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

Document Version	5.2.0
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History

Date	Version	Changed by	Change Description
31.03.2014	5.2.0	AUTOSAR Release Management	• Added functional description for DIDRange usage • Added support for bootloader interaction • Revised the header file structure • Editorial changes
31.10.2013	5.1.0	AUTOSAR Release Management	• Created API tables for service interfaces • Provided synchronous and asynchronous APIs for DataServices callouts • Harmonization for the length parameter interpretation all over RDBI, WDBI and RC services to be in bytes • Editorial changes • Removed chapter(s) on change documentation
12.03.2013	5.0.0	AUTOSAR Administration	• Added Response on Event support • Rework configuration for S/R communication • Rework OBD Service $06 management
01.12.2011	4.2.0	AUTOSAR Administration	• Change interaction with BswM module for mode management • Change of callout configuration management for services and sub-services processing • Synchronous and asynchronous clarification

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
24.11.2010	4.1.0	AUTOSAR Administration	• ComM_DCM_InactiveDiagnostic and ComM_DCM_ActiveDiagnostic has been defined as mandatory interfaces. • DcmDslPeriodicTxConfirmationPduId multiplicity changed and creation of DcmDslPeriodicConnection parameter in order to link the confirmation Id with TxPdu Id for PeriodicTransmission. • Dem_GetDTCOfOBDFreezeFrame, Dlt_ConditionCheckRead added as optional interfaces • DspInternal_<DiagnosticService> Api moved to mandatory internal interface to support the ECU Supplier diagnosis. • Rework of ReadData operation
08.12.2009	4.0.0	AUTOSAR Administration	• Add support of following UDS services : ReadMemoryByAdress, WriteMemoryByAdress, RequestDownload, RequestUpload, TransferData, RequestTransferExit, CommunicationControl, ResponseOnEvent. • Add of bootloader interaction • Add of BswM interaction • Add of IoHwAb interaction • Add of DLT interaction • Add of Signal based approach on RTE interfaces • Legal nvocation 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 nvocation revised

Document Change History

Date	Version	Changed by	Change Description
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
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

Advice for users

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

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

Table of Contents

Known limitations

The following limitations apply when using the DCM module:

- The DCM module does not provide any diagnostic multi-channel capabilities. 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 [16] shall be processed prior to any enhanced diagnostic requests, the DCM module provides a protocol switching mechanism based on protocol prioritization.
- 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 onComparisionOfValues of Service ResponseOnEvent is not supported in the current release.
- Subfunction onTimerInterrupt of Service ResponseOnEvent is not supported in the current release.
- UDS Service SecuredDataTransmission (0x84) is not supported in the current release.
- The DCM SWS does not cover any SAE J1939 related diagnostic requirements.
- Due to DEM limitation, the diagnostic service $19 05 is limited to the OBD legislative freeze frame.
- Management of IOControl service without InputOutputControlParameter in request and response is not supported
- The length of controlState parameter in IOControl request and response has to be of same size (due to the one configuration parameter DcmDspDataSize)
- Same layout of a DID which is used in RDBI, WDBI or IOCBI services
- The user optional parameter DTCSettingControlOptionRecord in the ControlDTCSetting request is only supported if it corresponds to a groupOfDTC value. In other cases it has to be managed in a vendor specific implementation.
- Only the ControlDTCSetting sub-functions 0x01 and 0x02 are supported.
- The handling of infrastructure errors reported by the RTE during DCM/DEM

  ↔ SW-C interactions is missing from the SWS and might have to be taken into account by implementers if they need it.
- The Dcm does not support DLT for ROE
- The ROE ServiceToRespondTo does not support PageBuffering
- ROE only supports sub-function listed in Table 2
- DID range feature cannot be applied for services DynamicallyDefineDataIdentifier, ReadDataByPeriodicIdentifier and InputOutputControlById

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.

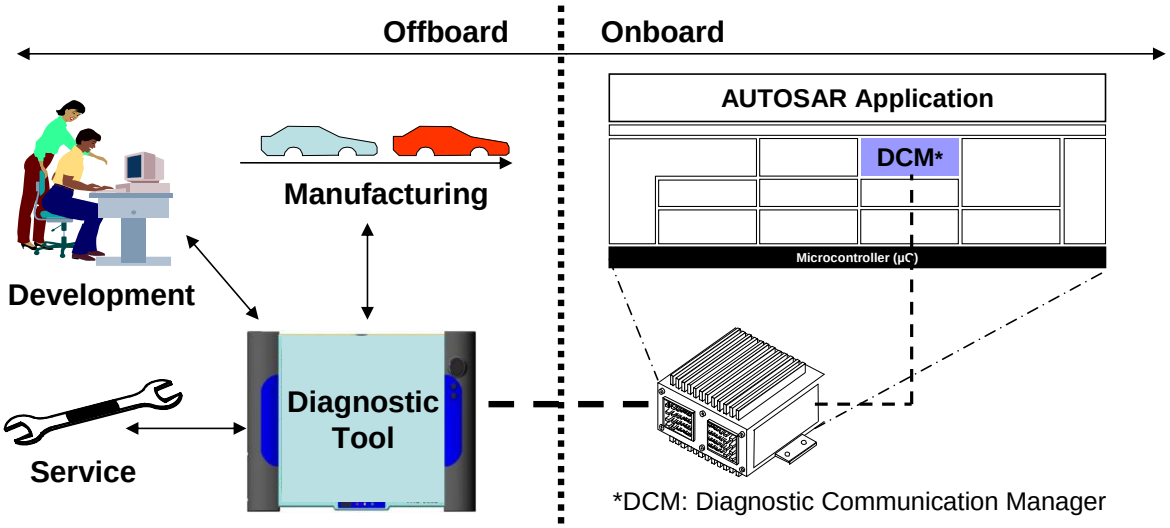


Figure 1 Overview of the communication between the external diagnostic tools and the onboard AUTOSAR Application

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 [15] services. In addition, the DCM module provides mechanisms to support the OBD services $01 - $0A defined in documents [20] and [16]. 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 [18]: Implementation of unified diagnostic services (UDS on CAN)
- ISO15765-4 [19]: Requirements for emission-related systems, Chapter 5 “Session Layer”

In the AUTOSAR Architecture the Diagnostic Communication Manager is located in the Communication Services (Service Layer).

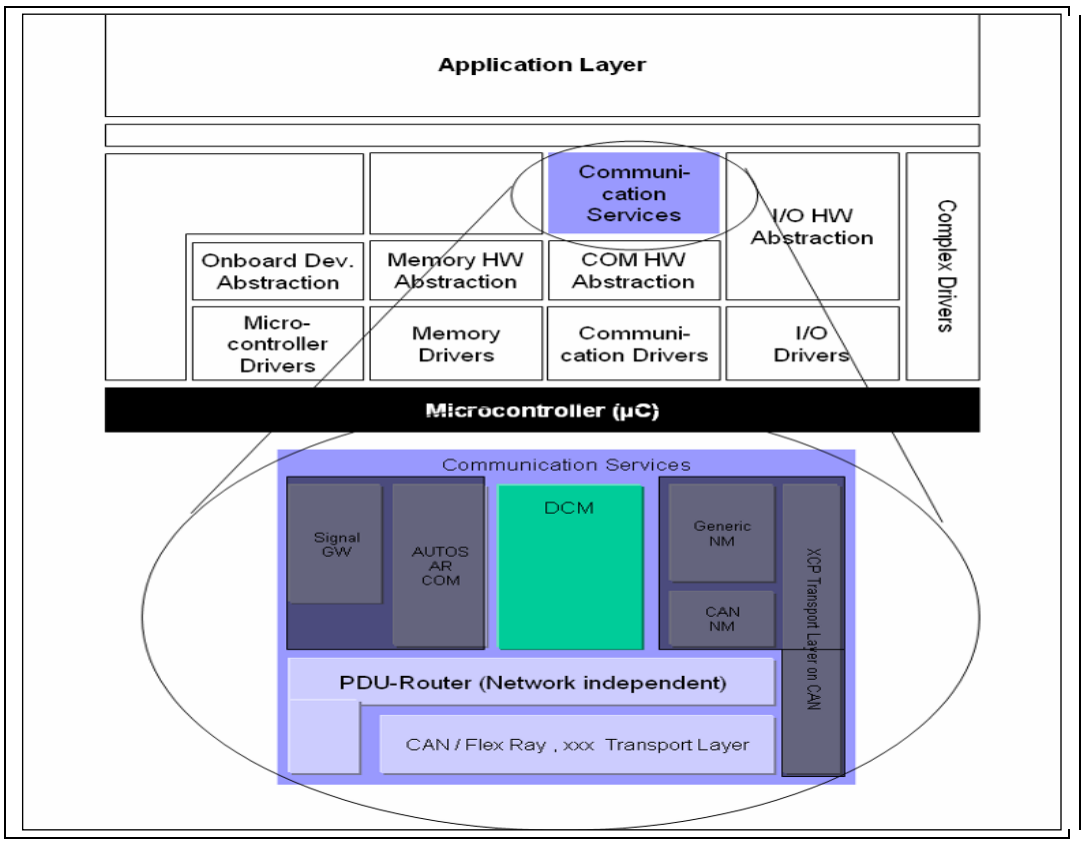


Figure 2 Position of the DCM module in AUTOSAR Architecture

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 [15])
OBD Service	This refers to an OBD Service as defined in ISO15031-5 (see [16])

Term	Description
AddressAndLengthFormatIdentifier	Defines the number of bytes used for the memoryAddress and memorySize parameter in the request messages
OBD Scan tool	See definition External Diagnostic Tool

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] General Requirements on Basic Software Modules,
AUTOSAR_SRS_General.pdf

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

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

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

[5] Layered Software Architecture,
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[6] Specification of ECU Configuration,
AUTOSAR_TPS_ECUConfiguration.pdf

[7] Specification of Diagnostic Event Manager,
AUTOSAR_SWS_DiagnosticEventManager.pdf

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

[9] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[10] Specification of NVRAM Manager,

AUTOSAR_SWS_NVRAMManager.pdf

[11] Specification of I/O Hardware Abstraction,
AUTOSAR_SWS_IOHardwareAbstraction.pdf

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

[13] Specification of Basic Software Mode Manager,
AUTOSAR_SWS_BSWModeManager.pdf

[14] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

[15] ISO14229-1 Unified diagnostic services (UDS) – Part 1: Specification and Requirements (Release 2006 12-01)
[16] ISO15031-5.4 Communication between vehicle and external equipment for emissions-related diagnostics – Part 5: Emission-related diagnostic services (2005-01-13)
[17] ISO15765-2: Road vehicles – Diagnostics on Controller Area Networks (CAN) – Part 2: Network layer services
[18] ISO15765-3: Diagnostics on controller area network (CAN) – Part 3: Implementation of unified diagnostic services (UDS on CAN) (Release 2004 10-06)
[19] ISO15765-4 Diagnostics on controller area network (CAN) – Part 4: Requirements for emission-related systems (Release 2005 01-04)
[20] SAE J1979 Rev May 2007
[21] OSEK/ VDX Communication Version 3.0.3 OSEKSWS_Com_00303

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [14] (SWS BSW General), which is also valid for Diagnostic Communication Manager.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Diagnostic Communication Manager.

4 Constraints and assumptions

4.1 Applicability to car domains

The DCM module can be used for all car domains.

4.2 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:

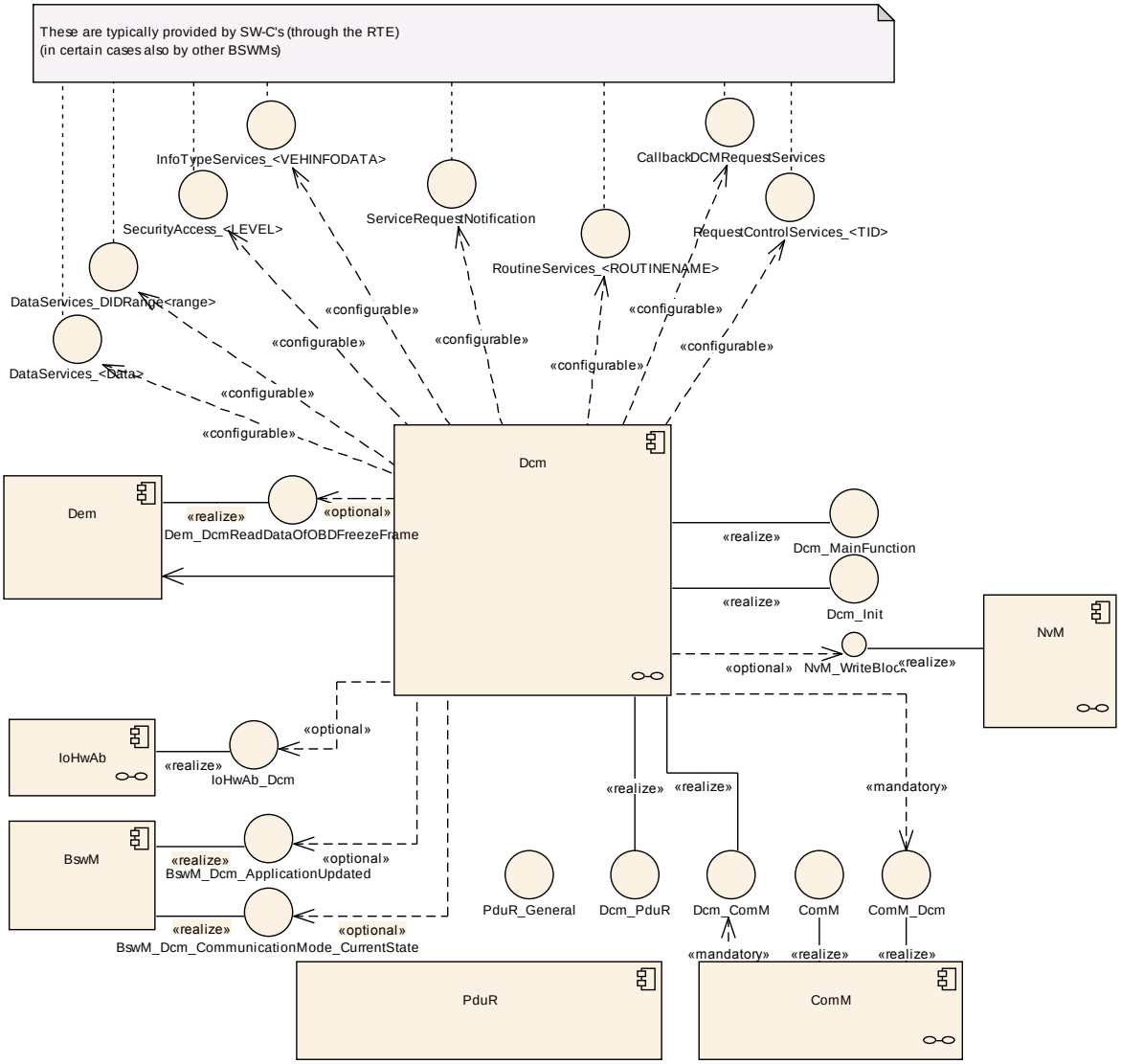


Figure 3 Interaction of the DCM with other modules

- 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 [15] 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.
- BswM: The Dcm notifies the BswM that the application was updated if the initialization of the DCM is the consequence of a jump from the bootloader . The Dcm also indicates to the BswM a communication mode change

5.1 File structure

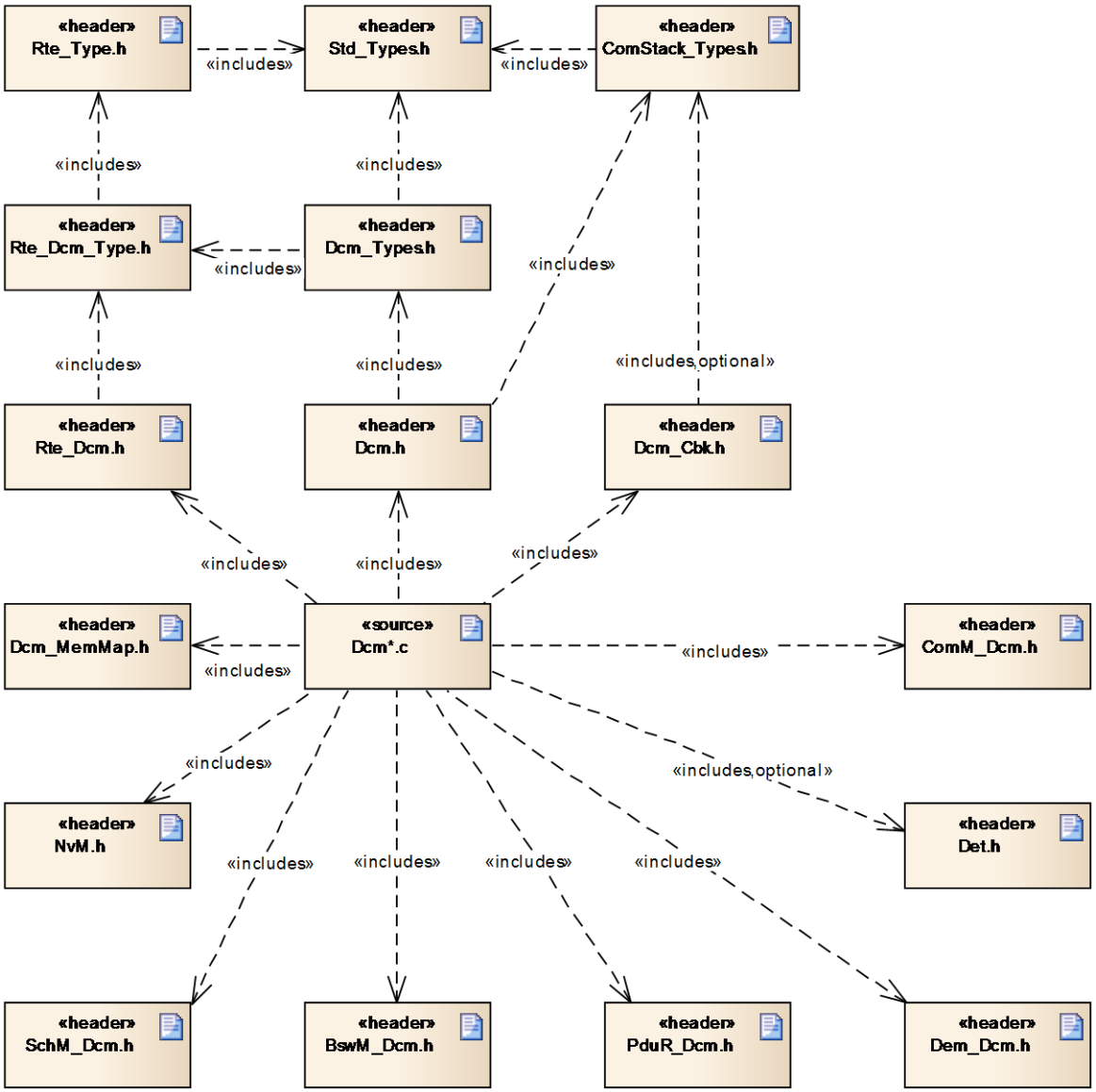


Figure 4: DCM module file structure

[SWS_Dcm_00055] ⌈ The Dcm module shall use the header file structure shown in Figure 4. ⌋(BSW00381, BSW00412, BSW00435, BSW00436, BSW00302)

[**SWS_Dcm_00683**] ⌈ The file Dcm_Types.h shall provide all Dcm types definition. ⌋()

Note: Dcm_Types.h will include types generated by RTE indirectly via inclusion of Rte_Dcm_Type.h.

[SWS_Dcm_01065]⌈ The file Dcm.h shall provide all type definitions and APIs used by other BSW modules for direct calls as described in chapter 8.4 Function definitions.⌋()

[SWS_Dcm_01066]⌈ The file Dcm_Cbk.h shall provide all Typedefinitions and APIs used by other BSW modules for direct calls as described in chapter 8.5 Callback Notifications.⌋()

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	constr_6000
-	-	constr_6001
-	-	constr_6002
-	-	constr_6003
-	-	constr_6004
-	-	constr_6005
-	-	constr_6006
-	-	constr_6007
-	-	constr_6008
-	-	constr_6009
-	-	constr_6010
-	-	constr_6011
-	-	constr_6012
-	-	constr_6013
-	-	constr_6014
-	-	constr_6015
-	-	constr_6016
-	-	constr_6017
-	-	constr_6018
-	-	constr_6019
-	-	constr_6020
-	-	constr_6021
-	-	constr_6022
-	-	constr_6023
-	-	constr_6024
-	-	SWS_Dcm_00039
-	-	SWS_Dcm_00052
-	-	SWS_Dcm_00079
-	-	SWS_Dcm_00084
-	-	SWS_Dcm_00085
-	-	SWS_Dcm_00092
-	-	SWS_Dcm_00093
-	-	SWS_Dcm_00094
-	-	SWS_Dcm_00111
-	-	SWS_Dcm_00112
-	-	SWS_Dcm_00113

-	-	SWS_Dcm_00114
-	-	SWS_Dcm_00115
-	-	SWS_Dcm_00117
-	-	SWS_Dcm_00118
-	-	SWS_Dcm_00119
-	-	SWS_Dcm_00120
-	-	SWS_Dcm_00121
-	-	SWS_Dcm_00122
-	-	SWS_Dcm_00123
-	-	SWS_Dcm_00125
-	-	SWS_Dcm_00126
-	-	SWS_Dcm_00127
-	-	SWS_Dcm_00128
-	-	SWS_Dcm_00131
-	-	SWS_Dcm_00132
-	-	SWS_Dcm_00133
-	-	SWS_Dcm_00134
-	-	SWS_Dcm_00135
-	-	SWS_Dcm_00136
-	-	SWS_Dcm_00139
-	-	SWS_Dcm_00140
-	-	SWS_Dcm_00141
-	-	SWS_Dcm_00145
-	-	SWS_Dcm_00146
-	-	SWS_Dcm_00147
-	-	SWS_Dcm_00148
-	-	SWS_Dcm_00149
-	-	SWS_Dcm_00150
-	-	SWS_Dcm_00151
-	-	SWS_Dcm_00152
-	-	SWS_Dcm_00153
-	-	SWS_Dcm_00154
-	-	SWS_Dcm_00155
-	-	SWS_Dcm_00156
-	-	SWS_Dcm_00157
-	-	SWS_Dcm_00159
-	-	SWS_Dcm_00160
-	-	SWS_Dcm_00161
-	-	SWS_Dcm_00162
-	-	SWS_Dcm_00163

-	-	SWS_Dcm_00164
-	-	SWS_Dcm_00165
-	-	SWS_Dcm_00166
-	-	SWS_Dcm_00167
-	-	SWS_Dcm_00168
-	-	SWS_Dcm_00169
-	-	SWS_Dcm_00170
-	-	SWS_Dcm_00178
-	-	SWS_Dcm_00192
-	-	SWS_Dcm_00193
-	-	SWS_Dcm_00195
-	-	SWS_Dcm_00196
-	-	SWS_Dcm_00197
-	-	SWS_Dcm_00198
-	-	SWS_Dcm_00201
-	-	SWS_Dcm_00202
-	-	SWS_Dcm_00203
-	-	SWS_Dcm_00204
-	-	SWS_Dcm_00211
-	-	SWS_Dcm_00217
-	-	SWS_Dcm_00218
-	-	SWS_Dcm_00221
-	-	SWS_Dcm_00222
-	-	SWS_Dcm_00223
-	-	SWS_Dcm_00224
-	-	SWS_Dcm_00225
-	-	SWS_Dcm_00228
-	-	SWS_Dcm_00231
-	-	SWS_Dcm_00232
-	-	SWS_Dcm_00235
-	-	SWS_Dcm_00236
-	-	SWS_Dcm_00237
-	-	SWS_Dcm_00238
-	-	SWS_Dcm_00240
-	-	SWS_Dcm_00241
-	-	SWS_Dcm_00249
-	-	SWS_Dcm_00251
-	-	SWS_Dcm_00253
-	-	SWS_Dcm_00254
-	-	SWS_Dcm_00255

-	-	SWS_Dcm_00256
-	-	SWS_Dcm_00257
-	-	SWS_Dcm_00258
-	-	SWS_Dcm_00259
-	-	SWS_Dcm_00260
-	-	SWS_Dcm_00269
-	-	SWS_Dcm_00271
-	-	SWS_Dcm_00272
-	-	SWS_Dcm_00273
-	-	SWS_Dcm_00275
-	-	SWS_Dcm_00287
-	-	SWS_Dcm_00297
-	-	SWS_Dcm_00300
-	-	SWS_Dcm_00302
-	-	SWS_Dcm_00307
-	-	SWS_Dcm_00311
-	-	SWS_Dcm_00321
-	-	SWS_Dcm_00323
-	-	SWS_Dcm_00324
-	-	SWS_Dcm_00325
-	-	SWS_Dcm_00333
-	-	SWS_Dcm_00334
-	-	SWS_Dcm_00342
-	-	SWS_Dcm_00344
-	-	SWS_Dcm_00345
-	-	SWS_Dcm_00346
-	-	SWS_Dcm_00350
-	-	SWS_Dcm_00351
-	-	SWS_Dcm_00352
-	-	SWS_Dcm_00353
-	-	SWS_Dcm_00354
-	-	SWS_Dcm_00356
-	-	SWS_Dcm_00358
-	-	SWS_Dcm_00360
-	-	SWS_Dcm_00371
-	-	SWS_Dcm_00372
-	-	SWS_Dcm_00373
-	-	SWS_Dcm_00377
-	-	SWS_Dcm_00379
-	-	SWS_Dcm_00386

-	-	SWS_Dcm_00387
-	-	SWS_Dcm_00392
-	-	SWS_Dcm_00394
-	-	SWS_Dcm_00395
-	-	SWS_Dcm_00396
-	-	SWS_Dcm_00397
-	-	SWS_Dcm_00398
-	-	SWS_Dcm_00399
-	-	SWS_Dcm_00400
-	-	SWS_Dcm_00401
-	-	SWS_Dcm_00402
-	-	SWS_Dcm_00403
-	-	SWS_Dcm_00404
-	-	SWS_Dcm_00405
-	-	SWS_Dcm_00407
-	-	SWS_Dcm_00408
-	-	SWS_Dcm_00409
-	-	SWS_Dcm_00418
-	-	SWS_Dcm_00419
-	-	SWS_Dcm_00420
-	-	SWS_Dcm_00422
-	-	SWS_Dcm_00423
-	-	SWS_Dcm_00433
-	-	SWS_Dcm_00434
-	-	SWS_Dcm_00435
-	-	SWS_Dcm_00436
-	-	SWS_Dcm_00437
-	-	SWS_Dcm_00438
-	-	SWS_Dcm_00439
-	-	SWS_Dcm_00440
-	-	SWS_Dcm_00442
-	-	SWS_Dcm_00443
-	-	SWS_Dcm_00444
-	-	SWS_Dcm_00459
-	-	SWS_Dcm_00460
-	-	SWS_Dcm_00462
-	-	SWS_Dcm_00463
-	-	SWS_Dcm_00467
-	-	SWS_Dcm_00468
-	-	SWS_Dcm_00469

-	-	SWS_Dcm_00470
-	-	SWS_Dcm_00473
-	-	SWS_Dcm_00474
-	-	SWS_Dcm_00481
-	-	SWS_Dcm_00482
-	-	SWS_Dcm_00483
-	-	SWS_Dcm_00488
-	-	SWS_Dcm_00489
-	-	SWS_Dcm_00490
-	-	SWS_Dcm_00491
-	-	SWS_Dcm_00492
-	-	SWS_Dcm_00493
-	-	SWS_Dcm_00494
-	-	SWS_Dcm_00495
-	-	SWS_Dcm_00511
-	-	SWS_Dcm_00512
-	-	SWS_Dcm_00516
-	-	SWS_Dcm_00517
-	-	SWS_Dcm_00518
-	-	SWS_Dcm_00520
-	-	SWS_Dcm_00521
-	-	SWS_Dcm_00527
-	-	SWS_Dcm_00528
-	-	SWS_Dcm_00529
-	-	SWS_Dcm_00530
-	-	SWS_Dcm_00537
-	-	SWS_Dcm_00539
-	-	SWS_Dcm_00540
-	-	SWS_Dcm_00541
-	-	SWS_Dcm_00543
-	-	SWS_Dcm_00544
-	-	SWS_Dcm_00556
-	-	SWS_Dcm_00557
-	-	SWS_Dcm_00558
-	-	SWS_Dcm_00560
-	-	SWS_Dcm_00561
-	-	SWS_Dcm_00562
-	-	SWS_Dcm_00563
-	-	SWS_Dcm_00564
-	-	SWS_Dcm_00565

-	-	SWS_Dcm_00566
-	-	SWS_Dcm_00567
-	-	SWS_Dcm_00568
-	-	SWS_Dcm_00569
-	-	SWS_Dcm_00570
-	-	SWS_Dcm_00571
-	-	SWS_Dcm_00574
-	-	SWS_Dcm_00575
-	-	SWS_Dcm_00576
-	-	SWS_Dcm_00578
-	-	SWS_Dcm_00579
-	-	SWS_Dcm_00580
-	-	SWS_Dcm_00581
-	-	SWS_Dcm_00587
-	-	SWS_Dcm_00588
-	-	SWS_Dcm_00589
-	-	SWS_Dcm_00590
-	-	SWS_Dcm_00594
-	-	SWS_Dcm_00595
-	-	SWS_Dcm_00601
-	-	SWS_Dcm_00614
-	-	SWS_Dcm_00616
-	-	SWS_Dcm_00617
-	-	SWS_Dcm_00620
-	-	SWS_Dcm_00621
-	-	SWS_Dcm_00622
-	-	SWS_Dcm_00623
-	-	SWS_Dcm_00624
-	-	SWS_Dcm_00625
-	-	SWS_Dcm_00628
-	-	SWS_Dcm_00632
-	-	SWS_Dcm_00638
-	-	SWS_Dcm_00639
-	-	SWS_Dcm_00640
-	-	SWS_Dcm_00641
-	-	SWS_Dcm_00642
-	-	SWS_Dcm_00643
-	-	SWS_Dcm_00644
-	-	SWS_Dcm_00645
-	-	SWS_Dcm_00646

-	-	SWS_Dcm_00647
-	-	SWS_Dcm_00651
-	-	SWS_Dcm_00652
-	-	SWS_Dcm_00653
-	-	SWS_Dcm_00654
-	-	SWS_Dcm_00655
-	-	SWS_Dcm_00656
-	-	SWS_Dcm_00659
-	-	SWS_Dcm_00660
-	-	SWS_Dcm_00668
-	-	SWS_Dcm_00669
-	-	SWS_Dcm_00670
-	-	SWS_Dcm_00671
-	-	SWS_Dcm_00672
-	-	SWS_Dcm_00673
-	-	SWS_Dcm_00674
-	-	SWS_Dcm_00677
-	-	SWS_Dcm_00678
-	-	SWS_Dcm_00680
-	-	SWS_Dcm_00681
-	-	SWS_Dcm_00682
-	-	SWS_Dcm_00683
-	-	SWS_Dcm_00684
-	-	SWS_Dcm_00685
-	-	SWS_Dcm_00686
-	-	SWS_Dcm_00687
-	-	SWS_Dcm_00688
-	-	SWS_Dcm_00690
-	-	SWS_Dcm_00691
-	-	SWS_Dcm_00692
-	-	SWS_Dcm_00694
-	-	SWS_Dcm_00696
-	-	SWS_Dcm_00697
-	-	SWS_Dcm_00698
-	-	SWS_Dcm_00699
-	-	SWS_Dcm_00700
-	-	SWS_Dcm_00701
-	-	SWS_Dcm_00702
-	-	SWS_Dcm_00703
-	-	SWS_Dcm_00704

-	-	SWS_Dcm_00705
-	-	SWS_Dcm_00706
-	-	SWS_Dcm_00707
-	-	SWS_Dcm_00708
-	-	SWS_Dcm_00715
-	-	SWS_Dcm_00716
-	-	SWS_Dcm_00718
-	-	SWS_Dcm_00719
-	-	SWS_Dcm_00720
-	-	SWS_Dcm_00721
-	-	SWS_Dcm_00722
-	-	SWS_Dcm_00723
-	-	SWS_Dcm_00724
-	-	SWS_Dcm_00725
-	-	SWS_Dcm_00726
-	-	SWS_Dcm_00727
-	-	SWS_Dcm_00728
-	-	SWS_Dcm_00729
-	-	SWS_Dcm_00732
-	-	SWS_Dcm_00733
-	-	SWS_Dcm_00734
-	-	SWS_Dcm_00735
-	-	SWS_Dcm_00738
-	-	SWS_Dcm_00739
-	-	SWS_Dcm_00740
-	-	SWS_Dcm_00741
-	-	SWS_Dcm_00742
-	-	SWS_Dcm_00751
-	-	SWS_Dcm_00752
-	-	SWS_Dcm_00753
-	-	SWS_Dcm_00754
-	-	SWS_Dcm_00755
-	-	SWS_Dcm_00756
-	-	SWS_Dcm_00757
-	-	SWS_Dcm_00758
-	-	SWS_Dcm_00759
-	-	SWS_Dcm_00760
-	-	SWS_Dcm_00761
-	-	SWS_Dcm_00762
-	-	SWS_Dcm_00763

