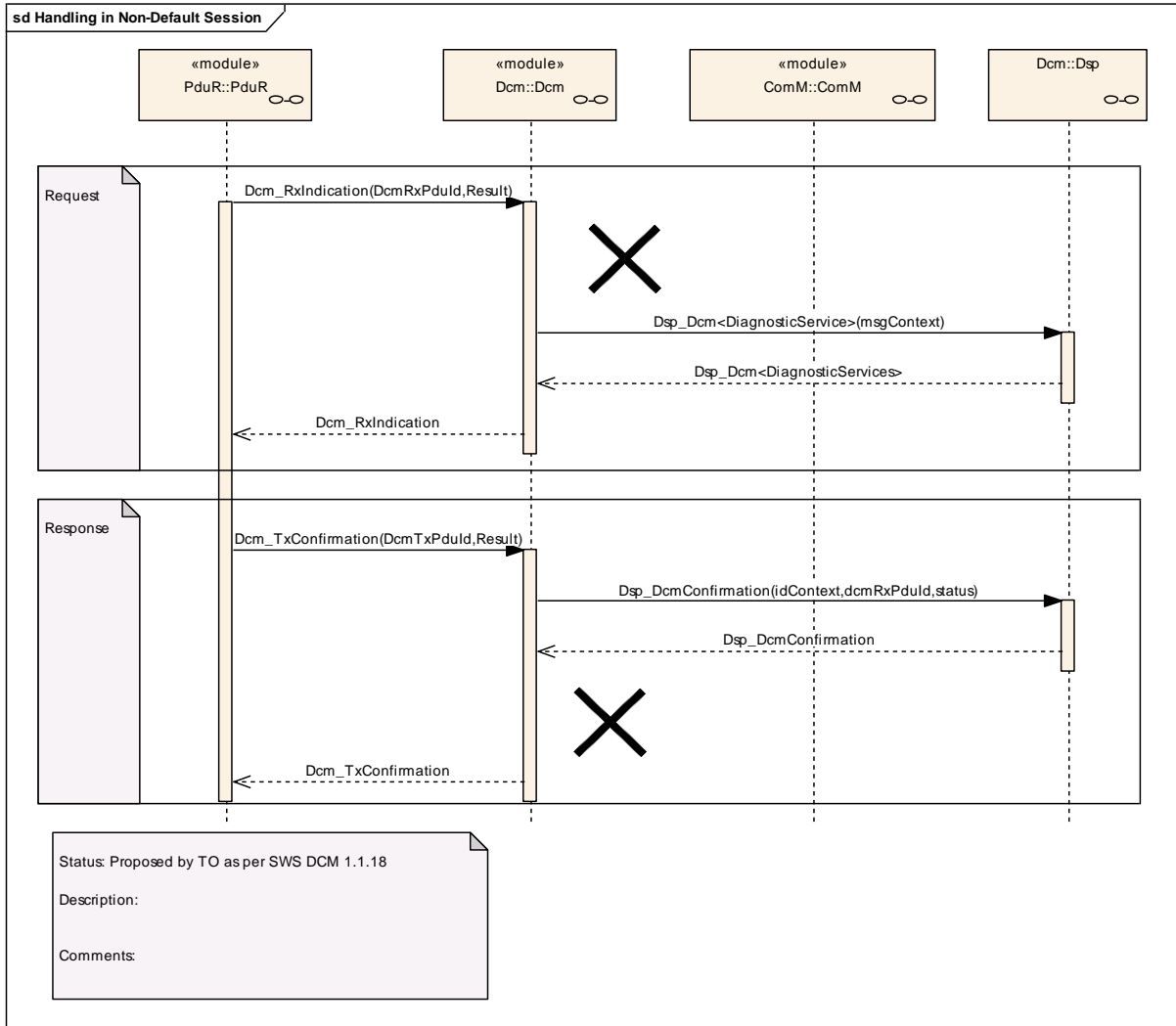
9.2.7.3 Session transitions
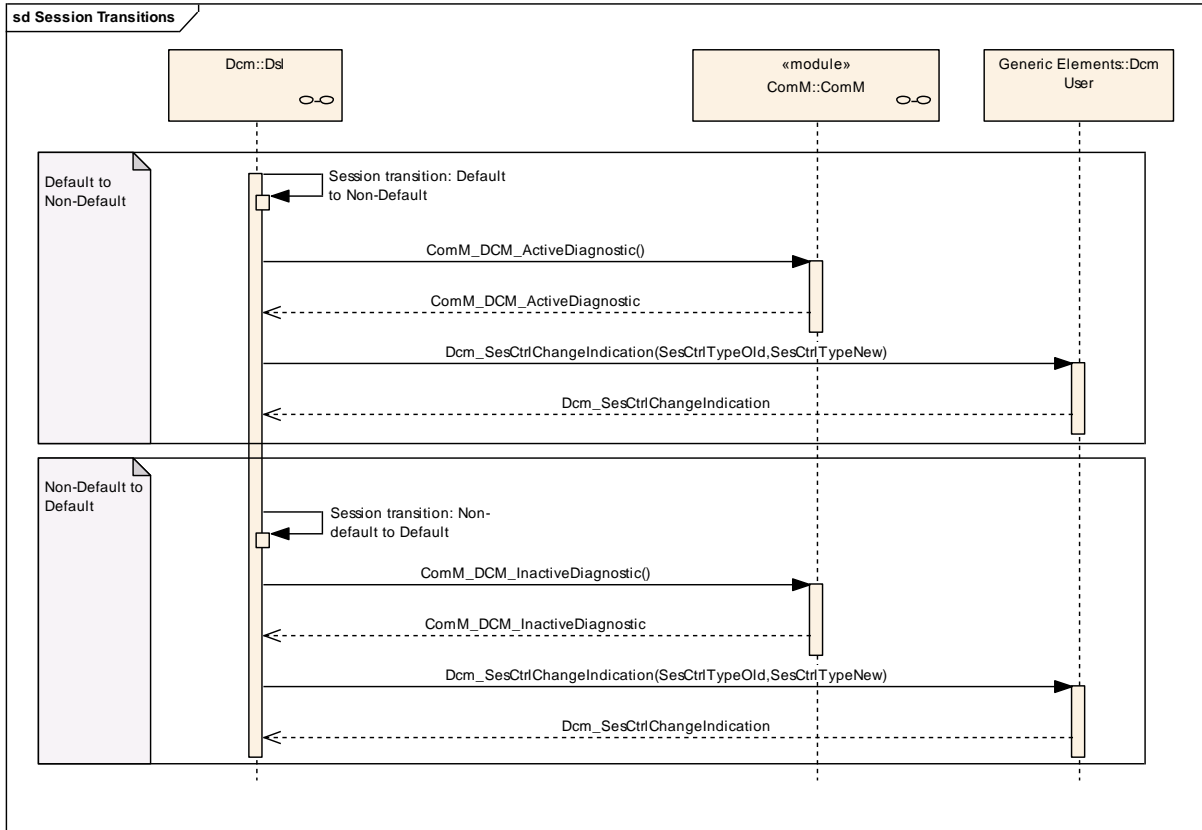
9.2.7.4 Communication States
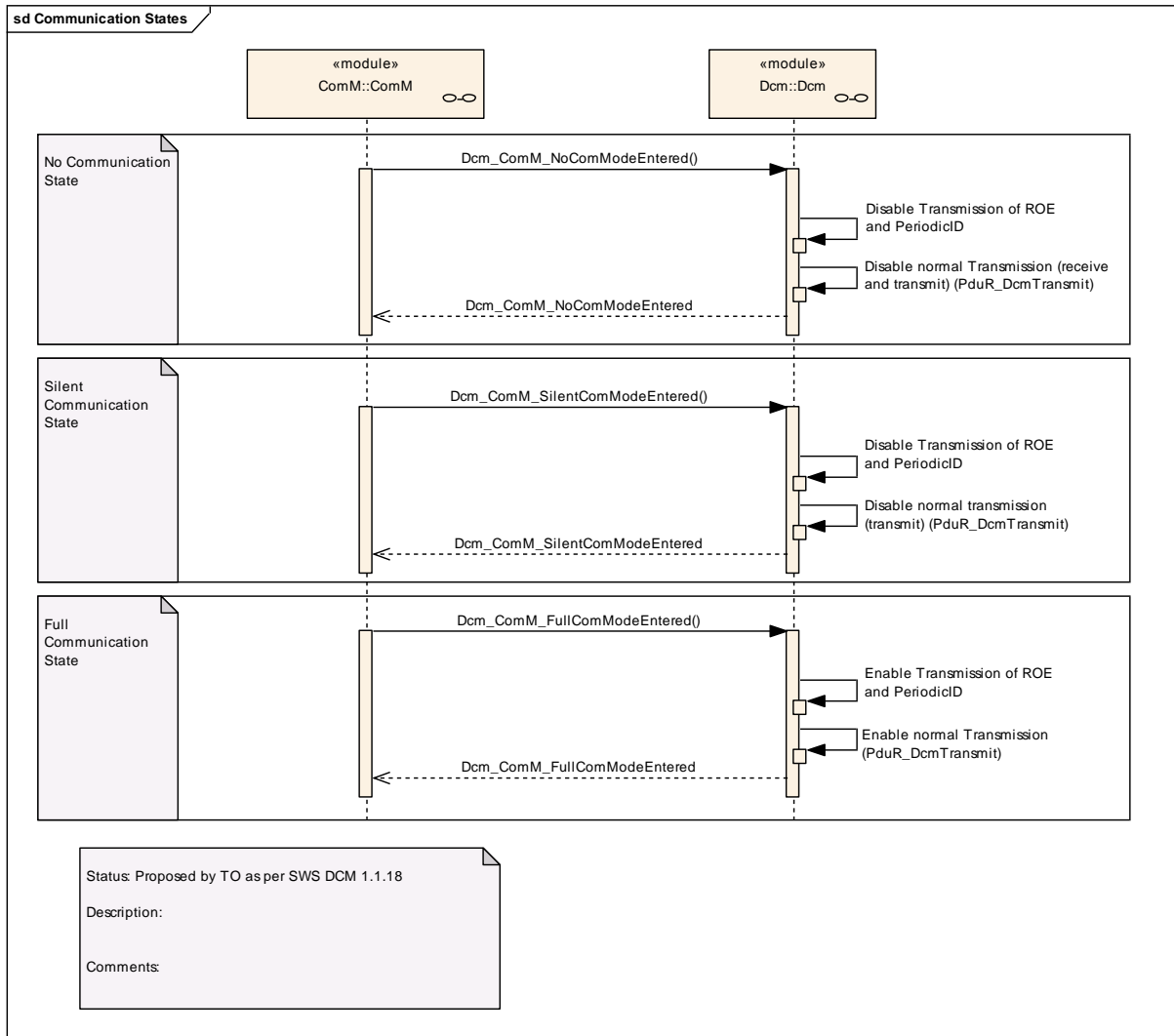
9.3 DSD (Diagnostic Service Dispatcher)

Receive a request message and transmit a positive response message – synchronous transmission
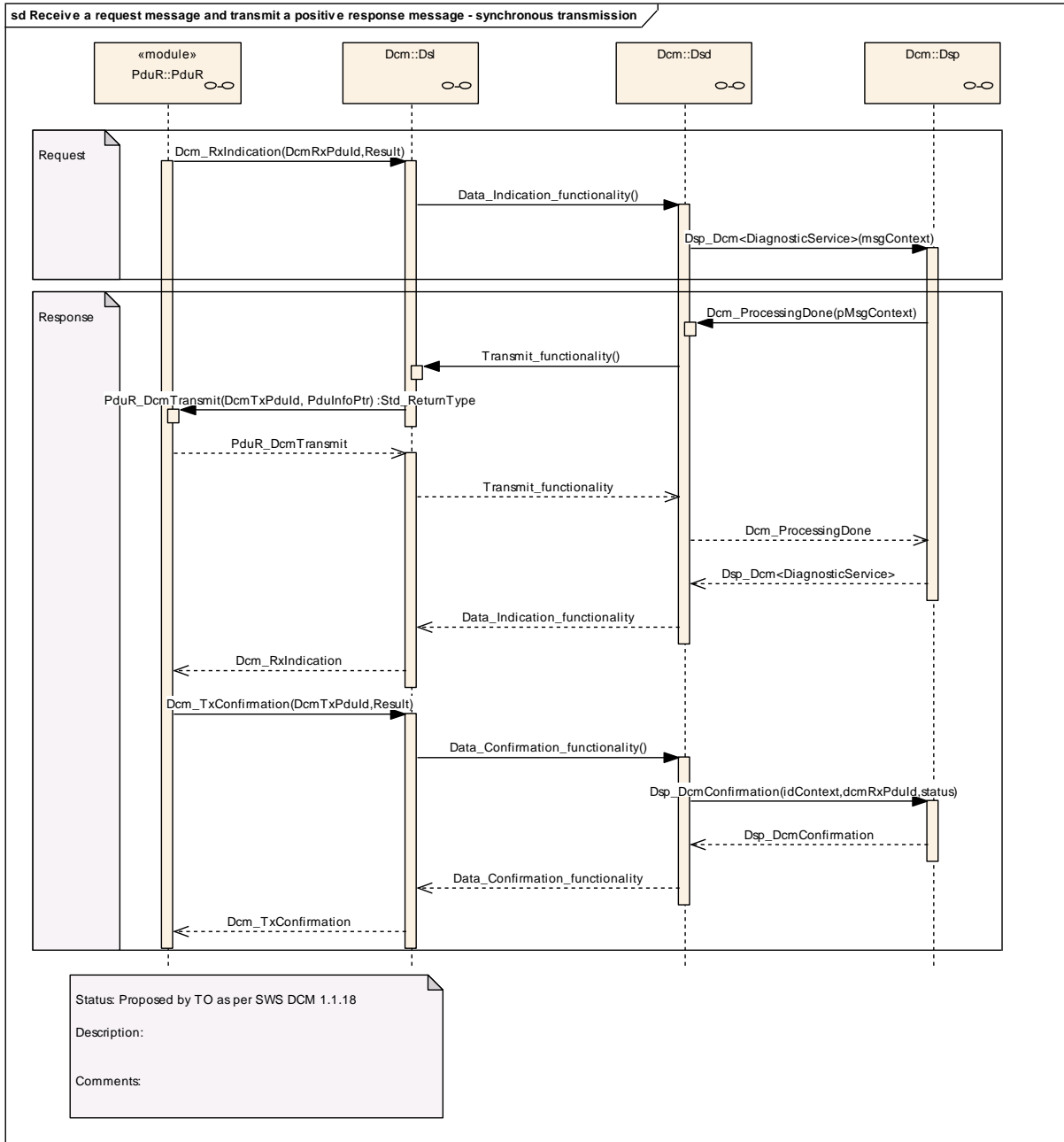
Receive a request message and transmit a positive response message –
asynchronous transmission

sd Receive a request message and transmit a positive response message - asynchronous transmission