-	-	SWS_Dcm_00764
-	-	SWS_Dcm_00766
-	-	SWS_Dcm_00768
-	-	SWS_Dcm_00769
-	-	SWS_Dcm_00770
-	-	SWS_Dcm_00773
-	-	SWS_Dcm_00774
-	-	SWS_Dcm_00775
-	-	SWS_Dcm_00776
-	-	SWS_Dcm_00777
-	-	SWS_Dcm_00778
-	-	SWS_Dcm_00779
-	-	SWS_Dcm_00780
-	-	SWS_Dcm_00781
-	-	SWS_Dcm_00782
-	-	SWS_Dcm_00783
-	-	SWS_Dcm_00784
-	-	SWS_Dcm_00785
-	-	SWS_Dcm_00786
-	-	SWS_Dcm_00788
-	-	SWS_Dcm_00789
-	-	SWS_Dcm_00790
-	-	SWS_Dcm_00793
-	-	SWS_Dcm_00794
-	-	SWS_Dcm_00796
-	-	SWS_Dcm_00797
-	-	SWS_Dcm_00798
-	-	SWS_Dcm_00799
-	-	SWS_Dcm_00800
-	-	SWS_Dcm_00801
-	-	SWS_Dcm_00802
-	-	SWS_Dcm_00803
-	-	SWS_Dcm_00804
-	-	SWS_Dcm_00805
-	-	SWS_Dcm_00806
-	-	SWS_Dcm_00807
-	-	SWS_Dcm_00808
-	-	SWS_Dcm_00809
-	-	SWS_Dcm_00810
-	-	SWS_Dcm_00811

-	-	SWS_Dcm_00812
-	-	SWS_Dcm_00813
-	-	SWS_Dcm_00814
-	-	SWS_Dcm_00815
-	-	SWS_Dcm_00818
-	-	SWS_Dcm_00819
-	-	SWS_Dcm_00820
-	-	SWS_Dcm_00821
-	-	SWS_Dcm_00822
-	-	SWS_Dcm_00823
-	-	SWS_Dcm_00824
-	-	SWS_Dcm_00825
-	-	SWS_Dcm_00826
-	-	SWS_Dcm_00827
-	-	SWS_Dcm_00828
-	-	SWS_Dcm_00829
-	-	SWS_Dcm_00830
-	-	SWS_Dcm_00831
-	-	SWS_Dcm_00832
-	-	SWS_Dcm_00833
-	-	SWS_Dcm_00834
-	-	SWS_Dcm_00835
-	-	SWS_Dcm_00836
-	-	SWS_Dcm_00837
-	-	SWS_Dcm_00838
-	-	SWS_Dcm_00839
-	-	SWS_Dcm_00840
-	-	SWS_Dcm_00841
-	-	SWS_Dcm_00843
-	-	SWS_Dcm_00845
-	-	SWS_Dcm_00848
-	-	SWS_Dcm_00849
-	-	SWS_Dcm_00851
-	-	SWS_Dcm_00852
-	-	SWS_Dcm_00853
-	-	SWS_Dcm_00854
-	-	SWS_Dcm_00855
-	-	SWS_Dcm_00856
-	-	SWS_Dcm_00857
-	-	SWS_Dcm_00858

-	-	SWS_Dcm_00859
-	-	SWS_Dcm_00860
-	-	SWS_Dcm_00861
-	-	SWS_Dcm_00862
-	-	SWS_Dcm_00863
-	-	SWS_Dcm_00864
-	-	SWS_Dcm_00865
-	-	SWS_Dcm_00866
-	-	SWS_Dcm_00867
-	-	SWS_Dcm_00868
-	-	SWS_Dcm_00869
-	-	SWS_Dcm_00870
-	-	SWS_Dcm_00871
-	-	SWS_Dcm_00872
-	-	SWS_Dcm_00873
-	-	SWS_Dcm_00874
-	-	SWS_Dcm_00875
-	-	SWS_Dcm_00876
-	-	SWS_Dcm_00877
-	-	SWS_Dcm_00878
-	-	SWS_Dcm_00879
-	-	SWS_Dcm_00880
-	-	SWS_Dcm_00884
-	-	SWS_Dcm_00885
-	-	SWS_Dcm_00886
-	-	SWS_Dcm_00887
-	-	SWS_Dcm_00888
-	-	SWS_Dcm_00889
-	-	SWS_Dcm_00890
-	-	SWS_Dcm_00891
-	-	SWS_Dcm_00892
-	-	SWS_Dcm_00893
-	-	SWS_Dcm_00894
-	-	SWS_Dcm_00895
-	-	SWS_Dcm_00896
-	-	SWS_Dcm_00897
-	-	SWS_Dcm_00898
-	-	SWS_Dcm_00900
-	-	SWS_Dcm_00901
-	-	SWS_Dcm_00902

-	-	SWS_Dcm_00903
-	-	SWS_Dcm_00905
-	-	SWS_Dcm_00906
-	-	SWS_Dcm_00907
-	-	SWS_Dcm_00908
-	-	SWS_Dcm_00909
-	-	SWS_Dcm_00912
-	-	SWS_Dcm_00913
-	-	SWS_Dcm_00914
-	-	SWS_Dcm_00915
-	-	SWS_Dcm_00918
-	-	SWS_Dcm_00920
-	-	SWS_Dcm_00921
-	-	SWS_Dcm_00922
-	-	SWS_Dcm_00923
-	-	SWS_Dcm_00924
-	-	SWS_Dcm_00925
-	-	SWS_Dcm_00926
-	-	SWS_Dcm_00927
-	-	SWS_Dcm_00928
-	-	SWS_Dcm_00929
-	-	SWS_Dcm_00930
-	-	SWS_Dcm_00931
-	-	SWS_Dcm_00933
-	-	SWS_Dcm_00934
-	-	SWS_Dcm_00940
-	-	SWS_Dcm_00941
-	-	SWS_Dcm_00942
-	-	SWS_Dcm_00943
-	-	SWS_Dcm_00944
-	-	SWS_Dcm_00945
-	-	SWS_Dcm_00946
-	-	SWS_Dcm_00947
-	-	SWS_Dcm_00948
-	-	SWS_Dcm_00949
-	-	SWS_Dcm_00950
-	-	SWS_Dcm_00951
-	-	SWS_Dcm_00952
-	-	SWS_Dcm_00953
-	-	SWS_Dcm_00954

-	-	SWS_Dcm_00956
-	-	SWS_Dcm_00957
-	-	SWS_Dcm_00958
-	-	SWS_Dcm_00962
-	-	SWS_Dcm_00963
-	-	SWS_Dcm_00964
-	-	SWS_Dcm_00965
-	-	SWS_Dcm_00966
-	-	SWS_Dcm_00967
-	-	SWS_Dcm_00968
-	-	SWS_Dcm_00969
-	-	SWS_Dcm_00970
-	-	SWS_Dcm_00971
-	-	SWS_Dcm_00972
-	-	SWS_Dcm_00973
-	-	SWS_Dcm_00974
-	-	SWS_Dcm_00976
-	-	SWS_Dcm_00977
-	-	SWS_Dcm_00978
-	-	SWS_Dcm_00979
-	-	SWS_Dcm_00980
-	-	SWS_Dcm_00981
-	-	SWS_Dcm_00982
-	-	SWS_Dcm_00983
-	-	SWS_Dcm_00984
-	-	SWS_Dcm_00985
-	-	SWS_Dcm_00986
-	-	SWS_Dcm_00987
-	-	SWS_Dcm_00988
-	-	SWS_Dcm_00989
-	-	SWS_Dcm_00990
-	-	SWS_Dcm_00991
-	-	SWS_Dcm_00992
-	-	SWS_Dcm_00993
-	-	SWS_Dcm_00994
-	-	SWS_Dcm_00995
-	-	SWS_Dcm_00996
-	-	SWS_Dcm_00997
-	-	SWS_Dcm_01000
-	-	SWS_Dcm_01001

-	-	SWS_Dcm_01002
-	-	SWS_Dcm_01003
-	-	SWS_Dcm_01004
-	-	SWS_Dcm_01005
-	-	SWS_Dcm_01010
-	-	SWS_Dcm_01011
-	-	SWS_Dcm_01012
-	-	SWS_Dcm_01013
-	-	SWS_Dcm_01014
-	-	SWS_Dcm_01015
-	-	SWS_Dcm_01017
-	-	SWS_Dcm_01018
-	-	SWS_Dcm_01019
-	-	SWS_Dcm_01020
-	-	SWS_Dcm_01021
-	-	SWS_Dcm_01022
-	-	SWS_Dcm_01023
-	-	SWS_Dcm_01024
-	-	SWS_Dcm_01025
-	-	SWS_Dcm_01026
-	-	SWS_Dcm_01027
-	-	SWS_Dcm_01028
-	-	SWS_Dcm_01029
-	-	SWS_Dcm_01030
-	-	SWS_Dcm_01031
-	-	SWS_Dcm_01032
-	-	SWS_Dcm_01033
-	-	SWS_Dcm_01034
-	-	SWS_Dcm_01035
-	-	SWS_Dcm_01036
-	-	SWS_Dcm_01037
-	-	SWS_Dcm_01038
-	-	SWS_Dcm_01039
-	-	SWS_Dcm_01040
-	-	SWS_Dcm_01041
-	-	SWS_Dcm_01042
-	-	SWS_Dcm_01043
-	-	SWS_Dcm_01044
-	-	SWS_Dcm_01045
-	-	SWS_Dcm_01046

-	-	SWS_Dcm_01047
-	-	SWS_Dcm_01048
-	-	SWS_Dcm_01050
-	-	SWS_Dcm_01051
-	-	SWS_Dcm_01052
-	-	SWS_Dcm_01053
-	-	SWS_Dcm_01055
-	-	SWS_Dcm_01056
-	-	SWS_Dcm_01057
-	-	SWS_Dcm_01058
-	-	SWS_Dcm_01059
-	-	SWS_Dcm_01060
-	-	SWS_Dcm_01061
-	-	SWS_Dcm_01062
-	-	SWS_Dcm_01065
-	-	SWS_Dcm_01066
-	-	SWS_Dcm_01067
BSW	-	SWS_Dcm_00999
BSW00302	-	SWS_Dcm_00055
BSW00307	-	SWS_Dcm_00999
BSW00314	-	SWS_Dcm_00999
BSW00321	-	SWS_Dcm_00999
BSW00326	-	SWS_Dcm_00999
BSW00328	-	SWS_Dcm_00999
BSW00334	-	SWS_Dcm_00999
BSW00336	-	SWS_Dcm_00999
BSW00338	-	SWS_Dcm_00040
BSW00339	-	SWS_Dcm_00999
BSW00341	-	SWS_Dcm_00999
BSW00342	-	SWS_Dcm_00999
BSW00347	-	SWS_Dcm_00999
BSW00358	-	SWS_Dcm_00037
BSW00361	-	SWS_Dcm_00999
BSW00369	-	SWS_Dcm_00044
BSW00373	-	SWS_Dcm_00053
BSW00375	-	SWS_Dcm_00999
BSW00376	-	SWS_Dcm_00053
BSW00378	-	SWS_Dcm_00999
BSW00381	-	SWS_Dcm_00055
BSW00383	-	SWS_Dcm_00999

BSW00385	-	SWS_Dcm_00999
BSW00386	-	SWS_Dcm_00999
BSW00387	-	SWS_Dcm_00999
BSW00406	-	SWS_Dcm_00999
BSW00407	-	SWS_Dcm_00065
BSW00409	-	SWS_Dcm_00999
BSW00412	-	SWS_Dcm_00055
BSW00413	-	SWS_Dcm_00999
BSW00414	-	SWS_Dcm_00037
BSW00415	-	SWS_Dcm_00999
BSW00416	-	SWS_Dcm_00999
BSW00417	-	SWS_Dcm_00999
BSW00422	-	SWS_Dcm_00999
BSW00423	-	SWS_Dcm_00999
BSW00424	-	SWS_Dcm_00053
BSW00425	-	SWS_Dcm_00999
BSW00426	-	SWS_Dcm_00999
BSW00427	-	SWS_Dcm_00999
BSW00428	-	SWS_Dcm_00999
BSW00429	-	SWS_Dcm_00999
BSW00432	-	SWS_Dcm_00999
BSW00433	-	SWS_Dcm_00999
BSW00435	-	SWS_Dcm_00055
BSW00436	-	SWS_Dcm_00055
BSW00437	-	SWS_Dcm_00999
BSW00438	-	SWS_Dcm_00037
BSW00439	-	SWS_Dcm_00999
BSW00440	-	SWS_Dcm_00999
BSW00442	-	SWS_Dcm_00506, SWS_Dcm_00507, SWS_Dcm_00508, SWS_Dcm_00509
BSW00443	-	SWS_Dcm_00999
BSW00444	-	SWS_Dcm_00999
BSW00445	-	SWS_Dcm_00999
BSW00446	-	SWS_Dcm_00999
BSW00447	-	SWS_Dcm_00999
BSW00453	-	SWS_Dcm_00999
BSW00455	-	SWS_Dcm_00999
BSW005	-	SWS_Dcm_00999
BSW010	-	SWS_Dcm_00999
BSW04001	-	SWS_Dcm_00243, SWS_Dcm_00244,

		SWS_Dcm_00245, SWS_Dcm_00246, SWS_Dcm_00410, SWS_Dcm_00411, SWS_Dcm_00414, SWS_Dcm_00417, SWS_Dcm_00421
BSW04003	-	SWS_Dcm_00030
BSW04005	-	SWS_Dcm_00020, SWS_Dcm_00252
BSW04006	-	SWS_Dcm_00022, SWS_Dcm_00250
BSW04011	-	SWS_Dcm_00338, SWS_Dcm_00339, SWS_Dcm_00340
BSW04015	-	SWS_Dcm_00027, SWS_Dcm_00143, SWS_Dcm_00144
BSW04016	-	SWS_Dcm_00024
BSW04017	-	SWS_Dcm_00028, SWS_Dcm_00038
BSW04019	-	SWS_Dcm_00547
BSW04020	-	SWS_Dcm_00001, SWS_Dcm_00200
BSW04021	-	SWS_Dcm_00015
BSW04033	-	SWS_Dcm_00496, SWS_Dcm_00499, SWS_Dcm_00502, SWS_Dcm_00503, SWS_Dcm_00504, SWS_Dcm_00505
BSW04058	-	SWS_Dcm_00004, SWS_Dcm_00005, SWS_Dcm_00077, SWS_Dcm_00279, SWS_Dcm_00295, SWS_Dcm_00296, SWS_Dcm_00378, SWS_Dcm_00382, SWS_Dcm_00383, SWS_Dcm_00384, SWS_Dcm_00385, SWS_Dcm_00388, SWS_Dcm_00389, SWS_Dcm_00393, SWS_Dcm_00465, SWS_Dcm_00475, SWS_Dcm_00476
BSW04065	-	SWS_Dcm_00004, SWS_Dcm_00005
BSW04067	-	SWS_Dcm_00378
BSW04079	-	SWS_Dcm_00441
BSW04082	-	SWS_Dcm_00243, SWS_Dcm_00244, SWS_Dcm_00245, SWS_Dcm_00246, SWS_Dcm_00410, SWS_Dcm_00411, SWS_Dcm_00414, SWS_Dcm_00417, SWS_Dcm_00421
BSW04098	-	SWS_Dcm_00531, SWS_Dcm_00532, SWS_Dcm_00535, SWS_Dcm_00536, SWS_Dcm_00592, SWS_Dcm_00767
BSW101	-	SWS_Dcm_00033, SWS_Dcm_00034, SWS_Dcm_00035, SWS_Dcm_00036, SWS_Dcm_00037
BSW159	-	SWS_Dcm_00999
BSW161	-	SWS_Dcm_00999
BSW162	-	SWS_Dcm_00999
BSW164	-	SWS_Dcm_00999
BSW168	-	SWS_Dcm_00999
BSW170	-	SWS_Dcm_00999

BSW172	-	SWS_Dcm_00999
SRS_Diag_04010	The DEM module and DCM module shall ensure interaction in order to fulfill ISO 14229-1 and ISO 15031-5	SWS_Dcm_00004, SWS_Dcm_00005, SWS_Dcm_00007, SWS_Dcm_00247, SWS_Dcm_00248, SWS_Dcm_00279, SWS_Dcm_00284, SWS_Dcm_00286, SWS_Dcm_00289, SWS_Dcm_00295, SWS_Dcm_00296, SWS_Dcm_00298, SWS_Dcm_00299, SWS_Dcm_00330, SWS_Dcm_00376, SWS_Dcm_00378, SWS_Dcm_00380, SWS_Dcm_00381, SWS_Dcm_00382, SWS_Dcm_00383, SWS_Dcm_00384, SWS_Dcm_00385, SWS_Dcm_00388, SWS_Dcm_00389, SWS_Dcm_00393, SWS_Dcm_00406, SWS_Dcm_00412, SWS_Dcm_00413, SWS_Dcm_00441, SWS_Dcm_00464, SWS_Dcm_00465, SWS_Dcm_00466, SWS_Dcm_00475, SWS_Dcm_00476, SWS_Dcm_00478, SWS_Dcm_00519, SWS_Dcm_01063, SWS_Dcm_01064

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	SWS_Dcm_00054
[BSW00419] Separate C-Files for pre-compile time configuration parameters	SWS_Dcm_00054
[BSW00381] Separate configuration header file for pre-compile time parameters	SWS_Dcm_00055
[BSW00412] Separate H-File for configuration parameters	SWS_Dcm_00055
[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	SWS_Dcm_00171, SWS_Dcm_00172, SWS_Dcm_00173
[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
[BSW00438] Post Build Configuration Data Structure	SWS_Dcm_00037
[BSW00402] Published information	Fulfilled by chapter 10
Wake-Up	--
[BSW00375] Notification of wake-up reason	Not applicable
Initialization	--
[BSW101] Initialization interface	SWS_Dcm_00033, SWS_Dcm_00034; SWS_Dcm_00035, SWS_Dcm_00036, SWS_Dcm_00037
[BSW00416] Sequence of Initialization	Not applicable
[BSW00406] Check module initialization	Not applicable
[BSW00437] NoInit–Area in RAM	Not applicable
Normal Operation	--
[BSW168] Diagnostic Interface of SW components	Not applicable
[BSW00407] Function to read out published parameters	SWS_Dcm_00065
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable
[BSW00424] BSW main processing function task allocation	SWS_Dcm_00053
[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
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable
[BSW00433] Calling of main processing functions	Not applicable
[BSW00450] Main Function Processing for Un-Initialized Module	SWS_Dcm_00593
[BSW00442] Debugging Support in Modules	SWS_Dcm_00043, SWS_Dcm_00484, SWS_Dcm_00485, SWS_Dcm_00486, SWS_Dcm_00487, SWS_Dcm_00506, SWS_Dcm_00507, SWS_Dcm_00508, SWS_Dcm_00509
Shutdown Operation	--

[BSW00336] Shutdown interface	Not applicable
Fault Operation and Error Detection	--
[BSW00337] Classification of errors	SWS_Dcm_00012, SWS_Dcm_00041
[BSW00338] Detection and Reporting of development errors	SWS_Dcm_00040, SWS_Dcm_00042
[BSW00369] Do not return development error codes via API	SWS_Dcm_00044
[BSW00339] Reporting of production relevant error status	Not applicable
[BSW00422] Debouncing of production relevant error status	Not applicable
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable
[BSW00323] API parameter checking	SWS_Dcm_00043
[BSW004] Version check	SWS_Dcm_00367
[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
[BSW00455] Implementation Conformance Class 1 and 2 (ICC1 and ICC2) Guidelines	Not applicable
Non-functional Requirements	--
Software Architecture Requirements	--
[BSW161] Microcontroller abstraction	Not applicable
[BSW162] ECU layout abstraction	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
[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
[BSW00453] Harmonization of BSW Modules	Not applicable
Software Module Design Requirements	--
Software quality	--
[BSW007] HIS MISRA C	Implementation specific

Naming conventions

Requirement	***Satisfied by***
[BSW00300] Module naming convention	Fulfilled by API definitions in chapter 8
[BSW00413] Accessing instances of BSW modules	Not applicable. (Only 1 instance of Dcm allowed)
[BSW00347] Naming separation of different instances of BSW drivers	Not applicable. (For driver only.)
[BSW00441] Enumeration literals and #define naming convention	Fulfilled by configuration chapter 8.2
[BSW00305] Self-defined data types naming convention	Fulfilled by type definitions in chapter 8
[BSW00307] Global variables naming convention	Not applicable (no global variables are specified for this module)
[BSW00310] API naming convention	SWS_Dcm_00548
[BSW00373] Main processing function naming convention	SWS_Dcm_00053

[BSW00327] Error values naming convention	SWS_Dcm_00364
[BSW00335] Status values naming convention	Fulfilled by API definitions in chapter 8
[BSW00350] Development error detection keyword	SWS_Dcm_00042
[BSW00408] Configuration parameter naming convention	Fulfilled by configuration in chapter 10
[BSW00410] Compiler switches shall have defined values	Fulfilled by configuration in chapter 10
[BSW00411] Get version info keyword	Fulfilled by configuration in chapter 10

Module file structure

Requirement	***Satisfied by***
[BSW00346] Basic set of module files	SWS_Dcm_00054
[BSW158] Separation of configuration from implementation	SWS_Dcm_00054
[BSW00314] Separation of interrupt frames and service routines	Not applicable (this module does not provide any ISRs)
[BSW00370] Separation of callback interface from API	SWS_Dcm_00054
[BSW00435] Header File Structure for the Basic Software Scheduler	SWS_Dcm_00055
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	SWS_Dcm_00055
[BSW00447] Standardizing Include file structure of BSW Modules Implementing Autosar Service	Not applicable. Implementation requirement

Standard header files

Requirement	***Satisfied by***
[BSW00348] Standard type header	See Section 8.1
[BSW00353] Platform specific type header	SWS_Dcm_00549
[BSW00361] Compiler specific language extension header	Not applicable (requirement on implementation, not on specification)

Module Design

Requirement	***Satisfied by***
[BSW00301] Limit imported information	SWS_Dcm_00054
[BSW00302] Limit exported information	SWS_Dcm_00055
[BSW00328] Avoid duplication of code	Not applicable (requirement on implementation, not on specification)
[BSW00312] Shared code shall be reentrant	Fulfilled by API definitions in chapter 8
[BSW006] Platform independency	SWS_Dcm_00555
[BSW00439] Declaration of interrupt handlers and ISRs	Not applicable
[BSW00448] Module SWS shall not contain requirements from Other Modules	Fulfilled by the whole document
[BSW00449] BSW Service APIs used by Autosar Application Software shall return a Std_ReturnType	Fulfilled by chapter 8.8.3 1 Configurable Interfaces

Types and keywords

Requirement	Satisfied by
[BSW00357] Standard API return type	Fulfilled by API definitions in chapter 8
[BSW00377] Module Specific API return type	Fulfilled by API definitions in chapter 8
[BSW00304] AUTOSAR integer data types	Fulfilled by API definitions in chapter 8
[BSW00355] Do not redefine AUTOSAR integer data types	Fulfilled by API definitions in chapter 8
[BSW00378] AUTOSAR Boolean type	Not applicable (Not used)
[BSW00306] Avoid direct use of compiler and platform specific keywords	SWS_Dcm_00551

Global data

Requirement	Satisfied by
[BSW00308] Definition of global data	SWS_Dcm_00554
[BSW00309] Global data with read-only constraint	SWS_Dcm_00552

Interface and API

Requirement	Satisfied by
[BSW00371] Do not pass function pointers via API	Fulfilled by API definitions in chapter 8
[BSW00358] Return type of init() functions	SWS_Dcm_00037
[BSW00414] Parameter of init function	SWS_Dcm_00037
[BSW00376] Return type and parameters of main processing functions	SWS_Dcm_00053
[BSW00359] Return type of callback functions	Fulfilled by API definitions in chapter 8
[BSW00360] Parameters of callback functions	Fulfilled by API definitions in chapter 8
[BSW00440] Function prototype for callback functions of AUTOSAR Services	Not applicable (requirement on implementation, not on specification)
[BSW00329] Avoidance of generic interfaces	Fulfilled by API definitions in chapter 8
[BSW00330] Usage of macros instead of functions	SWS_Dcm_00553
[BSW00331] Separation of error and status values	SWS_Dcm_00364

Safety Related Requirements

[BSW00443] Enabling / disabling defensive behavior of BSW	Not applicable
[BSW00444] Error reporting and logging for defensive behavior of BSW	Not applicable
[BSW00445] Protection against untimely call of BSW initialization	Not applicable
[BSW00446] Protection against untimely call of BSW de-initialization	Not applicable

Software Documentation Requirements

Requirement	*Satisfied by*
[BSW009] Module User Documentation	Fulfilled by the whole document
[BSW00401] Documentation of multiple instances of configuration parameters	Fulfilled by configuration chapter 0
[BSW172] Compatibility and documentation of scheduling strategy	Not applicable. (There is no scheduler in the DCM)
BSW010] Memory resource documentation	Not applicable. (requirement on implementation, not on specification)
[BSW00333] Documentation of callback function context	Fulfilled by API definitions in chapter 8
[BSW00374] Module vendor identification	SWS_Dcm_00335
[BSW00379] Module identification	SWS_Dcm_00335
[BSW003] Version identification	SWS_Dcm_00335
[BSW00318] Format of module version	SWS_Dcm_00335
[BSW00321] Enumeration of module version numbers	Not applicable. (requirement on implementation, not on specification)
[BSW00341] Microcontroller compatibility documentation	Not applicable. (requirement on implementation, not on specification)
[BSW00334] Provision of XML file	Not applicable. (requirement on implementation, not on specification)

Document: AUTOSAR requirements on Basic Software, cluster Diagnostic

Requirements on Basic Software Module Diagnostic	
Requirement	Satisfied by
General	--
[SRS_Diag_04010] Interface between Diagnostic service handling and Diagnostic event (error) management	SWS_Dcm_00007, SWS_Dcm_00247, SWS_Dcm_00005, SWS_Dcm_00248, SWS_Dcm_00376, SWS_Dcm_00293, SWS_Dcm_00378, SWS_Dcm_00380, SWS_Dcm_00381, SWS_Dcm_00295, SWS_Dcm_00296, SWS_Dcm_00478, SWS_Dcm_00475, SWS_Dcm_00476, SWS_Dcm_00479, SWS_Dcm_00382, SWS_Dcm_00480, SWS_Dcm_00298, SWS_Dcm_00299, SWS_Dcm_00383, SWS_Dcm_00384, SWS_Dcm_00385, SWS_Dcm_00441, SWS_Dcm_00429, SWS_Dcm_00430 SWS_Dcm_00431 SWS_Dcm_00388,

	SWS_Dcm_00389, , SWS_Dcm_00393, SWS_Dcm_00466, SWS_Dcm_00464, SWS_Dcm_00465, SWS_Dcm_00519, SWS_Dcm_00304, SWS_Dcm_00406, SWS_Dcm_00279, SWS_Dcm_00280, SWS_Dcm_00284, SWS_Dcm_00278, SWS_Dcm_00286, SWS_Dcm_00289, SWS_Dcm_00412, SWS_Dcm_00330, SWS_Dcm_00004, SWS_Dcm_00413
[BSW04082] Support of ISO15031-5 and SAE J1979	SWS_Dcm_00243, SWS_Dcm_00244, SWS_Dcm_00245, SWS_Dcm_00410, SWS_Dcm_00411, SWS_Dcm_00246, SWS_Dcm_00414, SWS_Dcm_00417, SWS_Dcm_00421
Diagnostic communication management (DCM)	--
[BSW04007] Provide Diagnostic service handling	Fulfilled by Section 7.4
[BSW04021] Switch diagnostic communication access	SWS_Dcm_00015
[BSW04032] Support of different diagnostic addresses	SWS_Dcm_00770_Conf, SWS_Dcm_00772_Conf
[BSW04058] Access to different event (fault) memories	SWS_Dcm_00077, SWS_Dcm_00005, SWS_Dcm_00293, SWS_Dcm_00378, SWS_Dcm_00295, SWS_Dcm_00296, SWS_Dcm_00475, SWS_Dcm_00476, SWS_Dcm_00382, SWS_Dcm_00383, SWS_Dcm_00384, SWS_Dcm_00385, SWS_Dcm_00429, SWS_Dcm_00388, SWS_Dcm_00389, SWS_Dcm_00390, SWS_Dcm_00393, SWS_Dcm_00465, SWS_Dcm_00279, SWS_Dcm_00004

[BSW04065] Clearing of events or event groups	SWS_Dcm_00005, SWS_Dcm_00004
[BSW04067] Counting and evaluation of events according to ISO 14229-1 DTCStatusMask	SWS_Dcm_00293, SWS_Dcm_00378
[BSW04097] Decentralized modular diagnostic configuration of SW-Cs	Fulfilled by chapter 8.7.3 Configurable Interfaces
[BSW04000] Support Diagnostic Standard UDS (ISO14229-1)	Fulfilled by the whole document
[BSW04001] Support Diagnostic Standard OBD (ISO15031-5)	SWS_Dcm_00243, SWS_Dcm_00244, SWS_Dcm_00245, SWS_Dcm_00410, SWS_Dcm_00411, SWS_Dcm_00246, SWS_Dcm_00414, SWS_Dcm_00417, SWS_Dcm_00421
[BSW04005] SecurityAccess level handling is managed by DCM	SWS_Dcm_00252, SWS_Dcm_00020
[BSW04006] Session handling is managed by DCM	SWS_Dcm_00022, SWS_Dcm_00250
[BSW04016] Provision of Busy Handling	SWS_Dcm_00024
[BSW04019] Application callback after transmit confirmation	SWS_Dcm_00547
[BSW04020] Suppression of Responses	SWS_Dcm_00001, SWS_Dcm_00200
[BSW04033] Upload/Download services for data handling	SWS_Dcm_00496, SWS_Dcm_00499, , SWS_Dcm_00502, SWS_Dcm_00503, SWS_Dcm_00504, SWS_Dcm_00505
[BSW04036] Format checking of diagnostic services	SWS_Dcm_00272_Conf, SWS_Dcm_00273_Conf, SWS_Dcm_00071_Conf, SWS_Dcm_00074_Conf
BSW04098 Standard bootloader interaction	SWS_Dcm_00531, SWS_Dcm_00532, SWS_Dcm_00533, SWS_Dcm_00535, SWS_Dcm_00538, SWS_Dcm_00536, SWS_Dcm_00537 , SWS_Dcm_00592
Timing Requirements	--
[BSW04015] Provision of timing handling according to ISO15765-3	SWS_Dcm_00027, SWS_Dcm_00143, SWS_Dcm_00144, SWS_Dcm_00311,

	SWS_Dcm_00750_Conf
Resource Usage	--
[BSW04017] Provide optimized buffer handling	SWS_Dcm_00028, SWS_Dcm_00038, SWS_Dcm_00775_Conf
Interface and API	--
[BSW04078] Interface to fault memory, fault status	Chapter 8
[BSW04011] Provide diagnostic state information	SWS_Dcm_00338, SWS_Dcm_00339, SWS_Dcm_00340
[BSW04003] Interface to PDU Router shall be network independent	SWS_Dcm_00030
[BSW04079] The size of a FreezeFrame shall be reported to the DCM by the DEM	SWS_Dcm_00441
Configuration	--
[BSW04059] Configuration of timing parameter	SWS_Dcm_00031_Conf
[BSW04024] Configurable size of transferred data	SWS_Dcm_00739_Conf

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.

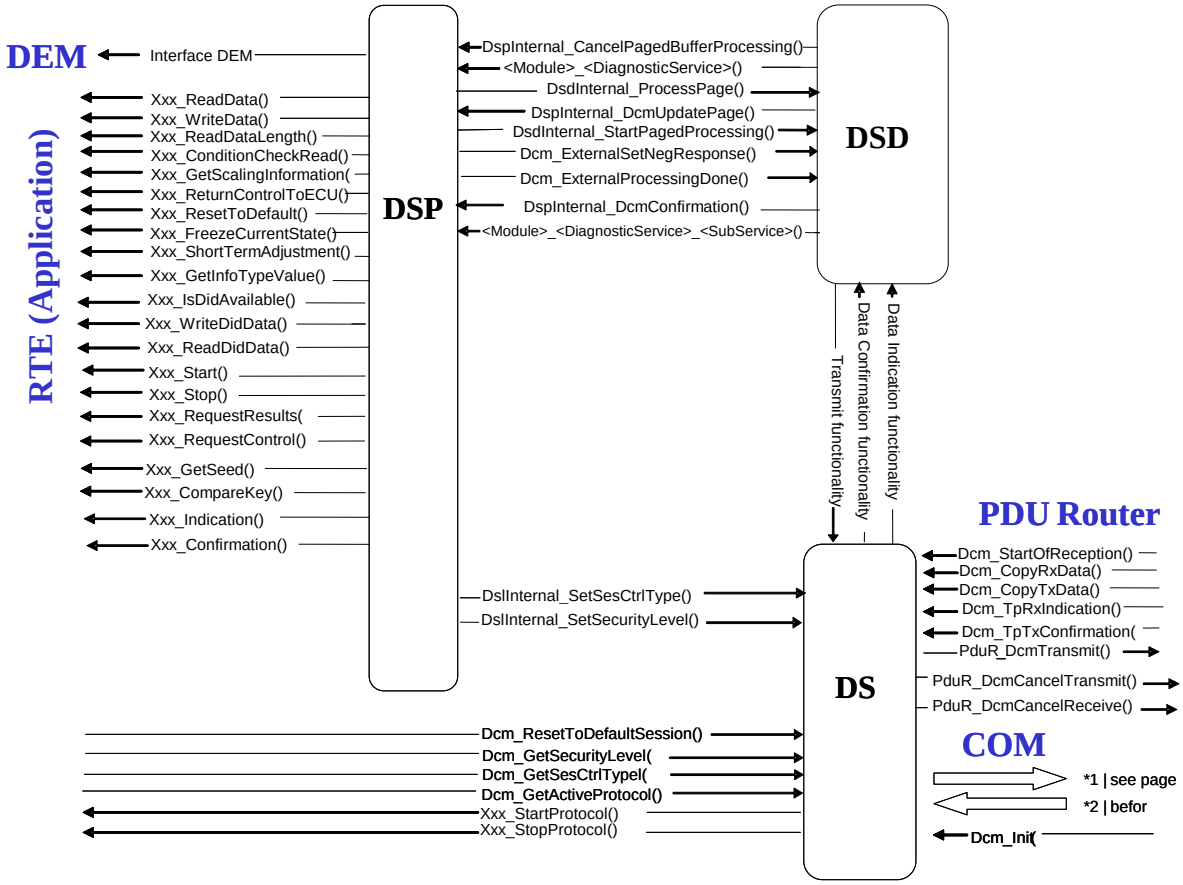


Figure 5 Possible interaction between the submodules in the DCM

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 negative response codes (NRCs).
The DCM SWS uses these NRCs in the interfaces between the DCM and other BSW modules and the SW-Cs. These NRCs are defined in the data type `Dcm_NegativeResponseCodeType`.

7.1.3 Non-volatile information

Several features of the Dcm require non-volatile information to be initialized. AUTOSAR does not describe how this information is accessed or if the information is already available when the Dcm is initialized. Therefore the access for the non-volatile information is implementation specific and has to be ensured during integration.

[SWS_Dcm_00870] ⌈The Dcm shall check if the NvM is read out correctly. If the non-volatile information could not read out correct the Dcm shall start a default reaction. The default reaction is described in the chapter were the usage of the non-volatile data is described.⌋()

[SWS_Dcm_01048] ⌈If the Dcm cancels a service with NvM access, it shall call NvM_CancelJobs().⌋()

The service is cancelled either by reaching the maximum number of RCRRP NRCs or by protocol preemption.

7.1.4 Data types

[SWS_Dcm_00968] ⌈The Dcm shall support the following data types:

- boolean
- uint8
- uint16
- uint32
- sint8
- sint16
- sint32
- uint8[n]

The type uint8[n] is mapped to either for fixed or variable data length.⌋()

[SWS_Dcm_00969] ⌈The Dcm shall treat non-integer data types (e.g. uint8[n]) either like integer data types of the matching size or leave their contents uninterpreted in case ***DcmDspDataEndianness*** is configured to OPAQUE.⌋()

[SWS_Dcm_00970] ⌈The Dcm module shall interpret opaque data as uint8[n] and shall always map it to an n-bytes sized signal.⌋()

For opaque data endianness, ***DcmDspDataEndianness*** has to be configured to OPAQUE.

[SWS_Dcm_00971] ⌈The Dcm shall extend the endianness conversion defined in [21] (Chapter 2.4), to signed data types.⌋()

In [21] (Chapter 2.4) the endianness conversion is defined for unsigned data types. The associated configurations can be found in the configuration 10.2.29 DcmDspData.

[constr_6002] Define the usage of DcmDspDataSize parameter「***DcmDspDataSize*** is always required, except ***DcmDspDataUsePort*** is set to {USE_DATA_SENDER_RECEIVER, USE_ECU_SIGNAL} and ***DcmDspDataType*** is set to {BOOLEAN, SINT8, SINT16, UINT32, SINT32}.」()

[constr_6003] Restrictions on bit-wise access「***DcmDspDataSize*** shall be a multiple of 8 if the value is greater than 15.」()

[constr_6004] UINT8 shall be used as (implementation) data type for bit lengths between 2 and 8「If ***DcmDspDataUsePort*** is of type USE_DATA_SENDER_RECEIVER or USE_ECU_SIGNAL and ***DcmDspDataSize*** is greater than 1 and less or equal 8, the ***DcmDspDataType*** shall use UINT8.」()

[constr_6005] UINT16 shall be used as (implementation) data type for bit lengths between 9 and 16「If ***DcmDspDataUsePort*** is of type USE_DATA_SENDER_RECEIVER or USE_ECU_SIGNAL and ***DcmDspDataSize*** is between 9 and 16 the ***DcmDspDataType*** shall use UINT16.」()

[constr_6006] Restrictions on bit-wise access「***DcmDspDataSize*** shall be a multiple of 8 and ***DcmDspDataUsePort*** is of USE_BLOCK_ID, USE_DATA_SYNCH_CLIENT_SERVER, USE_DATA_ASYNCH_CLIENT_SERVER, USE_DATA_SYNCH_FNC, USE_DATA_ASYNCH_FNC is used.」()

[constr_6007] Restrictions on bit-wise placement「***DcmDspDidDataPos*** Parameter shall address always a byte boundary, except ***DcmDspDataType*** is set to BOOLEAN, UINT8 or UINT16 with ***DcmDspDataSize*** lower than 16.」()

[constr_6008] Define the usage of DcmDspRoutineSignalLength parameter「***DcmDspRoutineSignalLength*** is only required if ***DcmDspRoutineFixedLength*** is set to false.」()

[constr_6009] Restrictions on bit-wise placement「***DcmDspRoutineSignalPos*** parameter shall address always a byte boundary, except ***DcmDspRoutineSignalType*** is set to BOOLEAN or UINT8.」()

[constr_6010] Restrictions on bit-wise access ⌜***DcmDspRoutineSignalLength*** shall not exceed the value of 8 in case of ***DcmDspRoutineSignalType*** set to UINT8.⌟()

[constr_6011] Only last parameters in RID may have a variable length ⌜***DcmDspRoutineSignalType*** with VARIABLE_LENGTH is only valid for the last signal and when ***DcmDspRoutineFixedLength*** is set to FALSE.⌟()

[constr_6012] Define the usage of DcmDspPidDataSize parameter ⌜***DcmDspPidDataSize*** is always required, except ***DcmDspPidDataUsePort*** is of type USE_DATA_SENDER_RECEIVER and ***DcmDspPidDataType*** is set to {BOOLEAN, SINT8,SINT16, UINT32, SINT32}.⌟()

[constr_6013] Restrictions on bit-wise access ⌜***DcmDspPidDataSize*** shall be a multiple of 8 if the value is greater than 15.⌟()

[constr_6014] UINT8 shall be used as (implementation) data type for bit lengths between 2 and 8 ⌜If ***DcmDspPidDataUsePort*** is of type USE_DATA_SENDER_RECEIVER and ***DcmDspPidDataSize*** is between 1 and 8 the ***DcmDspPidDataType*** shall use UINT8.⌟()

[constr_6015] UINT16 shall be used as (implementation) data type for bit lengths between 9 and 16 ⌜If ***DcmDspPidDataUsePort*** is of type USE_DATA_SENDER_RECEIVER and ***DcmDspPidDataSize*** is between 9 and 16 the ***DcmDspPidDataType*** shall use UINT16.⌟()

[constr_6016] Restrictions on bit-wise access ⌜***DcmDspPidDataSize*** shall be a multiple of 8 and ***DcmDspPidDataUsePort*** is of USE_DATA_SYNCH_CLIENT_SERVER, USE_DATA_SYNCH_FNC is used.⌟()

[constr_6017] Restrictions on bit-wise placement ⌜***DcmDspPidDataPos*** Parameter shall address always a byte boundary, except ***DcmDspPidDataType*** is set to BOOLEAN, UINT8 or UINT16 with ***DcmDspPidDataSize*** lower than 16.⌟()

[constr_6024] UINT8 shall be used as (implementation) data type for Client-Server interface「 In case ***DcmDspDataUsePort*** parameter is set to {USE_DATA_SYNCH_CLIENT_SERVER,USE_DATA_ASYNCH_CLIENT_SERVER }, ***DcmDspDataType*** shall use UINT8.」()

7.2 Diagnostic Session Layer (DSL)

7.2.1 Introduction

[SWS_Dcm_00030]「 All functional areas of the DSL submodule shall be in conformance with the specifications ISO14229-1 [15] and the network-independent part of ISO15765-3 [18].」(BSW04003)

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 [15] and ISO 15765-3 [18]),
- Application layer timing handling (as required by ISO14229-1 [15] and ISO 15765-3 [18]),
- Specific response behavior (as required by ISO14229-1 [15] and ISO 15765-3 [18]).

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.
- 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_StartOfReception()`, which inform the DCM module of the data size to be received and provides the data of the first frame or single frame, and allows the DCM to reject the reception if the data size overflows its buffer size, or if the requested service is not available. The further call to Dcm_CopyRxData request the DCM module to copy the data from the provided buffer to the DCM buffer.
If the reception of a diagnostic request is finished (when Dcm_StartOfReception succeeded) the PduR module will call `Dcm_TpRxIndication()` to give a receive indication to the DCM module.
The DCM shall be able to use generic connections, where the addressing information is provided to DCM by `Dcm_StartOfReception()` via the MetaData of the DcmRxPdu. This addressing information must be stored and used for the response and for detection of requests from the same tester. See section 7.2.4.5 Generic Connection Handling for further details.

[SWS_Dcm_00111] ⌈ The DSL submodule shall forward received data to the DSD submodule only after a call of `Dcm_TpRxIndication()` with parameter Result = E_OK (see SWS_Dcm_00093).⌋()

[SWS_Dcm_00241] ⌈ As soon as a request message is received (after a call of `Dcm_TpRxIndication()` with parameter Result = E_OK (see SWS_Dcm_00093) and until a call to `Dcm_TpTxConfirmation()` (see SWS_Dcm_00351) for the associated Tx-DcmPduId), the DSL submodule shall block the corresponding DcmPduId. During the processing of this request, no other request of the same DcmDslConnection (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 (except for concurrent TesterPresent requests). ⌋()

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,
- 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 [15]. 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:

[SWS_Dcm_00112] ⌈ When the PduR module calls `Dcm_TpRxIndication()` with parameter Result=E_OK (see SWS_Dcm_00093) 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). ⌋()

[SWS_Dcm_00113] ⌈ When the PduR module calls `Dcm_TpRxIndication()` with parameter Result = E_OK (see SWS_Dcm_00093) 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 SWS_Dcm_00113: 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

[SWS_Dcm_00114] ⌈ The DSD submodule shall request the DSL submodule for transmission of responses.⌋()

[SWS_Dcm_00115] ⌈ When the diagnostic response is ready 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 and, for generic connections, the addressing information, is given to the PduR module. After the DCM module has called successfully `PduR_DcmTransmit()`, the PduR module will call `Dcm_CopyTxData()` to request the DCM module to provide the data to be transmitted and will call `Dcm_TpTxConfirmation()` after the complete PDU has successfully been transmitted or an error occurred.
See section 7.2.4.5 Generic Connection Handling for further details on address information handling within generic connections."

[SWS_Dcm_00117] ⌈ 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_TpTxConfirmation()`, then the DSL submodule shall forward this confirmation to the DSD submodule.⌋()

[SWS_Dcm_00118] ⌈ In case of a failed transmission (failed `PduR_DcmTransmit()` request) or error confirmation (`Dcm_TpTxConfirmation()` with error), the DSD submodule shall not repeat the diagnostic response transmission.⌋()

Note: Dcm_TpTxConfirmation is only expected when PduR_DcmTransmit succeeded.

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 Generic Connection Handling

The DCM shall be able to handle generic connections, identified by DcmPdus with a MetaDataLength >= 1. These connections carry the actual tester address at run time. Generic connections are supported for CAN diagnostics using normal fixed or mixed 29 bit addressing formats according to ISO15765-2 [17]. Depending on the

actual layout of the CAN IDs, generic connections could also be used for extended or normal and mixed 11 bit addressing formats. The DCM is not aware of the actual addressing format used by CanTp.
The configuration parameter ***DcmDslProtocolRxTesterSourceAddr*** is not needed for generic connections.
Several connections may reference the same DcmPdus.

[**SWS_Dcm_00845**] ⌈ The configuration tool shall verify that generic connections are consistent by checking the MetaDataLength of all referenced PDUs of a DcmDslConnection (DcmDslProtocolRxPduRef, DcmDslProtocolTxPduRef, DcmDslPeriodicTxPduRef, DcmDslRoeTxPduRef). The TxPdu references may only be generic when DcmDslProtocolRxPduRef is also generic.⌋()

[**SWS_Dcm_00848**] ⌈ The source address of diagnostic requests received via a generic connection must be stored. It is provided in the first byte of the MetaData provided via Dcm_StartOfReception().⌋()

[**SWS_Dcm_00849**] ⌈ The stored source address shall be used as target address of responses transmitted via a generic connection. It shall be provided in the second byte of the MetaData provided to PduR_DcmTransmit().⌋()

7.2.4.6 Guarantee timing to tester by sending busy responses

[SWS_Dcm_00024] ⌈ 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) when reaching the response time (DcmDspSessionP2ServerMax -DcmTimStrP2ServerAdjust respectively DcmDspSessionP2StarServerMax -DcmTimStrP2StarServerAdjust)⌋(BSW04016)

Rationale for SWS_Dcm_00024: The DSL submodule guarantees the response timing to tester.

[SWS_Dcm_00119] ⌈ The DSL submodule shall send negative responses as required in SWS_Dcm_00024 from a separate buffer.⌋()

Rationale for SWS_Dcm_00119: 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.

[SWS_Dcm_00120] ⌈ If the number of negative responses for a requested diagnostic tasks (see SWS_Dcm_00024) reaches the value defined in the configuration parameter ***DcmDslDiagRespMaxNumRespPend***, the DCM module shall stop processing the active diagnostic request, inform the application or BSW (if this diagnostic task implies the call to a SW-C interface or a BSW interface) by setting OpStatus parameter, of active port interface, to DCM_CANCEL and shall send a negative response with NRC 0x10 (General reject).⌋()

7.2.4.7 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.

[SWS_Dcm_00121] ⌈ If a pending message for normal diagnostic responses (higher priority) exists, then the DSL submodule shall wait for the transmission confirmation (Call to `Dcm_TpTxConfirmation()`) 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***

[SWS_Dcm_00122] ⌈ The DCM module shall send responses for periodic transmissions using a separate protocol and a separate buffer of configurable size.⌋()

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

[SWS_Dcm_00123] ⌈ 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:

[SWS_Dcm_00125] ⌈ The DCM module shall discard periodic transmission events beside Full Communication Mode and shall not queue it for transmission.⌋()

[SWS_Dcm_00126]「 Periodic transmission events shall not activate the Full Communication Mode.」()

[SWS_Dcm_01044]「Only TYPE2 messages will support parallel execution of Diagnosis response.」()

7.2.4.8 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 (e.g: UDS Service ReadDataByIdentifier (0x22)).

[SWS_Dcm_00595]「 The ROE functionality is enabled only if the container DcmDslResponseOnEvent exists 」()

7.2.4.8.1 ResponseOnEvent StateChart

[SWS_Dcm_00871]「 The Dcm shall support several RoeEvents. Each RoeEvent can have the states “ROE cleared”, “ROE stopped” and “ROE started’’. The transitions from state to state are described in the following section. The Labels in Figure 6 represents the numbers of the sections.」()

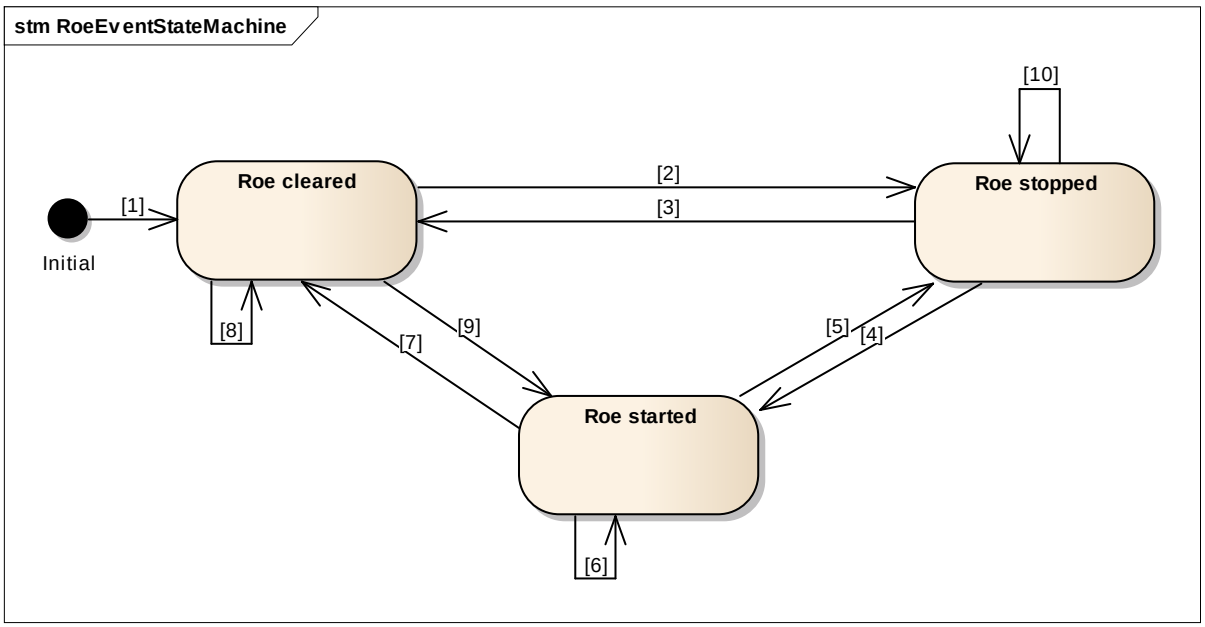


Figure 6 RoeEvent State Chart

1.1.1.1.1.1 Initializing Dcm (1)

[SWS_Dcm_00872] ⌈The Dcm changes the state of each event to ‘ROE cleared’ state during Dcm_Init().⌋()

1.1.1.1.1.2 Transition from ‚ROE cleared‘ to ‘ROE stopped‘ (2)

[SWS_Dcm_00873] ⌈By receiving a valid ROE setup request, the RoeEvent which is addressed in the request changes to the ‘ROE stopped’ state (See Table 2).⌋()

[SWS_Dcm_00874] ⌈If the RoeEvent was setup with the StorageState set to ‘storeEvent’ and no StartResponseOnEvent with StorageState set to ‘storeEvent’ and an EventWindowTime which is active over power cycles or clearResponseOnEvent has been received afterwards the Dcm will change to ‘ROE stopped’ state as soon as the non-volatile information is available.⌋()

Note: If an Event is initialized once with StorageState set to ‘StoreEvent’, it will stay initialized until it is cleared by a ClearResponseOnEvent request (See also **SWS_Dcm_00897**).

[SWS_Dcm_00951] ⌈If for a RoeEvent the configuration parameter DcmDspRoeInitialEventStatus is set to DCM_ROE_STOPPED, the Dcm will switch to ‘ROE stopped’ state immediatly in the initialisation.⌋()

Note: DcmDspRoeInitialEventStatus set defines an initialisation of a RoeEvent by configuration.

1.1.1.1.1.3 Transition from ‚ROE stopped‘ to ‚ROE cleared' (3)

[SWS_Dcm_00875] ⌈By receiving a valid ROE request with the sub-function clearResponseOnEvent (0x06) the RoeEvents change to the ‘ROE cleared’ state.⌋()

1.1.1.1.1.4 Transition from ‚ROE stopped‘ to ‚ROE started’ (4)

[SWS_Dcm_00876] ⌈By receiving a valid ROE request with the sub-function startResponseOnEvent (0x05) all stopped RoeEvents change to the ‘ROE started’ state.⌋()

[SWS_Dcm_00902] ⌈All RoeEvents which have been in ‘ROE started’ state when leaving the default session shall change back into ‘ROE started’ state when (re-) entering the default session.⌋()

[SWS_Dcm_00965] ⌈If a valid StartResponseOnEvent request is received with a storageState set to StoreEvent and the EventWindowTime supports the StorageState in a previous power cycle, the RoeEvent shall change from ‘ROE stopped’ state to ‘ROE started’ state as soon as the non-volatile data is available. (This ROEEvent was set to ‘ROE stopped’ according to **SWS_Dcm_00951**).⌋()

1.1.1.1.1.5 Transition from ‚ROE started‘ to 'ROE stopped’ (5)

[SWS_Dcm_00877] ⌈By receiving a valid ROE request with the sub-function stopResponseOnEvent (0x00) the stopped RoeEvents change to the ‘ROE stopped’ state.⌋()

[SWS_Dcm_00878] ⌈When the eventWindowTime times out the stopped RoeEvents change to the ‘ROE stopped’ state.⌋()

[SWS_Dcm_00879] ⌈By leaving the current session all started RoeEvents shall change to the ‘ROE stopped’ state.⌋()

Note: RoeEvents are stopped when the current session is left, independent if the session changes from a non-default session to the same or a different non-default session. By leaving the default session the current active RoeEvents are stopped and stored (in order to be re-started as soon the session changes back to the default session (See **SWS_Dcm_00902**)

[SWS_Dcm_00952] ⌈If a ROE request is received with the sub-function OnDTCStatusChange and the RoeEvent is ‘ROE started’, the RoeEvent for OnDTCStatusChange changes to ‘ROE stopped’ state and the ServiceToRespondTo shall be triggered by the DTCStatusMask which is set by the new request.⌋()

1.1.1.1.1.6 Transition from ‚ROE started’ to ‚ROE started’ (6)

[SWS_Dcm_00880] ⌈By receiving a valid ROE request with the sub-function StartResponseOnEvent (0x05) the Dcm answers positively and stays in ‘ROE started’ state.).⌋()

1.1.1.1.1.7 Transition from ‚ROE started‘ to 'ROE cleared’ (7)

[SWS_Dcm_00884] ⌈By receiving a valid ROE request with the sub-function clearResponseOnEvent (0x06) all started RoeEvents change to the ‘ROE cleared’ state. ⌋()

1.1.1.1.1.8 Transition from ‚ROE cleared‘ to 'ROE cleared’ (8)

[SWS_Dcm_00885] ⌈If all RoeEvents are in ‘ROE cleared’ state and a valid stopResponseOnEvent (0x00) request is received the Dcm shall reject the request with a negative Response with NRC 0x24 (requestSequenceError). ⌋()

[SWS_Dcm_00886] ⌈If all RoeEvents are in ‘ROE cleared’ state and a valid StartResponseOnEvent (0x05) request is received the Dcm shall reject the request with a negative Response with NRC 0x24 (requestSequenceError). ⌋()

[SWS_Dcm_00887] ⌈If all RoeEvents are in ‘ROE cleared’ state and a valid clearResponseOnEvent (0x06) request is received the Dcm anwers positively and the RoeEvents stay in ‘ROEcleared’ state.).⌋()

[SWS_Dcm_00888] ⌈If the non-volatile data could not be read correctly, all RoeEvents in ‘ROE cleared’ state remain in ‘ROE cleared’ state.⌋()

1.1.1.1.1.9 Transition from ‚ROE cleared‘ to 'ROE started’ (9)

[SWS_Dcm_00889] ⌈If the EventWindowTime is active over power cycles and not timed out, the Dcm shall reactivate all RoeEvents which were active in the default session during the last power cycle as soon as the non-volatile information is available.⌋()

[SWS_Dcm_00890] ⌈If a valid StartResponseOnEvent request is received with a storageState set to StoreEvent and the EventWindowTime supports the StorageState in a previous power cycle, the RoeEvent shall change to ‘ROE started’ state as soon as the non-volatile data is available. ⌋()

1.1.1.1.1.10 Transition from ‘ROE stopped’ to ‘ROE stopped‘ (10)

[SWS_Dcm_00891] ⌈If a RoeEvent is in 'ROE stopped' state and a valid stopResponseOnEvent (0x00) request is received the Dcm shall respond positively to the request and stay in the 'ROE stopped' state.⌋()

[SWS_Dcm_00953] ⌈If a ROE request is received with the sub-function OnDTCStatusChange and the RoeEvent is already 'ROE stopped' the RoeEvent for OnDTCStatusChange shall stay in 'ROE stopped' state and the ServiceToRespondTo shall be triggered by the DTCStatusMask which is set by the new request.⌋()

7.2.4.8.2 ROE sub-functions

[SWS_Dcm_00892] ⌈The Dcm shall support all ROE sub-functions marked as supported in Table 2.⌋()

Sub function ID	Sub-function name	Kind of sub-function	ServiceToRespondTo	Support status
0x00	stopResponseOnEvent	Control		Supported
0x01	onDTCStatusChange	Setup	0x19, 0x0E	Supported
0x02	onTimerInterrupt	Setup		Not supported
0x03	onChangeOfDataIdentifier	Setup	0x22	Supported
0x04	reportActivatedEvents	Control		Supported
0x05	StartResponseOnEvent	Control		Supported
0x06	clearResponseOnEvent	Control		Supported
0x07	onComparisonOfValues	Setup		Not supported
Other	OEM Specific	Setup		Not supported

Table 2: Supported sub function of Response on Event (0x86)

[SWS_Dcm_00893] ⌈For each setup sub function the Dcm shall only support the one fixed ServiceToRespondTo. The supported ServiceToRespondTo is listed in Table 2⌋()

7.2.4.8.3 EventWindowTime and StorageState

The EventWindowTime and StorageState are mandatory parameter in every ROE request. They can be contradicting between the setup request and the related control request.

[SWS_Dcm_00903] ⌈The Dcm shall evaluate the EventWindowTime from the setup request.⌋()

[SWS_Dcm_00894] ⌈The Dcm shall support in general the EventWindowTimes defined in Table 3.⌋()

Value	Name	Active over PowerCycles
0x02	Infinity	Storage State
0x03	CurrentCycle	No
0x04	CurrentAndFollowingCycle	Yes

Table 3 Supported ROE EventWindowTime

[SWS_Dcm_00895] ⌈The configuration parameter DcmDspRoeEventWindowTime shall contain a list of all EventWindowTimes supported for this specific Ecu.⌋()

[SWS_Dcm_00896] ⌈If the Roe request contains a different EventWindowTime than configured in DcmDspRoeEventWindowTime the Dcm shall reject the request with a negative response with the NRC 0x31 (RequestOutOfRange) .⌋()

[SWS_Dcm_00897] ⌈If a RoeEvent is setup with StorageState set to ‘storeEvent’ the initialization shall be stored non-volatile to be restored in every following driving cycle until it is cleared (See **SWS_Dcm_00874**).⌋()

[SWS_Dcm_00898] ⌈A RoeEvent shall change to ‘ROE started’ state at the beginning of each following power cycle until a stopResponseOnEvent request with storage StorageState set to StoreEvent is received if the RoeEvent fulfills all following conditions :

- The RoeEvent was started in default session
- The StartResponseOnEventRequest has a storageState set to ‘StoreEvent’
- The setup request has the EventWindowTime inifinity and the storageState was set to ‘StoreEvent’. ⌋()

[SWS_Dcm_00905] ⌈The EventWindowTime will end at the end of the current power cycle if all of the following conditions are fulfilled:

- The EventWindowTime is set to infinity (0x02) during the setup request
- The RoeEvent was started in default-session
- The storageState was not set in the StartResponseOnEvent request」()

[SWS_Dcm_00900]「If ResponseOnEvent started in default session with the EventWindowTime CurrentAndFollowingCycle the EventWindowTime shall end at the end of the next power cycle or with a clearResponseOnEvent/stopResponseOnEvent request.」()

[SWS_Dcm_00901]「If ResponseOnEvent is started in default session with an EventWindowTime currentCycle the EventWindowTime will end at the end of this power cycle or with a clearResponseOnEvent/stopResponseOnEvent. 」()

[SWS_Dcm_00906]「 If ResponseOnEvent is started in a non-default session, the EventWindowTime ends if one of the following conditions is fulfilled:

- The power cycle ends
- Receiving a clearResponseOnEvent request
- Receiving a stopResponseOnEvent request
- With any session change.」()

[SWS_Dcm_00907]「If the EventWindowTime times out and the power cycle is not ended, the Dcm shall send a final positive Response to the setup request.」()

For the EventWindowTime infinity (0x02), ThisCycle (0x03), ThisAndNextCycle (0x04) the Dcm will not send a final response because these EventWindow Times will end at the end of an power cycle. There will also no final response if the session changes or the service is stopped with a “stopResponseOnEvent” subfunction.

7.2.4.8.4 Pre-configuration of ResponseOnEvent

[SWS_Dcm_00908]「The Dcm shall only support Roe requests which where pre-configured in the configuration.」()

Note: The pre-configuration gives the Dcm the freedom to optimized not configured requests.

[SWS_Dcm_00909] ⌈The Dcm supports the configuration container DcmDspRoe to configure all supported ResponseOnEvent setup requests.⌋()

[SWS_Dcm_00954] ⌈If DcmDspRoeInitialEventStatus is set to DCM_ROE_STOPPED the Dcm shall behave like this RoeEvent was set-up with StorageState set to 'StoreEvent' and EvetenWindowTime set to infinity.⌋()

According to **SWS_Dcm_00954** and **SWS_Dcm_00897**, the pre-configuration of RoeEvents shall behave the same like received a received setup and start request in previous driving cycles. If the storageState is set in the start/stop/clearedResponseOnEventRequest the pre configuration will be replaced with the newly received request.

7.2.4.8.5 Handling of event-trigger
1.1.1.1.1.11 ROE event-trigger onDTCStatusChange (0x01)

If a RoeEvent is in 'ROE started' state and it is configured to onDTCStatusChange (See container DcmDspRoeEvent), the Dcm triggers a ServiceToResponseTo as soon as the Dem is reporting a DTCStatusChange which fits to the requested DTCStatusMask. According to **SWS_Dcm_00909**, the Dcm only supports preconfigured ROE requests. Therefore the container DcmDspRoeOnDTCStatusChange needs to be configured if onDTCStatusChange shall be used.

[SWS_Dcm_00912] ⌈If the state of one RoeEvent that is configured for onDTCStatusChange changes to 'ROE started' the Dcm shall call Dem_DcmControlDTCStatusChangedNotification (TRUE) to trigger DTC status change reports to the Dcm.⌋()

[SWS_Dcm_00913] ⌈If the state of the RoeEvent, configured to OnDTCStatusChange, leaves 'ROE started' the Dcm shall call Dem_DcmControlDTCStatusChangedNotification (FALSE) to stop triggering DTC status change reports to the Dcm.⌋()

[SWS_Dcm_00914] ⌈If the state of the RoeEvent is 'ROE started' for the sub-function OnDTCStatusChange shall trigger a serviceToRespondTo if Dcm_DemTriggerOnDTCStatus() is called and the DTCStatusNew fits to the corresponding DTCStatusMask.⌋()

[SWS_Dcm_00915] ⌈If an event is trigger for onDTCStatusChange, the Dcm shall execute a serviceToResponseTo 0x19 0x0E, if the DTCStatusNew fits to the corresponding DTCStatusMask.⌋()

1.1.1.1.1.12 ROE event-trigger onChangeOfDataIdentifier (0x03)

If a RoeEvent is in 'ROE started' state and it is configured to onChangeOfDataIdentifier (See container DcmDspRoeEvent) , the Dcm triggers a ServiceToResponseTo as soon as a SWC or a CDD is reporting a change of the DID referenced by DcmDspRoeDidRef (SWC or CCD reports DID change by call of Dcm_TriggerOnEvent). According to **SWS_Dcm_00909**, the Dcm only supports preconfigured ROE requests. Therefore the Did in the ROE setup request with onChangeOfDataIdentifier has to be linked as DcmDspRoeDidRef in the onChangeOfDataIdentifier configuration.