Receive a request message and suppress a positive response
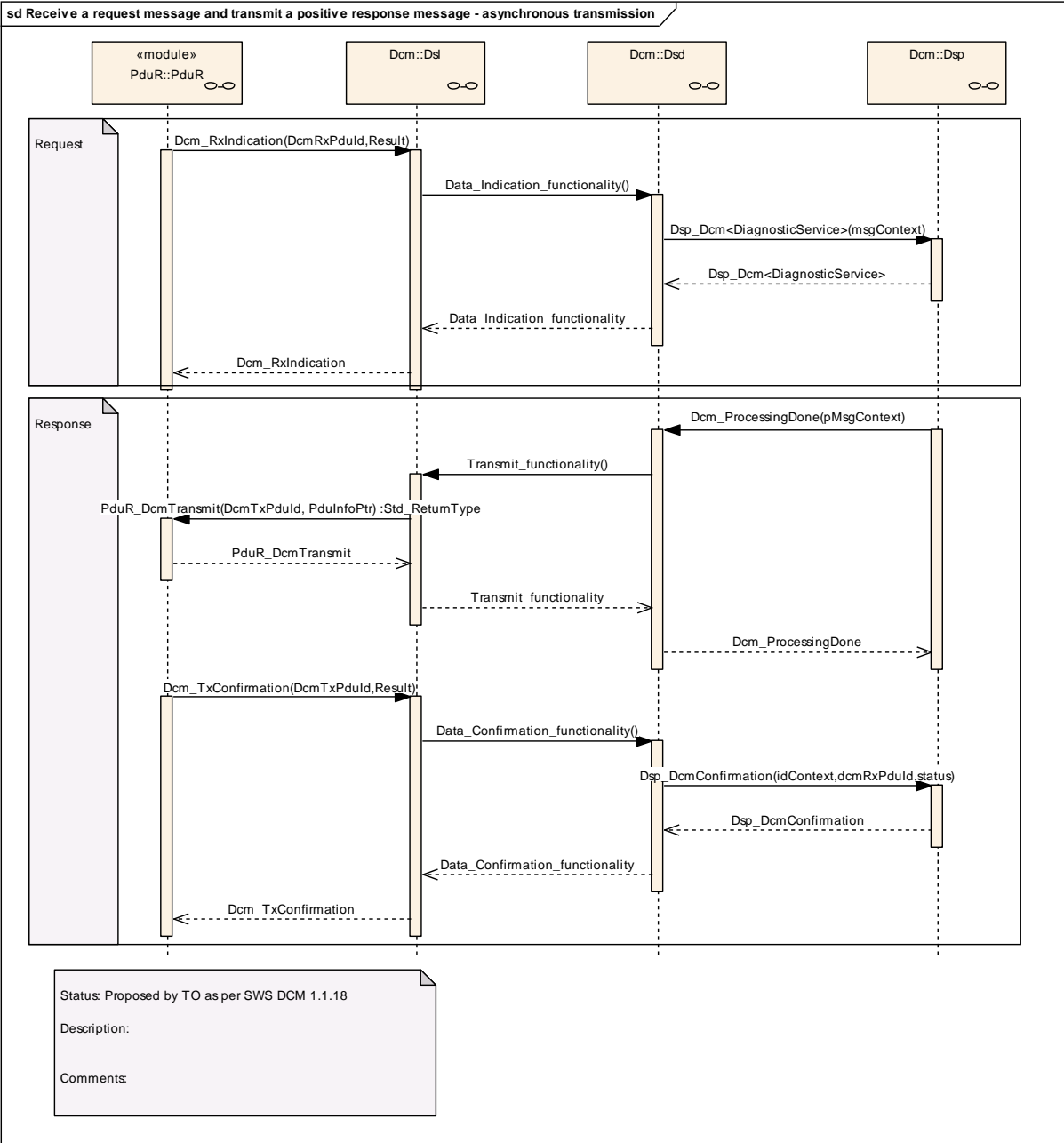
sd Receive a request message and suppress a positive response
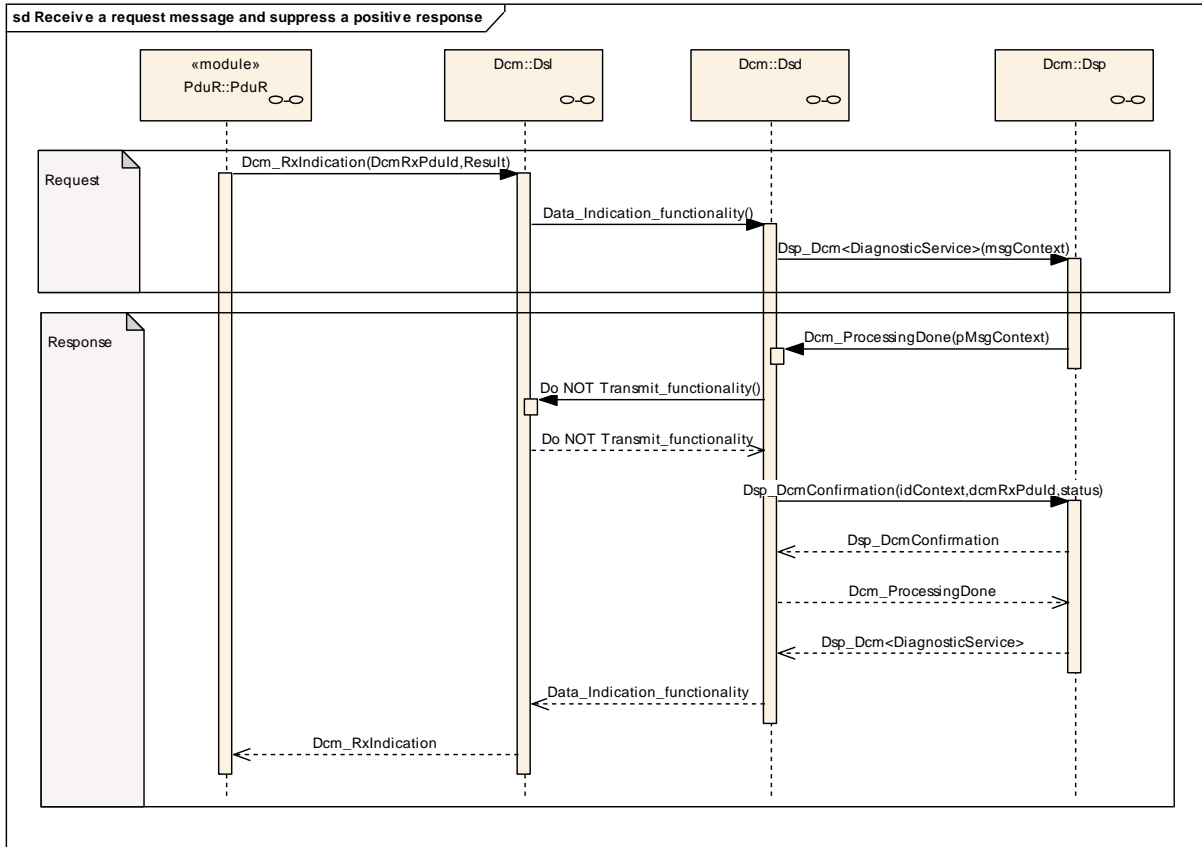
9.3.1 Receive request message and transmit negative response message
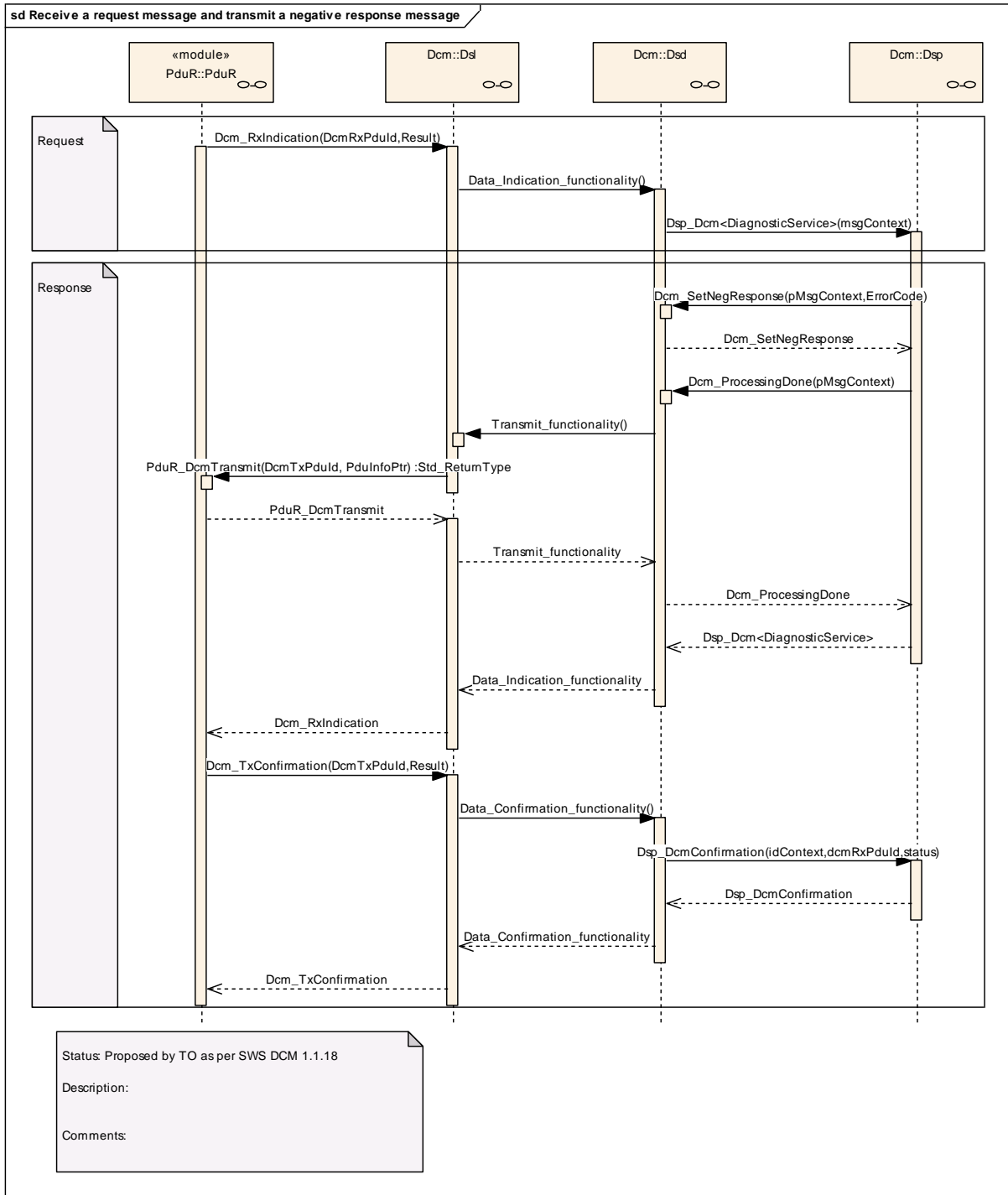
9.3.2 Process Service Request with paged-buffer
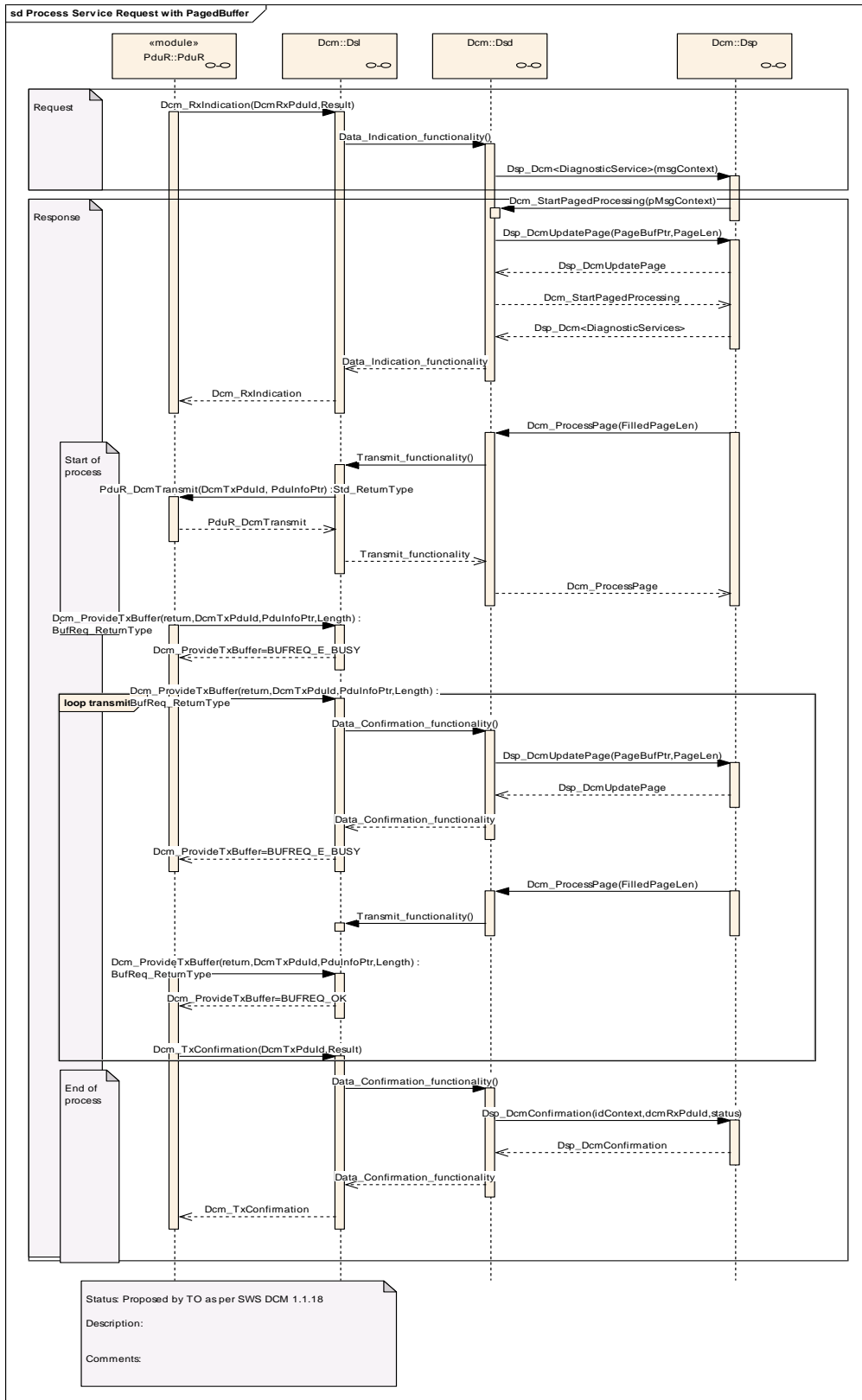
The following flow is processed in case no error occurs on the Application side:

Start of process:

4) `DsdInternal_StartPagedProcessing()`: With this API, the DSP submodule gives the complete response length to the DCM module and starts paged-buffer handling. This API starts no transmission!

5) UpdatePage(): The DCM module requests data to be transmitted.

6) `DsdInternal_ProcessPage()`: With this API, the DSP submodule requests transmission of the current page.

8) `PduR_DcmTransmit()`: The DCM module requests transmission to the lower layers.

9) `Dcm_ProvideTxBuffer()`: The buffer is filled and the DCM module shall return "BUFREQ_OK"(10).

Start of the loop:

11) `Dcm_ProvideTxBuffer()`: The PduR module requests the buffer but the buffer is not filled by the DSP submodule.

12 + 13) UpdatePage: The DCM module requests the DSP submodule to fill the next page.

14) By returning "BUFREQ_E_BUSY", the DCM module indicates that the buffer has to be filled by the DSP submodule.