[SWS_Dcm_00918] ⌈If a ResponseOnEvent is requested as onChangeOfDataIdentifier and the requested Did is not referred as DcmDspRoeDidRef for any DcmDspRoeEvent the Dcm shall reject the request with a negative response with NRC 0x31 RequestOutOfRange.⌋()

Note : The Dcm does not directly inform the SW-C about activation of ResponseOnEvent. The SW-C has to watch the change of the corresponding ModeDeclarationGroup DcmResponseOnEvent_<RoeEventID> and start reporting data identifier changes to the Dcm if the Mode is 'ROE started'

[SWS_Dcm_00920] ⌈If Dcm_TriggerOnEvent() is called and the passed RoeEvent is active, the Dcm shall trigger an Event for this RoeEvent.⌋()

[SWS_Dcm_00921] ⌈If an event is triggered for onChangeOfDataIdentifier, the Dcm shall execute a serviceToResponseTo 0x22 with the Did which is referred for this RoeEvent (DcmDspRoeDidRef) .⌋()

7.2.4.8.6 Trigger a ServiceToRespondTo

[SWS_Dcm_00922] ⌈If a ServiceToRespondTo is triggered by a RoeEvent the Dcm shall execute the ServiceToRespondTo as normal diagnostic service which is executed.⌋()

[SWS_Dcm_00558] ⌈If a ServiceToRespondTo is triggered while the Dcm is already executing a request on a different diagnostic Protocol the Dcm shall postpone the ServiceToRespondTo until the execution of the service is finalized.⌋()

[SWS_Dcm_00923] ⌈The Dcm shall only process the last ServiceToRespondTo. If already a ServiceToRespondTo is postponed due to another service execution the new respond shall overwrite the previous trigger.⌋()

[SWS_Dcm_00924] ⌈If a ServiceToRespondTo is executed while a Request on a different diagnostic protocol is received the ServiceToRespondTo shall be canceled.⌋()

[SWS_Dcm_00925] ⌈If ServiceToRespondTo are pending when the RoeEvent changes to the 'ROE cleared' state or 'ROE stopped' state the pending RoeEvent will be removed.⌋()

[SWS_Dcm_00127] ⌈ If the UDS service ResponseOnEvent (0x86) is received with the subservice StartResponseOnEvent, then the DSP sub-module shall store the respective configured sourceAddress of the received RxPduId for all RoeEvents which will be started until the eventWindowTime times out.⌋()

[SWS_Dcm_00128] ⌈The DSP submodule shall forward this stored sourceAddress as parameter in the DslInternal_ResponseOnOneEvent() function, where it is used to trigger a serviceToRespondTo⌋()

Note: The Dcm stores the sourceAddress of the protocol where the ROE request is received, independent if the serviceToResponseTo is send to a same or a different TxPduId. The sourceAddress links always the correct TxPduId, because there is only one TxPduId for ServiceToRespondTo linked to one protocol (See ConfigurationParameter DcmDslROEConnectionRef). If RoeEvents are active over power cycles the sourceAddress need to be stored over power cycles.

7.2.4.8.7 Send a ServiceToRespondTo

The Dcm supports the transmission from ServiceToResponseTo on the same TxPduId like the ROE response is send (TYPE 1) or on a different TxPduId (TYPE 2).

[SWS_Dcm_00131] ⌈ 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).⌋()

[SWS_Dcm_00926] ⌈If a ROE request is received on a protocol DcmDslMainConnection, the Dcm shall send the ServiceToRespondTo on the protocol which is referred as DcmDslROEConnectionRef. ⌋()

Note: if the EventWindowTime ist active over more than this power cycle, the Dcm has to store the protocol where the event was started.**[SWS_Dcm_00927]**「If the referred Protocol for ResponseOnEvent (DcmDslROEConnectionRef) is configured for TYPE1 the Dcm shall send the ServiceToRespondTo to the same TxPduID as the ROE response is send to. 」()

[SWS_Dcm_00928]「If the referred Protocol for ResponseOnEvent (DcmDslROEConnectionRef) is configured for TYPE2 the Dcm shall send the ServiceToRespondTo to the configured TxPduID (See configuration parameter DcmDslRoeTxPduRef). 」()

[SWS_Dcm_00132]「 The content of the pMsgContext pointer (ROE message) shall be copied into the buffer.」()

[SWS_Dcm_00133]「 ROE communication shall only be performed in Full Communication Mode.」()

[SWS_Dcm_00134]「 ROE events shall be disabled in any other Communication Mode except for the Full Communication Mode.」()

[SWS_Dcm_00135]「 ROE events occurring in a communication mode different from Full Communication Mode shall be discarded and not queued for later transmission. 」()

[SWS_Dcm_00136]「 ROE events requested by the Application shall not activate the Full Communication Mode.」()

1.1.1.1.1.13 Roe transmission cycle

[SWS_Dcm_00601]「 The DCM module shall respect a minimum time between two (2) consecutive Roe transmissions (see configuration parameter ***DcmDspRoeInterMessageTime***)」()

7.2.4.8.8 ResponseOnEvent in multiple client environments

[SWS_Dcm_00929]「If at least one RoeEvent is in ‘ROE started’ state the Dcm shall always process ROE request with the sub-function clearResponseOnEvent independent of the DcmDslProtocol where the request is received.」()

[SWS_Dcm_00930] ⌈If at least one RoeEvent is in 'ROE started' state the Dcm shall always process ROE request with the sub-function stopResponseOnEvent independent of the DcmDslProtocol where the request is received.⌋()

[SWS_Dcm_00940] ⌈If at least one RoeEvent is in 'ROE started' state the Dcm shall reject all ROE request received on a different DcmDslProtocol than the protocol where the RoeEvents were started with an NRC 0x22 (ConditionsNotCorrect), except for **SWS_Dcm_00929** and **SWS_Dcm_00930**.⌋()

[SWS_Dcm_01045] ⌈Only TYPE2 messages will support parallel execution of Diagnosis response.⌋()

7.2.4.9 Support of segmented response (paged-buffer)

[SWS_Dcm_00028] ⌈ If enabled (***DcmPagedBufferEnabled***=TRUE), the DCM module shall provide a mechanism to send responses larger than the configured and allocated diagnostic buffer.⌋(BSW04017)

[SWS_Dcm_01058]⌈ If DcmPagedBufferEnabled == TRUE and the generated Response for a Request is longer than DcmDslProtocolMaximumResponseSize, the Dcm shall respond with NRC 0x14 (DCM_E_RESPONSETOOLONG). ⌋()

[SWS_Dcm_01059]⌈ If DcmPagedBufferEnabled == FALSE and the generated Response for a Request is longer than Dcm_MsgContextType structure element resMaxDataLen, the Dcm shall respond with NRC 0x14 (DCM_E_RESPONSETOOLONG) .⌋()

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).

7.2.4.10 Support of ResponsePending response triggered by the Application

In some cases, e.g. in case of routine execution, 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).

When the DCM module calls an operation and gets an error status DCM_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.

7.2.4.11 Manage security level

[SWS_Dcm_00020] ⌈ The DSL submodule shall save the level of the current active security level.⌋(BSW04005)

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()`

[SWS_Dcm_00033] ⌈ During DCM initialization the security level is set to the value 0x00 (DCM_SEC_LEV_LOCKED).⌋(BSW101)

By SWS_Dcm_00033, the ECU is locked.

[SWS_Dcm_00139] ⌈ 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.

7.2.4.12 Manage session state

[SWS_Dcm_00022] ⌈ The DSL submodule shall save the state of the current active session.⌋(BSW04006)

For accessing this variable, the DSL submodule provides interfaces to:

- get the current active session: `Dcm_GetSesCtrlType()`
- set a new session: `DslInternal_SetSesCtrlType()`

[SWS_Dcm_00034] ⌈ During DCM initialization, the session state is set to the value 0x01 (“DefaultSession”).⌋(BSW101)

[SWS_Dcm_01062]⌈ The call to `Dcm_ResetToDefaultSession()` allows the application to reset the current session to Default session and invokes the mode switch of the ModeDeclarationGroupPrototype DcmDiagnosticSessionControl by calling SchM_Switch_<bsnp>_DcmDiagnosticSessionControl(RTE_MODE_DcmDiagnosticSessionControl_DEFAULT_SESSION).⌋()

Example: Automatic termination of an extended diagnostic session upon exceeding of a speed limit.

7.2.4.13 Keep track of active non-default sessions

[SWS_Dcm_00140] ⌈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) and invoke the the mode switch of the ModeDeclarationGroupPrototype DcmDiagnosticSessionControl by calling SchM_Switch_<bsnp>_DcmDiagnosticSessionControl(RTE_MODE_DcmDiagnosticSessionControl_DEFAULT_SESSION) .⌋()

Note: <bsnp> is the BSW Scheduler Name Prefix

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

[SWS_Dcm_00141] ⌈

Sub-sequent start	Completion of any final response message or an error indication (`Dcm_TpTxConfirmation()`: 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_TpRxIndication()`: indication of an error)
Sub-sequent stop	Start of a multi-frame request message (`Dcm_StartOfReception()`: indicates start of PDU reception)
	Reception of single-frame request message. (`Dcm_StartOfReception()`: indicates start of PDU reception)

“Start of S3Server” means reset the timer and start counting from the beginning.⌋()

7.2.4.14 Allow to modify timings

[SWS_Dcm_00027] ⌈ The DCM module shall handle the following protocol timing parameters in compliance with [18]: P2ServerMin, P2ServerMax, P2*ServerMin, P2*ServerMax, S3Server⌋(BSW04015)

[SWS_Dcm_00143] ⌈ P2min / P2*min and S3Server shall be set to defined values: P2min = 0ms, P2*min = 0ms, S3Server = 5s.⌋(BSW04015)

.

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,
- 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 ...).

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.

[SWS_Dcm_00035] ⌈ With every protocol initialization, the DSL submodule sets a link to the corresponding service table (see configuration parameter ***DcmDslProtocolSIDTable***).⌋(BSW101)

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.

[SWS_Dcm_00015] ⌈ A protocol with higher priority is allowed to preempt the already running protocol.⌋(BSW04021)

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

[SWS_Dcm_00459] ⌈ If a running diagnostic request is preempted by a higher priority request (of another protocol), the DSL submodule shall call all configured `Xxx_StopProtocol()` functions (see configuration parameter ***DcmDslCallbackDCMRequestService***).⌋()

[SWS_Dcm_00079] ⌈ In order to cancel pending transmission in lower-layer, related to the lower priority request, the DCM module shall call `PduR_DcmCancelTransmit()` with the following parameters:
PduId: the id of the Pdu to be canceled⌋()

[SWS_Dcm_00460] ⌈ 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. The current protocol shall be stopped and the new one started. ⌋()

[SWS_Dcm_01046] ⌈ If a running diagnostic request is preempted by a higher priority request (of another protocol), the DCM shall cancel all external pending operations with Dcm_OpStatus set to DCM_CANCEL ⌋()

[SWS_Dcm_01047] ⌈ In case an operation to the Dem is pending and the new request also requires an interaction with the Dem, the Dcm shall accept the new request and call the corresponding Dem API with the parameters from the new request. ⌋()

[SWS_Dcm_00575] ⌈ In order to cancel pending reception in lower-layer, related to the lower priority request, the DCM module shall call `PduR_DcmCancelReceive ()` with the following parameters:
PduId: the id of the Pdu to be canceled ⌋()

[SWS_Dcm_00576] ⌈ 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 receiverequest. The current protocol shall be stopped and the new one started. ⌋()

[SWS_Dcm_00625] ⌈ A Low-priority or same-priority request can preempt a higher priority protocol if this higher priority protocol is in default session and no active request is in execution phase. In this case the DSL submodule shall call all configured `Xxx_StopProtocol()` functions (see configuration parameter ***DcmDslCallbackDCMRequestService***). ⌋()

[SWS_Dcm_00728] ⌈The handling of protocols with equal priority shall be possible. ⌋()

[SWS_Dcm_00727] ⌈If a diagnostic request is already running and a second request (ClientB) can not be processed (e.g.due to priority assessment), the response behaviour depends on the configuration option parameter ***DcmDslDiagRespOnSecondDeclinedRequest*** (see SWS_Dcm_00914_Conf). If this configuration parameter is TRUE, a negative response with NRC 0x21 (BusyRepeatRequest) shall be issued for the second request (see **[SWS_Dcm_00788** and **[SWS_Dcm_00789**). If the configuration parameter is FALSE, no response shall be issued (see **[SWS_Dcm_00790**).⌋()

[**SWS_Dcm_00729**] ⌈In case of multiple clients with different PduIDs which are requesting the same protocol, as all the connections of the same protocol are having the same priority, a second request (with the different RxPduId) will not be processed. If the configuration parameter ***DcmDslDiagRespOnSecondDeclinedRequest*** is TRUE, a negative response with NRC 0x21 (BusyRepeatRequest) shall be issued for the second request. If the configuration parameter is FALSE, no response shall be issued.⌋()

[SWS_Dcm_01050] ⌈ In case of diagnostic parallel requests, with same / lower priority than the active request then the ComM APIs (ComM_DCM_ActiveDiagnostic, ComM_DCM_InactiveDiagnostic) shall not be called.⌋()

7.2.4.15.4 Detection of protocol start

[SWS_Dcm_00036] ⌈ With first request of a diagnostic protocol, the DSL submodule shall call all configured `Xxx_StartProtocol()` functions (see configuration parameter ***DcmDslCallbackDCMRequestService***).⌋(BSW101)

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

[SWS_Dcm_00144] ⌈ 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***). ⌋(BSW04015)

[SWS_Dcm_00145] ⌈ 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***).⌋()

[SWS_Dcm_00146] ⌈ After all `Xxx_StartProtocol()` functions have returned E_OK (meaning all components have allowed the start of the protocol), the security state is reset.⌋()

[SWS_Dcm_00147] ⌈After all `Xxx_StartProtocol()` functions have returned E_OK (meaning all components have allowed the start of the protocol), the session state is reset to default session. Furthermore the DCM module shall invoke the the mode switch of the ModeDeclarationGroupPrototype DcmDiagnosticSessionControl by calling SchM_Switch_<bsnp>_DcmDiagnosticSessionControl(RTE_MODE_DcmDiagnosticSessionControl_DEFAULT_SESSION) .⌋()

Note: <bsnp> is the BSW Scheduler Name Prefix

7.2.4.15.5 Protocol stop

A protocol stop can appear only in case of protocol preemption (See chapter 7.2.4.15.3 Preemption of protocol)

[SWS_Dcm_00624] ⌈ With the reception of `Dcm_TpTxConfirmation()` connected to the response given by the DSL submodule, the DCM shall not stop the current protocol (no call `to xxx_StopProtocol)`.⌋()

Note: A protocol (e.g. OBD) will be active till reset or other protocol preempts.

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 SWS_Dcm_00028).

7.2.4.17 Communication Mode Handling

7.2.4.17.1 No Communication

The ComM module will indicate the No Communication Mode to the DCM module by calling `Dcm_ComM_NoComModeEntered()`. In response, the DCM will immediately disable all transmissions (see the definition of `Dcm_ComM_NoComModeEntered()` for details).

7.2.4.17.2 Silent Communication

The ComM module will indicate the Silent Communication Mode to the DCM module by calling `Dcm_ComM_SilentComModeEntered()`. In response, the DCM will immediately disable all transmissions (see the definition of `Dcm_ComM_SilentComModeEntered()` for details).

7.2.4.17.3 Full Communication

The ComM module will indicate the Full Communication Mode to the DCM module by calling `Dcm_ComM_FullComModeEntered()`. In response, the DCM will enable all transmissions (see the definition of `Dcm_ComM_FullComModeEntered()` for details).

7.2.4.17.4 Default Session

[SWS_Dcm_00163] ⌈ With every accepted request (see **SWS_Dcm_01050**) in default session of a diagnostic protocol, the DCM shall call `ComM_DCM_ActiveDiagnostic(NetworkId)`,with the networkId associated to the received Pdu (see ***DcmDslProtocolComMChannelRef***), to inform the ComM module about the need to stay in Full Communication Mode.⌋()

[SWS_Dcm_00164] ⌈ With the reception of `Dcm_TpTxConfirmation()` connected to the response given by the DSL submodule, the DCM shall call `ComM_DCM_InactiveDiagnostic(NetworkId)`,with the networkId associated to the transmitted Pdu (see DcmDslProtocolComMChannelRef), to inform the ComM module that Full Communication is not longer needed.⌋()

[SWS_Dcm_00165] ⌈ The DCM shall not call `ComM_DCM_InactiveDiagnostic(NetworkId)` for NRC 0x78 (Response pending). The DCM shall only call `ComM_DCM_InactiveDiagnostic(NetworkId)` with the very last response (positive or negative) connected to the request.⌋()

[SWS_Dcm_00166] ⌈ If a “suppressPosRspMsgIndicationBit” is indicated and the positive response will be suppressed, the DCM shall call `ComM_DCM_InactiveDiagnostic(NetworkId).`⌋()

[SWS_Dcm_00697] ⌈ If a negative response is suppressed in case of functional addressing (see SWS_Dcm_00001), the DCM shall call `ComM_DCM_InactiveDiagnostic(NetworkId).`⌋()

7.2.4.17.5 Session Transitions

[SWS_Dcm_00167] ⌈ 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(NetworkId)`,with the networkId associated to the received Pdu, to inform the ComM module about the need to stay in Full Communication Mode.⌋()

[SWS_Dcm_00168] ⌈ 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 or protocol preemption), then the DCM shall call `ComM_DCM_InactiveDiagnostic(NetworkId)`,with the networkId associated to the received Pdu, to inform the ComM module that Full Communication is not longer needed. ⌋()

7.2.4.17.6 Non Default Session:

[SWS_Dcm_00169] ⌈ As long as the server is in a session other than the default session, the DCM shall not call `ComM_DCM_ActiveDiagnostic(NetworkId)`,with the networkId associated to the received Pdu, when receiving a request from a client provided by the PduR module. ⌋()

[SWS_Dcm_00170] ⌈ As long as the server is in a session other than the default session, the DCM shall not call `ComM_DCM_InactiveDiagnostic(NetworkId)`,with the networkId associated to the received Pdu, with the reception of `Dcm_TpTxConfirmation()` 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.

[SWS_Dcm_00178] ⌈ 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 suppress a negative 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 SWS_Dcm_00241). 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 SWS_Dcm_00200). This special suppression handling is completely performed within the DSD submodule.

7.3.2.3 Receive a request message and suppress a negative response

In case of functional addressing the DSD submodule shall suppress the negative response for NRC 0x11, 0x12 and 0x31 (see SWS_Dcm_00001)

7.3.2.4 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 if at least one of the parameters was processed successfully.

[SWS_Dcm_00827] ⌈ The DSD sub-module shall check the received diagnostic request in the order given by ISO14229-1. If one of the computations failed the DCM shall stop the execution of the NRC check sequence then stop or do not start the execution of the received diagnostic request and finally transmit the NRC for which the computation failed. ⌋()

7.3.2.5 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 [15]).
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.

7.3.2.6 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 `Dcm_ExternalProcessingDone()` 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 4 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 5 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.

[SWS_Dcm_00084] 「 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.

[SWS_Dcm_00192] 「 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.」()

[SWS_Dcm_00193] ⌈ 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.

[SWS_Dcm_00195] ⌈ The DSL submodule shall provide the current “Service Identifier Table” ⌋()

Rationale for SWS_Dcm_00195: 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.

[SWS_Dcm_00196] ⌈For the check, the DSD submodule shall scan the active “Service Identifier Table” for a newly received diagnostic service identifier. If this service identifier is supported and if the configuration parameter ***DcmDsdSidTabFnc*** (see ECUC_Dcm_00777) is not empty, the DSD submodule shall call the configured service interface (<Module>_<DiagnosticService>). If the configuration parameter is empty, the DCM shall call the internally implemented service interface. ⌋()

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

[SWS_Dcm_00197] ⌈ 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. ⌋()

[SWS_Dcm_00198] ⌈ 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 [15] Section 6.5: Server response implementation rules).

[SWS_Dcm_00200] ⌈ If the “suppressPosRspMsgIndicationBit” is TRUE, the DSD submodule shall NOT send a positive response message. ⌋(BSW04020)

[SWS_Dcm_00201] ⌈ The DSD submodule shall remove the “suppressPosRspMsgIndicationBit” (by masking the Bit) from the diagnostic message. ⌋()

[SWS_Dcm_00202] ⌈ The DCM module shall transport the information on a suppression of a positive response being active (between the layers) via the parameter Dcm_MsgContextType. ⌋()

[SWS_Dcm_00203] ⌈ In case of responsePending the DCM module shall clear the “suppressPosRspMsgIndicationBit.” ⌋()

Rationale for SWS_Dcm_00203: In the described case the final response (negative/positive) is required.

[SWS_Dcm_00204] ⌈ The DCM module shall only perform the “suppressPosRspMsgIndicationBit” handling when the configuration parameter ***DcmDsdSidTabSubfuncAvail*** is set for the newly received service identifier ⌋()

Note: The “suppressPosRspMsgIndicationBit” handling needs to be considered independent of the processing order in the request (like for RoutineControl service).

Rationale for SWS_Dcm_00204: 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

[SWS_Dcm_01000] ⌈In case a NRC is generated by DSD, the API DspInternal_DcmConfirmation is not called, but only XXX_Confirmation. ⌋()

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)) and sub-service 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.

[SWS_Dcm_00211] ⌈ 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 (serviceNotSupportedInActiveSession) to the DSL submodule.⌋()

[SWS_Dcm_00616] ⌈ If the newly received diagnostic service is allowed in the current Diagnostic Session (see SWS_Dcm_00211), but the requested subservice is not allowed in the current Diagnostic Session (according to the configuration parameter ***DcmDsdSubServiceSessionLevelRef***), the DSD submodule shall transmit a negative response with NRC 0x7E (subFunctionNotSupportedInActiveSession) 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()`.

[SWS_Dcm_00217] ⌈ If the newly received diagnostic service is not allowed in the current Security level (according to the configuration parameter ***DcmDsdSidTabSecurityLevelRef***), the DSD submodule shall transmit a negative response with NRC 0x33 (Security access denied) to the DSL submodule.⌋()

[SWS_Dcm_00617]「 If the newly received diagnostic service is allowed in the current Security level (see SWS_Dcm_00217), but the requested subservice is not allowed in the current Security level (according to the configuration parameter ***DcmDsdSubServiceSecurityLevelRef***), 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 Service mode dependencies

[SWS_Dcm_00773]「If the newly received diagnostic service is not allowed in the current mode condition (according to the configuration parameter ***DcmDsdSidTabModeRuleRef***), the DSD submodule shall transmit the calculated negative response of the referenced DcmModeRule to the DSL submodule.」()

[SWS_Dcm_00774]「If the newly received diagnostic service is allowed in the current mode condition (**[SWS_Dcm_00773**), but the requested subservice is not allowed in the current mode condition (according to the configuration parameter ***DcmDsdSubServiceModeRuleRef***), the DSD submodule shall transmit the calculated negative response of the referenced DcmModeRule to the DSL submodule.」()

7.3.4.4 Check format and subfunction support

The DSD submodule checks whether a specific subfunction is supported before executing the requested command.

[SWS_Dcm_00273]「 The DSD submodule shall trigger a negative response with NRC 0x12 (SubFunction not supported), when the analysis of the request message results in subfunction not supported. This analysis shall not be done for UDS Service RoutineControl (0x31)」()

The DSD submodule will check for the minimum message length before executing the requested command.

[SWS_Dcm_00696]「 The DSD submodule shall trigger a negative response with NRC 0x13 (Incorrect message length or invalid format), if the length of the request is inferior to the minimul length of the request.」()

7.3.4.4.1 Verification of the Manufacturer Application environment/permission

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

[SWS_Dcm_00218] ⌈ If configured (configuration parameter ***DcmDsdRequestManufacturerNotificationEnabled***=TRUE), the DSD submodule shall call the operation `Xxx_Indication()` on all configured ServiceRequestIndication ports (see configuration parameter ***DcmDsdServiceRequestManufacturerNotification***).⌋()

[SWS_Dcm_00462] ⌈ If at least a single `Xxx_Indication()` function called according to SWS_Dcm_00218 returns E_REQUEST_NOT_ACCEPTED, the DSD submodule shall give no response.⌋()

Note: in case of **[SWS_Dcm_00462**, DspInternal_DcmConfirmation is not called but only `Xxx_Confirmation()` is called.

[SWS_Dcm_00463] ⌈ If at least a single `Xxx_Indication()` function called according to SWS_Dcm_00218 has returned E_NOT_OK and no function has returned E_REQUEST_NOT_ACCEPTED, the DSD submodule shall trigger a negative response with NRC from the `ErrorCode` parameter.⌋()

7.3.4.4.2 Verification of the Supplier Application environment/permission

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

[SWS_Dcm_00516] ⌈ If configured (configuration parameter ***DcmDsdRequestSupplierNotificationEnabled***=TRUE), the DSD submodule shall call the operation `Xxx_Indication()` on all configured ServiceRequestIndication ports (see configuration parameter ***DcmDsdServiceRequestSupplierNotification***).⌋()

[SWS_Dcm_00517] ⌈ If at least a single `Xxx_Indication()` function called according to SWS_Dcm_00516 returns E_REQUEST_NOT_ACCEPTED, the DSD submodule shall give no response.⌋()

[SWS_Dcm_00518] ⌈ If at least a single `Xxx_Indication()` function called according to SWS_Dcm_00516 has returned E_NOT_OK and no function has returned E_REQUEST_NOT_ACCEPTED, the DSD submodule shall trigger a negative response with NRC from the `ErrorCode` parameter.⌋()

7.3.4.5 Distribution of diagnostic message to DSP submodule

[SWS_Dcm_00221] ⌈ 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. ⌋()

7.3.4.6 Assemble positive or negative response

[SWS_Dcm_00222] ⌈ 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).

7.3.4.6.1 Positive Response

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

[SWS_Dcm_00223] ⌈ The DSD submodule shall add the response service identifier and the response data stream (returned by the Application) in the parameter “Dcm_MsgContextType”. ⌋()

[SWS_Dcm_00224] ⌈ 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. ⌋()

[SWS_Dcm_00225] ⌈ The DSD submodule shall execute the “Initiate transmission” functionality in the next execution step ⌋()

7.3.4.6.2 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 [15] (see Section 4.2.4 Response code parameter definition Table 12) and ISO15031-5 [16]. The DSP and the Application have to take care of the correct use of NRC of the executed Service ID.

[SWS_Dcm_00228] ⌈ The DSD submodule shall handle all NRCs supported from the Application and defined in `Dcm_NegativeResponseCodeType`. ⌋()

7.3.4.6.3 Suppression of response

[SWS_Dcm_00231] ⌈ 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. ⌋()

[SWS_Dcm_00001] ⌈ 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). ⌋(BSW04020)

7.3.4.7 Initiate transmission

[SWS_Dcm_00232] ⌈ The DSD submodule shall forward the diagnostic (response) message (positive or negative response) to the DSL submodule. ⌋()

[SWS_Dcm_00237] ⌈ 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.

[SWS_Dcm_00235] ⌈ The DSL submodule shall forward the received confirmation from the PduR module to the DSD submodule. ⌋()

[SWS_Dcm_00236] ⌈ The DSD submodule shall forward the confirmation via the internal function `DspInternal_DcmConfirmation()` to the DSP submodule. ⌋()

[SWS_Dcm_00238] ⌈ 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()`.

[SWS_Dcm_00240] ⌈In case the request has been fully processed by the Dcm, The DSD submodule shall finish the processing of one Diagnostic Message of the Diagnostic Service Dispatcher by calling `DspInternal_DcmConfirmation()`.⌋()

Rationale for SWS_Dcm_00240: 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()`. (Refer to 8.11.3 DspInternal_DcmConfirmation)

[SWS_Dcm_00741] ⌈The DSD submodule shall call the operation Xxx_Confirmation() on all ports using the ServiceRequestNotification interface (see configuration parameter DcmDsdServiceRequestManufacturerNotification and DcmDsdServiceRequestSupplierNotification) ⌋()

[SWS_Dcm_00742] ⌈The call of Xxx_Confirmation() shall be done right after the call of `DspInternal_DcmConfirmation()`⌋()

[SWS_Dcm_00770] ⌈The DSD submodule shall call the operation Xxx_Confirmation() on all ports using the ServiceRequestNotification interface (see configuration parameter DcmDsdServiceRequestManufacturerNotification and DcmDsdServiceRequestSupplierNotification).⌋()

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

[SWS_Dcm_00272] ⌈ 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

[SWS_Dcm_00039] ⌈ The DSP submodule shall assemble the response message excluding response service identifier and determine the response message length. ⌋()

[SWS_Dcm_00038] ⌈ 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. ⌋(BSW04017)

Requirement SWS_Dcm_00038 is needed because of segmented diagnostic data transmission on CAN using ISO15765-2 [17], 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 7.2.4.9 for details about the paged-buffer mechanism).

[SWS_Dcm_00269] ⌈ The DSP submodule shall confirm the completion of the request processing with the function call `Dcm_ExternalProcessingDone()`. ⌋()

7.4.1.3 Additional Negative Response Codes (NRCs)

[SWS_Dcm_00271] ⌈Unless another particular NRC is specified,the DSP submodule shall trigger a negative response with NRC 0x10 (generalReject), when the API calls made to execute the service do not return OK. ⌋()

[SWS_Dcm_00275] ⌈ 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. ⌋()

7.4.1.4 Diagnostic mode declaration groups

[SWS_Dcm_00775] ⌈The DCM shall act as a mode manager for the diagnostic modes:

1) DcmDiagnosticSessionControl (service 0x10)

2) DcmEcuReset (partly service 0x11)

3) DcmModeRapidPowerShutDown (partly service 0x11)

4) DcmCommunicationControl_<symbolic name of ComMChannelId>. (service 0x28)

5) DcmControlDTCSetting (service 0x85)

6) DcmResponseOnEvent_<RoeEventID> (service 0x86)

⌋()

Note: The RTE/SchM will prefix the names with “MODE_”, wherefore the names do not include the MODE keyword.

[SWS_Dcm_00776] ⌈The DCM SWS defines the mode of its ***ModeDeclarationGroups***.⌋()

[SWS_Dcm_00778] ⌈ For ***ModeDeclarationGroup*** DcmDiagnosticSessionControl, the mode declaration shall be identical to the shortname of the container DcmDspSessionRow:

```
ModeDeclarationGroup DcmDiagnosticSessionControl {
   {
   DEFAULT_SESSION,
   PROGRAMMING_SESSION,
   EXTENDED_SESSION,
   etc.
   }
        initialMode = DEFAULT_SESSION
};⌋()
```

Note: According **constr_6001** there are standardized mode declaration which are part of the standardized AUTOSAR interface.
Note: Refer [ecuc_sws_2108] defining the symbolic name prefix

[SWS_Dcm_00806] ⌈The DCM shall define the ***ModeDeclarationGroupPrototype*** DcmDiagnosticSessionControl as provided-ModeGroup based on the ***ModeDeclarationGroup*** DcmDiagnosticSessionControl. ⌋()

[SWS_Dcm_00777] ⌈The DCM shall define the ***ModeDeclarationGroupPrototype*** DcmEcuReset as provided-ModeGroup based on the following ***ModeDeclarationGroup***:
ModeDeclarationGroup DcmEcuReset {
{
NONE,
HARD
KEYONOFF,
SOFT,
JUMPTOBOOTLOADER,
JUMPTOSYSSUPPLIERBOOTLOADER ,
EXECUTE
}
initialMode = NONE
};⌋()

[SWS_Dcm_00807] ⌈The DCM shall define the ***ModeDeclarationGroupPrototype*** DcmRapidPowerShutDown as provided-ModeGroup based on the following ***ModeDeclarationGroup***:
ModeDeclarationGroup DcmModeRapidPowerShutDown {
{
ENABLE_RAPIDPOWERSHUTDOWN,
DISABLE_RAPIDPOWERSHUTDOWN

}
initialMode = ENABLE_RAPIDPOWERSHUTDOWN
};⌋()

[SWS_Dcm_00779] ⌈For ***ModeDeclarationGroup*** Dcm_CommunicationControl the mode declarations shall be identical to the enumarations of the type Dcm_CommunicationModeType. The initial mode shall be set to DCM_ENABLE_RX_TX_NORM⌋()

[SWS_Dcm_00780] ⌈The DCM shall define for each network which is considered in the CommunicationControl service a separate ***ModeDeclarationGroupPrototype*** with the naming convention DcmCommunicationControl_<symbolic name of ComMChannelId>.⌋()

[SWS_Dcm_00781] ⌈The DCM shall define the ***ModeDeclarationGroupPrototype*** DcmControlDTCSetting as provided-ModeGroup based on the following ***ModeDeclarationGroup***:
ModeDeclarationGroup DcmControlDTCSetting {
{
ENABLEDTCSETTING,
DISABLEDTCSETTING

```
}
        initialMode = ENABLEDTCSETTING
};」()
```

[SWS_Dcm_00931]「For the **ModeDeclarationGroup** DcmResponseOnEvent_<RoeEventID>, the mode declaration shall be identical to the symbolic names of the state of the state machine of the RoeEvent:

```
ModeDeclarationGroup DcmResponseOnEvent_<RoeEventID>  {
{
  EVENT_STARTED,
  EVENT_STOPPED,
  EVENT_CLEARED
}
        initialMode = EVENT_CLEARED
}」()
```

[SWS_Dcm_00933]「ModeSwitchInterface

```
SchM_Switch_<bsnp>_DcmResponseOnEvent_<RoeEventId> {
  isService = true;
  RoeMode currentMode;
};」()
```

Note: <bsnp> is the BSW Scheduler Name Prefix

The Dcm provides a state machine for each RoeEvent (see Figure 6). The state for a RoeEvent is needed by SWC to activate event reporting or report the Roe status to a Did. Therefore the Dcm provides for each state of each RoeEvent a ModeDeclarationGroup which reports the current state of the state machine as mode.