15) `DsdInternal_ProcessPage()`: With this API, the DSP submodule requests transmission of the current page.

17) Then, on the next call of `Dcm_ProvideTxBuffer()` the buffer is filled and the DCM module shall return "BUFREQ_OK" (18).

LOOP: The flow 10 to 18 is repeated as long data can be sent.

End of the loop:

n-2 -> n) `Dcm_TxConfirmation()` When all data is send, the PduR module indicates the sending with a confirmation, which is given to the DSP submodule.

The APIs 4, 5 and 6 are needed only for paged-buffer transmission.

Page buffer timeout handling:

The DCM module reacts in the following described way, when the DSP submodule starts paged-buffer handling, but is not able to process further on filling the response data. E.g. there are problems to access data from an EEPROM device.

When providing the Pagebuffer to the DSP submodule (13: UpdatePage()), the DCM module starts a timeout supervision. If timeout (value configured in **DcmPagedBufferTimeout**) occurs, before the DSP submodule requests next page (`DsdInternal_ProcessPage()`) following error handling is carried out in the DCM module:

- The DCM module stops further processing of paged-buffer (item 15),
- The DCM module requests the DSP submodule (14: `DspInternal_CancelPagedBufferProcessing()`) to stop further processing of PagedBuffer, and
- The DCM module will cancel ongoing transmission in lower layers (done with return value BUFREQ_E_NOT_OK in next `Dcm_ProvideTxBuffer()` request, item 17).

sd Process Service Request with PagedBuffer and timeout
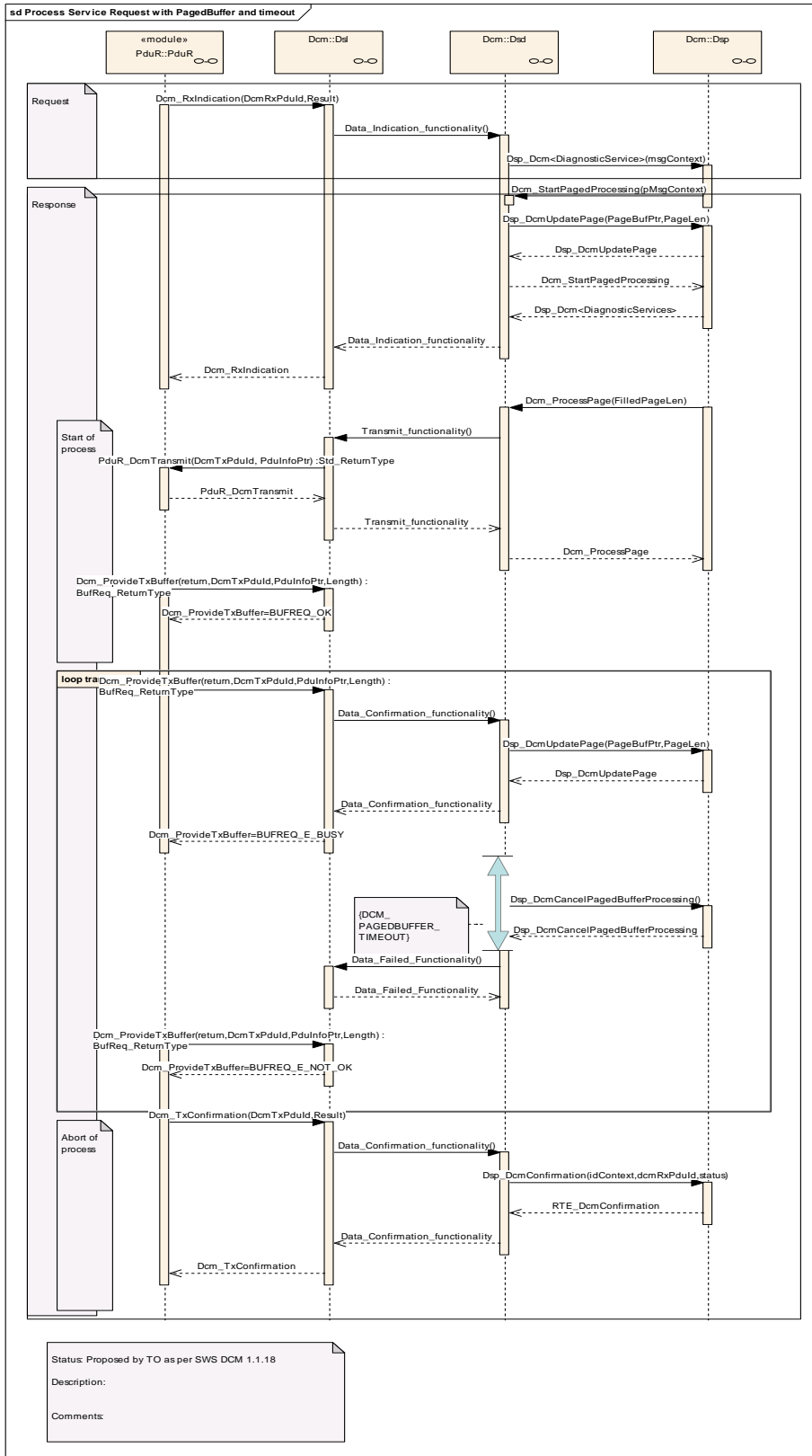
9.4 DSP (Diagnostic Service Processing)

9.4.1 Interface DSP – DEM (service 0x19, 0x14, 0x85)

Please refer to Section 9. in [6].

9.4.2 Interface special services

9.4.2.1 Process Diagnostic Session Control
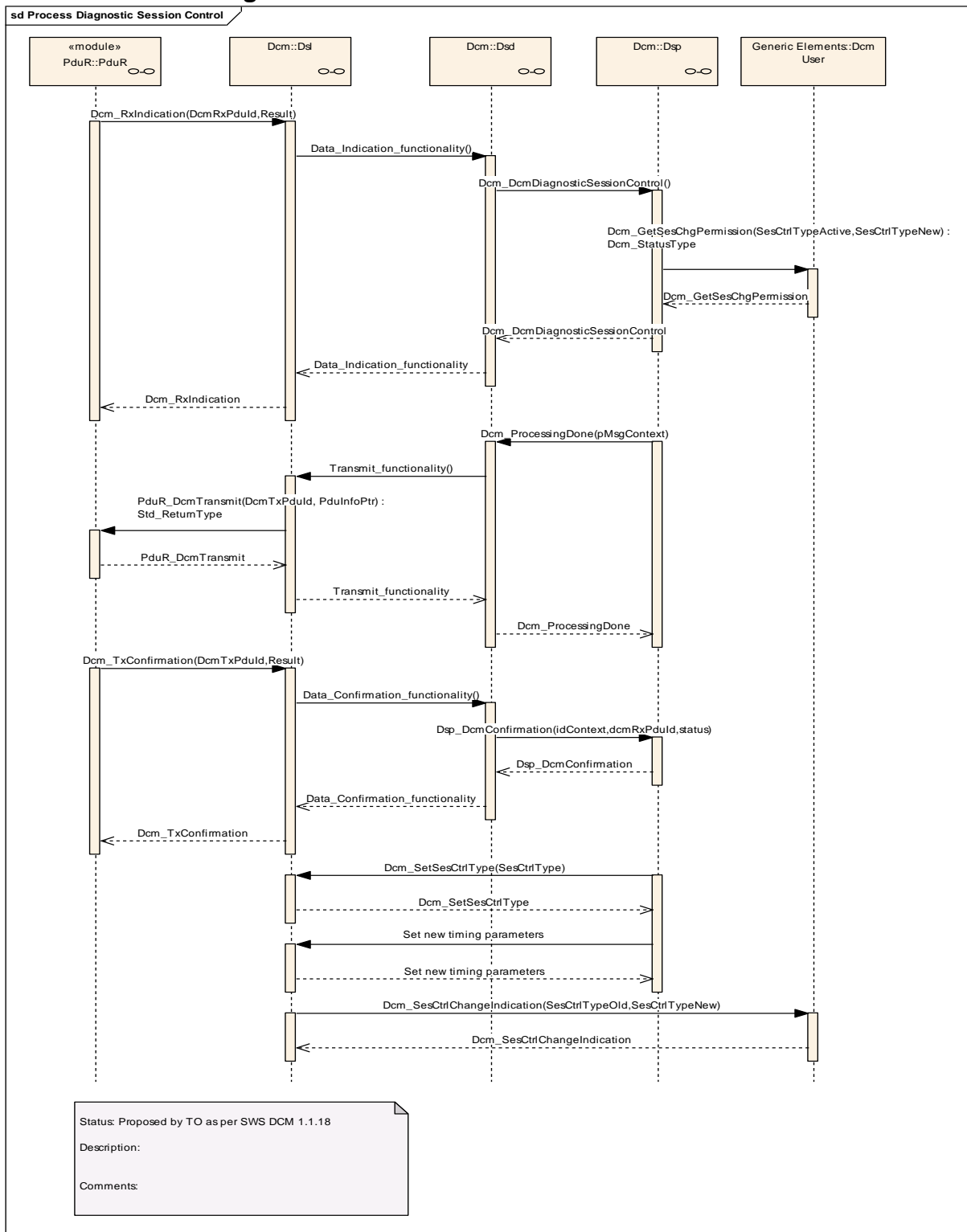
On a Diagnostic Session Control request from a tester, the DSP submodule requests the Application to get the permission to change the session.
With the permission from the Application, a positive response is given to the PduR module. In the data confirmation function the new session type and the new timing values are set.

The DSL submodule indicates the session change to the Application
(`Xxx_ChangeIndication()`).

9.4.2.2 Process Tester Present
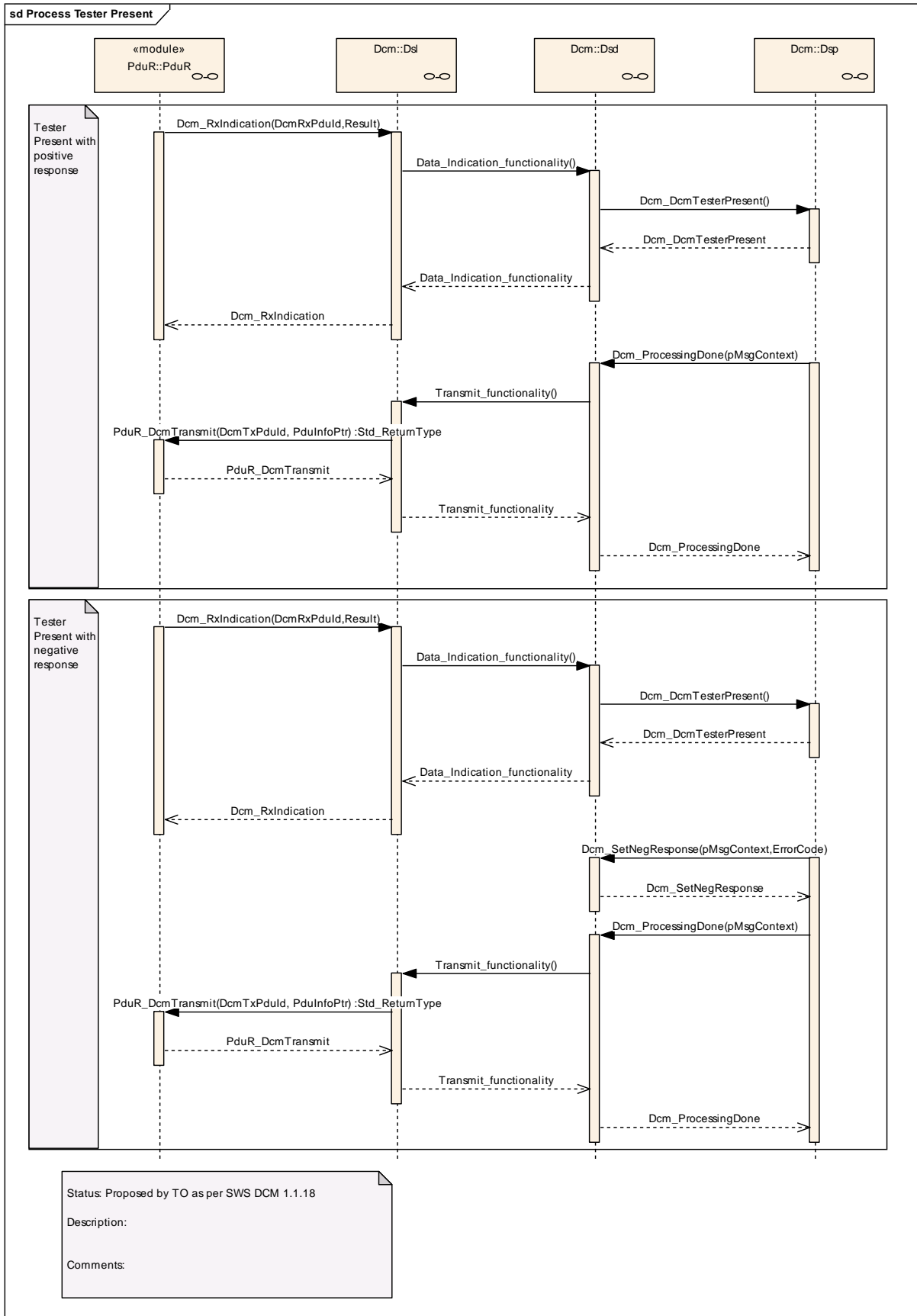
Above sequence diagram shows processing of TesterPresent commands, which are not of type functional addressed with subfunction 0x80. These TesterPresent commands are interpreted in the DSL submodule (more details can be found in Section 7.2.4.3 Concurrent "TesterPresent" ("keep alive logic"))
All the other TesterPresent commands are processed in the following way:
On a command TesterPresent the DSD submodule calls the DSP submodule with the function TesterPresent(). The sequence chart also shows the case when an error occurs and a negative response is sent.