[SWS_Dcm_00934]「The ModeDeclarationGroup shall represent the current state of the ROE state machine for this RoeEvent.」()

7.4.1.5 mode dependent request execution

The execution of a request can be limited depending on mode condition. This enables the DCM to formalize environmental checks.

Similar to a session/security check a further check (see **[SWS_Dcm_00773** and **[SWS_Dcm_00774**) can be configured to the DCM. The referenced mode rule is arbitrating on to several mode declarations of a mode declaration groups in which the request can be processed. Otherwise a configurable NRC (see **[SWS_Dcm_00812**) is responded.

[SWS_Dcm_00808] ⌈The DcmModeRule shall evaluate all referenced DcmModeConditions and/or nested DcmModeRules either by a logical AND in case ***DcmLogicalOperator*** is set to DCM_AND or by a logical OR in case the ***DcmLogicalOperator*** is set to DCM_OR. In case only a single ***DcmModeCondition*** or ***DcmModeRule*** is referenced the ***DcmLogicalOperator*** shall not be present and therefore not be used.⌋()

[SWS_Dcm_00809] ⌈Each DcmModeConditions shall either have a ***DcmSwcModeRef*** or a ***DcmBswModeRef***.⌋()

[SWS_Dcm_00810] ⌈The DcmModeConditions shall evaluate if the referenced Mode-Declaration is set in case of ***DcmConditionType*** is set to DCM_EQUALS or is not set in case of ***DcmConditionType*** is set to DCM_EQUALS_NOT.⌋()

Note: The current mode of the referenced ModeDeclarationGroupPrototypes could be read by either the API SchM_Mode (in case of DcmBswModeRef) or by the API Rte_Mode (in case of DcmSwcModeRef).

[SWS_Dcm_00811] ⌈In case multiple DcmModeConditions are referenced within a ***DcmModeRule*** they shall be evaluated in order of the index attributes of the EcucReferenceValues for ***DcmArgumentRef***. ⌋()

Note: This implies the priority of NRCs

[SWS_Dcm_00782] ⌈The DcmModeRule shall have an optional parameter DcmModeRuleNrcValue to define the NegativeResponseCode which is sent in case the service execution is prohibited due to the DcmModeRule.⌋()

[SWS_Dcm_00812] ⌈In case a nested DcmModeRule contains also a ***DcmModeRuleNrcValue*** parameter, this NRC shall be used prior the higher-level NRC.⌋()

[SWS_Dcm_00813] ⌈In case DcmLogicalOperator is set to DCM_AND, the first failed DcmModeRule with an explicit configured NRC (DcmModeRuleNrcValue) shall be used to define the NRC for the response message. ⌋()

[SWS_Dcm_00814] ⌈In case DcmLogicalOperator is set to DCM_OR, the last DcmModeRule with an explicit configured NRC (DcmModeRuleNrcValue)shall be used to define the NRC for the response message.⌋()

Note: The difference in the AND and OR logical operation is to allow an optimized implementation.

[SWS_Dcm_00815]「In case the complete evaluation result in no specific NRC the NRC 0x22 (ConditionsNotCorrect) shall be used.」()

[SWS_Dcm_00942]「The DCM shall create for commonly used ModeDeclarationGroupPrototype of each DcmSwcModeRef of DcmModeConditions a require mode switch port referencing this ModeDeclarationGroupPrototype. The name pattern of this port prototype shall be DcmModeUser_<ModeDeclarationGroupPrototype>" in case the ModeDeclarationGroupPrototype shortname is unique. Otherwise the name pattern is implementation specific, except the required prefix "DcmModeUser_".」()

Note: ModeDeclarationGroupPrototypes are not necessarily unique, wherefore the exception is required to avoid name clashes in the DCM Service-SWC.

Examples on using mode dependent request execution:

General assumptions:
1) DcmModeRule1 consists of DcmModeCondition1, DcmModeRule2 and DcmModeRule3
2) DcmModeRule1 defines NRC 0x22
3) DcmModeRule2 and DcmModeRule3 do not have any sub-rules
4) DcmModeRule2 defines NRC 0x72
5) DcmModeRule3 does not define a NRC value

Example 1:
1) DcmModeRule1 uses an OR combination (DcmModeCondition1 OR DcmModeRule2 OR DcmModeRule3)
a) DcmModeCondition1 is failing
--> NRC 0x22 is returned
b) DcmModeRule2 is failing
--> NRC 0x72 is returned
c) DcmModeRule3 is failing
--> NRC 0x22 is returned
d) DcmModeCondition1, DcmModeRule2 and DcmModeRule3 are failing
--> NRC 0x72 is returned
e) DcmModeCondition1 and DcmModeRule3 are failing
--> NRC 0x22 is returned

Example 2:
1) DcmModeRule1 uses an AND combination (DcmModeCondition1 AND DcmModeRule2 AND DcmModeRule3)
a) DcmModeCondition1 is failing
--> NRC 0x22 is returned
b) DcmModeRule2 is failing
--> NRC 0x72 is returned
c) DcmModeRule3 is failing
--> NRC 0x22 is returned
d) DcmModeCondition1, DcmModeRule2 and DcmModeRule3 are failing

--> NRC 0x22 is returned
e) DcmModeCondition1 and DcmModeRule3 are failing
--> NRC 0x22 is returned
e) DcmModeRule2 and DcmModeRule3 are failing
--> NRC 0x72 is returned

7.4.1.6 Sender/Receiver Communication

[SWS_Dcm_00962]「The Dcm shall create for each configured DcmDspData element having a sender/receiver interface (if parameter ***DcmDspDataUsePort*** is set to *USE_DATA_SENDER_RECEIVER*) which is read (***DcmDspDidRead***) a corresponding R-Port DataInterface with one data element having an ImplementationDataType of type ***DcmDspDataType***.」()

[SWS_Dcm_00963]「The Dcm shall create for each configured DcmDspData element having a sender/reciever (if parameter ***DcmDspDataUsePort*** is set to *USE_DATA_SENDER_RECEIVER*) which is written (***DcmDspDidWrite***) a corresponding P-Port with one data element DataInterface having an ImplementationDataType of type ***DcmDspDataType***.」()

[SWS_Dcm_00964]「The created ports of **SWS_Dcm_00962** and **SWS_Dcm_00963** shall derive the CompuMethod from the DcmDspDiagnosisScaling container (if present) and add them to the DataType in the port interface.」()

7.4.2 UDS Services

[SWS_Dcm_00442]「 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		supported (callout)
0x24	ReadScalingDataByIdentifier		Supported
0x27	SecurityAccess		Supported
0x28	CommunicationControl		Supported
0x2A	ReadDataByPeriodicIdentifier		Supported
0x2C	DynamicallyDefineDataIdentifie		Supported

0x2E	WriteDataByIdentifier		Supported
0x2F	InputOutputControlByIdentifier		Supported
0x31	RoutineControl		Supported
0x34	RequestDownload		supported (callout)
0x35	RequestUpload		supported (callout)
0x36	TransferData		supported
0x37	RequestTransferExit		supported
0x3D	WriteMemoryByAddress		supported (callout)
0x3E	TesterPresent		Supported
0x83	AccessTimingParameter		NRC “ServiceNotSupported”
0x84	SecuredDataTransmission		NRC “ServiceNotSupported”
0x85	ControlDTCSetting	On, off	supported
0x86	ResponseOnEvent	All excepted onComparisionOfValues and OnTimerInterrupt	supported
0x87	LinkControl		user optional

Table 6: Support of UDS Services」()

7.4.2.1 General behavior using DEM interfaces

[SWS_Dcm_00007]「The Dcm module shall retrieve the DTCStatusAvailabilityMask by using the function `Dem_DcmGetDTCStatusAvailabilityMask()`.」(SRS_Diag_04010)

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

Note : Masking is performed in the module Dem and does not need to be done on Dcm side (see SWS_Dem_00657 in [7])

[SWS_Dcm_00371]「 To avoid updating data values while reading out extended data records, the DCM module shall call the following API sequence: first lock updates by calling `Dem_DcmDisableDTCRecordUpdate`() (which is an exclusive area function concerning to BSW00434) then call `Dem_DcmGetSizeOfExtendedDataRecordByDTC()`, then `Dem_DcmGetExtendedDataRecordByDTC(RecNum)` and finally shall re-enable updates by calling `Dem_DcmEnableDTCRecordUpdate()`.」()

[SWS_Dcm_00372] ⌈ To avoid updating data values while reading out freeze frame records, the DCM module shall call the following API sequence: first lock updates by calling `Dem_DcmDisableDTCRecordUpdate`() (which is an exclusive area function concerning to BSW00434) then call `Dem_DcmGetSizeOfFreezeFrameByDTC(RecNum)`, then `Dem_DcmGetFreezeFrameDataByDTC(RecNum)` and finally shall re-enable updates by calling `Dem_DcmEnableDTCRecordUpdate()`.⌋()

[SWS_Dcm_00702] ⌈ If function Dem_DcmDisableDTCRecordUpdate() 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.

[SWS_Dcm_00700] ⌈ 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_DcmSetDTCFilter()`.⌋()

Note: The parameter DTCFormat of the functions Dem_DcmClearDTC(), Dem_DcmSetDTCFilter(), Dem_DcmSetFreezeFrameRecordFilter() and Dem_DcmGetNextFilteredDTCAndFDC() defines the output-format of the requested DTC values for the sub-sequent API calls.
For the 2-byte ISO15031-6 DTC format, the DTCFormat parameter shall be equal to DEM_DTC_FORMAT_OBD. For the 2-byte ISO14229-1 DTC format, the DTCFormat parameter shall be equal to DEM_DTC_FORMAT_UDS.

[SWS_Dcm_00835] ⌈The Dcm shall call Dem_DcmSetDTCFilter prior to Dem_DcmGetNumberOfFilteredDTC, any sequence of Dem_DcmGetNextFilteredDTC, any sequence of Dem_DcmGetNextFilteredDTCAndFDC, as well as any sequence of Dem_DcmGetNextFilteredDTCAndSeverity.⌋()

[SWS_Dcm_00836] ⌈The Dcm shall call Dem_DcmSetFreezeFrameRecordFilter prior to any sequence of Dem_DcmGetNextFilteredRecord.⌋()

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

[**SWS_Dcm_00250**] ⌈ The DCM module shall implement the UDS Service 0x10 ⌋(BSW04006)

[SWS_Dcm_00307] ⌈ 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 [15] Section 8.2).

[SWS_Dcm_00311] ⌈The send confirmation function shall set the new diagnostic session type with `DslInternal_SetSesCtrlType()` and shall set the new timing parameters (P2ServerMax, P2ServerMax*) (see configuration parameters ***DcmDspSessionP2ServerMax*** and ***DcmDspSessionP2StarServerMax***) and do the mode switch of the ***ModeDeclarationGroupPrototype*** DcmDiagnosticSessionControl by calling SchM_Switch_<bsnp>_DcmDiagnosticSessionControl() with the new diagnostic session type (see **SWS_Dcm_00778**).⌋()BSW04015)

[SWS_Dcm_00085] ⌈ The DSP submodule shall manage internally a read access for the dataIdentifier 0xF186 (ActiveDiagnosticSessionDataIdentifier) defined in ISO14229-1 [15].⌋()

7.4.2.3 UDS Service 0x11 - ECUReset

UDS Service ECUReset (0x11) allows an external tester to request a server reset. The service request contains parameter:

- resetType

[SWS_Dcm_00260] ⌈ The DCM module shall implement the UDS Service ECUReset (0x11) ⌋()

[**SWS_Dcm_00373**] ⌈On reception of a request for UDS Service 0x11 with the sub functions other than enableRapidPowerShutDown (0x04) or disableRapidPowerShutDown (0x05), the DCM module shall trigger the mode switch of ModeDeclarationGroupPrototype DcmEcuReset equal to the received resetType. After the mode switch is requested the DCM shall trigger the start of the positive response message transmission.
Sub function hardReset (0x01) to HARD
Sub function keyOffOnReset (0x02) to KEYONOFF,
Sub function softReset (0x03) to SOFT⌋()

Note: By this mode switch the DCM informs the BswM to prepare the shutdown of the ECU.

[**SWS_Dcm_00594**] ⌈On the transmit confirmation (call to Dcm_TpTxConfirmation) of the positive response, the DCM module shall trigger the mode switch of ModeDeclarationGroupPrototype DcmEcuReset to the mode EXECUTE (via SchM_Switch_<bsnp>_DcmEcuReset(RTE_MODE_DcmEcuReset_EXECUTE)).⌋()

Note: By this final mode switch the DCM request the BswM to finally shutdown the ECU and to to perform the reset.

[SWS_Dcm_00818] ⌈On reception of a request for UDS Service 0x11 with the sub functions enableRapidPowerShutdown (0x04) or disableRapidPowerShutdown (0x05), the DCM module shall trigger the mode switch of ModeDeclarationGroupPrototype DcmRapidPowerShutDown:

Sub function enableRapidPowerShutDown (0x04) to ENABLE_RAPIDPOWERSHUTDOWN,
Sub function disableRapidPowerShutDown (0x05) to DISABLE_RAPIDPOWERSHUTDOWN⌋()

Note: If EnableRapidPowerShutdown is enabled, the ECU should shorten its powerdown time.

[SWS_Dcm_00589] ⌈ In case the parameter *DcmDspPowerDownTime* is present, the DCM shall set the powerDownTime in positive response to sub-service *enableRapidPowerShutDown* with value set in *DcmDspPowerDownTime*.⌋()

[SWS_Dcm_00834] ⌈After sending the positive response of EcuReset (call of Dcm_TpTxConfirmation) the DCM shall ignore all further requests during reset-processing.⌋()

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.[

SWS_Dcm_00247] ⌈ The DCM module shall implement UDS Service 0x14.⌋(SRS_Diag_04010)

[SWS_Dcm_00005] ⌈ Upon reception of a UDS Service ClearDiagnosticInformation (0x14) request with parameter groupOfDTC, the DCM module shall call the operation `DcmClearDTC` with the following parameter values:
DTC: groupOfDTC from the service request
DTCFormat: DEM_DTC_FORMAT_UDS
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY⌋(SRS_Diag_04010, BSW04058, BSW04065)

[SWS_Dcm_00705] ⌈ In case Dem_DcmClearDTC() returns DEM_CLEAR_OK, the DCM module shall send a positive response.⌋()

[SWS_Dcm_00706] ⌈ In case Dem_DcmClearDTC() returns DEM_CLEAR_PENDING, the DCM shall invoke Dem_DcmClearDTC() on next Dcm_MainFunction call again. It is up to the DCM to send NRC 78 to respect the response behaviour⌋()

[SWS_Dcm_00707] ⌈ In case Dem_DcmClearDTC() returns DEM_CLEAR_FAILED, the DCM shall send a negative response 0x22 – conditionsNotCorrect.⌋()

[SWS_Dcm_00708] ⌈ In case Dem_DcmClearDTC() returns DEM_CLEAR_WRONG_DTC, the DCM shall send a negative response 0x31 – requestOutOfRange.⌋()

[SWS_Dcm_00966] ⌈ In case Dem_DcmClearDTC() returns DEM_CLEAR_BUSY, the DCM shall send a negative response 0x22 – ConditionsNotCorrect.⌋()

Note: Dem_DcmClearDTC typically triggers further callbacks through the RTE. To indicate the respective call-tree for these runnables, a work-around is used: The Dcm triggers the DTC deletion using the Dem interface DcmIf (operation DcmClearDTC, refer to chapter 8.9) instead of a direct C call.

[SWS_Dcm_01060]⌈ In case `Dem_DcmClearDTC()` returns DEM_CLEAR_MEMORY_ERROR, the Dcm shall trigger a negative response with NRC 0x72 (GeneralProgrammingFailure).⌋() ⌋()

7.4.2.5 UDS Service 0x19 – Read DTC Information

[SWS_Dcm_00248]⌈ The DCM module shall implement the UDS Service 0x19.⌋(SRS_Diag_04010)

[SWS_Dcm_00739]⌈If a DEM_STATUS_PENDING value is returned from Dem_DcmGetStatusOfDTC, the DCM shall call the API (at least on each Dcm_MainFunction cycle) again as lon&g as DEM_STATUS_PENDING is returned⌋()

[SWS_Dcm_00740]⌈If a DEM_FILTERED_PENDING value is returned from Dem_DcmGetNextFilteredRecord or Dem_DcmGetNextFilteredDTCAndFDC, the DCM shall call the API (at least on each Dcm_MainFunction cycle) again as long as DEM_FILTERED_PENDING is returned⌋()

[SWS_Dcm_01043]⌈If DEM_WRONG_FILTER value is returned from Dem_ReturnSetFilterType, the Dcm module shall send a negative response with NRC 0x31 (Request out of range).⌋()

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:

- DTCSeverityMask

- DTCStatusMask

[SWS_Dcm_00376] ⌈ 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: ⌋(SRS_Diag_04010)

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see SWS_Dcm_00007)
DTCFormatIdentifier	value returned by `Dem_DcmGetTranslationType()`
DTCCount	value calculated according to SWS_Dcm_00293

[SWS_Dcm_00293] ⌈ When responding to UDS Service 0x19 with subfunction 0x01, 0x07, 0x11 or 0x12, the DCM module shall calculate the number of DTCs using `Dem_DcmGetNumberOfFilteredDTC()` after having set the DEM-filter with `Dem_DcmSetDTCFilter()` using the parameter values of the following table:

	reportNumberOfDTCByStatusMask	reportNumberOfDTCBySeverityMaskRecord	reportNumberOfMirrorMemoryDTCByStatusMask	reportNumberOfEmissionsRelatedOBDDTCByStatusMask
	0x01	0x07	0x11	0x12
DTCStatusMask	DTCStatusMask from request (see SWS_Dcm_00700)	DTCStatusMask from request (see SWS_Dcm_00700)	DTCStatusMask from request (see SWS_Dcm_00700)	DTCStatusMask from request (see SWS_Dcm_00700)
DTCKind	DEM_DTC_KIND_ALL_DTCS	DEM_DTC_KIND_ALL_DTCS	DEM_DTC_KIND_ALL_DTCS	DEM_DTC_KIND_EMISSION_REL_DTCS
DTCFormat	DEM_DTC_FORMAT_UDS	DEM_DTC_FORMAT_UDS	DEM_DTC_FORMAT_UDS	DEM_DTC_FORMAT_UDS
DTCOrigin	PRIMARY_MEMORY	PRIMARY_MEMORY	MIRROR_MEMORY	PRIMARY_MEMORY
FilterWithSeverity	NO	YES	NO	NO
DTCSeverityMask	Not relevant	DTCSeverityMask from request	Not relevant	Not relevant
FilterForFaultDetectionCounter	NO	NO	NO	NO

⌋(SRS_Diag_04010, BSW04058, BSW04067)

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.

[SWS_Dcm_00377]「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 SWS_Dcm_00007)
DTCAndStatusRecord	As defined in SWS_Dcm_00008 and SWS_Dcm_00378

[SWS_Dcm_00008]「On reception of a UDS Service 0x19 request with subfunction 0x02, 0x0A, 0x0F, 0x13 or 0x15 and if DTCStatusAvailabilityMask (see SWS_Dcm_00007) is equal to 0, the DCM module shall answer positively with 0 DTC.」()

[SWS_Dcm_00378]「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_DcmGetNextFilteredDTC()` after having configured the filter with `Dem_DcmSetDTCFilter()` using the parameter values of the following table:

	reportDTCByStatusMask	**reportSupportedDTCs**	**reportMirrorMemoryDTCByStatusMask**	**reportEmissionsRelatedOBDDTCByStatusMask**	**reportDTCWithPermanentStatus**
	0x02	0x0A	0x0F	0x13	0x15
DTCStatusMask	DTCStatusMask from request (see SWS_Dcm_00700)	0x00	DTCStatusMask from request (see SWS_Dcm_00700)	DTCStatusMask from request (see SWS_Dcm_00700)	0x00
DTCKind	DEM_DTC_KIND_ALL_DTCS	DEM_DTC_KIND_ALL_DTCS	DEM_DTC_KIND_ALL_DTCS	DEM_DTC_KIND_EMISSION_REL_DTCS	DEM_DTC_KIND_ALL_DTCS
DTCFormat	DEM_DTC_FORMAT_UDS	DEM_DTC_FORMAT_UDS	DEM_DTC_FORMAT_UDS	DEM_DTC_FORMAT_UDS	DEM_DTC_FORMAT_UDS
DTCOrigin	PRIMARY_MEMORY	PRIMARY_MEMORY	MIRROR_MEMORY	PRIMARY_MEMORY	PERMANENT_MEMORY
FilterWithSeverity	NO	NO	NO	NO	NO
DTCSeverityMask	Not relevant	Not relevant	Not relevant	Not relevant	Not relevant
FilterForFaultDetectionCounter	NO	NO	NO	NO	NO

」(SRS_Diag_04010, BSW04058, BSW04067)

Note:

- The DCM module can get an indication of the number of records that will be found using `Dem_DcmGetNextFilteredDTC()` by using `Dem_DcmGetNumberOfFilteredDTC()`. 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_DcmSetDTCFilter()`.

[**SWS_Dcm_00828**] ⌈ In case of paged buffer support is disabled, the Dcm module shall not insert zero-padded DTCs to the response of UDS Service 0x19 with subfunctions 0x02, 0x0A, 0x0F, 0x13 or 0x15.⌋()

When using paged buffer mechanism, in some case, it's possible that the number of DTC matching the filter change between the calculation of the total size, needed for the first page transmission, and the sending of the further pages. For this reason, the following requirement apply :

[SWS_Dcm_00587] ⌈In case of paged buffer support is enabled, The DCM shall limit the response size to the size calculated when sending the first page. If more DTCs match the filter after this sending, the additional DTCs shall not be considered.⌋()

[SWS_Dcm_00588] ⌈In case of paged buffer support is enabled,The DCM shall pad the response with the size calculated when sending the first page. If less DTC match the filter after this sending, the missing DTCs shall be padded with 0 value as defined in 15031-6.⌋()

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

[SWS_Dcm_00379] ⌈ 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 SWS_Dcm_00007)
DTCAndSeverityRecord	As defined in SWS_Dcm_00380

⌋()

[SWS_Dcm_00380] ⌈ When responding to UDS Service 0x19 with subfunction 0x08, the DCM module shall obtain the DTCAndSeverityRecords by repeatedly calling `Dem_DcmGetNextFilteredDTCAndSeverity()` after having configured the filter with `Dem_DcmSetDTCFilter()` using the parameter values of the following table:

	reportDTCBySeverityMaskRecord
DTCStatusMask	DTCStatusMask from request (see SWS_DCM_00700)
DTCKind	DEM_DTC_KIND_ALL_DTCS
DTCFormat	DEM_DTC_FORMAT_UDS
DTCOrigin	PRIMARY_MEMORY
FilterWithSeverity	YES
DTCSeverityMask	DTCSeverityMask from request
FilterForFaultDetectionCounter	NO

⌋(SRS_Diag_04010)

Note: The DCM module can get an indication of the number of records that will be found using `Dem_DcmGetNextFilteredDTCAndSeverity()` by using `Dem_DcmGetNumberOfFilteredDTC()`.

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

[SWS_Dcm_00381] ⌈ 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 SWS_Dcm_00007)

DTCAndSeverityRecord	DTCSeverityMask : use `Dem_DcmGetSeverityOfDTC()`
	DTCFunctionalUnit : use `Dem_DcmGetFunctionalUnitOfDTC()`
	DTCxxx : the given DTC of the request
	statusOfDTC : use `Dem_DcmGetStatusOfDTC()` with parameters value:
	`DTC: DTC from the request`
	`DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY`

」(SRS_Diag_04010)

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)

[SWS_Dcm_00297] 「 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 SWS_Dcm_00295
DTCExtendedDataRecordNumber	see SWS_Dcm_00296
DTCExtendedDataRecord	see SWS_Dcm_00296

」()

[SWS_Dcm_00295] 「 When responding to UDS Service 0x19 with subfunction 0x06 or 0x10, the DCM module shall calculate the statusOfDTC by calling `Dem_DcmGetStatusOfDTC()` with the following parameters:

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

」(SRS_Diag_04010, BSW04058)

[SWS_Dcm_00296] 「 When responding to UDS Service 0x19 with subfunction 0x06 or 0x10, the DCM module shall assemble the ExtendedDataRecords by calling `Dem_DcmGetExtendedDataRecordByDTC()` repeatedly (only If `Dem_DcmGetExtendedDataRecordByDTC()` returns DEM_RECORD_OK and BufSize different from 0 (not empty buffer) (where ExtendedDataNumber goes from 0x01 to 0xEF) with the following parameter values:

	reportDTCExtendedDataRecordByDTCNumber	reportMirrorMemoryDTCExtendedDataRecordByDTCNumber
	0x06	0x10
DTC	DTCMaskRecord from request	DTCMaskRecord from request
DTCOrigin	PRIMARY_MEMORY	MIRROR_MEMORY
ExtendedDataNumber	0x01 to 0xEF	0x01 to 0xEF

⌋(SRS_Diag_04010, BSW04058)
The DTCExtendedDataRecordNumber is equal to the ExtendedDataNumber, which is used an IN parameter in the DEM interface Dem_DcmGetExtendedDataRecordByDTC()

[SWS_Dcm_00478] ⌈ The DCM module shall obtain the size of the data returned by DEM in `Dem_DcmGetExtendedDataRecordByDTC()` call by using `Dem_DcmGetSizeOfExtendedDataRecordByDTC()`.⌋(SRS_Diag_04010)

To get the size of all extended datas (corresponding to the sum of the size of extended record number(s) and the size of extended record(s) data), the DCM module call `Dem_DcmGetSizeOfExtendedDataRecordByDTC()` with ExtendedDataNumber set to 0xFF.

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)

[SWS_Dcm_00474] ⌈ 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 SWS_Dcm_00475
DTCExtendedDataRecordNumber	see SWS_Dcm_00476
DTCExtendedDataRecord	see SWS_Dcm_00476

⌋()

[SWS_Dcm_00475] ⌈ When responding to UDS Service 0x19 with subfunction 0x06 or 0x10, the DCM module shall calculate the statusOfDTC by calling `Dem_DcmGetStatusOfDTC()` with the following parameters:

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

⌋(SRS_Diag_04010, BSW04058)

[SWS_Dcm_00476] ⌈ When responding to UDS Service 0x19 with subfunction 0x06 or 0x10, the DCM module shall assemble the ExtendedDataRecords by calling `Dem_DcmGetExtendedDataRecordByDTC()` repeatedly (only If `Dem_DcmGetExtendedDataRecordByDTC()` returns DEM_RECORD_OK and BufSize different from 0 (not empty buffer) (where ExtendedDataNumber goes from 0x90 to 0xEF) with the following parameter values:

	reportDTCExtendedDataRecordByDTCNumber	**reportMirrorMemoryDTCExtendedDataRecordByDTCNumber**
	0x06	0x10
DTC	DTCMaskRecord from request	DTCMaskRecord from request
DTCOrigin	PRIMARY_MEMORY	MIRROR_MEMORY
ExtendedDataNumber	0x90 to 0xEF	0x90 to 0xEF

⌋(SRS_Diag_04010, BSW04058)

The DTCExtendedDataRecordNumber is equal to the ExtendedDataNumber, which is used an IN parameter in the DEM interface Dem_DcmGetExtendedDataRecordByDTC().

As required in SWS_Dcm_00478, the DCM module shall obtain the size of the extended data record by using `Dem_DcmGetSizeOfExtendedDataRecordByDTC()`.

To get the size of all OBD related extended data (corresponding to the sum of the size of extended record number(s) and the size of extended record(s) data), the DCM module call Dem_DcmGetSizeOfExtendedDataRecordByDTC() with ExtendedDataNumber set to 0xFE.

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)

[SWS_Dcm_00386] ⌈ 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 SWS_Dcm_00295
DTCExtendedDataRecordNumber	DTCExtendedDataRecordNumber from request only if Dem_DcmGetExtendedDataRecordByDTC returns DEM_RECORD_OK and BufSize different from 0. Else

	see **[SWS_Dcm_00841**
DTCExtendedDataRecord	see SWS_Dcm_00382

」()

[SWS_Dcm_00841]「If `Dem_DcmGetExtendedDataRecordByDTC` returns DEM_RECORD_OK and BufSize 0 (empty buffer), the Dcm module shall omit the DTCExtendedDataRecordNumber for the related record in the response of service 0x19 0x06/0x10.」()

[SWS_Dcm_00382]「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_DcmGetExtendedDataRecordByDTC()` with the following parameter values.」(SRS_Diag_04010, BSW04058)

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

As required in SWS_Dcm_00478, the DCM module shall obtain the size of the extended data record by using `Dem_DcmGetSizeOfExtendedDataRecordByDTC()`.

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.

[SWS_Dcm_00300]「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 SWS_Dcm_00299

」()

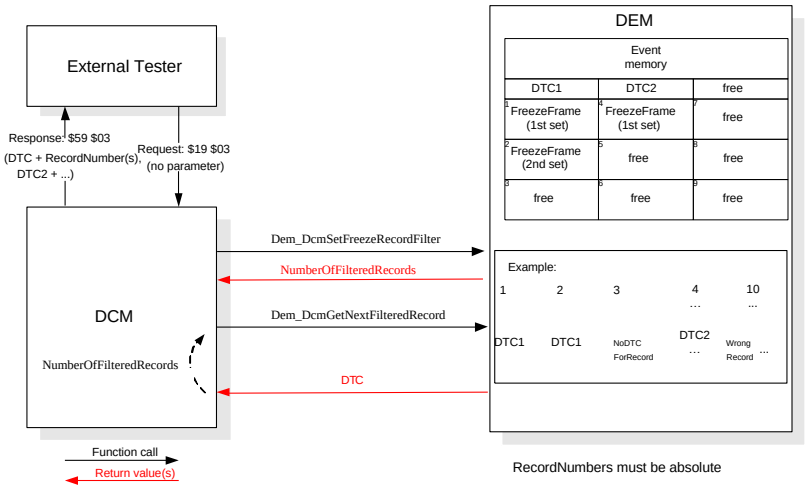


Figure 7: Request DTC Snapshot Record Identification

[SWS_Dcm_00298]⌈ The DSP submodule shall call `Dem_DcmSetFreezeFrameRecordFilter()` that returns the NumberOfFilteredRecords value with DTCFormat equal to DEM_DTC_FORMAT_UDS.⌋(SRS_Diag_04010)

[SWS_Dcm_00299]⌈ When responding to UDS Service 0x19 with subfunction 0x03, the DCM module shall obtain the consecutive DTCs and DTCSnapshotRecordNumbers by repeatedly calling `Dem_DcmGetNextFilteredRecord()`.⌋(SRS_Diag_04010)

When using paged buffer mechanism, in some case, it's possible that the number of DTC matching the filter change between the calculation of the total size, needed for the first page transmission, and the sending of the further pages. For this reason, the requirement SWS_Dcm_00587 and SWS_Dcm_00588 shall be considered for the implementation of this subservice.

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

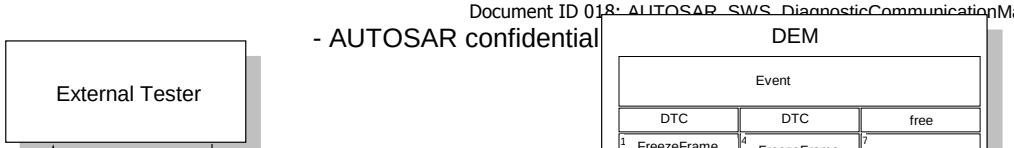


Figure 8: Request DTC Snapshot Record by Snapshot Record Number

[SWS_Dcm_00302] ⌈ 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 SWS_Dcm_00383
DTCSnapshotRecordNumber	The DTCSnapshotRecordNumber is contained in the output buffer from the `Dem_DcmGetFreezeFrameDataByDTC()` call. See SWS_Dcm_00384
DTCSnapshotRecordNumberOfIdentifiers/ DTCSnapshotRecord	As defined in SWS_Dcm_00384

⌋()

[SWS_Dcm_00387] ⌈ 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 SWS_Dcm_00383
DTCSnapshotRecordNumber	The DTCSnapshotRecordNumber is contained in the output buffer from the `Dem_DcmGetFreezeFrameDataByDTC()` call. See SWS_Dcm_00385

DTCSnapshotRecordNumberOfIdentifiers/ DTCSnapshotRecord	As defined in SWS_Dcm_00385

」()

[SWS_Dcm_00383] 「 When responding to UDS Service 0x19 with subfunction 0x04, the DCM module shall obtain the status of the DTC by calling `Dem_DcmGetStatusOfDTC()` with the following parameters:
DTC: DTC from the request
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY」(SRS_Diag_04010, BSW04058)

[SWS_Dcm_00384] 「 Upon reception of UDS Service 0x019 with subfunction 0x04 and DTCSnapshotRecordNumber not 0xff, DCM module shall obtain the "DTCSnapshotRecordNumberOfIdentifiers" and the FreezeFrame by calling `Dem_DcmGetFreezeFrameDataByDTC()` with the following parameter values:
DTC: DTCMaskRecord from the request in UDS format
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY
RecordNumber: DTCSnapshotRecordNumber from the request」(SRS_Diag_04010, BSW04058)

[SWS_Dcm_00385] 「 Upon reception of UDS Service 0x19 with subfunction 0x04 and DTCSnapshotRecordNumber 0xff, the DCM module shall cycle through all FreezeFrame numbers from 0x00 to 0xfe and obtain the corresponding "DTCSnapshotRecordNumberOfIdentifiers" and FreezeFrame by calling `Dem_DcmGetFreezeFrameDataByDTC()` with the following parameter values:
DTC: DTCMaskRecord from the request in UDS format
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY
RecordNumber: value from 0x00 -> 0xFE」(SRS_Diag_04010, BSW04058)

[SWS_Dcm_00441] 「 The DCM module shall obtain the size of the data returned by DEM in `Dem_DcmGetFreezeFrameDataByDTC()` call by using `Dem_DcmGetSizeOfFreezeFrameByDTC()`.」(SRS_Diag_04010, BSW04079)

To get the size of all FreezeFrame data, the DCM module call `Dem_DcmGetSizeOfFreezeFrameByDTC()` with RecordNumber set to 0xFF.

7.4.2.5.10 UDS Service 0x19 with subfunction 0x05 (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

Due to DEM limitation, the diagnostic service $19 05 is limited to the OBD legislative freeze frame.

[SWS_Dcm_00632] ⌈ On reception of service 0x19 with subfunction 0x05, if the record number of the diagnostic request is different from 0x00, the DCM module shall send a negative response with NRC 0x31 (request out of range).⌋()

[SWS_Dcm_00574] ⌈ When sending a positive response to UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber is 0x00, the DCM module shall use the following data in the response message:

Parameter name	Value
DTCSnapshotRecordNumber	DTCSnapshotRecordNumber from request (0x00)
DTCAndStatusRecord	DTC according to SWS_Dcm_00388, statusOfDTC according to SWS_Dcm_00389
DTCSnapshotRecordNumberOfIdentifiers / DTCSnapshotRecord	As defined in SWS_Dcm_00388

⌋()

[SWS_Dcm_00388] ⌈ When responding to UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber is 0x00, the DCM module shall obtain the DTC, the DTCSnapshotRecordNumberOfIdentifiers and the DTCSnapshotRecord by calling `Dem_DcmGetOBDFreezeFrameData ()` with the following parameter value:

DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY
BufSize: Data size available in the transmit buffer
DestBuffer: Address where the DEM shall copy the freeze frame data⌋(SRS_Diag_04010, BSW04058)

[SWS_Dcm_00389] ⌈ When responding to UDS Service 0x19 with subfunction 0x05 and DTCSnapshotRecordNumber is 0x00, the DCM module shall obtain the status of the DTC by calling `Dem_DcmGetStatusOfDTC()` with the following parameters:
DTC: DTC as defined in SWS_Dcm_00388
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY⌋(SRS_Diag_04010, BSW04058)

The function `Dem_DcmGetOBDFreezeFrameData ()` copy the DTCSnapshotRecord data and DTCSnapshotRecordNumberOfIdentifiers, as defined by UDS for the response message to Service 0x19 subfunction 0x05, to the buffer provided by the DCM module.
The DCM module can know the size of data returned in `Dem_DcmGetOBDFreezeFrameData ()` using parameter BufSize.

7.4.2.5.11 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.

[SWS_Dcm_00392] ⌈ 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 SWS_Dcm_00007)
DTCAndStatusRecord	The DTC is obtained according to SWS_Dcm_00466, the StatusOfDtc is obtained according to SWS_Dcm_00393

⌋()

[SWS_Dcm_00393] ⌈ 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_DcmGetStatusOfDTC()` with the following parameter values:
DTC: the DTC value as defined in SWS_Dcm_00466
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY⌋(SRS_Diag_04010, BSW04058)

[SWS_Dcm_00466] ⌈ For the purpose of responding to UDS Service 0x19 with subfunctions 0x0B, 0x0C, 0x0D or 0x0E, the DCM shall obtain the DTC with `Dem_DcmGetDTCByOccurrenceTime()` using the parameter values of the following table:

	reportFirstTestFailedDTC	**reportFirstConfirmedDTC**	**reportMostRecentTestFailedDTC**	**reportMostRecentConfirmedDTC**
	0x0B	**0x0C**	**0x0D**	**0x0E**
DTCRequest	DEM_FIRST_FAILED_DTC	DEM_FIRST_DET_CONFIRMED_DTC	DEM_MOST_RECENT_FAILED_DTC	DEM_MOST_REC_DET_CONFIRMED_DTC

⌋(SRS_Diag_04010)

[SWS_Dcm_00766] ⌈If the Dcm received DEM_OCCURR_NOT_AVAILABLE by calling `Dem_DcmGetDTCByOccurrenceTime` it shall response with a positive and empty response⌋()

7.4.2.5.12 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.

[SWS_Dcm_00464] ⌈ 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_DcmGetNextFilteredDTCAndFDC()`
DTCFaultDetectionCounter	The DTCFaultDetectionCounter is obtained according from the call to `Dem_DcmGetNextFilteredDTCAndFDC()`

⌋(SRS_Diag_04010)

[SWS_Dcm_00465] ⌈ 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_DcmGetNextFilteredDTCAndFDC()` after having configured the filter with `Dem_DcmSetDTCFilter()` using the parameter values of the following table:

DTCStatusMask	0x00
DTCKind	DEM_DTC_KIND_ALL_DTCS
DTCFormat	DEM_DTC_FORMAT_UDS
DTCOrigin	PRIMARY_MEMORY
FilterWithSeverity	NO
DTCSeverityMask	Not relevant
FilterForFaultDetectionCounter	YES

⌋(SRS_Diag_04010, BSW04058)

[SWS_Dcm_00681] ⌈ The DCM module shall obtain the number of records that will be found using `Dem_DcmGetNextFilteredDTCAndFDC()` by using `Dem_DcmGetNumberOfFilteredDTC().`⌋()

[SWS_Dcm_00519] ⌈ The calls to `Dem_DcmSetDTCFilter()` with parameter FilterForFaultDetectionCounter set to YES shall be done in the context of the Dcm_MainFunction()⌋(SRS_Diag_04010)

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

When using paged buffer mechanism, in some case, it’s possible that the number of DTC matching the filter change between the calculation of the total size, needed for the first page transmission, and the sending of the further pages. For this reason, the

requirement SWS_Dcm_00587 and SWS_Dcm_00588 shall be considered for the implementation of this subservice.

7.4.2.6 UDS Service 0x22 - ReadDataByIdentifier

[SWS_Dcm_00253] ⌈ The DCM module shall implement the UDS Service ReadDataByIdentifier (0x22) ⌋()

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

[SWS_Dcm_00438] ⌈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*** and ***DcmDspDidRange***) If none of the requested DIDs is supported, the DCM module shall send NRC 0x31 (Request out of range). ⌋()

[SWS_Dcm_00651] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22) with DID in the range 0xF200 to 0xF3FF, the DCM module shall check if the DID has been dynamically configured. If not, the DCM module shall send NRC 0x31 (Request out of range). ⌋()

[SWS_Dcm_00652] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22) with DID in the range 0xF200 to 0xF3FF, if verification has been successfully done (see **SWS_Dcm_00651**) and the dynamic DID has been defined with a DID source (See **SWS_Dcm_00646**), the DCM module shall use the configuration of this DID source to read the data. ⌋()

[SWS_Dcm_00864] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22) with DID in the range 0xF200 to 0xF3FF, if verification has been successfully done (see **SWS_Dcm_00651**) and the dynamic DID has been defined with a DID source (See **SWS_Dcm_00646**), the DCM module shall do the session, security and mode dependencies checks for all source DIDs in case the configuration parameter ***DcmDspDDDIDcheckPerSourceDID*** is set to TRUE. ⌋()

[SWS_Dcm_00865] ⌈ In case the configuration parameter ***DcmDspDDDIDcheckPerSourceDID*** is set to FALSE, there is no session, security or mode dependencies check for the source DIDs. ⌋()

Note: In case there is a need to validate the session or security dependencies always, the DDDID should be cleared by any security and session transitions.

[SWS_Dcm_00653] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22) with DID in the range 0xF200 to 0xF3FF, if verification has been successfully done (see **SWS_Dcm_00651**) and the dynamic DID has been defined with a memory address (See **SWS_Dcm_00646**), the DCM module shall use the callout Dcm_ReadMemory to read the data.⌋()

[SWS_Dcm_00561] ⌈ If a DID is set as unused (DcmDspDidUsed set to FALSE), the DCM shall consider the DID as not supported (according to SWS_Dcm_00438)⌋()

[SWS_Dcm_00433] ⌈ 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 none of the DID has a Read access, the DCM module shall send NRC 0x31 (Request out of range).⌋()

[SWS_Dcm_00434] ⌈ 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 none of the DID can be renden in the current session, the DCM module shall send a NRC 0x31 (Request out of Range).⌋()

[SWS_Dcm_00435] ⌈ 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).⌋()

[SWS_Dcm_00819] ⌈ 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 mode condition (according to the configuration parameter ***DcmDspDidReadModeRuleRef***). If not, the DCM module shall send the calculated negative response of the referenced DcmModeRule.⌋()

[SWS_Dcm_00440] ⌈ If the requested DID references other DID using ***DcmDspDidRef***, the DCM module shall process the verification and the reading of every referenced DID and concatenate the response data without any gaps based on the sequence in the configuration.⌋()

[constr_6022] ***DcmDspDidRef*** shall not be a mixture between references to DIDs or the DID definitions itself「 ***DcmDspDid*** container shall either have ***DcmDspDidRef*** parameters or ***DcmDspDidSignal*** containers, but not both.」()

[constr_6023] ***DcmDspDidRef*** shall not reference the same DID reference twice「 DcmDspDid container shall not include the same ***DcmDspDidRef*** parameters more than once. 」()

7.4.2.6.1 Non-OBD DID

[SWS_Dcm_00578]「 On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID outside the OBD range (F400-F8FF), after all verification (see SWS_Dcm_00433, SWS_Dcm_00434 and SWS_Dcm_00435), If the data is configured as a “ECU signal” of the IoHwAb (parameter DcmDspDataUsePort), the DCM shall call the Api IoHwAb_Dcm_Read<EcuSignalName >() (parameter ***DcmDspDataReadEcuSignal***) to get the Data. In this case, the requirements SWS_Dcm_00439, SWS_Dcm_00436 and SWS_Dcm_00437 shall not apply.」()

[SWS_Dcm_00439]「 On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID outside the OBD range (F400-F8FF), the DCM module shall request the application if the DID can be read by calling the configured function (if parameter ***DcmDspDataUsePort*** set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC*;* see configuration parameter ***DcmDspDataConditionCheckReadFnc***) on each data of the DID or call the associated *ConditionCheckRead* operation (if parameter ***DcmDspDataUsePort*** set to *USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER*). If not (one function returns E_NOT_OK) , the DCM module shall send a negative response with NRC set to value from the parameter “ErrorCode” of ***DcmDspDataConditionCheckReadFnc*** function or **ConditionCheckRead** operation.」()

[SWS_Dcm_00436]「 On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID outside the OBD range (F400-F8FF), the DCM module shall check if the datas of the DID have a fixed data length (see configuration parameter ***DcmDspDataFixedLength***): if not (Parameter set to False) the DCM module shall call the configured function (if parameter ***DcmDspDataUsePort*** set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC; see configuration parameter ***DcmDspDataReadDataLengthFnc***) to get the data length in byte or call the associated *ReadDataLength* operation (if parameter ***DcmDspDataUsePort*** set to *USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER*).」()

[SWS_Dcm_00437] ⌈ After all verification (see SWS_Dcm_00433, SWS_Dcm_00434, SWS_Dcm_00435 and SWS_Dcm_00436) the DCM module shall get for every requested DID outside the OBD range (F400-F8FF), all the data values by calling all the configured function (if parameter ***DcmDspDataUsePort*** set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC; see configuration parameter ***DcmDspDataReadFnc***) or call all the associated *ReadData* operations (if parameter ***DcmDspDataUsePort*** set to *USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER*) or read all the associated *SenderReceiver* interfaces (if parameter ***DcmDspDataUsePort*** set to *USE_DATA_SENDER_RECEIVER*). ⌋()

[SWS_Dcm_00560] ⌈ If the data is configured as a BlockId of the NvRam (parameter *DcmDspDataUsePort*), the DCM shall call the Api NvM_ReadBlock() with the BlockId (parameter *DcmDspDataBlockIdRef*) ⌋()

[SWS_Dcm_00638] ⌈ To serialize the required AUTOSAR data types (signed- and unsigned integer) into the response message of ReadDataByIdentifier responses the target endianness configured in ***DcmDspDataEndianness*** shall be considered for ***DcmDspData*** elements having ***DcmDspDataUsePort*** set to {USE_DATA_SENDER_RECEIVER, USE_ECU_SIGNAL}. In case ***DcmDspDataEndianness*** is not present, the ***DcmDspDataDefaultEndianness*** shall be used instead. ⌋()

7.4.2.6.2 OBD DID

[SWS_Dcm_00481] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID inside the OBD range (F400-F4FF), the DCM module shall get the DID value as defined for OBD Service $01 (See SWS_Dcm_00407 , SWS_Dcm_00408) ⌋()

[SWS_Dcm_00482] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID inside the OBD Monitor range (F600-F6FF), the DCM module shall get the DID value as defined for OBD Service $06 (See **SWS_Dcm_00957** and **SWS_Dcm_00958**) ⌋()

[SWS_Dcm_00483] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID inside the OBD InfoType range (F800-F8FF), the DCM module shall get the DID value as defined for OBD Service $09 (See SWS_Dcm_00422 , SWS_Dcm_00423) ⌋()

7.4.2.7 UDS Service 0x24 - ReadScalingDataByIdentifier

[SWS_Dcm_00258] ⌈ The DCM module shall implement the UDS Service ReadScalingDataByIdentifier (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***)
- For every data of the DID :
  - The function GetScalingInformation (see configuration parameters ***DcmDspDataGetScalingInfoFnc*** and ***DcmDspDataUsePort***)
  - The length of the ScalingInfo returned by the GetScalingInformation function (see configuration parameter ***DcmDspDataScalingInfoSize***)

[SWS_Dcm_00394] ⌈ On reception of a request for UDS Service ReadScalingByIdentifier, the DCM module shall call every function `Xxx_GetScalingInformation()` configured for everay data of the DID received in the request and return the data received in the response ⌋()

7.4.2.8 UDS Service 0x27 - SecurityAccess

[SWS_Dcm_00252] ⌈ The DCM module shall implement the UDS Service SecurityAccess (0x27) ⌋(BSW04005)

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.

[SWS_Dcm_00321] ⌈ 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). ⌋()

[SWS_Dcm_00323] ⌈ If the requested subfunction value is configured and a service with subfunction type “requestSeed “(= 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. ⌋()

[SWS_Dcm_00324] [In the other case than the one described in **[SWS_Dcm_00323** (access type is not active or “send key” request), if ***DcmDspSecurityUsePort*** is set to USE_ASYNCH_CLIENT_SERVER or USE_SYNCH_CLIENT_SERVER, 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).⌋()

[SWS_Dcm_00862] ⌈ On reception of the UDS Service SecurityAccess (0x27) with subfunction type “requestSeed” and if the requested access type is not already active, the DCM module shall request a seed by calling the configured Xxx_GetSeed() function (if the configuration parameter ***DcmDspSecurityUsePort*** is set to USE_SYNCH_FNC or USE_ASYNCH_FNC, refer to configuration parameter ***DcmDspSecurityGetSeedFnc***). ⌋()

[SWS_Dcm_00863] ⌈ On reception of the UDS Service SecurityAccess (0x27 with subfunction type “sendKey”, if the requested access type is not already active and if the “request seed” for the related access type was executed successfully, the DCM module shall request the result of a key comparison by calling the configured Xxx_CompareKey() function (if the configuration parameter ***DcmDspSecurityUsePort*** is set to USE_SYNCH_FNC or USE_ASYNCH_FNC, refer to configuration parameter ***DcmDspSecurityCompareKeyFnc***). ⌋()

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 ***DcmDspSecurityNumAttDelay***), and
- InvalidKey (NRC 0x35), when invalid key is send at “send key”.

[SWS_Dcm_00325] ⌈ If no errors are detected, the DSP submodule shall set the new access type with `DslInternal_SetSecurityLevel()`.⌋()

7.4.2.9 UDS Service 0x2A - ReadDataByPeriodicIdentifier

[SWS_Dcm_00254] ⌈ The DSP submodule shall implement the UDS Service ReadDataByPeriodicIdentifier (0x2A)⌋()

[SWS_Dcm_00721]: ⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), for every requested periodicDID the DCM module shall check if the periodicDID can be read in the current session (see configuration parameter ***DcmDspDidReadSessionRef***). If none of the periodicDID can be read in the current session, the DCM module shall send a NRC 0x31 (Request out of Range).⌋()

[SWS_Dcm_00722]: ⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), for every requested periodicDID the DCM module shall check if the periodicDID can be read in the current security level (see configuration parameter ***DcmDspDidReadSecurityLevelRef***). If not, the DCM module shall send NRC 0x33 (Security access denied).⌋()

[SWS_Dcm_00820] ⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), for every requested periodicDID the DCM module shall check if the periodicDID can be read in the current mode condition (see configuration parameter ***DcmDspDidReadModeRuleRef***). If not, the DCM module shall send the calculated negative response code of the reference ***DcmModeRule***⌋()

[SWS_Dcm_00716] ⌈ To serialize the required AUTOSAR data types (signed- and unsigned integer) into the response message of ReadDataByPeriodicIdentifier responses the target endianness configured in ***DcmDspDataEndianness*** shall be considered for ***DcmDspData*** elements having ***DcmDspDataUsePort*** set to {USE_DATA_SENDER_RECEIVER, USE_ECU_SIGNAL}. In case ***DcmDspDataEndianness*** is not present, the ***DcmDspDataDefaultEndianness*** shall be used instead.⌋()

[SWS_Dcm_00843] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier(0x2A),the DCM module shall check if the periodicDataIdentifiers requested in a single request do not exceed the configured DcmDspMaxPeriodicDidToRead. Otherwise (in case the number of elements are exceeded) the DCM module shall send a NRC 0x31 (Request out of Range).⌋()

[SWS_Dcm_00851] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier(0x2A), the DCM module shall check if the requested periodicDataIdentifiers in a single request could all be added to the scheduler considering the size of the scheduler (see configuration parameter ***DcmDspMaxPeriodicDidScheduler***). Otherwise (in case the requested periodicDataIdentifiers can not be added to the scheduler) the DCM module shall send a NRC 0x31 (Request out of Range).⌋()

7.4.2.10 UDS Service 0x2C - DynamicallyDefineDataIdentifier

[SWS_Dcm_00259] ⌈ The DSP submodule shall implement the DynamicallyDefineDataIdentifier (service 0x2C, diagnostic data access) of the Unified Diagnostic Services. ⌋()

The DynamicallyDefineDataIdentifier service is implemented internally in DCM module.

[SWS_Dcm_00866] ⌈ If ***DcmDDDIDStorage*** configuration parameter is set to FALSE, the DCM shall initialize all DDDIDs as not present at power-up (Dcm_Init). ⌋()

[SWS_Dcm_00867] ⌈ If ***DcmDDDIDStorage*** configuration parameter is set to TRUE, the DCM shall restore the DDDID definition from NvM at power-up (Dcm_Init). ⌋()

[SWS_Dcm_00868] ⌈ If ***DcmDDDIDStorage*** configuration parameter is set to TRUE, the DCM shall trigger the storage of the DDDID definition to NvRam (via NvM_SetRamBlockStatus). ⌋()

[SWS_Dcm_00646] ⌈ On reception of service DynamicallyDefineDataIdentifier with subservice defineByIdentifier or defineByMemoryAddress, the DCM module shall configure this new DID with associated information receive from the diagnostic request: Memory address and memory length or DID source, position and size. ⌋()

[SWS_Dcm_00861] ⌈ On reception of the UDS Service DynamicallyDefineDataIdentifier (0x2C), the DCM module shall check if the DDDID will not exceed the configured parameter value ***DcmDspDDDIDMaxElements***. Otherwise (in case the number of elements will be exceeded) the DCM module shall send a NRC 0x31 (Request out of Range). ⌋()

[SWS_Dcm_00854] ⌈On reception of the UDS Service DynamicallyDefineDataIdentifier (0x2C) with subservice defineByMemoryAddress, the DCM shall check if the requested AddressAndLengthFormatIdentifier is supported (refer to configuration parameter ***DcmDspSupportedAddressAndLengthFormatIdentifier***), Otherwise the NRC 0x31 (requestOutOfRange) shall be responded. In case the container AddressAndLengthFormatIdentifier is not present, the DCM shall accept all possible AddressAndLengthFormatIdentifiers. ⌋()

[SWS_Dcm_00647] ⌈ On reception of service DynamicallyDefineDataIdentifier with subservice clearDynamicallyDefinedDataIdentifier, the DCM module shall remove the configuration of this DID.⌋()

[SWS_Dcm_00723]: ⌈On reception of the UDS Service DynamicallyDefineDataIdentifier (0x2C),the DCM module shall check if the DDDID can be defined in the current session (see configuration parameter ***DcmDspDidReadSessionRef***). If not, the DCM module shall send a NRC 0x31 (Request out of Range).⌋()

[SWS_Dcm_00724]: ⌈On reception of the UDS Service DynamicallyDefineDataIdentifier (0x2C), the DCM module shall check if the DDDID can be defined in the current security level (see configuration parameter ***DcmDspDidReadSecurityLevelRef***). If not, the DCM module shall send NRC 0x33 (Security access denied). ⌋()

[SWS_Dcm_00725]: ⌈On reception of the UDS Service DynamicallyDefineDataIdentifier (0x2C), the DCM module shall check if the requested Source-DIDs are supported in the current session (see configuration parameter of referenced DID ***DcmDspDidReadSessionRef***)). If not, the DCM module shall send a NRC 0x31 (Request out of Range).⌋()

[SWS_Dcm_00726]: ⌈On reception of the UDS Service DynamicallyDefineDataIdentifier (0x2C), the DCM module shall check if the requested Source-DID or the memoryRange are supported in the current security level (see configuration parameter of referenced DID ***DcmDspDidReadSecurityLevelRef*** or memoryRange ***DcmDspReadMemoryRangeSecurityLevelRef***). If not, the DCM module shall send a NRC 0x33 (Security access denied).⌋()

[SWS_Dcm_00821] ⌈On reception of the UDS Service DynamicallyDefineDataIdentifier (0x2C), the DCM module shall check if the requested Source-DID or the memoryRange are supported in the current mode condition (see configuration parameter of referenced DID ***DcmDspDidReadModeRuleRef*** or memoryRange ***DcmDspReadMemoryRangeModeRuleRef***). If not, the DCM module shall send the calculated negative response code of the referenced DcmModeRule.⌋()

In case of memory address(es), on reception of ReadDataByIdentifier or ReadDataByPeriodicIdentifier request for a dynamically defined DID, the DCM will

use the callout Dcm_ReadMemory for all contained memory addresses to access the data.

[SWS_Dcm_01051] ⌈On reception of the UDS Service DynamicallyDefineDataIdentifier (0x2C), if the request message contains different MemoryIdValue compare to the configured values in ***DcmDspMemoryIdInfo*** container, the Dcm shall send a NRC 0x31 (Request out of Range).⌋()

In case of DID source(s), on reception of ReadDataByIdentifier or ReadDataByPeriodicIdentifier request for a dynamically defined DID, the DCM will use the configuration of the contained DIDs to read the data.

7.4.2.11 UDS Service 0x2E - WriteDataByIdentifier

[SWS_Dcm_00255] ⌈ 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.
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***)
- For every data of the DID:
  - The function WriteData to be used for this data (see configuration parameters ***DcmDspDataWriteFnc*** and ***DcmDspDataUsePort***)

[SWS_Dcm_00467] ⌈ On reception of the UDS Service WriteDataByIdentifier (0x2E), the DCM module shall check if the DID is supported (see configuration parameter ***DcmDspDid*** and ***DcmDspDidRange***) If not, the DCM module shall send NRC 0x31 (Request out of range) .⌋()

[SWS_Dcm_00562] ⌈ If a DID is set as unused (DcmDspDidUsed set to FALSE), the DCM shall consider the DID as not supported (according to SWS_Dcm_00467)⌋()

[SWS_Dcm_00468] ⌈ 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).⌋()

[SWS_Dcm_00469] ⌈ On reception of the UDS Service WriteDataByIdentifier (0x2E), the DCM module shall check if the DID can be written in the current session (see configuration parameter ***DcmDspDidWriteSessionRef***). If not, the DCM module shall send a NRC 0x31 (Request Out of Range).⌋()

[SWS_Dcm_00470] ⌈ On reception of the UDS Service WriteDataByIdentifier (0x2E), the DCM module shall check if the DID can be written in the current security level (see configuration parameter ***DcmDspDidWriteSecurityLevelRef***). If not, the DCM module shall send NRC 0x33 (Security access denied).⌋()

[SWS_Dcm_00822] ⌈On reception of the UDS Service WriteDataByIdentifier (0x2E), the DCM module shall check if the DID can be written in the current mode condition (see configuration parameter ***DcmDspDidWriteModeRuleRef***). If not, the DCM module shall send the calculated negative response code of the referenced ***DcmModeRule***. ⌋()

[SWS_Dcm_00473] ⌈ On reception of the UDS Service WriteDataByIdentifier (0x2E), if all data of the DID have fixed length (See configuration parameter ***DcmDspDataFixedLength***), the DCM module shall check if the received data length corresponds to the DID data length (addition of all DcmDspDataSize)⌋()

[SWS_Dcm_00395] ⌈ After all verification (see SWS_Dcm_00467, SWS_Dcm_00468, SWS_Dcm_00469, SWS_Dcm_00470, SWS_Dcm_00473) the DCM module shall write all the data of the DID by calling the configured functions (if parameter ***DcmDspDataUsePort*** set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC; see configuration parameter ***DcmDspDataWriteFnc***) or call all the associated *WriteData* operations (if parameter ***DcmDspDataUsePort*** set to *USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER*) or write all the associated *SenderReceiver* interfaces (if parameter ***DcmDspDataUsePort*** set to *USE_DATA_SENDER_RECEIVER*) with the following parameter values:
Data: the dataRecord form the request
DataLength: the number of bytes in the dataRecord (get from the configuration if the data has fixed length (See configuration parameter ***DcmDspDataFixedLength***) or from the diagnostic request length if the data has dynamic length)⌋()

[SWS_Dcm_00541] ⌈ If the data is configured as a BlockId of the NvRam (parameter ***DcmDspDataUsePort*** set to USE_BLOCK_ID), the Dcm shall :

1. Request NvM_SetBlockLockStatus(<***DcmDspDataBlockIdRef***>, FALSE), to temporarily unlock the NvM Block (It might be locked by executing this procedure before).

2. Request NvM_WriteBlock(<***DcmDspDataBlockIdRef*** >, <DataBuffer>) with BlockId corresponding to the configuration parameter ***DcmDspDataBlockIdRef***)

3. Poll for completion of write request, using NvM_GetErrorStatus()

4.a) On success (NVM_REQ_OK), the DCM shall issue NvM_SetBlockLockStatus(<***DcmDspDataBlockIdRef*** >, TRUE) (to lock the NvM block against further updates from the application) and send a positive response message.

4.b) Otherwise (on any NvM failure) the DCM module shall trigger a negative response with NRC 0x10 (GeneralReject).」()

[SWS_Dcm_00620]「 The data of a DID can have dynamic datalength (See configuration parameter ***DcmDspDataFixedLength***) only if this DID contains only one data. 」()

In other case the DCM won't be able to split the data from the request

[SWS_Dcm_00639]「 To serialize the required AUTOSAR data types (signed- and unsigned integer) into the response message of WriteDataByIdentifier responses the target endianness configured in ***DcmDspDataEndianness*** shall be considered for ***DcmDspData*** elements having ***DcmDspDataUsePort*** set to {USE_DATA_SENDER_RECEIVER}. In case ***DcmDspDataEndianness*** is not present, the ***DcmDspDataDefaultEndianness*** shall be used instead.」()

[constr_6018]「DcmDspData elements used in service 0x2E shall not have DcmDspDataUsePorts set to USE_ECU_SIGNAL」()

7.4.2.12 UDS Service 0x2F - InputOutputControlByIdentifier

[SWS_Dcm_00256]「 The DCM module shall implement the UDS Service InputOutputControlByIdentifier (0x2F).」()

When using Service 0x2F, the request of the tester contains a 2-byte DID.

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

- The 2-bytes DID (see configuration parameter ***DcmDspDidIdentifier***)
- For every data of the DID :
  - The function `Xxx_ReturnControlToECU()` for this data (see configuration parameters ***DcmDspDataReturnControlToEcuFnc*** and ***DcmDspDataUsePort***)
  - The function `Xxx_ResetToDefault()` for this data (see configuration parameters ***DcmDspDataResetToDefaultFnc*** and ***DcmDspDataUsePort***)
  - The function `Xxx_FreezeCurrentState()` for this DID (see configuration parameters ***DcmDspDataFreezeCurrentStateFnc*** and ***DcmDspDataUsePort***)
  - The function `Xxx_ShortTermAdjustment()` for this DID (see configuration parameters ***DcmDspDataShortTermAdjustmentFnc*** and ***DcmDspDataUsePort***)
  - The sizes of the control record used in the function `Xxx_ShortTermAdjustment()` (see configuration parameter and ***DcmDspDataSize***)

[SWS_Dcm_00579] ⌈ The DCM shall support the following InputOutputControlParameter definitions:

Hex	Description
00	returnControlToECU
01	resetToDefault
02	freezeCurrentState
03	shortTermAdjustment

⌋()

[SWS_Dcm_00563] ⌈ 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).⌋()

[SWS_Dcm_00564] ⌈ If a DID is set as unused (DcmDspDidUsed set to FALSE), the DCM shall consider the DID as not supported (according to SWS_Dcm_00563)⌋()

[SWS_Dcm_00565] ⌈ On reception of the UDS Service InputOutputControlByIdentifier (0x2F), the DCM module shall check if the DID has a Control access configured (see configuration parameter ***DcmDspDidControl*** in ***DcmDspDidAccess***). If not, the DCM module shall send NRC 0x31 (Request out of range).⌋()

[SWS_Dcm_00566] ⌈ On reception of the UDS Service InputOutputControlByIdentifier (0x2F), the DCM module shall check if the DID can be control in the current session (see configuration parameter ***DcmDspDidControlSessionRef***). If not, the DCM module shall send a NRC 0x31 (Request Out of Range).⌋()

[SWS_Dcm_00567] ⌈ On reception of the UDS Service InputOutputControlByIdentifier (0x2F), the DCM module shall check if the DID can be control in the current security level (see configuration parameter ***DcmDspDidControlSecurityLevelRef***). If not, the DCM module shall send NRC 0x33 (Security access denied).⌋()

[SWS_Dcm_00823] ⌈On reception of the UDS Service InputOutputControlByIdentifier (0x2F), the DCM module shall check if the DID can be control in the current mode condition (see configuration parameter ***DcmDspDidControlModeRuleRef***). If not, the DCM module shall send the calculated negative response code of the referenced DcmModeRule.⌋()

[SWS_Dcm_00580] ⌈ On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) , if all verifications have been successfully done (see SWS_Dcm_00563, SWS_Dcm_00565, SWS_Dcm_00566, SWS_Dcm_00567) and if the data is configured as a “ECU signal” of the IoHwAb (parameter *DcmDspDataUsePort*), the DCM shall call the Api IoHwAb_Dcm_<symbolic name of ECU signal (parameter DcmDspDataEcuSignal)>() with InputOutputControlParameter for the ‘action’ parameter and in case of InputOutputControlParameter is set to ‘shortTermAdjustment’ the signal value for the “signal” parameter. In this case the requirements SWS_Dcm_00396**,** SWS_Dcm_00397**,** SWS_Dcm_00398 and SWS_Dcm_00399 doesn’t apply.⌋()

[SWS_Dcm_00581] ⌈ In case of more than one supported I/O signal per DataIdentifier, the DCM shall internally consider the parameter **controlEnableMaskRecord** and control only the included signals in the request message. ⌋()

The controlMask information are not forwarded to the SW-Cs (DCM internal usage only).

[SWS_Dcm_00680] ⌈ The following mapping shall be used for the controlMask management: First bit of controlMask maps to first DID data element) ⌋()

The **controlEnableMaskRecord** is only present, if the DataIdentifer supports more than one signal.