9.4.2.3 Process Security Access
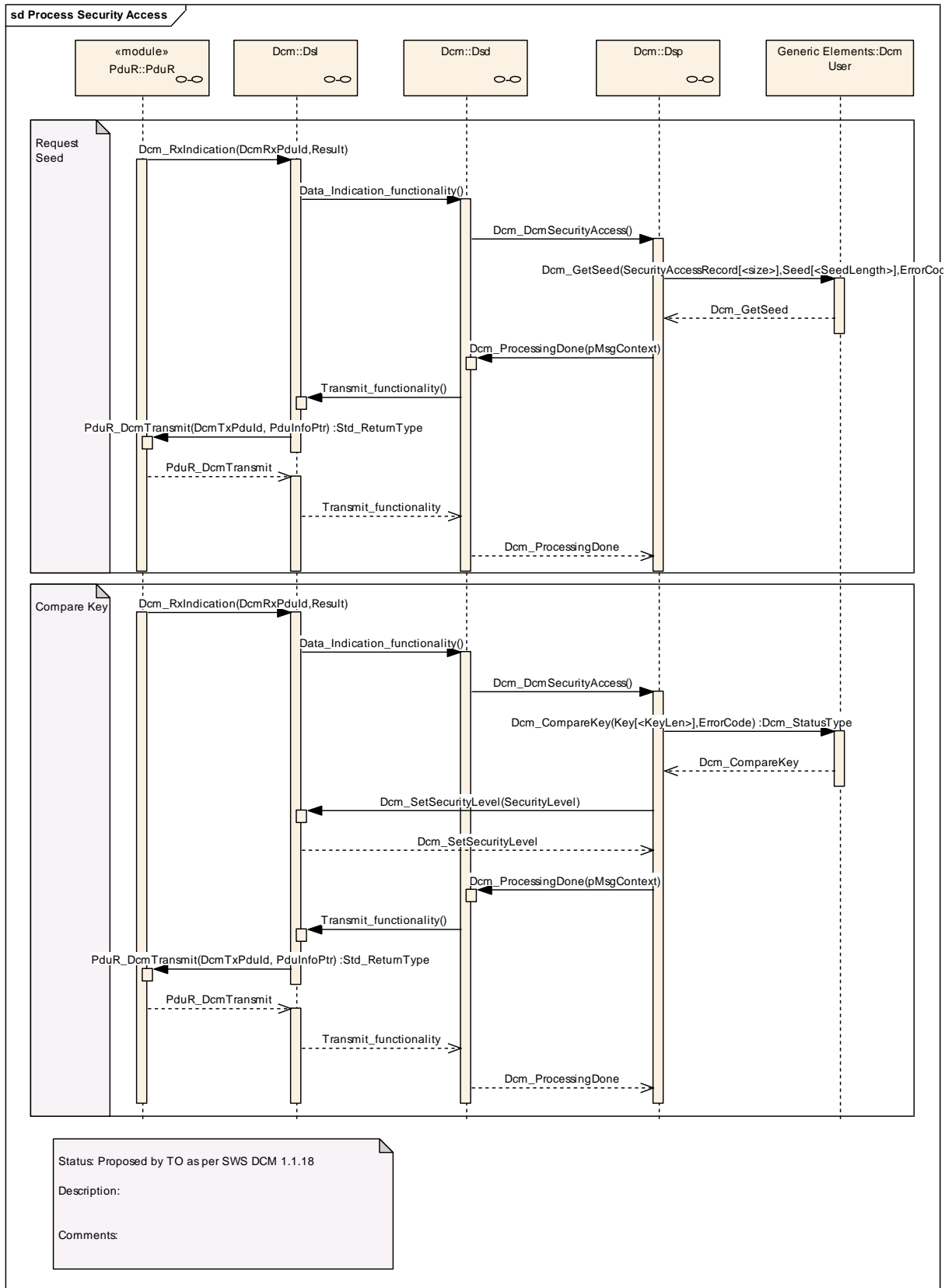
To get the security access, the DSD submodule has to call the DSP submodule to get the seed value from the application. If no error is detected, the seed value is sent in the positive response.

In a second step, the DSP submodule gets the key calculated by the tester and requests the application to compare this key with the internal calculated key. If no error occurs, the new access type is set in the DSL submodule and a positive response is sent.

10 Configuration specification

10.1 How to read this section

In addition to this section, it is highly recommended to read the documents:
- AUTOSAR Layered Software Architecture [5]
- AUTOSAR ECU Configuration Specification [6]. 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.

Pre-compile time   specifies whether the configuration parameter shall be of configuration class Pre-compile time or not.

Label	Description
x	The configuration parameter shall be of configuration class Pre-compile time.
--	The configuration parameter shall never be of configuration class Pre-compile time.

Link time   specifies whether the configuration parameter shall be of configuration class Link time or not.

Label	Description
x	The configuration parameter shall be of configuration class Link time.
--	The configuration parameter shall never be of configuration class Link time.

Post Build   specifies whether the configuration parameter shall be of configuration class Post Build or not.

Label	Description
x	The configuration parameter shall be of configuration class Post Build and no specific implementation is required.
L	Loadable - the configuration parameter shall be of configuration class Post Build and only one configuration parameter set resides in the ECU.
M	Multiple - the configuration parameter shall be of configuration class Post Build and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class Post

Label	Description
	Build.

10.1.2 Variants

The DCM module has the following variants:

Dcm171: Variant A:This variant is limited to pre-compile configuration parameters only.

Dcm172: Variant B:This variant is limited to link time configuration parameters only. Variant B shall be used when the DCM implementation is delivered in object code only.

Dcm173: Variant C: This variant allows a mix of pre-compile-time- and post-build-time- configuration parameters.

In variant C the pre-compile parameter shall be used to enable or disable the functionality (e.g. ROE transmission DCM_ROE_ENABLED). With the post build parameter the functionality shall be configured (e.g. DCM_ROE_TRANS_TYPE). Please note: This pre-compile configuration parameters are mandatory for all variants. The pre-compile parameter shall be used to enable or disable DCM functionality (e.g. ROE transmission DCM_ROE_ENABLED) or are DCM global configuration data that shall be configured at pre complile time (e.g. memory influence).

10.1.3 Containers

The 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.

Containers have names indicating what kinds of parameters are handled inside.

10.2 DCM configurations

10.2.1 Dcm

Module Name	*Dcm*
Module Description	Configuration of the Dcm (Diagnostic Communications Manager) module.

Included Containers		
Container Name	*Multiplicity*	*Scope / Dependency*
DcmConfigSet	1	This container contains the configuration parameters and sub

		containers of the DCM module supporting multiple configuration sets. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
DcmDsd	1	These parameters apply to Diagnostic Service Dispatcher. All parameters for all service dispatchers are included in this one configuration container. upperMultiplicity: Service dispatcher is multiple times configurable in this one configuration container. lowerMultiplicity: There must always be one service dispatcher in a DCM.
DcmGeneral	1	This container contains the configuration (parameters) for Component wide parameters
DcmPageBufferCfg	1	This container contains the configuration (parameters) for Page Buffer handling

10.2.2 DcmDsd

SWS Item	ECUC_Dcm_00688 :
Container Name	DcmDsd
Description	These parameters apply to Diagnostic Service Dispatcher. All parameters for all service dispatchers are included in this one configuration container. upperMultiplicity: Service dispatcher is multiple times configurable in this one configuration container. lowerMultiplicity: There must always be one service dispatcher in a DCM.
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDsdServiceTable	1..256	This container contains the configuration (DSD parameters) for Service Identifier Table.

10.2.3 DcmDsdServiceTable

SWS Item	ECUC_Dcm_00732 :	
Container Name	DcmDsdServiceTable{SERVICE_IDENTIFIER_TABLE}	
Description	This container contains the configuration (DSD parameters) for Service Identifier Table.	
Configuration Parameters		

SWS Item	ECUC_Dcm_00736 :		
Name	DcmDsdSidTabId {DCM_SIDTAB_ID}		
Description	Due the fact of using one or more service tables the member of the Service Identifier Table includes a unique id for the Service Identifier Table.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME

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

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDsdService	1..*	This container contains the configuration (DSD parameters) for Service.

10.2.4 DcmDsdService

SWS Item	ECUC_Dcm_00689 :
Container Name	DcmDsdService
Description	This container contains the configuration (DSD parameters) for Service.
Configuration Parameters	

SWS Item	Dcm777 :		
Name	DcmDsdSidTabFnc		
Description	Callback function of the DSP or ECU Supplier specific component for the particular DcmDsdSidTabServiceId. This parameter is related to the interface DspInternal_<DiagnosticService>.		
Multiplicity	1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_00735 :		
Name	DcmDsdSidTabServiceId {DCM_SIDTAB_SERVICEID}		
Description	Id of the Service identifier in hex. The possible Service identifier are predefined in the ISO 14229-1 and ISO 15031-5 and in Table 5 and Table 6.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_00737 :
Name	DcmDsdSidTabSubfuncAvial {DCM_SIDTAB_SUBFUNC_AVAIL}
Description	Information whether the DcmDsdSidTabServiceId includes Sub functions or not. Used for the Handling of "suppressPosRspMsgIndicationBit" ISO14229-1 can be referenced here, as this specification gives fix definition, if an SID includes Subfunction or not. true = sub-function available false = sub-function not available
Multiplicity	1
Type	BooleanParamDef
Default value	--

Document ID 018: AUTOSAR_SWS_DCM

ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDsdSidTabServiceId		

SWS Item	ECUC_Dcm_00733 :		
Name	DcmDsdSidTabSecurityLevelRef {DCM_SIDTAB_SEC_LEVEL_ROW}		
Description	Link to the Security Access Levels needed for execution of the DcmDsdService. Please refer to the ISO 14229-1, ISO 15031-5 and "Verification of the Security Access levels". Please note, that it shall be provided to configure several DcmDsdSidTabSecurityLevelRef per DcmDsdService.		
Multiplicity	1..*		
Type	Reference to [DcmDspSecurityRow]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_00734 :		
Name	DcmDsdSidTabSessionLevelRef {DCM_SIDTAB_SESSION_LEVEL_ROW}		
Description	Link to the Session Control needed for execution of the DcmDsdService. Please refer to the ISO 14229-1, ISO 15031-5 and "Verification of the Diagnostic Session". Please note, that it shall be provided to configure several DcmDsdSidTabSessionLevelRef per DcmDsdService.		
Multiplicity	1..*		
Type	Reference to [DcmDspSessionRow]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.5 DcmConfigSet

SWS Item	Dcm819_Conf :
Container Name	DcmConfigSet [Multi Config Container]
Description	This container contains the configuration parameters and sub containers of the DCM 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
DcmDsl	1	These parameters apply to a Diagnostic Session Layer. There

		may be a parameter set (DSL Configuration) per protocol. upperMultiplicity: Each DCM configuration must have exactly one DSL configuration. lowerMultiplicity: Each DCM configuration must have exactly one DSL configuration.
DcmDsp	1	These parameters apply to Diagnostic Service Processing. There will always be one set of these parameters per DCM.

10.2.6 DcmDsl

SWS Item	ECUC_Dcm_00690 :
Container Name	DcmDsl
Description	These parameters apply to a Diagnostic Session Layer. There may be a parameter set (DSL Configuration) per protocol. upperMultiplicity: Each DCM configuration must have exactly one DSL configuration. lowerMultiplicity: Each DCM configuration must have exactly one DSL configuration.
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDslBuffer	1..256	This container contains the configuration (parameters) for the diagnostic buffer.
DcmDslCallbackDCMRequestService	1..*	The name of this container is the name of the R-Port through which the DCM requires the interface CallBackDCMRequestServices. This name should adhere to the naming convention defined in the DCM SWS.
DcmDslDiagResp	1	This container contains the configuration (parameters) for the ResponsePending handling
DcmDslProtocol	1	This container contains the configuration (parameters) for the protocol configuration (for each protocol) The following parameters needs to be configured per protocol.
DcmDslProtocolTiming	1	This container contains the configuration (parameters) for Protocol timing. This container contains rows of DcmDslProtocolTimingRow containers. Hint: DcmDslProtocolTiming is only relevant in case of using the UDS service AccessTimingParameter.
DcmDslServiceRequestIndication	0..*	The name of this container is the name of the port through which the DCM requires the port-interface ServiceRequestIndication. The lowerMultiplicity is 0: If DcmRequestIndicationEnabled = false the Indication API is not available.
DcmDslSessionControl	1..*	The name of this container is the name of the R-port through which the DCM requires the PortInterface SessionControl

10.2.7 DcmDslBuffer

SWS Item	ECUC_Dcm_00739 :

Container Name	DcmDslBuffer{DIAGNOSTIC_BUFFER_CFG}
Description	This container contains the configuration (parameters) for the diagnostic buffer.
Configuration Parameters	

SWS Item	DCM032 :		
Name	DcmDslBufferID {DCM_BUFFER_ID}		
Description	Identifier of Buffer.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_00738 :		
Name	DcmDslBufferSize {DCM_BUFFER_SIZE}		
Description	Size of Diagnostic Buffer (in Bytes) For a linear buffer: size of the buffer shall be as large as the longest message (request or response) For a paged buffer (only Tx possible): size has impacts on the application performance Please note: max. range is the valid range for a CAN network. We assume a FlexRay (or other networks) implementation will work with this range (and the page buffer mechanism) without any problems.		
Multiplicity	1		
Type	IntegerParamDef		
Range	8 .. 4095		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.8 DcmDslCallbackDCMRequestService

SWS Item	DCM679 :
Container Name	DcmDslCallbackDCMRequestService
Description	The name of this container is the name of the R-Port through which the DCM requires the interface CallBackDCMRequestServices. This name should adhere to the naming convention defined in the DCM SWS.
Configuration Parameters	

No Included Containers

10.2.9 DcmDslDiagResp

SWS Item	ECUC_Dcm_00691 :
Container Name	DcmDslDiagResp{DIAGNOSTIC_RESPPEND_CFG}
Description	This container contains the configuration (parameters) for the

	ResponsePending handling

Configuration Parameters

SWS Item	ECUC_Dcm_01032 :		
Name	DcmDslDiagRespForceRespPendEn		
Description	Allow to enable (TRUE) or disable (FALSE) the mechanism of directly triggering of ResponsePending by application.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	ECUC_Dcm_00693 :		
Name	DcmDslDiagRespMaxNumRespPend {DCM_NUM_MAX_RESPPEND}		
Description	Maximum number of negative responses with response code 0x78 (requestCorrectlyReceived-ResponsePending) allowed per request DCM will send a negative response with response code 0x10 (generalReject), in case the limit value gets reached.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.10 DcmDslProtocol

SWS Item	ECUC_Dcm_00694 :
Container Name	DcmDslProtocol{DIAGNOSTIC_PROTOCOL_TABLE}
Description	This container contains the configuration (parameters) for the protocol configuration (for each protocol) The following parameters needs to be configured per protocol.

Configuration Parameters

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDslProtocolRow	1..*	Definition of a single Row of configuration for the protocol configuration (for each protocol)

10.2.11 DcmDslProtocolRow

SWS Item	ECUC_Dcm_00695 :
Container Name	DcmDslProtocolRow{DIAGNOSTIC_PROTOCOL_TABLE_ROW}

Description	Definition of a single Row of configuration for the protocol configuration (for each protocol)
Configuration Parameters	

SWS Item	ECUC_Dcm_01043 :			
Name	DcmDslProtocolBlockSize			
Description	Allow to configure the size returned for the first call of Dcm_ProvideRxBuffer. This size will by used by the TP to calculate the Block size. When this parameter is not configured, the buffer size used is the one configured in DcmDslBufferSize configuration parameter.			
Multiplicity	0..1			
Type	IntegerParamDef			
Range	1 .. 65535			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time		X	VARIANT-LINK-TIME
	Post-build time		--	
Scope / Dependency	scope: Module			

SWS Item	ECUC_Dcm_00696 :			
Name	DcmDslProtocolID {DCM_PROTOCOL_ID}			
Description	The SWS for DCM defines this as enum type of maxsize uint8 with values {OBD, UDS, ...}. It is effectively the name of the diagnostic protocol type for the DCM DSL protocol that is being configured. 0x00 : DCM_OBD_ON_CAN OBD on CAN (ISO15765-4; ISO15031-5) 0x01 : DCM_UDS_ON_CAN UDS on CAN (ISO15765-3; ISO14229-1) 0x02 : DCM_UDS_ON_FLEXRAY UDS on FlexRay (Manufacturer specific; ISO14229-1) 0x03 : DCM_ROE_ON_CAN 0x04 : DCM_ROE_ON_FLEXRAY 0x05 : DCM_PERIODIC_ON_CAN 0x06 : DCM_PERIODIC_ON_FLEXRAY Implementation Type: Dcm_ProtocolType 0x07..0xEF : Reserved for further AUTOSAR implementation 0xF0..0xFF : Reserved for SW supplier specific			
Multiplicity	1			
Type	IntegerParamDef (Symbolic Name generated for this parameter)			
Range	0 .. 255			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME
	Post-build time		L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolRxAddrType			

SWS Item	ECUC_Dcm_00697 :			
Name	DcmDslProtocolIsParallelExecutab			
Description	Enables the parallel processing of ROE or Periodic Transmission protocol. Only these both protocols are allowed to run in parallel to normal protocol (UDS, OBD).			
Multiplicity	1			
Type	BooleanParamDef			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time		X	VARIANT-LINK-TIME
	Post-build time		--	
Scope / Dependency	scope: Module			

- AUTOSAR confidential -

SWS Item	ECUC_Dcm_00698 :		
Name	DcmDslProtocolPreemptTimeout {DCM_PROTOCOL_PREEMPT_TIMEOUT}		
Description	This is the value for the timeout (in milliseconds) of preempting protocol until protocol needs to be started. This is defined in the AUTOSAR SWS for DCM as a uint16. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM. min: A negative value is not allowed. upperMultiplicity: Exactly one DcmDslProtocolPreemptTimeout value per DSL protocol timing structure. lowerMultiplicity: Exactly one DcmDslProtocolPreemptTimeout value per DSL timing structure. origin: Standard AUTOSAR configuration parameter.		
Multiplicity	1		
Type	FloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolID		

SWS Item	ECUC_Dcm_00699 :		
Name	DcmDslProtocolPriority {DCM_PROTOCOL_PRIO}		
Description	The SWS for DCM defines this item as uint8. This is the protocol priority that is used during protocol preemption handling. 0 = Highest priority (protocol may not be preempted by other protocols) 1, 2, 3... = Reducing priority. Protocol may be preempted by other protocols with lower priority value. upperMultiplicity: Exactly one priority must be provided per Tx Protocol. lowerMultiplicity: Exactly one priority must be provided per Tx Protocol. origin: Standard AUTOSAR configuration parameter.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolID		

SWS Item	ECUC_Dcm_00700 :		
Name	DcmDslProtocolTransType		
Description	Selects the transmission type for protocol.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	Type1	Messages on the DcmTxPduId already used for normal diagnostic responses. The outgoing messages must be synchronized with 'normal outgoing messages', which have a higher priority.	
	Type2	Messages on a separate DcmTxPduId.	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolRxPduId, DcmDslProtocol.DcmDslProtocolRow.DcmDslProtocolID		

SWS Item	Dcm910_Conf :		
Name	DcmSendRespPendOnTransToBoot		
Description	Parameter specifying if the ECU should send a NRC 0x78 (response pending) before transitioning to the bootloader (parameter set to TRUE) or if the transition shall be initiated without sending NRC 0x78 (parameter set to FALSE).		
Multiplicity	1		
Type	BooleanParamDef		
Default value	true		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	ECUC_Dcm_00701 :		
Name	DcmDslProtocolRxBufferID		
Description	Link to buffer configuration (DcmDslBufferID) for configuration of protocol buffer used for Rx actions. upperMultiplicity / lowerMultiplicity:: Exactly one Rx buffer is required for protocol reception		
Multiplicity	1		
Type	Reference to [DcmDslBuffer]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolRxPduRef, DcmDslBuffer.DcmDslBufferID		

SWS Item	ECUC_Dcm_00702 :		
Name	DcmDslProtocolSIDTable {DCM_PROTOCOL_IDENTIFIER_TABLE_ID}		
Description	Link to the used diagnostic service table for this protocol. upperMultiplicity: Must have exactly one service table for the protocol. lowerMultiplicity: Must have exactly one service table for the protocol.		
Multiplicity	1		
Type	Reference to [DcmDsdServiceTable]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolID, DcmDsdServiceIdTable.DcmDsdSidTabId		

SWS Item	ECUC_Dcm_01033 :		
Name	DcmDslProtocolTimeLimit {DCM_LIMITS}		
Description	Link to the extended / limit timing structure. Note: limits are checked when using AccessTimingParameter service. upperMultiplicity: Exactly one extended timing structure must be provided. lowerMultiplicity: Exactly one extended timing structure must be provided.		
Multiplicity	0..1		
Type	Reference to [DcmDslProtocolTimingRow]		
ConfigurationClass	Pre-compile time	--	
	Link time	--	

	Post-build time	--	
Scope / Dependency			

SWS Item	ECUC_Dcm_00704 :		
Name	DcmDslProtocolTxBufferID		
Description	Link to buffer configuration (DcmDslBufferID) for configuration of protocol buffer used for Tx actions. upperMultiplicity / lowerMultiplicity:: Exactly one Tx buffer is required for protocol transmission.		
Multiplicity	1		
Type	Reference to [DcmDslBuffer]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolTxPduRef, DcmDslBuffer.DcmDslBufferID		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDslConnection	1..*	This container contains links between Diagnostic Protocol (=DcmDslProtocolRow) and the according Rx or Tx channel. Because of the usecase to allow more then one diagnostic tester (using different CAN channels) it is necessary to allow configuration of multiple DcmDslConnections.

10.2.12 DcmDslConnection

SWS Item	ECUC_Dcm_00705 :
Choice container Name	DcmDslConnection
Description	This container contains links between Diagnostic Protocol (=DcmDslProtocolRow) and the according Rx or Tx channel. Because of the usecase to allow more then one diagnostic tester (using different CAN channels) it is necessary to allow configuration of multiple DcmDslConnections.

Container Choices		
Container Name	Multiplicity	Scope / Dependency
DcmDslMainConnection	1	This container contains configuration for Diagnostic Main Connection (DcmDslProtocolTx, DcmDslProtocolRx). In addition it contains links to the ROE connection (DcmDslROEConnectionRef) and the Periodic Transmission Connection (DcmDslPeriodicTranmissionConRef) relevant for protocols with DcmDslProtocolTransType = TYPE2.
DcmDslPeriodicTransmission	0..1	This container contains the configuration (parameters) for Periodic Transmission service. Hint: Periodic Transmission request comes via DcmDslMainConnection and PeriodicTransmission Event response is given via DcmDslPeriodicTransmission.
DcmDslResponseOnEvent	0..1	This container contains the configuration (parameters) for ResponseOnEvent service. Hint: ROE service request is recept via DcmDslMainConnection and that Event response is given via DcmDslResposeOnEvent

10.2.13 DcmDslMainConnection

SWS Item	ECUC_Dcm_00706 :
Container Name	DcmDslMainConnection{DIAGNOSTIC_CONNECTION_TABLE}
Description	This container contains configuration for Diagnostic Main Connection (DcmDslProtocolTx, DcmDslProtocolRx). In addition it contains links to the ROE connection (DcmDslROEConnectionRef) and the Periodic Transmission Connection (DcmDslPeriodicTranmissionConRef) relevant for protocols with DcmDslProtocolTransType = TYPE2.
Configuration Parameters	

SWS Item	ECUC_Dcm_00707 :		
Name	DcmDslPeriodicTranmissionConRef		
Description	Configures the link to the connection of PeriodicTransmission protocol for the processing of the PeriodicTransmission events.		
Multiplicity	0..1		
Type	Reference to [DcmDslPeriodicTransmission]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_00708 :		
Name	DcmDslROEConnectionRef		
Description	Configures the link to the connection of ROE protocol for processing of the ROE events.		
Multiplicity	0..1		
Type	Reference to [DcmDslResponseOnEvent]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDslProtocolRx	1..*	This container contains the configuration (parameters) for the protocol configuration of RX channel (for each protocol) The following parameters needs to be configured per protocol. Please keep in mind, that the parameter DcmDslProtocolRxPduId can be given several times per protocol and that the combination from DcmDslProtocolRxPduId and DcmDslProtocolRxAddrType is unique. Only one physical protocol is allowed per connection. upperMultiplicity: More than one receive PDU ID may be configured for reception (e.g. one for func requests, one for phys requests). lowerMultiplicity: At least one receive PDU ID configuration must be provided per protocol.
DcmDslProtocolTx	1	This container contains the configuration (parameters) for the protocol configuration of TX channel (for each protocol) The included parameters needs to be configured per protocol.

10.2.14 DcmDslProtocolRx

SWS Item	ECUC_Dcm_00709 :
Container Name	DcmDslProtocolRx{DIAGNOSTIC_PROTOCOL_RX_TABLE}
Description	This container contains the configuration (parameters) for the protocol configuration of RX channel (for each protocol) The following parameters needs to be configured per protocol. Please keep in mind, that the parameter DcmDslProtocolRxPduId can be given several times per protocol and that the combination from DcmDslProtocolRxPduId and DcmDslProtocolRxAddrType is unique. Only one physical protocol is allowed per connection. upperMultiplicity: More than one receive PDU ID may be configured for reception (e.g. one for func requests, one for phys requests). lowerMultiplicity: At least one receive PDU ID configuration must be provided per protocol.
Configuration Parameters	

SWS Item	ECUC_Dcm_00710 :		
Name	DcmDslProtocolRxAddrType {DCM_PROTOCOL_RX_ADDR_TYPE}		
Description	Declares the communication type of this DCM_PROTOCOL_DCMRXPDUID. PHYSICAL is used for 1 to 1 communication FUNCTIONAL is used for 1 to n communication		
Multiplicity	1		
Type	EnumerationParamDef		
Range	DCM_FUNCTIONAL_TYPE	FUNCTIONAL = 1 to n communication	
	DCM_PHYSICAL_TYPE	PHYSICAL = 1 to 1 communications using physical addressing	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolRxPduId, DcmDslProtocol.DcmDslProtocolRow.DcmDslProtocolID		

SWS Item	Dcm688 :		
Name	DcmDslProtocolRxPduId {DCM_DSL_PROTOCOL_RX_PDU_ID}		
Description	DcmRxPduId value for reception of requests.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_00906 :		
Name	DcmDslProtocolRxComMChannelRef {DCM_DSL_PROTOCOL_RX_COMM_CHANNEL_REF}		
Description	Reference to the ComMChannel on which the DcmDslProtocolRxPdu is received.		
Multiplicity	1		
Type	Reference to [ComMChannel]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME

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

SWS Item	ECUC_Dcm_00770 :			
Name	DcmDslProtocolRxPduRef {DCM_PROTOCOL_DCMRXPDUID}			
Description	DcmRxPduId reference for reception of requests / Multiple links shall be allowed. (e.g. one DcmRxPduId to receive func requests, one DcmRxPduId to receive phys requests)			
Multiplicity	1			
Type	Reference to [Pdu]			
ConfigurationClass	Pre-compile time	X		VARIANT-PRE-COMPILE
	Link time	X		VARIANT-LINK-TIME
	Post-build time	L		VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolRxAddrType, DcmDslProtocolRxPduRef			

No Included Containers

10.2.15 DcmDslProtocolTx

SWS Item	ECUC_Dcm_00711 :
Container Name	DcmDslProtocolTx{DIAGNOSTIC_PROTOCOL_TX_TABLE}
Description	This container contains the configuration (parameters) for the protocol configuration of TX channel (for each protocol) The included parameters needs to be configured per protocol.

Configuration Parameters

SWS Item	ECUC_Dcm_00772 :			
Name	DcmDslProtocolTxPduRef {DCM_PROTOCOL_DCMTXPDUID}			
Description	DcmRxPduId reference for transmission of responses			
Multiplicity	1			
Type	Reference to [Pdu]			
ConfigurationClass	Pre-compile time	X		VARIANT-PRE-COMPILE
	Link time	X		VARIANT-LINK-TIME
	Post-build time	L		VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocol.DcmDslProtocolRow.DcmDslProtocolID			

No Included Containers

10.2.16 DcmDslPeriodicTransmission

SWS Item	ECUC_Dcm_00741 :
Container Name	DcmDslPeriodicTransmission{PERIODIC_TRANSMISSION_PARAMETERS}
Description	This container contains the configuration (parameters) for Periodic Transmission service. Hint: Periodic Transmission request comes via DcmDslMainConnection and

	PeriodicTransmission Event response is given via DcmDslPeriodicTransmission.

Configuration Parameters

SWS Item	ECUC_Dcm_00742 :		
Name	DcmDslPeriodicTxPduRef {DCM_PERIODIC_TRANS_DCMTXPDUID}		
Description	This is the reference to the PDU ID to be used by this DCM when sending Periodic Transmissions. It is only needed for Type2 Periodic Transmissions configurations. upperMultiplicity: One or several DcmDslPeriodicTxPduRef must be defined if Periodic Transmission is enabled and TYPE2 Periodic Transmissions is configured. lowerMultiplicity: DcmDslPeriodicTxPduRef does not need to be defined if Periodic Transmission is not enabled or TYPE1 Periodic Transmissions is configured.		
Multiplicity	0..*		
Type	Reference to [Pdu]		
ConfigurationClass	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolTransType		

No Included Containers

10.2.17 DcmDslResponseOnEvent

SWS Item	ECUC_Dcm_00744 :
Container Name	DcmDslResponseOnEvent{RESPONSE_ON_EVENT_PARAMETERS}
Description	This container contains the configuration (parameters) for ResponseOnEvent service. Hint: ROE service request is recept via DcmDslMainConnection and that Event response is given via DcmDslResposeOnEvent

Configuration Parameters

SWS Item	ECUC_Dcm_00743 :		
Name	DcmDslRoeTxPduRef {DCM_ROE_DCMTXPDUID}		
Description	Reference to the PDU for transmission of ROE response (only needed for ROE Transmission Type is TYPE2) upperMultiplicity: One PDU required if ROE Transmission Type is TYPE2. lowerMultiplicity: No PDU required if ROE Transmission Type is TYPE1.		
Multiplicity	0..1		
Type	Reference to [Pdu]		
ConfigurationClass	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolTransType		

No Included Containers

10.2.18 DcmDslProtocolTiming

SWS Item	ECUC_Dcm_01021 :
Container Name	DcmDslProtocolTiming{PROTOCOL_TIMING_STRUCTURE}
Description	This container contains the configuration (parameters) for Protocol timing. This container contains rows of DcmDslProtocolTimingRow containers. Hint: DcmDslProtocolTiming is only relevant in case of using the UDS service AccessTimingParameter.
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDslProtocolTimingRow	0..*	Definition of a single row of configuration for protocol timing

10.2.19 DcmDslProtocolTimingRow

SWS Item	ECUC_Dcm_01020 :
Container Name	DcmDslProtocolTimingRow{PROTOCOL_TIMING_STRUCTURE_ROW}
Description	Definition of a single row of configuration for protocol timing
Configuration Parameters	

SWS Item	ECUC_Dcm_01022 :		
Name	DcmTimStrP2ServerMax {DCM_TIMSTR_P2SERVER_MAX}		
Description	This is the value for P2ServerMax in milliseconds. This is defined in the AUTOSAR SWS for DCM as a uint16. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM.		
Multiplicity	1		
Type	FloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_01023 :		
Name	DcmTimStrP2ServerMin {DCM_TIMSTR_P2SERVER_MIN}		
Description	This is the value for P2ServerMin in milliseconds. This is defined in the AUTOSAR SWS for DCM as a uint16. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM.		
Multiplicity	1		
Type	FloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_01024 :

Name	DcmTimStrP2StarServerMax {DCM_TIMSTR_P2STARSERVER_MAX}			
Description	This is the value for P2*ServerMax in milliseconds. This is defined in the AUTOSAR SWS for DCM as a uint16. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM.			
Multiplicity	1			
Type	FloatParamDef			
Range	0 .. INF			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME
	Post-build time		L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU			