[SWS_Dcm_00396] ⌈ On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to returnControlToEcu, if all verifications have been successfully done (see SWS_Dcm_00563, SWS_Dcm_00565, SWS_Dcm_00566, SWS_Dcm_00567), the DCM module shall invoke all impacted configured function of the controlEnableMaskRecord (if parameter ***DcmDspDataUsePort*** set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC; see configuration parameter ***DcmDspDataReturnControlToEcuFnc***). Alternatively call all the associated *ReturnControlToECU* operations (if parameter ***DcmDspDataUsePort*** set to *USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER*) for every data of the DID received in the request. ⌋()

[SWS_Dcm_00397] ⌈ On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to resetToDefault, if all verifications have been successfully done (see SWS_Dcm_00563, SWS_Dcm_00565, SWS_Dcm_00566, SWS_Dcm_00567), the DCM module shall invoke all impacted configured function of the controlEnableMaskRecord (if parameter ***DcmDspDataUsePort*** set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC; see configuration parameter ***DcmDspDataResetToDefaultFnc***). Alternatively call all the associated *ResetToDefault* operations (if parameter ***DcmDspDataUsePort*** set to *USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER*) for every data of the DID received in the request. ⌋()

[SWS_Dcm_00398]「 On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to freezeCurrentState, if all verifications have been successfully done (see SWS_Dcm_00563, SWS_Dcm_00565, SWS_Dcm_00566, SWS_Dcm_00567), the DCM module shall invoke all impacted configured function of the controlEnableMaskRecord (if parameter ***DcmDspDataUsePort*** set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC; see configuration parameter ***DcmDspDataFreezeCurrentStateFnc***). Alternatively call all the associated *FreezeCurrentState* operations (if parameter ***DcmDspDataUsePort*** set to *USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER*) for every data of the DID received in the request.」()

[SWS_Dcm_00399]「 On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to shortTermAdjustment, if all verifications have been successfully done (see SWS_Dcm_00563, SWS_Dcm_00565, SWS_Dcm_00566, SWS_Dcm_00567), the DCM module shall invoke all impacted configured function of the controlEnableMaskRecord (if parameter ***DcmDspDataUsePort*** set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC; see configuration parameter ***DcmDspDataShortTermAdjustmentFnc***). Alternatively call all the associated *ShortTermAdjustment* operations (if parameter ***DcmDspDataUsePort*** set to *USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER*) for every data of the DID received in the request.」()

[SWS_Dcm_00858]「 On any session transition, the DCM shall stop all controls in progress which are not support in the new session anymore:

- For every DID signals with DcmDspDataUsePort set to USE_ECU_SIGNAL and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: call to IoHwAb_Dcm_<symbolic name of ECU signal (parameter **DcmDspDataEcuSignal)**>() with ‘action’ parameter set to returnControlToEcu.

- For every DID signals with DcmDspDataUsePort set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_SYNCH_CLIENT_SERVER and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: call the port interface operation “ReturnControlToECU”

- For every DID signals with DcmDspDataUsePort set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: call the configured function xxx_ReturnControlToECU (See parameter *DcmDspDataReturnControlToEcuFnc*)」()

[SWS_Dcm_00628]「On a session transition to default session (either from default session or from non-default session), the DCM shall stop all the control in progress:

- For every DID signals with DcmDspDataUsePort set to USE_ECU_SIGNAL and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: call to IoHwAb_Dcm_<symbolic name of ECU signal (parameter **DcmDspDataEcuSignal)**>() with 'action' parameter set to returnControlToEcu.

- For every DID signals with DcmDspDataUsePort set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_SYNCH_CLIENT_SERVER and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: call the port interface operation "ReturnControlToECU"

- For every DID signals with DcmDspDataUsePort set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: call the configured function xxx_ReturnControlToECU (See parameter *DcmDspDataReturnControlToEcuFnc*)」()

[SWS_Dcm_00859]「On any security level change, the DCM shall stop all controls in progress which are not support by the new security level anymore:

- For every DID signals with DcmDspDataUsePort set to USE_ECU_SIGNAL and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: call to IoHwAb_Dcm_<symbolic name of ECU signal (parameter **DcmDspDataEcuSignal)**>() with 'action' parameter set to returnControlToEcu.

- For every DID signals with DcmDspDataUsePort set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_SYNCH_CLIENT_SERVER and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: call the port interface operation “ReturnControlToECU”

- For every DID signals with DcmDspDataUsePort set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: call the configured function xxx_ReturnControlToECU (See parameter *DcmDspDataReturnControlToEcuFnc*)()

[SWS_Dcm_00640] ⌈ To serialize the required AUTOSAR data types (signed- and unsigned integer) into the response message of InputOutputControlByIdentifier responses the target endianness configured in ***DcmDspDataEndianness*** shall be considered for ***DcmDspData*** elements having ***DcmDspDataUsePort*** set to {USE_ECU_SIGNAL}. In case ***DcmDspDataEndianness*** is not present, the ***DcmDspDataDefaultEndianness*** shall be used instead.⌋()

[SWS_Dcm_00682] ⌈ The ControlStatusRecord for the IoControl response shall be retrieved using the associated ReadData operations⌋()

[constr_6019] ⌈DcmDspData elements used in service 0x2F shall not have DcmDspDataUsePorts set to USE_DATA_SENDER_RECEIVER⌋()

7.4.2.13 UDS Service 0x31 - RoutineControl

[SWS_Dcm_00257] ⌈ The DCM module shall implement the UDS Service RoutineControl (0x31) for subFunctions startRoutine, stopRoutine and requestsRoutineResults.⌋()

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***)

- List of signal available in the request and in the response (see configuration parameters ***DcmDspStartRoutineIn*** and ***DcmDspStartRoutineOut***)
- The function `Xxx_Stop()` associated with this routineIdentifier (see configuration parameters ***DcmDspStopRoutineFnc*** and ***DcmDspRoutineUsePort***)
- List of signal available in the request and in the response (see configuration parameters ***DcmDspRoutineStopIn*** and ***DcmDspRoutineStopOut***)
- The function `Xxx_RequestResults()` associated with this routineIdentifier (see configuration parameters ***DcmDspRequestResultsRoutineFnc*** and ***DcmDspRoutineUsePort***)
- List of signal available in the response (see configuration parameter ***DcmDspRoutineRequestResOut***)

[SWS_Dcm_00568] ⌈ 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).⌋()

[SWS_Dcm_00569] ⌈ If a Routine is set as unused (DcmDspRoutineUsed set to FALSE), the DCM shall consider the Routine as not supported (according to SWS_Dcm_00568)⌋()

[SWS_Dcm_00570] ⌈ On reception of the UDS Service RoutineControl (0x31), the DCM module shall check if the Routine can be executed in the current session (see configuration parameter ***DcmDspRoutineSessionRef***). If not, the DCM module shall send a NRC 0x31 (Request Out of Range).⌋()

[SWS_Dcm_00571] ⌈ On reception of the UDS Service RoutineControl (0x31), the DCM module shall check if the Routine can be executed in the current security level (see configuration parameter ***DcmDspRoutineSecurityLevelRef***). If not, the DCM module shall send NRC 0x33 (Security access denied).⌋()

[SWS_Dcm_00824] ⌈On reception of the UDS Service RoutineControl (0x31), the DCM module shall check if the Routine can be executed in the current mode condition (see configuration parameter ***DcmDspRoutineModeRuleRef***). If not, the DCM module shall send the calculated negative reponse code of the referenced DcmModeRule.⌋()

[SWS_Dcm_00869] ⌈On reception of the UDS Service RoutineControl (0x31), the DCM module shall check if the SubFunction to the corresponding Routine is supported (see existence of configuration container ***DcmDspStartRoutineIn*** and ***DcmDspStartRoutineOut*** for SubFunction 0x01; ***DcmDspRoutineStopIn*** and ***DcmDspRoutineStopOut*** for SubFunction 0x02; ***DcmDspRoutineRequestResOut*** for SubFunction 0x03). If not, the DCM module shall send NRC 0x12 (SubFunction not supported) .⌋()

[SWS_Dcm_00590] ⌈ When receiving a request for UDS Service RoutineControl (0x31) if all verifications have been successfully done (see SWS_Dcm_00568, SWS_Dcm_00570, SWS_Dcm_00571), the DCM module shall split the routineControlOptionRecord received according of the list of input signal configured for this routine (See configuration parameters ***DcmDspStartRoutineIn and DcmDspRoutineStopIn***)⌋()

[SWS_Dcm_00400] ⌈ When receiving a request for UDS Service RoutineControl (0x31) with subfunction startRoutine, if all verifications have been successfully done (see SWS_Dcm_00568, SWS_Dcm_00570, SWS_Dcm_00571), the DCM module shall call the configured `Xxx_Start()` function passing the dataIn, calculated from routineControlOptionRecord (see SWS_Dcm_00590), and the dataOut reference according of the list of output signal configured for this routine (See configuration parameter ***DcmDspStartRoutineOut***). The datalength of the dataIn can be fixed or dynamic according to ***DcmDspRoutineFixedLength***. If dynamic, the datalength shall be provided in the parameter currentDataLength who holds the length in bytes of the last dataIn parameter. The datalength can be dynamic only on the last dataIn parameter.⌋()

[SWS_Dcm_00401] ⌈ Upon completing SWS_Dcm_00400, when `Xxx_Start()` returns no ErrorCode, the DCM module shall reply with a positive response with the data returned by `Xxx_Start()` in the dataOut as routineStatusRecord (dataOut are merged according to the list of output signal configured for this routine (See configuration parameter ***DcmDspStartRoutineOut***)). The datalength of the dataOut can be fixed or dynamic according to ***DcmDspRoutineFixedLength***. If dynamic, the datalength shall be provided in the parameter currentDataLength who holds the length in bytes of the last dataOut parameter. The datalength can be dynamic only on the last dataOut parameter.⌋()

[SWS_Dcm_00402] ⌜ When receiving a request for UDS Service RoutineControl (0x31) with subfunction stopRoutine, if all verifications have been successfully done (see SWS_Dcm_00568, SWS_Dcm_00570, SWS_Dcm_00571), the DCM module shall call the configured `Xxx_Stop()` function passing the dataIn, calculated from routineControlOptionRecord (see SWS_Dcm_00590), and the dataOut reference according of the list of output signal configured for this routine (See configuration parameter ***DcmDspRoutineStopOut***). The datalength of the dataIn can be fixed or dynamic according to ***DcmDspRoutineFixedLength***. If dynamic, the datalength shall be provided in the parameter currentDataLength who holds the length in bytes of the last dataIn parameter. The datalength can be dynamic only on the last dataIn parameter. ⌟()

[SWS_Dcm_00403] ⌜ Upon completing SWS_Dcm_00402, when `Xxx_Stop()` returns no ErrorCode, the DCM module shall reply with a positive response with the data returned by `Xxx_Stop()` in the dataOut as routineStatusRecord (dataOut are merged according to the list of output signal configured for this routine (See configuration parameter ***DcmDspStopRoutineOut***)).The datalength of the dataOut can be fixed or dynamic according to ***DcmDspRoutineFixedLength***. If dynamic, the datalength shall be provided in the parameter currentDataLength who holds the length in bytes of the last dataOut parameter.The datalength can be dynamic only on the last dataOut parameter. ⌟()

[SWS_Dcm_00404] ⌜ When receiving a request for UDS Service RoutineControl (0x31) with subfunction requestRoutineResults, if all verifications have been successfully done (see SWS_Dcm_00568, SWS_Dcm_00570, SWS_Dcm_00571), the DCM module shall call the configured `Xxx_RequestResults()` function and provide the dataOut reference according of the list of output signal configured for this routine (See configuration parameter ***DcmDspRoutineRequestResOut***). ⌟()

[SWS_Dcm_00405] ⌜ Upon completing SWS_Dcm_00404, when `Xxx_RequestResults()` returns no ErrorCode, the DCM module shall reply with a positive response with the data returned by `Xxx_RequestResults()` in the dataOut as routineStatusRecord (dataOut are merged according to the list of output signal configured for this routine (See configuration parameter ***DcmDspRoutineRequestResOut***)).The datalength of the dataOut can be fixed or dynamic according to ***DcmDspRoutineFixedLength***. If dynamic, the datalength shall be provided in the parameter currentDataLength who holds the length in bytes of the last dataOut parameter.The datalength can be dynamic only on the last dataOut parameter. ⌟()

[SWS_Dcm_00641] ⌈ To support the required AUTOSAR data types (signed- and unsigned integer), if configuration parameter *DcmDslProtocolEndiannessConvEnabled* is set to TRUE, the DCM module must perform endianness conversion of all types mentioned in **SWS_Dcm_00968** for data received and transmitted in RoutineControl service.⌋()

[SWS_Dcm_00701] ⌈ On reception of the UDS Service RoutineControl (0x31), for every requested TID inside the OBD range (E000-E0FF), the DCM module shall get the TID value as defined for OBD Service $08 (See SWS_Dcm_00418 , SWS_Dcm_00419). ⌋()

7.4.2.14 UDS Service 0x3E - Tester Present

[SWS_Dcm_00251] ⌈ 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.

7.4.2.15 UDS Service 0x85 - ControlDTCSetting

[SWS_Dcm_00249] ⌈ 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.

[SWS_Dcm_00304] ⌈On reception of the UDS Service 0x85 with DTCSettingType=on and the optional parameter DTCSettingControlOptionRecord is NOT present in the request message, the DCM module shall call Dem_DcmEnableDTCSetting(DTCGroup, DTCKind)with DTCGroup = DEM_DTC_GROUP_ALL_DTCS and DTCKind = DEM_DTC_KIND_ALL_DTCS.SRS_Diag_04010)

[SWS_Dcm_01063]⌈ On reception of the UDS Service 0x85 with DTCSettingType=on and the optional parameter DTCSettingControlOptionRecord is present in the request message, the DCM module shall call Dem_DcmEnableDTCSetting(DTCGroup, DTCKind)with DTCGroup = DTCSettingControlOptionRecord of the request message and DTCKind = DEM_DTC_KIND_ALL_DTCS. ⌋ (SRS_Diag_04010)

This requirement is only valid if DcmSupportDTCSettingControlOptionRecord is set to true (see **SWS_Dcm_00829** and **SWS_Dcm_00852**)

[SWS_Dcm_01064]⌜ On reception of the UDS Service 0x85 with DTCSettingType=off and the optional parameter DTCSettingControlOptionRecord is NOT present in the request message, the DCM module shall call Dem_DcmDisableDTCSetting(DTCGroup, DTCKind)with DTCGroup = DEM_DTC_GROUP_ALL_DTCS and DTCKind = DEM_DTC_KIND_ALL_DTCS.⌟(SRS_Diag_04010)

[SWS_Dcm_00783] ⌜ In case of `Dem_DcmEnableDTCSetting` returns DEM_CONTROL_DTC_SETTING_OK (see **SWS_Dcm_00304**), the DCM shall invoke a mode switch of the ModeDeclarationGroupPrototype DcmControlDTCSetting by calling SchM_Switch_<bsnp>_DcmControlDTCSetting (RTE_MODE_DcmControlDTCSetting_ENABLEDTCSETTING).⌟()

[SWS_Dcm_00406] ⌜On reception of the UDS Service 0x85 with DTCSettingType=off and the optional parameter DTCSettingControlOptionRecord is present in the request message, the DCM module shall call Dem_DcmDisableDTCSetting(DTCGroup, DTCKind)with DTCGroup = DTCSettingControlOptionRecord of the request message and DTCKind = DEM_DTC_KIND_ALL_DTCS.⌟(SRS_Diag_04010)

This requirement is only valid if DcmSupportDTCSettingControlOptionRecord is set to true (see **SWS_Dcm_00829** and **SWS_Dcm_00852**)

[SWS_Dcm_00784] ⌜ In case of Dem_DcmDisableDTCSetting returns DEM_CONTROL_DTC_SETTING_OK (see **[SWS_Dcm_00406]**), the DCM shall invoke a mode switch of the ModeDeclarationGroupPrototype DcmControlDTCSetting by calling SchM_Switch_<bsnp>_DcmControlDTCSetting (RTE_MODE_DcmControlDTCSetting_DISABLEDTCSETTING).⌟()

[SWS_Dcm_00751] ⌜ In case the DTCSetting is disabled and a transitions to default session or upon any diagnostic session change where the new session do not support UDS Service ControlDTCsetting anymore, the DCM module shall call `Dem_DcmEnableDTCSetting`(DTCGroup, DTCKind)with DTCGroup = DEM_DTC_GROUP_ALL_DTCS and DTCKind = DEM_DTC_KIND_ALL_DTCS; as well as switch the mode DcmControlDTCSetting to ENABLEDTCSETTING.⌟()

For some use-cases the DCM may re-enable the controlDTCsetting due to external changed mode conditions:

[SWS_Dcm_00752]⌈ In case the DTCSetting is disabled and at least one referenced arbitrary ModeDeclarationGroupPrototypes (see configuration parameter DcmDspControlDTCSettingReEnableModeRuleRef) for service ControlDTCSetting (0x85) with DTCSettingType off (0x02) are not fulfilled anymore; the DCM module shall call `Dem_DcmEnableDTCSetting`(DTCGroup, DTCKind)with DTCGroup = DEM_DTC_GROUP_ALL_DTCS and DTCKind = DEM_DTC_KIND_ALL_DTCS as well as switch the mode DcmControlDTCSetting to ENABLEDTCSETTING.⌋()

Note: This active observation of the referenced mode declaration groups can either be achieved by polling the mode condtion each MainFuntion cycle or by attaching to the change notification of mode declaration group (SchM will trigger a BSWEntitiy in DCM on changes of this mode declaration group)

[SWS_Dcm_00829]⌈In case the configuration parameter ***DcmSupportDTCSettingControlOptionRecord*** is set to true and the length of the optional parameter DTCSettingControlOptionRecord in the request is different from 3 bytes, the Dcm shall return NRC 0x13 (Incorrect message length or invalid format) to the tester.⌋()

[SWS_Dcm_00852]⌈In case the configuration parameter ***DcmSupportDTCSettingControlOptionRecord*** is set to false the DCM shall return NRC 0x13 (Incorrect message length or invalid format) if any data is present after the subFunction.⌋()

[SWS_Dcm_00830]⌈In case of `Dem_DcmDisableDTCSetting` or `Dem_DcmEnableDTCSetting` returns DEM_CONTROL_DTC_WRONG_DTCGROUP (wrong groupOfDTC), the Dcm shall return NRC 0x31 (RequestOutOfRange).⌋()

7.4.2.16 UDS Service 0x3D – WriteMemoryByAddress

[SWS_Dcm_00488]⌈ The DCM module shall implement the WriteMemoryByAddress (service 0x3D) of the Unified Diagnostic Services.⌋()

This service is used to write data using a physical memory address.

[SWS_Dcm_00855] ⌈On reception of the UDS Service WriteMemoryByAddress (0x3D), the DCM shall check if the requested AddressAndLengthFormatIdentifier is supported (refer to configuration parameter ***DcmDspSupportedAddressAndLengthFormatIdentifier***), Otherwise the NRC 0x31 (requestOutOfRange) shall be responded. In case the container AddressAndLengthFormatIdentifier is not present, the DCM shall accept all possible AddressAndLengthFormatIdentifiers.⌋()

[SWS_Dcm_00489] ⌈ On reception of the UDS Service WriteMemoryByAddress (0x3D), the DCM shall check if the complete memory range to write (from 'memoryAddress' parameter to 'memoryAddress + memorySize -1') is inside the allowed memory ranges (check of ***DcmDspWriteMemoryRangeLow*** and ***DcmDspWriteMemoryRangeHigh*** parameters for each DcmDspWriteMemoryRangeInfo container). If not, the DCM module shall send NRC 0x31 (Request out of range).⌋()

[SWS_Dcm_00490] ⌈ On reception of the UDS Service WriteMemoryByAddress (0x3D), the DCM shall check if the complete memory range (from 'memoryAddress' parameter to 'memoryAddress + memorySize -1') can be written in the current security level (see ***DcmDspWriteMemoryRangeSecurityLevelRef***). If security level is not correct, the DCM module shall send NRC 0x33 (securityAccessDenied).⌋()

[SWS_Dcm_00825] ⌈On reception of the UDS Service WriteMemoryByAddress (0x3D), the DCM shall check if the complete memory range (from 'memoryAddress' parameter to 'memoryAddress + memorySize -1') can be written in the current mode condition (see ***DcmDspWriteMemoryRangeModeRuleRef***). If mode condition is not correct, the DCM module shall send the calculated negative response code of the referenced dcmModeRule.⌋()

[SWS_Dcm_00491] ⌈ On reception of the UDS Service WriteMemoryByAddress (0x3D), and after verification of the validity of the request (see **[SWS_Dcm_00489** and **[SWS_Dcm_00490**) the DCM module shall call the callout Dcm_WriteMemory().⌋()

[SWS_Dcm_01052] ⌈On reception of the UDS Service WriteMemoryByAddress (0x3D), if the request message contains different MemoryIdValue compare to the configured values in ***DcmDspMemoryIdInfo*** container, the Dcm shall send a NRC 0x31 (Request out of Range).⌋()

[SWS_Dcm_01056] ⌈The configured ranges of memory address (DcmDspWriteMemoryRangeHigh and DcmDspWriteMemoryRangeLow) should not overlap each other.⌋()

7.4.2.17 UDS Service 0x23 – ReadMemoryByAddress

[SWS_Dcm_00492] ⌈ The DCM module shall implement the ReadMemoryByAddress (service 0x23) of the Unified Diagnostic Services.⌋()

This service is used to read data using a physical memory address.

[SWS_Dcm_00853] ⌈ On reception of the UDS Service ReadMemoryByAddress (0x23), the DCM shall check if the requested AddressAndLengthFormatIdentifier is supported (refer to configuration parameter ***DcmDspSupportedAddressAndLengthFormatIdentifier***), Otherwise the NRC 0x31 (requestOutOfRange) shall be responded. In case the container DcmDspAddressAndLengthFormatIdentifier is not present, the DCM shall accept all possible AddressAndLengthFormatIdentifiers.).⌋()

[SWS_Dcm_00493] ⌈ On reception of the UDS Service ReadMemoryByAddress (0x23), the DCM shall check if the complete memory range to read (from ‘memoryAddress’ parameter to ‘memoryAddress + memorySize -1’) is inside the allowed memory ranges (check of ***DcmDspReadMemoryRangeLow*** and ***DcmDspReadMemoryRangeHigh*** parameters for each ***DcmDspReadMemoryRangeInfo*** container). If not, the DCM module shall send NRC 0x31 (Request out of range).⌋()

[SWS_Dcm_00494] ⌈ On reception of the UDS Service ReadMemoryByAddress (0x23), the DCM shall check if the complete memory range (from ‘memoryAddress’ parameter to ‘memoryAddress + memorySize -1’) can be readen in the current security level (see ***DcmDspReadMemoryRangeSecurityLevelRef***). If security level is not correct, the DCM module shall send NRC 0x33 (securityAccessDenied).⌋()

[SWS_Dcm_00826] ⌈ On reception of the UDS Service ReadMemoryByAddress (0x23), the DCM shall check if the complete memory range (from ‘memoryAddress’ parameter to ‘memoryAddress + memorySize -1’) can be readen in the current mode condition (see ***DcmDspReadMemoryRangeModeRuleRef***). If mode condition is not correct, the DCM module shall send calclulated negative response code of the referenced DcmModeRule.⌋()

[SWS_Dcm_00495] ⌈ On reception of the UDS Service ReadMemoryByAddress (0x23), and after verification of the validity of the request (see **[SWS_Dcm_00493** and **[SWS_Dcm_00494**) the DCM module shall call the callout Dcm_ReadMemory().⌋()

[SWS_Dcm_01053] ⌈On reception of the UDS Service ReadMemoryByAddress (0x23), if the request message contains different MemoryIdValue compare to the configured values in ***DcmDspMemoryIdInfo*** container, the Dcm shall send a NRC 0x31 (Request out of Range).⌋()

[SWS_Dcm_01056] ⌈The configured ranges of memory address (DcmDspReadMemoryRangeHigh and DcmDspReadMemoryRangeLow) should not overlap each other.⌋()

7.4.2.18 UDS Service 0x34 – RequestDownload

[SWS_Dcm_00496] ⌈ The DCM module shall implement the RequestDownload (service 0x34) of the Unified Diagnostic Services.⌋(BSW04033)

[SWS_Dcm_00856] ⌈On reception of the UDS ServiceRequestDownload (0x34), the DCM shall check if the requested AddressAndLengthFormatIdentifier is supported (refer to configuration parameter ***DcmDspSupportedAddressAndLengthFormatIdentifier***), Otherwise the NRC 0x31 (requestOutOfRange) shall be responded. In case the container AddressAndLengthFormatIdentifier is not present, the DCM shall accept all possible AddressAndLengthFormatIdentifiers.⌋()

[SWS_Dcm_01057] ⌈On reception of the UDS ServiceRequestDownload (0x34), if the request message contains different MemoryIdValue compare to the configured values in ***DcmDspMemoryIdInfo*** container, the Dcm shall send a NRC 0x31 (Request out of Range).⌋()

This service is used to request the start of a download process.

7.4.2.19 UDS Service 0x35 – RequestUpload

[SWS_Dcm_00499] ⌈ The DCM module shall implement the RequestUpload (service 0x35) of the Unified Diagnostic Services.⌋(BSW04033)

[SWS_Dcm_00857] ⌈On reception of the UDS RequestUpload (0x35), the DCM shall check if the requested AddressAndLengthFormatIdentifier is supported (refer to configuration parameter ***DcmDspSupportedAddressAndLengthFormatIdentifier***), Otherwise the NRC 0x31 (requestOutOfRange) shall be responded. In case the container AddressAndLengthFormatIdentifier is not present, the DCM shall accept all possible AddressAndLengthFormatIdentifiers.⌋()

[SWS_Dcm_01055] ⌈On reception of the UDS RequestUpload (0x35), if the request message contains different MemoryIdValue compare to the configured values in ***DcmDspMemoryIdInfo*** container, the Dcm shall send a NRC 0x31 (Request out of Range).⌋()

This service is used to request the start of a upload process.

7.4.2.20 UDS Service 0x36 – TransferData

[SWS_Dcm_00502] ⌈ The DCM module shall implement the TransferData (service 0x36) of the Unified Diagnostic Services.⌋(BSW04033)

This service is used to transfer data during a download or upload process.

[SWS_Dcm_00503] ⌈ On reception of the UDS Service TransferData (0x36), if a download process is running (RequestDownload service has been previously received) and the request format is correct, the DCM module shall call the callout Dcm_WriteMemory().⌋(BSW04033)

[SWS_Dcm_00504] ⌈ On reception of the UDS Service TransferData (0x36), if an upload process is running (RequestUpload service has been previously received) and the request format is correct, the DCM module shall call the callout Dcm_ReadMemory().⌋(BSW04033)

[SWS_Dcm_00645] ⌈ On reception of the UDS Service TransferData (0x36), if a block sequence error is detected, the DCM module shall trigger a negative response with NRC 0x73 (*WrongBlockSequenceCounter*)⌋()

7.4.2.21 UDS Service 0x37 – RequestTransferExit

[SWS_Dcm_00505] ⌈ The DCM module shall implement the RequestTransferExit (service 0x37) of the Unified Diagnostic Services.⌋(BSW04033)

This service is used to terminate a download or upload process.

7.4.2.22 UDS Service 0x28 – CommunicationControl

[SWS_Dcm_00511] ⌈ The DCM module shall implement the CommunicationControl (service 0x28) of the Unified Diagnostic Services.⌋()

[SWS_Dcm_00512] ⌈ On invocation of the sent confirmation function of the UDS Service CommunicationControl (0x28) from DSD with the subnet parameter of the request equal to 0x00, the DCM shall do for each NetworkHandle (see DcmDspAllComMChannelRef) which is configured in ***DcmDspComControlAllChannel***:

1) trigger the mode switch Dcm_CommunicationControl_<Network> ModeDeclarationGroupPrototype to the mode corresponding the communicationType and controlType parameter from the CommunicationControl request.
2) call the Api BswM_Dcm_CommunicationMode_CurrentState

with the parameters NetworkHandleType and Dcm_CommunicationModeType corresponding to the communicationType and controlType parameter from the CommunicationControl request (see Dcm_CommunicationModeType definition).⌋()

[SWS_Dcm_00785] ⌈On invocation of the sent confirmation function of the UDS Service CommunicationControl (0x28) from DSD with the subnet parameter of the request equal to 0x0F(CommunicationControl on the network which request is received on), the DCM shall do for the NetworkHandle (see ***DcmDslProtocolComMChannelRef***) of the current received ***DcmDslProtocolRxPduRef:***

1) trigger the mode switch Dcm_CommunicationControl_<Network> ModeDeclarationGroupPrototype to the mode corresponding to the communicationType and controlType parameter from the CommunicationControl request.
2) call the Api BswM_Dcm_CommunicationMode_CurrentState with the parameters NetworkHandleType and Dcm_CommunicationModeType corresponding to the communicationType and controlType parameter from the CommunicationControl request (see Dcm_CommunicationModeType definition)⌋()

[SWS_Dcm_00786] ⌈On invocation of the sent confirmation function of the UDS Service CommunicationControl (0x28) from DSD with the subnet parameter of the request between 0x01 and 0x0E, the DCM shall check if the received subnet parameter (see ***DcmDspSubnetNumber***) is supported. In case it is not supported a NegativeResponse code 0x31 shall be sent. In case it is supported the DCM shall do for the corresponding NetworkHandle (see ***DcmDspSpecificComMChannelRef)*** of the received subnet parameter (see ***DcmDspSubnetNumber***):

1) tigger the mode switche Dcm_CommunicationControl_<Network> ModeDeclarationGroupPrototype to the mode corresponding the communicationType and controlType parameter from the CommunicationControl request.
2) call the Api BswM_Dcm_CommunicationMode_CurrentState the parameters NetworkHandleType and with Dcm_CommunicationModeType corresponding the communicationType and controlType parameter from the CommunicationControl request (see Dcm_CommunicationModeType definition) 」()

For some use-cases the DCM may re-enable the CommunicationControl due to external changed mode conditions:

[SWS_Dcm_00753] 「The DCM module shall do for all NetworkHandles wich are currently under control (state other DCM_ENABLE_RX_TX_NORM_NM) in case at least one referenced arbitrary ***ModeDeclarationGroupPrototype*** (see configuration parameter ***DcmDspComControlCommunicationReEnableModeRuleRef***) is not fulfilled anymore:

1) trigger the mode switches of each Dcm_CommunicationControl_<Network> ModeDeclarationGroupPrototype to mode **DCM_ENABLE_RX_TX_NORM_NM**
2) call BswM_Dcm_CommunicationMode_CurrentState with the parameters NetworkHandleType and RequestedCommunicationMode set to **DCM_ENABLE_RX_TX_NORM_NM** 」()

[SWS_Dcm_00860] 「If on any network CommunicationControl is unequal to **DCM_ENABLE_RX_TX_NORM_NM** and the DCM is transitioning to default session or upon any diagnostic session change where the new session do not support UDS Service CommunicationControl anymore, the DCM shall:

1) trigger the mode switches of each Dcm_CommunicationControl_<Network> ModeDeclarationGroupPrototype to mode **DCM_ENABLE_RX_TX_NORM_NM**
2) call BswM_Dcm_CommunicationMode_CurrentState with the parameters NetworkHandleType and RequestedCommunicationMode set to **DCM_ENABLE_RX_TX_NORM_NM** 」()

7.4.2.23 UDS Service 0x87 – LinkControl

This service is used to gain bus bandwidth for diagnostic purposes

The Service LinkControl (0x87) is user optional. There are different project specific use cases which are not handled in the default Dcm. One use case is to switch the bandwidth in application an other use case performs an OEM bootloader jump.

Therefore the service LinkControl needs to be implemented project specific as external service (refer to Chapter 8.9 DCM as Service-Component)

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 1: Support for OBD services in the DCM

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.

In a vehicle there can be 3 different types of OBD ECUs:

- Master ECU (one per vehicle)
- Primary ECU (several per vehicle)
- Dependent / Secondary ECUs (several per vehicle)

From the Basic Software point of view Dependent / Secondary ECUs doesn't need any specific OBD functionality. In Dependent / Secondary ECUs OBD-relevant information will not be stored in the Basic Software (e.g. no direct communication

with the scan tool). The respective OBD functionality might be handeled in Dependent / Secondary ECUs by a SWC.

The following OBD requirements are only valid for Master and Primary ECUs.
If necessary the OBD requirements differenciate between Master and Primary Requirement.

The following table gives an overview about which OBD functionality must be supported in a Master ECU, Primary ECU or Dependent / Secondary ECU:

Functionality	Master ECU	Primary ECU	Dependent / Secondary ECU
OBD Scantool Communication	Yes	Yes	No

Table 7: Overview about OBD functionality in different OBD ECUs

[SWS_Dcm_00077] ⌜ 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⌟(BSW04058)

7.4.3.3 OBD Service $01 – Request Current Powertrain diagnostic Data

[SWS_Dcm_00243] ⌜ The DCM module shall implement the OBD service $01 (Request Current Powertrain diagnostic Data) in compliance to all provisions of the OBD standard.⌟(BSW04082, BSW04001)

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.

The DCM collects the PID information from 1 to n SW-Cs. This applies in particular for PIDs which contain several data values for potentially different sources. Example: PID$83 reports Nox Sensor Data for sensor 1 and sensor 2 in one composed PID which might come from different SW-C.