SWS Item	ECUC_Dcm_01025 :			
Name	DcmTimStrP2StarServerMin {DCM_TIMSTR_P2STARSERVER_MIN}			
Description	This is the value for P2*ServerMin in milliseconds. This is defined in the AUTOSAR SWS for DCM as a uint16. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM.			
Multiplicity	1			
Type	FloatParamDef			
Range	0 .. INF			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME
	Post-build time		L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU			

SWS Item	ECUC_Dcm_01026 :			
Name	DcmTimStrS3Server {DCM_TIMSTR_S3SERVER}			
Description	This is the value for S3Server in milliseconds. This is defined in the AUTOSAR SWS for DCM as a uint16. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM.			
Multiplicity	1			
Type	FloatParamDef			
Range	0 .. INF			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME
	Post-build time		L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU			

No Included Containers

10.2.20   DcmDslServiceRequestIndication

SWS Item	DCM681 :
Container Name	DcmDslServiceRequestIndication
Description	The name of this container is the name of the port through which the DCM

	requires the port-interface ServiceRequestIndication. The lowerMultiplicity is 0: If DcmRequestIndicationEnabled = false the Indication API is not available.
Configuration Parameters	

No Included Containers

10.2.21 DcmDslSessionControl

SWS Item	DCM680 :
Container Name	DcmDslSessionControl
Description	The name of this container is the name of the R-port through which the DCM requires the PortInterface SessionControl
Configuration Parameters	

No Included Containers

10.2.22 DcmDsp

SWS Item	ECUC_Dcm_00712 :
Container Name	DcmDsp
Description	These parameters apply to Diagnostic Service Processing. There will always be one set of these parameters per DCM.
Configuration Parameters	

SWS Item	Dcm638 :		
Name	DcmDspMaxDidToRead		
Description	Indicates the maximum allowed DIDs in a single "ReadDataByIdentifier" request. If set to 0, then no limitation is applied.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	*Multiplicity*	*Scope / Dependency*
DcmDspComControl	0..1	--
DcmDspDid	0..*	This container contains the configuration (parameters) of the DID.
DcmDspDidInfo	0..*	This container contains the configuration (parameters) of the DID's Info
DcmDspEcuReset	0..*	There is a container for each R-Port through which the DCM accesses the ResetService PortInterface. The name of the container is the name of the port. lowerMultiplicity: DcmDspEcuReset does not need to be defined if the EcuReset service is not supported.
DcmDspObdService04	0..1	This container contains the configuration (parameters) of the

		Clear/Reset Emission-Related Diagnostic Information service (Service $04)
DcmDspPid	0..*	This container defines the availability of a PID to the DCM. In case the PID is accessed through a port, the name of this container is the port-name.
DcmDspReadDTC	1	This container contains the configuration (parameters) of the service Read DTC Information - 0x19 (per sub-function type) This container contains Rows of DcmDspReadDTCRow
DcmDspRequestControl	0..*	This container contains the configuration (parameters) of the "Request control of on-board system, test or component" service (Service $08). The name of this container is the name of the port through which the DCM has access to a RequestControlServices interface.
DcmDspRoutine	0..*	This container contains the configuration (parameters) for Routines
DcmDspRoutineInfo	0..*	This container contains the configuration (parameters) for Routine's Info.
DcmDspSecurity	1	This container contains the configuration (DSP parameter) for security level configuration (per security level) Description This container contains Rows of DcmDspSecurityRow
DcmDspSession	1	This container contains the configuration (DSP parameter) session control configuration (per session control) This container contains Rows of DcmDspSessionRow
DcmDspTestResultByObdmid	0..*	This container contains the configuration (parameters) of the "Request on-board monitoring test results" service (Service $06).
DcmDspVehInfo	0..*	This container contains the configuration (parameters) of the "Request vehicle information service" (service $09).

10.2.23 DcmDspComControl

SWS Item	ECUC_Dcm_00900 :
Container Name	DcmDspComControl
Description	--
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspComControlAllChannel	0..*	Collection of ComM channels which shall be controlled if all networks are addressed.
DcmDspComControlSpecificChannel	0..*	Assigns subnet number to ComM channel which will be controlled.

10.2.24 DcmDspComControlAllChannel

SWS Item	ECUC_Dcm_00901 :
Container Name	DcmDspComControlAllChannel
Description	Collection of ComM channels which shall be controlled if all networks are addressed.
Configuration Parameters	

SWS Item	ECUC_Dcm_00902 :
Name	DcmDspAllComMChannelRef

{DCM_DSL_PROTOCOL_RX_COMM_CHANNEL_REF}			
Description	Reference to ComM channel.		
Multiplicity	1		
Type	Reference to [ComMChannel]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.25   DcmDspComControlSpecificChannel

SWS Item	ECUC_Dcm_00903 :	
Container Name	DcmDspComControlSpecificChannel	
Description	Assigns subnet number to ComM channel which will be controlled.	
Configuration Parameters		

SWS Item	ECUC_Dcm_00905 :		
Name	DcmDspSubnetNumber		
Description	Subnet Number which controls the specific ComMChannel.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 14		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_00904 :		
Name	DcmDspSpecificComMChannelRef {DCM_DSL_PROTOCOL_RX_COMM_CHANNEL_REF}		
Description	Reference to ComM channel.		
Multiplicity	1		
Type	Reference to [ComMChannel]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.26   DcmDspDid

SWS Item	Dcm601 :
Container Name	DcmDspDid

Description	This container contains the configuration (parameters) of the DID.
Configuration Parameters	

SWS Item	Dcm677 :		
Name	DcmDspDidConditionCheckReadFnc		
Description	Function name to demand application if the conditions (e.g. System state) to read the DID are correct. (ConditionCheckRead-function). Multiplicity shall be equal to parameter DcmDspDidReadFnc. Only relevant if DcmDspDidUsePort==FALSE.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDidReadFnc		

SWS Item	Dcm678 :		
Name	DcmDspDidConditionCheckWriteFnc		
Description	Function name to demand application if the conditions (e.g. System state) to write the DID are correct. (ConditionCheckWrite-function). Only relevant if DcmDspDidUsePort==FALSE.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDidWriteFnc		

SWS Item	Dcm674 :		
Name	DcmDspDidFreezeCurrentStateFnc		
Description	Function name to request to application to freeze the current state of an IOControl. (FreezeCurrentState-function). Only relevant if DcmDspDidUsePort==FALSE.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDidFreezeCurrentStateAvailable		

SWS Item	Dcm676 :
Name	DcmDspDidGetScalingInfoFnc

Description	Function name to request to application the scaling information of the DID. (GetScalingInformation-function). Only relevant if DcmDspDidUsePort==FALSE.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm602 :		
Name	DcmDspDidIdentifier		
Description	2 byte Identifier of the DID		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm671 :		
Name	DcmDspDidReadDataLengthFnc		
Description	Function name to request from application the data length of a DID. (ReadDataLength-function). Only relevant if DcmDspDidUsePort==FALSE.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDidFixedLength		

SWS Item	Dcm669 :		
Name	DcmDspDidReadFnc		
Description	Function name to request from application the data value of a DID. (ReadData-function). Multiplicity shall be equal to parameter DcmDspDidConditionCheckReadFnc. Only relevant if DcmDspDidUsePort==FALSE.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build	--	

time	
Scope / Dependency	scope: ECU dependency: DcmDspDidConditionCheckReadFnc

SWS Item	Dcm673 :		
Name	DcmDspDidResetToDefaultFnc		
Description	Function name to request to application to reset an IOControl to default value. (ResetToDefault-function). Only relevant if DcmDspDidUsePort==FALSE.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDidResetToDefaultAvailable		

SWS Item	Dcm672 :		
Name	DcmDspDidReturnControlToEcuFnc		
Description	Function name to request to application to return control to ECU of an IOControl. (ReturnControlToECU-function). Only relevant if DcmDspDidUsePort==FALSE.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDidReturnControlToEcuAvailable		

SWS Item	Dcm675 :		
Name	DcmDspDidShortTermAdjustmentFnc		
Description	Function name to request to application to adjuste the IO signal. (ShortTermAdjustment-function). Only relevant if DcmDspDidUsePort==FALSE.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDidShortTermAdjustementAvailable		

SWS Item	Dcm605 :
Name	DcmDspDidSize
Description	Length of data associated to the DID. If DID has variable datalength, that corresponds to the maximum datalength.

Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_01039 :		
Name	DcmDspDidUsePort		
Description	If this parameter is TRUE, the DCM will access the DID using an R-Port requiring a PortInteface DidServices_<DID>. The name of this R-Port is the name of the container DcmDspDid. If this parameter is FALSE, the DCM will access the DID using the functions that are defined in parameters of type FunctionNameDef in the DcmDspDid container.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm805_Conf :		
Name	DcmDspDidUsed		
Description	Allow to activate or deactivate the usage of a DID, for multi purpose ECUs true = DID available false = DID not available		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	Dcm670 :		
Name	DcmDspDidWriteFnc		
Description	Function name to request application to write the data value of a DID. (WriteData-function). Multiplicity shall be equal to parameter DcmDspDidConditionCheckWriteFnc. Only relevant if DcmDspDidUsePort==FALSE.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: ECU
	dependency: DcmDspDidConditionCheckWriteFnc

SWS Item	Dcm604 :		
Name	DcmDspDidInfoRef		
Description	Reference to DcmDspDidInfo containing information on this DID.		
Multiplicity	1		
Type	Reference to [DcmDspDidInfo]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm606 :		
Name	DcmDspDidRef		
Description	Reference to DcmDspDid in case this DID refer to one or serveral other DID's		
Multiplicity	0..*		
Type	Reference to [DcmDspDid]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDspDidControlRecordSizes	0..*	This container defines the sizes of the data sent and received with the DID control functions

10.2.27 DcmDspDidControlRecordSizes

SWS Item	ECUC_Dcm_01037 :
Container Name	DcmDspDidControlRecordSizes
Description	This container defines the sizes of the data sent and received with the DID control functions
Configuration Parameters	

SWS Item	ECUC_Dcm_01035 :		
Name	DcmDspDidControlEnableMaskRecordSize		
Description	This defines the size of the controlEnableMaskRecord as defined in ISO14229-1 for UDS Service 0x2F		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_01036 :			
Name	DcmDspDidControlOptionRecordSize			
Description	This defines the size of the controlOptionRecord (as defined in ISO14229-1 for service UDS 0x2F) without the InputOutputControlParameter			
Multiplicity	1			
Type	IntegerParamDef			
Range	0 .. 255			
Default value	--			
ConfigurationClass	Pre-compile time	X	All Variants	
	Link time	--		
	Post-build time	--		
Scope / Dependency	scope: ECU			

SWS Item	ECUC_Dcm_01038 :			
Name	DcmDspDidControlStatusRecordSize			
Description	This defines the size of the controlStatusRecord (as defined in ISO14229-1 for UDS Service 0x2F) WITHOUT the InputOutputControlParameter			
Multiplicity	1			
Type	IntegerParamDef			
Range	0 .. 255			
Default value	--			
ConfigurationClass	Pre-compile time	X	All Variants	
	Link time	--		
	Post-build time	--		
Scope / Dependency	scope: ECU			

No Included Containers

10.2.28 DcmDspDidInfo

SWS Item	Dcm607 :
Container Name	DcmDspDidInfo
Description	This container contains the configuration (parameters) of the DID's Info
Configuration Parameters	

SWS Item	Dcm612 :			
Name	DcmDspDidDynamicallyDefined			
Description	Indicates if this DID can be dynamically defined true = DID can be dynamically defined false = DID can not be dynamically defined			
Multiplicity	1			
Type	BooleanParamDef			
Default value	--			
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD	
	Link time	X	VARIANT-LINK-TIME	
	Post-build time	--		
Scope / Dependency	scope: ECU			

SWS Item	Dcm608 :
Name	DcmDspDidFixedLength
Description	Indicates if the datalength of the DID is fixed true = datalength of the DID is fixed false = datalength of the DID is variable

Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm611 :		
Name	DcmDspDidScalingInfoSize		
Description	If Scaling information service is available for this DID, it provides the size of the scaling information.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDspDidAccess	1	This container contains the configuration (parameters) of the DID access

10.2.29 DcmDspDidAccess

SWS Item	Dcm609 :
Container Name	DcmDspDidAccess
Description	This container contains the configuration (parameters) of the DID access
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDspDidControl	0..1	This container contains the configuration (parameters) of the DID control.
DcmDspDidRead	0..1	This container contains the configuration (parameters) of the DID read.
DcmDspDidWrite	0..1	This container contains the configuration (parameters) of the DID write.

10.2.30 DcmDspDidControl

SWS Item	Dcm619 :
Container Name	DcmDspDidControl
Description	This container contains the configuration (parameters) of the DID control.
Configuration Parameters	

SWS Item	Dcm620 :		
Name	DcmDspDidControlSecurityLevelRef		
Description	Reference to DcmDspSecurityRow Security levels allowed to control this DID.		
Multiplicity	1..*		
Type	Reference to [DcmDspSecurityRow]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm621 :		
Name	DcmDspDidControlSessionRef		
Description	Reference to DcmDspSessionRow Sessions allowed to control this DID.		
Multiplicity	1..*		
Type	Reference to [DcmDspSessionRow]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm624 :		
Name	DcmDspDidFreezeCurrentState		
Description	This indicates the presence of "FreezeCurrentState" and refers to a container defining the sizes of the parameters		
Multiplicity	0..1		
Type	Reference to [DcmDspDidControlRecordSizes]		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm623 :		
Name	DcmDspDidResetToDefault		
Description	This indicates the presence of "ResetToDefault" and refers to a container defining the sizes of the parameters		
Multiplicity	0..1		
Type	Reference to [DcmDspDidControlRecordSizes]		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm622 :		
Name	DcmDspDidReturnControlToEcu		
Description	This indicates the presence of "ReturnControlToEcu" and refers to a container defining the sizes of the parameters		
Multiplicity	0..1		

Type	Reference to [DcmDspDidControlRecordSizes]		
ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm625 :		
Name	DcmDspDidShortTermAdjustement		
Description	This indicates the presence of "ShortTermAdjustment" and refers to a container defining the sizes of the parameters		
Multiplicity	0..1		
Type	Reference to [DcmDspDidControlRecordSizes]		
ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.31 DcmDspDidRead

SWS Item	Dcm613 :
Container Name	DcmDspDidRead
Description	This container contains the configuration (parameters) of the DID read.
Configuration Parameters	

SWS Item	Dcm614 :		
Name	DcmDspDidReadSecurityLevelRef		
Description	Reference to DcmDspSecurityRow Security levels allowed to read this DID.		
Multiplicity	1..*		
Type	Reference to [DcmDspSecurityRow]		
ConfigurationClass	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm615 :		
Name	DcmDspDidReadSessionRef		
Description	Reference to DcmDspSessionRow Sessions allowed to read this DID.		
Multiplicity	1..*		
Type	Reference to [DcmDspSessionRow]		
ConfigurationClass	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers	

10.2.32   DcmDspDidWrite

SWS Item	Dcm616 :
Container Name	DcmDspDidWrite
Description	This container contains the configuration (parameters) of the DID write.
Configuration Parameters	

SWS Item	Dcm617 :		
Name	DcmDspDidWriteSecurityLevelRef		
Description	Reference to DcmDspSecurityRow Security levels allowed to write this DID.		
Multiplicity	1..*		
Type	Reference to [DcmDspSecurityRow]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm618 :		
Name	DcmDspDidWriteSessionRef		
Description	Reference to DcmDspSessionRow Sessions allowed to write this DID.		
Multiplicity	1..*		
Type	Reference to [DcmDspSessionRow]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers	

10.2.33   DcmDspDidInfo

SWS Item	Dcm607 :
Container Name	DcmDspDidInfo
Description	This container contains the configuration (parameters) of the DID's Info
Configuration Parameters	

SWS Item	Dcm612 :		
Name	DcmDspDidDynamicallyDefined		
Description	Indicates if this DID can be dynamically defined true = DID can be dynamically defined false = DID can not be dynamically defined		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile	X	VARIANT-PRE-COMPILE,

Document ID 018: AUTOSAR_SWS_DCM

- AUTOSAR confidential -

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

SWS Item	**Dcm608 :**		
Name	DcmDspDidFixedLength		
Description	Indicates if the datalength of the DID is fixed true = datalength of the DID is fixed false = datalength of the DID is variable		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dcm611 :**		
Name	DcmDspDidScalingInfoSize		
Description	If Scaling information service is available for this DID, it provides the size of the scaling information.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspDidAccess	1	This container contains the configuration (parameters) of the DID access

10.2.34 DcmDspEcuReset

SWS Item	**Dcm657 :**
Container Name	DcmDspEcuReset
Description	There is a container for each R-Port through which the DCM accesses the ResetService PortInterface. The name of the container is the name of the port. lowerMultiplicity: DcmDspEcuReset does not need to be defined if the EcuReset service is not supported.
Configuration Parameters	

No Included Containers

10.2.35 DcmDspPid

SWS Item	Dcm626 :
Container Name	DcmDspPid
Description	This container defines the availability of a PID to the DCM. In case the PID is accessed through a port, the name of this container is the port-name.
Configuration Parameters	

SWS Item	Dcm629 :		
Name	DcmDspGetPidValFnc		
Description	Function name for reading value by PID (GetPIDValue-function). This is only relevant if DcmDspPidUsePort==FALSE.		
Multiplicity	1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm627 :		
Name	DcmDspPidIdentifier		
Description	2 bytes Identifier of the PID		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm628 :		
Name	DcmDspPidSize		
Description	Length of data associated to the PID.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_01040 :
Name	DcmDspPidUsePort
Description	If this is false, the DCM will use the function defined in DcmDspGetPidValFnc to get the pid-value. If this is true, the PID will have an R-Port requiring the interface PidServices_<PID>. The name of this port is the name of the container.
Multiplicity	1

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

SWS Item	Dcm806_Conf :		
Name	DcmDspPidUsed		
Description	Allow to activate or deactivate the usage of a PID, for multi purpose ECUs true = PID available false = PID not available		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.36   DcmDspReadDTC

SWS Item	Dcm074 :
Container Name	DcmDspReadDTC{DCM_READDTC_SUB_FUNCTION_TABLE}
Description	This container contains the configuration (parameters) of the service Read DTC Information - 0x19 (per sub-function type) This container contains Rows of DcmDspReadDTCRow
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDspReadDTCRow	0..*	This container contains the configuration (parameters) of the service Read DTC Information - 0x19 (per sub-function type)

10.2.37   DcmDspReadDTCRow

SWS Item	ECUC_Dcm_01027 :
Container Name	DcmDspReadDTCRow{DCM_READDTC_SUB_FUNCTION_TABLE_ROW}
Description	This container contains the configuration (parameters) of the service Read DTC Information - 0x19 (per sub-function type)
Configuration Parameters	

SWS Item	ECUC_Dcm_01028 :
Name	DcmDspDTCInfoSubFuncLevel {DCM_DTC_INFO_SUB_FUNCTION_LEVEL}
Description	hex value of the sub-function.
Multiplicity	1
Type	IntegerParamDef

Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	ECUC_Dcm_01029 :		
Name	DcmDspDTCInfoSubFuncSupp {DCM_DTC_INFO_SUB_FUNCTION_SUPPORT}		
Description	Indicates whether the respective sub-function is supported. TRUE = sub-function supported. FALSE = sub-function not supported.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	ECUC_Dcm_01034 :		
Name	DcmDspDTCInfoSecLevelRef {DCM_SIDTAB_SEC_LEVEL_ROW}		
Description	Link to the Security Access Levels needed for execution of the DcmDspReadDTCRow. Please refer to the ISO 14229-1, ISO 15031-5 and "Verification of the Security Access levels". Please note, that it shall be provided to configure several DcmDsdSidTabSecLevelRef.		
Multiplicity	1..*		
Type	Reference to [DcmDspSecurityRow]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.38   DcmDspRequestControl

SWS Item	Dcm637 :
Container Name	DcmDspRequestControl
Description	This container contains the configuration (parameters) of the "Request control of on-board system, test or component" service (Service $08). The name of this container is the name of the port through which the DCM has access to a RequestControlServices interface.
Configuration Parameters	

SWS Item	ECUC_Dcm_00722 :
Name	DcmDspRequestControlInBufferSize
Description	--
Multiplicity	1

Type	IntegerParamDef			
Range	0 .. 255			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time		X	VARIANT-LINK-TIME
	Post-build time		--	
Scope / Dependency	scope: ECU			

SWS Item	ECUC_Dcm_00723 :			
Name	DcmDspRequestControlOutBufferSize			
Description	--			
Multiplicity	1			
Type	IntegerParamDef			
Range	0 .. 255			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time		X	VARIANT-LINK-TIME
	Post-build time		--	
Scope / Dependency	scope: ECU			

SWS Item	Dcm656 :			
Name	DcmDspRequestControlTestId			
Description	Test Id for Service $08			
Multiplicity	1			
Type	IntegerParamDef			
Range	0 .. 255			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time		X	VARIANT-LINK-TIME
	Post-build time		--	
Scope / Dependency	scope: ECU			

No Included Containers

10.2.39   DcmDspRoutine

SWS Item	Dcm640 :
Container Name	DcmDspRoutine
Description	This container contains the configuration (parameters) for Routines
Configuration Parameters	

SWS Item	ECUC_Dcm_00753 :			
Name	DcmDspRequestResultsRoutineFnc			
Description	Function name for request to application the results of a routine. (Routine_RequestResults-function)			
Multiplicity	0..1			
Type	FunctionNameDef			
Default value	--			
regularExpression	--			
ConfigurationClass	Pre-compile		X	All Variants

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

SWS Item	Dcm641 :		
Name	DcmDspRoutineIdentifier		
Description	2 bytes Identifier of the RID		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_00724 :		
Name	DcmDspRoutineUsePort		
Description	If this is true, the DCM uses a port (whose name is the name of the container) to access the interface RoutineServices. In that case, the configuration must not provide function names in DcmDspStartRoutineFnc, DcmDspStopRoutineFnc or DcmDspRequestResultsRoutineFnc. If this is false, the DCM expects to find the names of the functions to be used in DcmDspStartRoutineFnc, DcmDspStopRoutineFnc or DcmDspRequestResultsRoutineFnc.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm807_Conf :		
Name	DcmDspRoutineUsed		
Description	Allow to activate or deactivate the usage of a Routine, for multi purpose ECUs true = Routine available false = Routine not available		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	Dcm664 :	
Name	DcmDspStartRoutineFnc	
Description	Function name for request to application to start a routine. (Routine_Start-function)	
Multiplicity	0..1	

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

SWS Item	ECUC_Dcm_00752 :		
Name	DcmDspStopRoutineFnc		
Description	Function name for request to application to stop a routine. (Routine_Stop-function)		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm642 :		
Name	DcmDspRoutineInfoRef		
Description	Reference to DcmDspRoutineInfo containing information on this routine.		
Multiplicity	1		
Type	Reference to [DcmDspRoutineInfo]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.40 DcmDspRoutineInfo

SWS Item	Dcm643 :
Container Name	DcmDspRoutineInfo
Description	This container contains the configuration (parameters) for Routine's Info.
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDspRoutineAuthorization	1	This container contains the configuration (parameters) for the Routine Authorization.
DcmDspRoutineRequestRes	0..1	This container contains the configuration (parameters) of Routine Result
DcmDspRoutineStop	0..1	This container contains the configuration (parameters) of Routine Stop

DcmDspStartRoutine	1	This container contains the configuration (parameters) of Routine Start.

10.2.41   DcmDspRoutineAuthorization

SWS Item	Dcm644 :	
Container Name	DcmDspRoutineAuthorization	
Description	This container contains the configuration (parameters) for the Routine Authorization.	
Configuration Parameters		

SWS Item	Dcm648 :		
Name	DcmDspRoutineSecurityLevelRef		
Description	Reference to DcmDspSecurityRow Security levels allowed to control this RID.		
Multiplicity	1..*		
Type	Reference to [DcmDspSecurityRow]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm649 :		
Name	DcmDspRoutineSessionRef		
Description	Reference to DcmDspSessionRow Sessions allowed to control this RID.		
Multiplicity	1..*		
Type	Reference to [DcmDspSessionRow]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.42   DcmDspRoutineRequestRes

SWS Item	Dcm646 :
Container Name	DcmDspRoutineRequestRes
Description	This container contains the configuration (parameters) of Routine Result
Configuration Parameters	

SWS Item	Dcm652 :	
Name	DcmDspReqResRtnCtrlOptRecSize	
Description	Size of optional record in the RequestResult response	
Multiplicity	1	
Type	IntegerParamDef	
Range	0 .. 255	
Default value	--	

ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.43 DcmDspRoutineStop

SWS Item	Dcm645 :
Container Name	DcmDspRoutineStop
Description	This container contains the configuration (parameters) of Routine Stop
Configuration Parameters	

SWS Item	Dcm650 :		
Name	DcmDspStopRoutineCtrlOptRecSize		
Description	Size of optional record in the Stop request		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm651 :		
Name	DcmDspStopRoutineStsOptRecSize		
Description	Size of optional record in the Stop response		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.44 DcmDspStartRoutine

SWS Item	Dcm647 :
Container Name	DcmDspStartRoutine
Description	This container contains the configuration (parameters) of Routine Start.
Configuration Parameters	

SWS Item	Dcm654 :

Name	DcmDspStartRoutineCtrlOptRecSize {Dcm655}		
Description	Size of optional record in the Routine Start request		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm655 :		
Name	DcmDspStartRoutineStsOptRecSize		
Description	Size of optional record in the Routine Start response		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.45 DcmDspSecurity

SWS Item	ECUC_Dcm_00764 :
Container Name	DcmDspSecurity{DCM_SEC_LEVEL_TABLE}
Description	This container contains the configuration (DSP parameter) for security level configuration (per security level) Description This container contains Rows of DcmDspSecurityRow
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDspSecurityRow	0..31	Definition of a single Row of configuration for security level onfiguration (per security level) The name of this container is the name of the port through which the DCM will require a PortInterface of type SecurityAccess_<LEVEL>

10.2.46 DcmDspSecurityRow

SWS Item	ECUC_Dcm_00759 :
Container Name	DcmDspSecurityRow{DCM_SEC_LEVEL_TABLE_ROW}
Description	Definition of a single Row of configuration for security level onfiguration (per security level) The name of this container is the name of the port through which the DCM will require a PortInterface of type SecurityAccess_<LEVEL>
Configuration Parameters	

SWS Item	ECUC_Dcm_00725 :			
Name	DcmDspSecurityADRSize			
Description	Size of the AccessDataRecord used in GetSeed			
Multiplicity	0..1			
Type	IntegerParamDef			
Range	1 .. 255			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time		X	VARIANT-LINK-TIME
	Post-build time		--	
Scope / Dependency	scope: Module			

SWS Item	ECUC_Dcm_00757 :			
Name	DcmDspSecurityDelayTime {DCM_SEC_DELAY_INV_KEY}			
Description	Delay time after failed security access (in ms). This is started after DcmDspSecurityNumAttDelay number of failed security accesses. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM. min: A negative value is not allowed. upperMultiplicity: Exactly one delay time must be specified per security levbel configuration. lowerMultiplicity: Exactly one delay time must be specified per security levbel configuration. origin: Standard AUTOSAR configuration parameter.			
Multiplicity	1			
Type	FloatParamDef			
Range	0 .. INF			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time		X	VARIANT-LINK-TIME
	Post-build time		--	
Scope / Dependency	scope: Module			

SWS Item	ECUC_Dcm_00726 :			
Name	DcmDspSecurityDelayTimeOnBoot			
Description	Start delay timer on power on (in ms) The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM. min: A negative value is not allowed. upperMultiplicity: Exactly one delay time must be specified per security level configuration. lowerMultiplicity: Exactly one delay time must be specified per security level configuration.			
Multiplicity	1			
Type	FloatParamDef			
Range	0 .. INF			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time		X	VARIANT-LINK-TIME
	Post-build time		--	
Scope / Dependency	scope: Module			

SWS Item	ECUC_Dcm_00760 :
Name	DcmDspSecurityKeySize {DCM_SEC_NUM_KEY}

Description	size of the security key (in Bytes).		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	ECUC_Dcm_00754 :		
Name	DcmDspSecurityLevel {DCM_SEC_LEVEL, DCM_SEC_NAME}		
Description	The locked state cannot be configured explicitly. 1,2,3...63: configuration dependent - Conversion formula to calculate SecurityLevel out of tester requested SecurityAccessType parameter: SecurityLevel = (SecurityAccessType + 1) / 2		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	1 .. 63		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	ECUC_Dcm_00762 :		
Name	DcmDspSecurityNumAttDelay {DCM_SEC_NUM_MAX_ATT_DELAY}		
Description	Number of failed security accesses after which the delay time is activated		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module dependency: DcmDspSecurityNumAttLock, DcmDspSecurityDelayTime		

SWS Item	ECUC_Dcm_01030 :		
Name	DcmDspSecurityNumAttLock {DCM_SEC_NUM_MAX_ATT_LOCK}		
Description	Number of failed security accesses after which security access is locked. 0 indicates that the security access will be left untouched and the ECU never locks.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspSecurityNumAttDelay		

SWS Item	ECUC_Dcm_00755 :			
Name	DcmDspSecuritySeedSize {DCM_SEC_NUM_SEED}			
Description	size of the security seed (in Bytes).			
Multiplicity	1			
Type	IntegerParamDef			
Range	1 .. 255			
Default value	--			
ConfigurationClass	*Pre-compile time*		X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time		X	VARIANT-LINK-TIME
	Post-build time		--	
Scope / Dependency	scope: Module			

No Included Containers

10.2.47 DcmDspSession

SWS Item	ECUC_Dcm_00769 :
Container Name	DcmDspSession{DCM_SESSION_LEVEL_TABLE}
Description	This container contains the configuration (DSP parameter) session control configuration (per session control) This container contains Rows of DcmDspSessionRow
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspSessionRow	0..31	Definition of a single Row of session control configuration (per session control)

10.2.48 DcmDspSessionRow

SWS Item	ECUC_Dcm_00767 :
Container Name	DcmDspSessionRow{DCM_SESSION_LEVEL_TABLE_ROW}
Description	Definition of a single Row of session control configuration (per session control)
Configuration Parameters	

SWS Item	Dcm815_Conf :	
Name	DcmDspSessionForBoot	
Description	This parameter defines whether this diagnostic session allows to jump to Bootloader (OEM Bootloader or System Supplier Bootloader). If this diagnostic session doesn't allow to jump to Bootloader the value DCM_NO_BOOT shall be chosen.	
Multiplicity	1	
Type	EnumerationParamDef	
Range	DCM_NO_BOOT	This diagnostic session doesn't allow to jump to Bootloader.
	DCM_OEM_BOOT	This diagnostic session allows to jump to OEM Bootloader.
	DCM_SYS_BOOT	This diagnostic session allows to jump to System Supplier Bootloader.
ConfigurationClass	*Pre-compile time*	X VARIANT-PRE-COMPILE,

			VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ECUC_Dcm_00765 :**		
Name	DcmDspSessionLevel {DCM_SESSION_LEVEL, DCM_SESSION_NAME}		
Description	hex value of the Session control. The symbolicName represents the Name of the Session SWS defines the Dcm_SesCtrlType: 0x01 DEFAULT_SESSION 0x02 PROGRAMMING_SESSION 0x03 EXTENDED_DIAGNOSTIC_SESSION 0x04 SAFETY_SYSTEM_DIAGNOSTIC_SESSION 0x05...0x7F <configuration dependent (according "diagnosticSessionType" parameter DiagnosticSessionControl request> 0x80...0xFE Reserved by Document 0xFF ALL_SESSION_LEVEL		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ECUC_Dcm_00766 :**		
Name	DcmDspSessionP2ServerMax {DCM_SESSION_P2SERVER_MAX}		
Description	This is the session value for P2ServerMax in milliseconds (per Session Control). This is defined in the AUTOSAR SWS for DCM as a uint16. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM.		
Multiplicity	1		
Type	FloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ECUC_Dcm_00768 :**		
Name	DcmDspSessionP2StarServerMax {DCM_SESSION_P2STRSERVER_MAX}		
Description	This is the session value for P2*ServerMax in milliseconds (per Session Control). This is defined in the AUTOSAR SWS for DCM as a uint16. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM.		
Multiplicity	1		
Type	FloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	

Scope / Dependency	scope: Module

No Included Containers

10.2.49 DcmDspTestResultByObdmid

SWS Item	ECUC_Dcm_00771 :	
Container Name	DcmDspTestResultByObdmid	
Description	This container contains the configuration (parameters) of the "Request on-board monitoring test results" service (Service $06).	
Configuration Parameters		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDspTestResultObdmidTid	1..*	This container contains the configuration (parameters) of a OBDMID and the assigend TIDs for for the "Request on-board monitoring test results" service (Service $06).
DcmDspTestResultTid	1..*	This container contains the configuration (parameters) of a single TID for the "Request on-board monitoring test results" service (Service $06). The name of this container is the R-Port through which the DCM requires a PortInterface DtrServices.

10.2.50 DcmDspTestResultObdmidTid

SWS Item	ECUC_Dcm_00773 :
Container Name	DcmDspTestResultObdmidTid
Description	This container contains the configuration (parameters) of a OBDMID and the assigend TIDs for for the "Request on-board monitoring test results" service (Service $06).
Configuration Parameters	

SWS Item	Dcm684 :			
Name	DcmDspTestResultObdmid			
Description	OBDMID for Service $06			
Multiplicity	1			
Type	IntegerParamDef			
Range	0 .. 255			
Default value	--			
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD	
	Link time	X	VARIANT-LINK-TIME	
	Post-build time	--		
Scope / Dependency	scope: ECU			

SWS Item	Dcm685 :		
Name	DcmDspTestResultObdmidTidRef		
Description	Link to reported TIDs of this OBDMID upperMultiplicity / lowerMultiplicity:: At least one TID must be reported for an OBDMID		
Multiplicity	1..*		
Type	Reference to [DcmDspTestResultTid]		
ConfigurationClass	Pre-compile	X	VARIANT-PRE-COMPILE

Document ID 018: AUTOSAR_SWS_DCM

	time		
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolRxPduRef, DcmDslBuffer.DcmDslBufferID		

No Included Containers

10.2.51   DcmDspTestResultTid

SWS Item	**Dcm634 :**
Container Name	DcmDspTestResultTid
Description	This container contains the configuration (parameters) of a single TID for the "Request on-board monitoring test results" service (Service $06). The name of this container is the R-Port through which the DCM requires a PortInterface DtrServices.
Configuration Parameters	

SWS Item	**Dcm635 :**		
Name	DcmDspTestResultTestId		
Description	Test Id for Service $06		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dcm686 :**		
Name	DcmDspTestResultUaSid		
Description	Unit And Scaling ID for Service $06		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.52   DcmDspVehInfo

SWS Item	**Dcm630 :**

Container Name	DcmDspVehInfo
Description	This container contains the configuration (parameters) of the "Request vehicle information service" (service $09).
Configuration Parameters	

SWS Item	Dcm633 :		
Name	DcmDspGetVehInfoTypeFnc		
Description	Function name for reading the associated INFOTYPE. (GetInfotype-function)		
Multiplicity	1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm631 :		
Name	DcmDspVehInfoInfoType		
Description	INFOTYPE for Service $09		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Dcm632 :		
Name	DcmDspVehInfoSize		
Description	Length of data of associated INFOTYPE.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Dcm_00727 :		
Name	DcmDspVehInfoUsePort		
Description	When this is true, the DCM uses a port (whose name is the name of the DcmDspVehInfo. In that case, the DcmDspGetVehInfoTypeFnc must be an empty string as the RTE APIs are used. When this is false, the DCM calls the funtion defined in DcmDspGetVehInfoTypeFnc.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile	X	All Variants

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

No Included Containers

10.2.53 DcmDspObdService04

SWS Item	Dcm01020_Conf :
Container Name	DcmDspObdService04
Description	This container contains the configuration (parameters) of the Clear/Reset Emission-Related Diagnostic Information service (Service $04)
Configuration Parameters	

SWS Item	Dcm01019_Conf :		
Name	DcmObdService04Kind		
Description	The configuration parameter DcmObdService04Kind makes it possible to configure the DCM module so that the OBD Service $04 clears either all DTCs (DcmObdService04Kind=DEM_DTC_KIND_ALL_DTCS) or only the OBD-relevant DTCs (DcmObdService04Kind=DEM_DTC_KIND_EMISSION_REL_DTCS).		
Multiplicity	1		
Type	EnumerationParamDef		
Range	DEM_DTC_KIND_ALL_DTCS	Service 04 clears all DTCs.	
	DEM_DTC_KIND_EMISSION_REL_DTCS	Service 04 clears all OBD-relevant DTCs	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.54 DcmGeneral

SWS Item	ECUC_Dcm_00822 :
Container Name	DcmGeneral{COMPONENT_WIDE_PARAMETERS}
Description	This container contains the configuration (parameters) for Component wide parameters
Configuration Parameters	

SWS Item	ECUC_Dcm_00823 :
Name	DcmDevErrorDetect {DCM_DEV_ERROR_DETECT}
Description	Preprocessor switch to enable or disable the Development Error Detection (DET) mechanism.
Multiplicity	1
Type	BooleanParamDef

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

SWS Item	**ECUC_Dcm_01031 :**		
Name	DcmRequestIndicationEnabled		
Description	Allows to enable or disable the requested indication mechanism.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dcm600 :**		
Name	DcmRespondAllRequest		
Description	If set to FALSE the DCM will not respond to diagnostic request that contains a service ID which is in the range from 0x40 to 0x7F or in the range from 0xC0 to 0xFF (Response IDs).		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_01041 :**		
Name	DcmStartupNumberOfRetries {DCM_STARTUP_NUMBER_OF_RETRIES}		
Description	If the transmission of the response message fails, the Dcm retries the transmission for a number of times that is configured by this parameter.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**ECUC_Dcm_01042 :**		
Name	DcmStartupTimerDuration {DCM_STARTUP_DURATION}		
Description	Startup Timer Duration in seconds.		
Multiplicity	0..1		
Type	FloatParamDef		
Range	0 .. 2		

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

SWS Item	**ECUC_Dcm_00820 :**		
Name	DcmTaskTime {DCM_TASK_TIME}		
Description	Allow to configure the time for the periodic cyclic task (in ms). Please note: This configuration value shall be equal to the value in the ScheduleManager module. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM. min: A negative value is not allowed. upperMultiplicity: Exactly one TaskTime must be specified per configuration. lowerMultiplicity: Exactly one TaskTime must be specified per configuration.		
Multiplicity	1		
Type	FloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**ECUC_Dcm_00821 :**		
Name	DcmVersionInfoApi {DCM_VERSION_INFO_API}		
Description	Preprocessor switch to enable or disable the output Version info of the functionality.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

No Included Containers

10.2.55 DcmPageBufferCfg

SWS Item	**ECUC_Dcm_00775 :**
Container Name	DcmPageBufferCfg{PAGE_BUFFER_CFG}
Description	This container contains the configuration (parameters) for Page Buffer handling
Configuration Parameters	

SWS Item	**ECUC_Dcm_00776 :**
Name	DcmPagedBufferEnabled {DCM_PAGEDBUFFER_ENABLED}
Description	Allow to enable or disable the Page buffer mechanism. true = Page buffer handling enabled false = Page Buffer handling disabled

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

SWS Item	ECUC_Dcm_00774 :		
Name	DcmPagedBufferTimeout {DCM_PAGEDBUFFER_TIMEOUT}		
Description	Allow to configure the Timeout (in ms) towards the application for filling the next page. This parameter is only relevant if the Page Buffer handling is enabled. (DcmPagedBufferEnabled = TRUE) Defined in the DCM SWS as uint16, this parameter contains a value in milliseconds. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM. min: A negative value is not allowed. upperMultiplicity: Exactly one Timeout must be specified per configuration. lowerMultiplicity: Exactly one Timeout must be specified per configuration.		
Multiplicity	0..1		
Type	FloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmPagedBufferEnabled		

No Included Containers

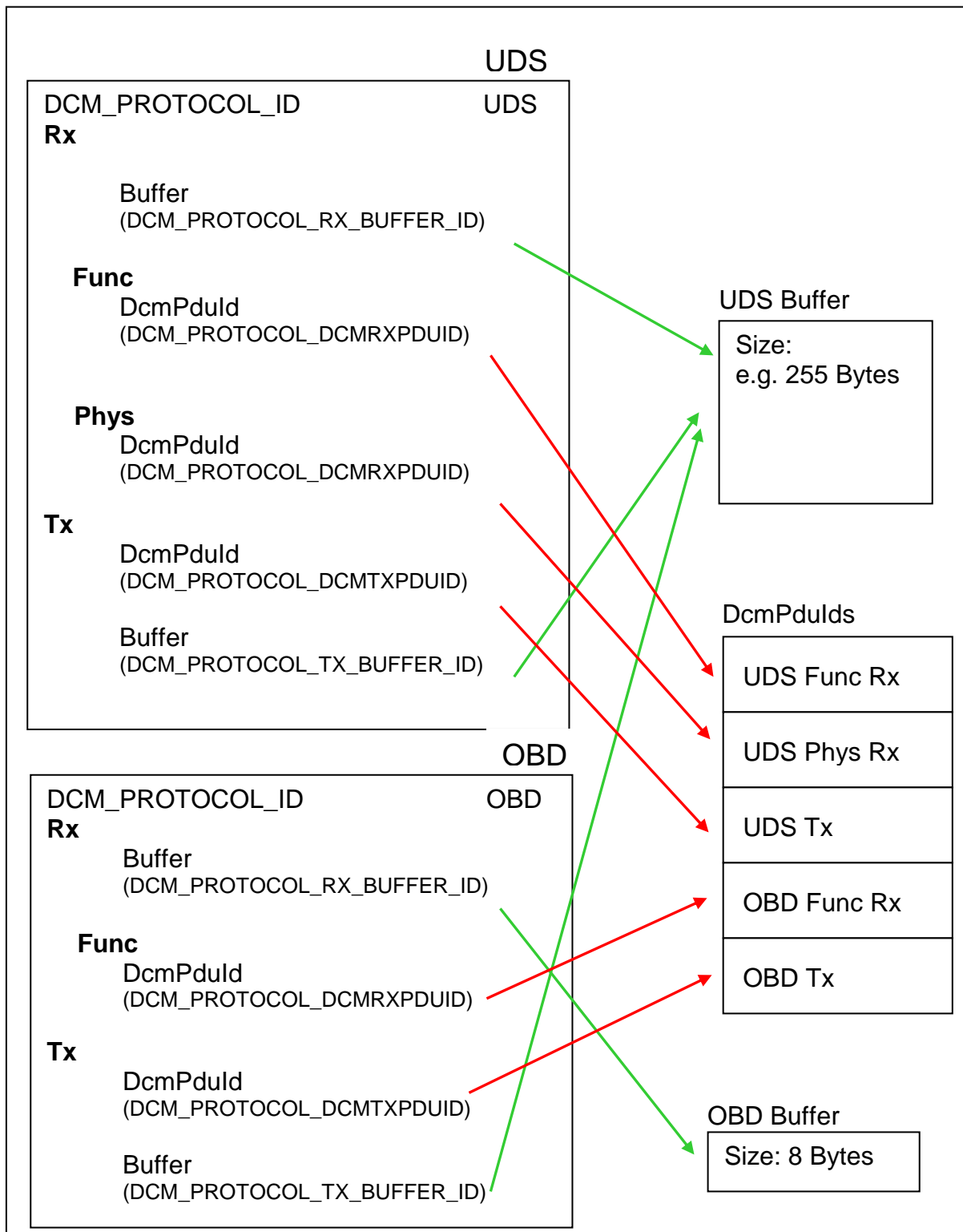## 10.3 Protocol Configuration Example

Figure 52 Examples of protocol configuration with focus on buffer / DcmPduId settings

Above example shows protocol configuration at the use cases examples OBD and UDS (used for customer enhanced diagnosis). It is assumed that for UDS

communication, there are functional and physical requests. There will be separate DcmPduRxIds for functional and physical reception.

Concerning buffer configuration it is proposed to use a separate buffer for the functional requests. This in correspondence to support the keep alive logic with functional addressed TesterPresent commands.

It is also proposed to use a separate receive buffer for the OBD commands. This in reference to support the protocol switch functionality.

It is allowed to share for both protocols the transmit buffer.

Please note:

The DcmDslProtocolRx has two possible configurations:

- functional
- physical

The physical shall have a 1:1 (or 1:0) dependency to the DcmDslMainConnection. (which means: **DcmDslProtocolRxPduRef** in combination DCM_PROTOCOL_RX_ADDR_TYP = physical can exist only once per "Module")

The functional shall have a 1:n dependency to the DcmDslMainConnection. (which means: **DcmDslProtocolRxPduRef** in combination DCM_PROTOCOL_RX_ADDR_TYP = functional can exist several times per "Module")

The DcmDslProtocolTx shall exist only once per "Module"

10.4 Published Information

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

The standard common published information like

- vendorId (<Module>_VENDOR_ID),
- moduleId (<Module>_MODULE_ID),
- arMajorVersion (<Module>_AR_MAJOR_VERSION),
- arMinorVersion (<Module>_ AR_MINOR_VERSION),
- arPatchVersion (<Module>_ AR_PATCH_VERSION),
- swMajorVersion (<Module>_SW_MAJOR_VERSION),
- swMinorVersion (<Module>_ SW_MINOR_VERSION),
- swPatchVersion (<Module>_ SW_PATCH_VERSION),
- vendorApiInfix (<Module>_VENDOR_API_INFIX)

is provided in the BSW Module Description Template.