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

- The PID Identifier (see configuration parameter ***DcmDspPidIdentifier***)
- Indication of the PID is used or not (for postbuild configuration) (see configuration parameter ***DcmDspPidUsed***)
- The size of the PID (see configuration parameter ***DcmDspPidSize***)
- The supported information for this PID (see configuration parameter ***DcmDspPidSupportInfo***)

- List of data (***DcmDspPidData***) for the PID with the following configuration for every data
  - The length of the data associated with the PID (see configuration parameter ***DcmDspPidDataSize***)
  - The position of the data in the PID (see configuration parameter ***DcmDspPidDataPos***)
  - The reference to the supported information container (see configuration parameter ***DcmDspPidDataSupportInfo)***
  - The `Xxx_ReadData()` function that the DCM must call to obtain the current value of the data 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 ***DcmDspPidDataReadFnc*** and ***DcmDspPidDataUsePort***)

[SWS_Dcm_00407] ⌜ On reception of an OBD Service $01 request with only "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_ReadData()` functions for every data of the PID.
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 data of every PID that is needed by the DCM.

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

[SWS_Dcm_00408] ⌜ On reception of an OBD Service $01 request with only PIDs that are not "availability PIDs", the DCM shall obtain the current value of these PIDs by invoking the configured `Xxx_ReadData()` functions for every data of the PID and shall return these values as response to Service $01. ⌟()

[SWS_Dcm_00943] ⌜ On reception of an OBD Service $01 request with a mixure of "avalibility PIDs" and not "avalibility PIDs", this request shall be ignored by the Dcm. ⌟()

The entity providing the actual implementation of the `Xxx_ReadData()` function for a specific signal of a 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_ReadData()` function point to the correct function provided by the DEM.

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.

Note: For PIDs where Dem provides the function, DcmDspPidDataUsePort for that PID should be set to USE_DATA_SYNCH_FNC and DcmDspPidDataReadFnc shall point to the function Dem_DcmReadDataOfPID<NN> where <NN> represents the Id of the PID.

The data byte A of the PIDs contain the support status of the subsequent data bytes. Since not all data values might be available due to the particular vehicle configuration (e.g. there is only a Nox-sensor 1 available in the vehicle in the example above), the PID response contains in this data byte A the information about the support status of the following data values. Note, that the PIDs always contain the same number of bytes – even if not all values are really available.

[SWS_Dcm_00621] ⌈ If a PID contains support information (presence of ***DcmDspPidDataSupportInfo*** container) the DCM shall add the support information in the diagnostic response.⌋()

[SWS_Dcm_00622] ⌈ If a PID contains support information (presence of ***DcmDspPidDataSupportInfo*** container) the DCM shall calculate the support information value according to the available data for this PID: for every ***DcmDspPidData*** container existing for this PID, the associated support information bits, referenced in ***DcmDspPidDataSupportInfo***, shall be set to one⌋()

The response to the OBD-tester needs to be composed out of the available data values. Data bytes that are not provided by an SW-C need to be replaced with fill-byte to obtain a complete PID contents.

[SWS_Dcm_00623] ⌈When responding to OBD Service $01, the DCM shall put fill-bytes between ***DcmDspPidData*** in the PID whenever content bytes are missing in order to fit to the PID size (see configuration parameter ***DcmDspPidSize***).⌋()

[SWS_Dcm_00944] ⌈ The Dcm shall set the fill bytes to 0x00.⌋()

Note: If other fill-bytes than 0x00 are needed by legistlation, the application has to provide the value of the fill-byte.

[SWS_Dcm_00718] ⌈ To support the required AUTOSAR data types (signed- and unsigned integer), if configuration parameter *DcmDslProtocolEndiannessConvEnabled* is set to TRUE, the DCM module must perform endianness conversion of all types mentioned in **SWS_Dcm_00968** for data transmitted in Service 1 response (when sender/receiver interface DataServices_{Data} is used).⌋()

7.4.3.4 OBD Service $02 – Request Power Train FreezeFrame Data

[SWS_Dcm_00244] ⌈ The DCM shall implement OBD Service $02 (Request Power Train FreezeFrame Data) in compliance to all provisions of the OBD standard.⌋(BSW04082, BSW04001)

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

[SWS_Dcm_00409] ⌈ The DCM shall ignore all requests regarding record-numbers that are not 0⌋()

[SWS_Dcm_00972] ⌈ On reception of an OBD Service $02 request with a mixure of “avalibility PIDs” and not “avalibility PIDs”, this request shall be ignored by the Dcm.⌋()

[SWS_Dcm_00973] ⌈When responding to OBD Service $02, the DCM shall put fill-bytes between ***DcmDspPidData*** in the PID whenever content bytes are missing in order to fit to the PID size (see configuration parameter ***DcmDspPidSize***).⌋()

[SWS_Dcm_00974] ⌈ The Dcm shall set the fill bytes to 0x00.⌋()

Note: If other fill-bytes than 0x00 are needed by legistlation, the application has to provide the value of the fill-byte.

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.

[SWS_Dcm_00279] ⌈ On reception of a request for Service $02 with PID $02, the DCM shall call `Dem_DcmGetDTCOfOBDFreezeFrame()` with FrameNumber set to 0x00 to get the DTC number⌋(SRS_Diag_04010, BSW04058)

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

[SWS_Dcm_01061]⌈ If `Dem_DcmGetDTCOfOBDFreezeFrame` returns E_NOT_OK, the Dcm shall answer positively with $0000 (indicates no stored freeze frame data).⌋()

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”.

[SWS_Dcm_00284] ⌈ On reception of a service $02 request with an “availability PID”, the DCM shall respond with the corresponding supported (=configured) PIDs encoded according to the OBD standard. ⌋(SRS_Diag_04010)

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.

[SWS_Dcm_00286] ⌈ 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_DcmReadDataOfOBDFreezeFrame()` for every data of the PID 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
DataElementIndexOfPid = implicit index (from 0 to n) of the DataElement calculated by DCM according to the order of the DataElement positions in the PID (see parameter DcmDspPidDataPos) ⌋(SRS_Diag_04010)

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

[SWS_Dcm_00287] ⌈ Upon the completion of SWS_Dcm_00286, 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

[SWS_Dcm_00245] ⌈ The DCM module shall implement OBD Service $03 (Request emission-related diagnostic trouble codes) in compliance to all provisions of the OBD standard. ⌋(BSW04082, BSW04001)

[SWS_Dcm_00410] ⌈ 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. ⌋(BSW04082, BSW04001)

[SWS_Dcm_00411] ⌈ 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. ⌋(BSW04082, BSW04001)

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.

[SWS_Dcm_00289] ⌈ 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_DcmSetDTCFilter()` and `Dem_DcmGetNextFilteredDTC()`.⌋(SRS_Diag_04010)

[SWS_Dcm_00412] ⌈ 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_DcmSetDTCFilter()` and `Dem_DcmGetNextFilteredDTC()`.⌋(SRS_Diag_04010)

[SWS_Dcm_00330] ⌈ 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_DcmSetDTCFilter()` and `Dem_DcmGetNextFilteredDTC()`.⌋(SRS_Diag_04010)

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

Parameters to Dem_DcmSetDTCFilter			
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
DTCFormat	DEM_DTC_FORMAT_OBD	DEM_DTC_FORMAT_OBD	DEM_DTC_FORMAT_OBD
DTCOrigin	DEM_DTC_ORIGIN_ PRIMARY_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
DTCSeverityMask	Not relevant	Not relevant	Not relevant
FilterForFault DetectionCounter	DEM_FILTER_ FOR_FDC_NO	DEM_FILTER_ FOR_FDC_NO	DEM_FILTER_ FOR_FDC_NO

When using paged buffer mechanism, in some case, it's possible that the number of DTC matching the filter change between the calculation of the total size, needed for the first page transmission, and the sending of the further pages. For this reason, the requirement SWS_Dcm_00587 and SWS_Dcm_00588 shall be considered for the implementation of this service.

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

[SWS_Dcm_00246] ⌈ The DCM module shall implement OBD Service $04 (Clear/reset emission-related diagnostic information) in compliance to all provisions of the OBD standard.⌋(BSW04082, BSW04001)

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

[SWS_Dcm_00004] ⌈ When receiving a request for OBD Service $04, the DCM module shall call the operation `DcmClearDTC` with the following parameter values:
DTC = DEM_DTC_GROUP_ALL_DTCS
DTCFormat: DEM_DTC_FORMAT_OBD
DTCOrigin = DEM_DTC_ORIGIN_PRIMARY_MEMORY⌋(SRS_Diag_04010, BSW04058, BSW04065)

[SWS_Dcm_00413] ⌈ In case `Dem_DcmClearDTC()` returns DEM_CLEAR_OK, the DCM module shall send a positive response.⌋(SRS_Diag_04010)

[SWS_Dcm_00703] ⌈ In case `Dem_DcmClearDTC()` returns DEM_CLEAR_PENDING, the DCM shall invoke `Dem_DcmClearDTC()` on next Dcm_MainFunction call again. It is up to the DCM to send NRC 78 to respect the response behaviour⌋()

[SWS_Dcm_00704] ⌈ In case `Dem_DcmClearDTC()` returns DEM_CLEAR_FAILED, the DCM shall send a negative response 0x22 – conditionsNotCorrect.⌋()

[SWS_Dcm_00967] ⌈ In case Dem_DcmClearDTC() returns DEM_CLEAR_BUSY, the DCM shall send a negative response 0x22 – ConditionsNotCorrect.⌋()

[SWS_Dcm_01067]⌈ In case Dem_DcmClearDTC() returns DEM_CLEAR_MEMORY_ERROR, the DCM module shall send a negative response 0x22 - ConditionNotCorrect.⌋()

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

7.4.3.7.1 General requirements

[SWS_Dcm_00414] ⌈ 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.⌋(BSW04082, BSW04001)

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.

A tester request for supported OBDMIDs may contain up to six (6) "availability OBDMIDs".

[SWS_Dcm_00945] ⌈ On reception of an OBD Service $06 request with "availability OBDMIDs" together with other OBDMIDs as parameter, the DCM module shall ignore the request.⌋()

[SWS_Dcm_00956] ⌈ On reception of an OBD Service $06 request with multiple non-availability OBDMIDs, the DCM module shall ignore the request.⌋()

[SWS_Dcm_00946] ⌈ Data A - E shall be filled with $00 if unused⌋()

7.4.3.7.2 Test results obtained via DEM interaction

The maintenance of the DTRs lies within the responsibility of the DEM. SW-Cs reporting DTRs use dedicated interfaces offered by the DEM. Upon requests from the tester the DCM retrieves the information from the DEM using dedicated DEM interfaces. There is no direct interaction between the DCM and SW-Cs.

[SWS_Dcm_00957] ⌈On reception of an OBD Service $06 request with only "availability OBDMID(s)" as parameter(s), the DCM module shall obtain the supported OBDMIDs by calling the DEM interface `Dem_DcmGetAvailableOBDMIDs()` for each "availability OBDMID ($00, $20, …)" contained within the request and concatenate the results within the response message.⌋()

[SWS_Dcm_00958] ⌈ On reception of an OBD Service $06 request with an OBDMID that is not an “availability OBDMID”, the DCM module shall call the DEM interface `Dem_DcmGetNumTIDsOfOBDMID()` to obtain the TIDs available for the requested OBDMID and then recurrently call the interface `Dem_DcmGetDTRData()` for the number of reported TIDs to obtain the associated DTR data. ⌋()

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

[SWS_Dcm_00417] ⌈ 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.⌋(BSW04082, BSW04001)

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.

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***)

To provide OBD Service $08, the DCM relies on external functions configured per TID

[SWS_Dcm_00418] ⌈ 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.⌋()

[SWS_Dcm_00947] ⌈ On reception of an OBD Service $08 request “availability TIDs” together with other TIDs as parameter, the DCM module shall ignore the request.⌋()

[SWS_Dcm_00419] ⌈ 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)⌋()

[SWS_Dcm_00420] ⌈ After the execution of SWS_Dcm_00419, the Dcm module shall respond to the service request using the data stored by the RequestControl function in the OutBuffer.⌋()

[SWS_Dcm_00948] ⌈ As specified in [20], unused data bytes shall be filled with $00.⌋()

7.4.3.9 OBD Service $09 – Request Vehicle Information

[SWS_Dcm_00421] ⌈ The DCM module shall implement OBD Service $09 (Request Vehicle Information) in compliance to all provisions of the OBD standard.⌋(BSW04082, BSW04001)

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.

The DCM module’s configuration defines the InfoTypes and associated data that are available on one or several SW-C. The configuration defines for each such InfoType:

- The value of InfoType (see configuration parameter ***DcmDspVehInfoInfoType***)
- For every data of the InfoType:
  - The position of this data in the InfoType (see configuration parameter ***DcmDspVehInfoDataOrder***)
  - the size of the value of the InfoType data (see configuration parameter ***DcmDspVehInfoDataSize***)
  - the function that the DCM module must call to obtain the value for this InfoType data OR the port-name through which the DCM module can obtain the value for this InfoType data (see configuration parameter ***DcmDspVehInfoDataReadFnc*** and ***DcmDspVehInfoDataUsePort***).

To provide OBD Service $09, the DCM relies on external functions that allow it to

obtain the value of an InfoType data. There is one such function per InfoType data that is needed by the DCM.
When invoking a `Xxx_GetInfotypeValueData()` function, the DCM module provides a buffer of the correct size in which the value of the InfoType data can be stored. The entity providing the actual implementation of the `Xxx_GetInfotypeValueData()` function for a specific InfoType data 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 needed by the DCM to provide Service $09 are provided by the DEM. This is handled in the DCM configuration.

[SWS_Dcm_00422] ⌈ 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.⌋()

[SWS_Dcm_00949] ⌈ On reception of an OBD Service $09 request "availability InfoTypes" together with other InfoTypes as parameter, the DCM module shall ignore the request.⌋()

[SWS_Dcm_00423] ⌈ 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 all the configured `Xxx_GetInfotypeValueData()` function for every data of this InfoType and shall return the value as response to Service $09⌋()

[SWS_Dcm_00684] ⌈ Additional to collecting the available InfoType value contributions from the individual SW-C, the DCM shall compute the data byte NofDataItems in the diagnostic response, which defines the number of DataItems included in one InfoType .⌋()

Note: The Calculation of the Calibration Identification (CAL-ID) and Calibration Verification Number (CVN) is not a BSW Task and will not handled within the DCM.

7.4.4 Bootloader interaction

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

[SWS_Dcm_00531] ⌈ A jump to bootloader is possible only with services DiagnosticSessionControl and LinkControl services ⌋(BSW04098)

[SWS_Dcm_00532] ⌈ 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 prepare the jump to the OEM bootloader (see SWS_Dcm_00535) by triggering the mode switch of *ModeDeclarationGroupPrototype* DcmEcuReset to JUMPTOBOOTLOADER. ⌋(BSW04098)

Note: By this mode switch the DCM informs the BswM to prepare the jump to the bootloader.

[SWS_Dcm_00592] ⌈ 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 prepare the jump to the System Supplier bootloader (see SWS_Dcm_00535) by triggering the mode switch of *ModeDeclarationGroupPrototype* DcmEcuReset to JUMPTOSYSSUPPLIERBOOTLOADER ⌋(BSW04098)

Note: By this mode switch the DCM informs the BswM to prepare the jump to the bootloader.

[SWS_Dcm_00654] ⌈In case the ***ModeDeclarationGroupPrototype*** DcmEcuReset is switched to mode JUMPTOBOOTLOADER or JUMPTOSYSSUPPLIERBOOTLOADER and the configuration parameter *DcmSendRespPendOnTransToBoot* is set to TRUE, the DCM shall trigger transmission of NRC 0x78 – RCR-RP.
In the sent confirmation of this NRC 0x78 the DCM shall trigger the mode switch of the ***ModeDeclarationGroupPrototype*** DcmEcuReset to EXECUTE.⌋()

Note: This final transmission of NRC 0x78 before switching to Bootloader shall reload the P2* timout in the client.

[SWS_Dcm_00719] ⌈In case the ***ModeDeclarationGroupPrototype*** DcmEcuReset is switched to mode JUMPTOBOOTLOADER or JUMPTOSYSSUPPLIERBOOTLOADER the and the configuration parameter *DcmSendRespPendOnTransToBoot* is set to FALSE, the DCM shall shall trigger the mode switch of the ***ModeDeclarationGroupPrototype*** DcmEcuReset to EXECUTE in the next Dcm_MainFunction cycle without sending a NRC 0x78.⌋()

In case of **[SWS_Dcm_00719**, the exact response handling depends on the state of the ‘suppressPosRspMsgIndicationBit’ (TRUE or FALSE) in the request message.

[SWS_Dcm_00535]「If the jump to bootloader is requested (see **[SWS_Dcm_00532**, **[SWS_Dcm_00592** and the configuration parameter ***DcmSendRespPendOnTransToBoot*** is set to TRUE (See **[SWS_Dcm_00654**), the DCM shall call Dcm_SetProgConditions() after a successful transmission of NRC 0x78 (Response pending).」(BSW04098)

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

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.

[SWS_Dcm_00995] 「If the NRC 0x78 (Response Pending) response in SWS_Dcm_00535 is not sent successfully the Dcm shall cancel the current request.」()

[SWS_Dcm_00997] 「If the NRC 0x78 (Response Pending) response in **[SWS_Dcm_00535** is not sent successfully no jump to the bootloader shall performed 」()

Note: If the NRC 0x78 request has not been sent correctly the Dcm will stay in the application and wait for the next request from the Client.

[SWS_Dcm_00720]「If the jump to bootloader is requested (see **[SWS_Dcm_00532**, **[SWS_Dcm_00592** and the configuration parameter DcmSendRespPendOnTransToBoot is set to FALSE (see **[SWS_Dcm_00719**, the DCM shall call Dcm_SetProgConditions() immediately.」()

[SWS_Dcm_00715]「If the jump to bootloader is requested (see **[SWS_Dcm_00532, [SWS_Dcm_00592**)and if the call to *Dcm_SetProgConditions()* returns E_NOT_OK (see **[SWS_Dcm_00535 and [SWS_Dcm_00720**) the DCM shall not request any reset, shall not perform the jump to bootloader, shall not switch the ModeDeclarationGroupPrototype DcmEcuReset to EXECUTE and shall answer negatively to the request with NRC 0x22 (Conditions not correct).」()

7.4.4.2 Jump from Bootloader

[SWS_Dcm_00536] ⌈ At DCM initialization, the DCM shall call Dcm_GetProgConditions() to know if the initialization is the consequence of a jump from the bootloader. ⌋(BSW04098)

Note: It is the responsibility of the software integrator to ensure that the data contained in Dcm_ProgConditionsType 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.

[SWS_Dcm_00537] ⌈ If the initialization of the DCM is the consequence of a jump from the bootloader (see **[SWS_Dcm_00536**, the DCM shall call ComM_DCM_ActiveDiagnostic(`NetworkId`) to request the ComManager for the full communication mode. ⌋()

[SWS_Dcm_00767] ⌈ When the ComM reports full communication to the Dcm, the Dcm shall send the Response to the Service Id passed in the Dcm_ProgConditionsType. ⌋(BSW04098)

[SWS_Dcm_00768] ⌈ If the initialization of the DCM is the consequence of a jump from the bootloader (see **[SWS_Dcm_00536** and 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. ⌋()

7.4.4.3 Flags management

7.4.4.3.1 Jump to Bootloader

On reception of a UDS Service 0x10 request (Diagnostic Session Control) with subfunction 0x02 (Start Programming Session), the Dcm will set the ***ReprogramingRequest*** flag and, if indicated for this service, the ***ResponseRequired*** flag by calling `Dcm_SetProgConditions()`.
After an ECU reset, when the bootloader is started, it will clear the ReprogramingRequest flag and, if required, send a response and clear the ***ResponseRequired*** flag.

The `Dcm_SetProgConditions()` API shall be called again in the next Dcm main function cycle if previous return status was E_PENDING.
In case that, during jump to Bootloader, the `Dcm_SetProgConditions()` API returns E_NOT_OK, a DET error shall be reported (DCM_E_SET_PROG_CONDITIONS_FAIL) and normal functionality will resume.

7.4.4.3.2 Jump from Bootloader

After successful reprograming of the application software, the bootloader will update the ***ApplUpdated*** flag and the ***ResponseRequired*** flags.

After an ECU reset, when the newly programmed application is started for the first time, the Dcm will read the **ApplUpdated** and **ResponseRequired** flag by calling `Dcm_GetProgConditions()`. During this function call the **ApplUpdated** and **ResponseRequired** flags are cleared by the integration code.

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.

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

[SWS_Dcm_00044] ⌈ The used return values shall be the same for development and production. Only the values given by the DCM SWS shall be used. ⌋(BSW00369)

[SWS_Dcm_00040] ⌈ The following errors and exceptions shall be detectable by the DCM module depending on its build version (development/production mode). ⌋(BSW00338)

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 SWS_Dcm_00120)	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
DCM API service invoked with NULL POINTER as parameter	Development	DCM_E_PARAM_POINTER	0x07

7.6 Error notification

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).

[SWS_Dcm_00052] ⌈ The header file of the DCM, DCM.h, shall provide a module ID called DCM_MODULE_ID set to the value 0x35. ⌋()

7.7 Debugging

[SWS_Dcm_00506]⌈ The current status of diagnostic activity (linked to ComM_DCM_InactiveDiagnostic(`NetworkId`) and ComM_DCM_ActiveDiagnostic(`NetworkId`) call) shall be available for debugging.⌋(BSW00442)

[SWS_Dcm_00507]⌈ The current security level shall be available for debugging.⌋(BSW00442)

[SWS_Dcm_00508]⌈ The current session state shall be available for debugging.⌋(BSW00442)

[SWS_Dcm_00509]⌈ The current protocol shall be available for debugging.⌋(BSW00442)

7.8 Synchronous and Asynchronous implementation

The DCM can access data using an R-Port requiring either a synchronous or an asynchronous ClientServertInterface DataServices_{Data}. In the DCM SWS, the parameter DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER.

In case of USE_DATA_SYNCH_CLIENT_SERVER, the interface shall be compatible with the Dem interface "DataServices_<SyncDataElement>" (no OpStatus parameter).
The parameter OpStatus and return parameter DCM_E_PENDING shall only be available in case of USE_DATA_ASYNCH_CLIENT_SERVER.

Note: a Dcm implementation using AsynchronousServerCallPoint or SynchronousServerCallPoint when calling service processors is completely an implementation decision. This only indicates that the operation uses the status of the operation to allow an asynchronous processing by the SW-C (initiating a request, checking if a request is still pending, or cancelling a pending request, see SWS_Dcm_00686).
There is no correlation to the operation signature (i.e. existence of OpStatus parameter and DCM_E_PENDING return code) that demands AsynchronousServerCallPoint or SynchronousServerCallPoint usage.

7.9 DID configuration

The configuration of the DCM contains a list of supported DIDs wich can be configured in two ways:

- The individual DID configuration,which required one port-connection per configured DID to access to the datas (reading and writing). The interface DataServices should be used for each DID in this case.

- The DID range configuration, used to handle a set of DIDs sharing the same behavior uniformly in one SW-C with only one port-connection. The interface DataServices_DIDRange_{Range} should be used in this case. Using this configuration allows an interface optimization.
  The following parameters shall be configured in order to use the DIDRange optimization: ***DcmDspDidRangeIdentifierLowerLimit*** and ***DcmDspDidRangeIdentifierUpperLimit*** which delimited the range of the DIDs. ***DcmDspDidRangeMaxDataLength*** and ***DcmDspDidRangeHasGaps*** .

7.9.1 Did ranges

DID ranges are in general the same as the 'normal' DID read and write function, except that the DID is also passed as a parameter.
This allows to treat the DID range in a switch/case in the read or the write function.

The ranges can be applied for reading (ReadDataByIdentifier 0x22) and writing (WriteDataByIdentifier 0x2E) DIDs.

The ranges can be configured in **ECUC_Dcm_00937 :** DcmDspDidRange.
Each configured range is by default accessible by service 0x22 and 0x2E.
In case the range should be limited to reading or writing, the referenced DcmDspDidInfo container should define the ***DcmDspDidAccess*** accordingly.

It is also possible to define gaps within the range (***DcmDspDidRangeHasGaps***). By activating this feature, the Dcm invokes each time a DID is requested within the configured range, the operation IsDidAvailable has to check the current availability. And as the DIDs of the specified range can have different length, the length of the longest DID has to be configured (***DcmDspDidRangeMaxDataLength***) in order to reserve enough buffer passed to the respective function.

In general, the range functionality can also be used for a single DID if you specifically want to pass the DID as a parameter. Then lower DID and upper DID should be the same.

[**constr_6020**] Definition of allowed DID access「Any defined range shall only reference via ***DcmDspDidRangeInfoRef***. The ***DcmDspDidInfo*** defines the access ***DcmDspDidAccess***. The sub-containers ***DcmDspDidControl*** and ***DcmDspDidDefine*** shall not be used] .」()

[constr_6021] DID ranges cannot be mapped on DDDIDs, because service 0x2C DDDID do not support the range feature. Practically ***DcmDspDidRangeIdentifierLowerLimit*** and ***DcmDspDidRangeIdentifierUpperLimit*** should not include DIDs of the range 0xF200 till 0xF3FF.「Any defined range shall only reference ***DcmDspDidInfo*** via DcmDspDidRangeInfoRef, having set ***DcmDspDidDynamicallyDefined*** == False.」()

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.
[SWS_Dcm_00333]

⌜

Module	*Imported Type*
ComStack_Types	BufReq_ReturnType
	NetworkHandleType
	PduIdType
	PduLengthType
	RetryInfoType
	TPParameterType
	PduInfoType
Dem	Dem_DTCFormatType
	Dem_DTCKindType
	Dem_DTCOriginType
	Dem_DTCRequestType
	Dem_DTCSeverityType
	Dem_DTCTranslationFormatType
	Dem_ReturnControlDTCSettingType
	Dem_ReturnDisableDTCRecordUpdateType
	Dem_ReturnGetDTCByOccurrenceTimeType
	Dem_ReturnGetExtendedDataRecordByDTCType
	Dem_ReturnGetFreezeFrameDataByDTCType
	Dem_ReturnGetFunctionalUnitOfDTCType
	Dem_ReturnGetNextFilteredElementType
	Dem_ReturnGetNumberOfFilteredDTCType
	Dem_ReturnGetSeverityOfDTCType
	Dem_ReturnGetSizeOfDataByDTCType
	Dem_ReturnGetStatusOfDTCType
	Dem_ReturnSetFilterType
	Dem_UdsStatusByteType
GENERIC TYPES	<EcuSignalDataType>
	<datatype>
NvM	NvM_BlockIdType
SchM	SchM_ReturnType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌟()

8.2 Exported types

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

[SWS_Dcm_01001] ⌈ ImplementationDataType Dcm_OpStatusType

Name	Dcm_OpStatusType		
Kind	Type		
Derived from	uint8		
Description	--		
Range	DCM_INITIAL	0x00	Indicates the initial call to the operation
	DCM_PENDING	0x01	Indicates that a pending return has been done on the previous call of the operation
	DCM_CANCEL	0x02	Indicates that the DCM requests to cancel the pending operation
	DCM_FORCE_RCRRP_OK	0x03	Confirm a response pending transmission
Variation	--		

⌋()

[SWS_Dcm_01002] ⌈ ImplementationDataType Dcm_ConfirmationStatusType

Name	Dcm_ConfirmationStatusType		
Kind	Type		
Derived from	uint8		
Description	--		
Range	DCM_RES_POS_OK	0x00	--
	DCM_RES_POS_NOT_OK	0x01	--
	DCM_RES_NEG_OK	0x02	--
	DCM_RES_NEG_NOT_OK	0x03	--
Variation	--		

⌋()

[SWS_Dcm_01003] ⌈ ImplementationDataType _Dcm_SecLevelType

Name	Dcm_SecLevelType
Kind	Type

Derived from	uint8		
Description	Security Level type definition		
Range	DCM_SEC_LEV_LOCKED	0x00	--
	configuration dependent	0x01...0x3F	--
	Reserved by Document	0x40...0xFF	--
Variation	--		

」()

[SWS_Dcm_01004] 「 ImplementationDataType Dcm_SecCtrlType

Name	Dcm_SesCtrlType		
Kind	Type		
Derived from	uint8		
Description	Session type definition. 0, 127 and all values above 127 are reserved by ISO.		
Range	DCM_DEFAULT_SESSION	0x01	--
	DCM_PROGRAMMING_SESSION	0x02	--
	DCM_EXTENDED_DIAGNOSTIC_SESSION	0x03	--
	DCM_SAFETY_SYSTEM_DIAGNOSTIC_SESSION	0x04	--
	configuration dependent	0x40...0x7E	(according to "diagnosticSessionType" parameter of DiagnosticSessionControl request)
Variation	--		

」()

[SWS_Dcm_01005] 「 ImplementationDataType Dcm_ProtocolType

Name	Dcm_ProtocolType		
Kind	Type		
Derived from	uint8		
Description	Protocol type definition		
Range	DCM_OBD_ON_CAN	0x00	OBD on CAN (ISO15765-4; ISO15031-5)
	DCM_OBD_ON_FLEXRAY	0x01	(OBD on Flexray (Manufacturer specific;

			ISO15031-5))
	DCM_OBD_ON_IP	0x02	(OBD on Internet Protocol (Manufacturer specific; ISO15031-5))
	DCM_UDS_ON_CAN	0x03	UDS on CAN (ISO15765-3; ISO14229-1)
	DCM_UDS_ON_FLEXRAY	0x04	UDS on FlexRay (Manufacturer specific; ISO14229-1)
	DCM_UDS_ON_IP	0x05	(UDS on Internet Protocol (Manufacturer specific; ISO14229-1))
	DCM_ROE_ON_CAN	0x06	Response On Event on CAN
	DCM_ROE_ON_FLEXRAY	0x07	Response On Event on FlexRay
	DCM_ROE_ON_IP	0x08	(Response on Event on Internet Protocol)
	DCM_PERIODICTRANS_ON_CAN	0x09	Periodic Transmission on CAN
	DCM_PERIODICTRANS_ON_FLEXRAY	0x0A	Periodic Transmission on FlexRay
	DCM_PERIODICTRANS_ON_IP	0x0B	(Periodic Transmission on Internet Protocol)
	DCM_NO_ACTIVE_PROTOCOL	0x0C	No protocol has been started
	Reserved for further AUTOSAR implementation	0x0D..0xEF	--
	DCM_SUPPLIER_1	0xF0	Reserved for SW supplier specific.
	DCM_SUPPLIER_2	0xF1	Reserved for SW supplier specific.
	DCM_SUPPLIER_3	0xF2	Reserved for SW supplier specific.
	DCM_SUPPLIER_4	0xF3	Reserved for SW supplier specific.
	DCM_SUPPLIER_5	0xF4	Reserved for SW supplier specific.
	DCM_SUPPLIER_6	0xF5	Reserved for SW supplier specific.
	DCM_SUPPLIER_7	0xF6	Reserved for SW supplier specific.

	DCM_SUPPLIER_8	0xF7	Reserved for SW supplier specific.
	DCM_SUPPLIER_9	0xF8	Reserved for SW supplier specific.
	DCM_SUPPLIER_10	0xF9	Reserved for SW supplier specific.
	DCM_SUPPLIER_11	0xFA	Reserved for SW supplier specific.
	DCM_SUPPLIER_12	0xFB	Reserved for SW supplier specific.
	DCM_SUPPLIER_13	0xFC	Reserved for SW supplier specific.
	DCM_SUPPLIER_14	0xFD	Reserved for SW supplier specific.
	DCM_SUPPLIER_15	0xFE	Reserved for SW supplier specific.
Variation	--		

」()

[SWS_Dcm_01010] 「 ImplementationDataType Dcm_NegativeResponseCodeType

Name	Dcm_NegativeResponseCodeType		
Kind	Type		
Derived from	uint8		
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).		
Range	range of values 0x01..0x0F reserved by ISO 14229	0x01..0x0F	ISOSAERESRVD
	DCM_E_GENERALREJECT	0x10	GR
	DCM_E_SERVICENOTSUPPORTED	0x11	SNS
	DCM_E_SUBFUNCTIONNOTSUPPORTED	0x12	SFNS
	DCM_E_INCORRECTMESSAGELENGTHORINVALIDFORMAT	0x13	IMLOIF
	DCM_E_RESPONSETOOLONG	0x14	RTL
	range of values 0x15..0x20 reserved by ISO 14229	0x15..0x20	ISOSAERESRVD
	DCM_E_BUSYREPEATREQUEST	0x21	BRR
	DCM_E_CONDITIONSNOTCORRECT	0x22	CNC

	value 0x23 reserved by ISO 14229	0x23	ISOSAERESRVD
	DCM_E_REQUESTSEQUENCEERROR	0x24	RSE
	DCM_E_NORESPONSEFROMSUBNETCOMPONENT	0x25	NRFSC
	DCM_E_FAILUREPREVENTSEXECUTIONOFREQUESTEDACTION	0x26	FPEORA
	range of values 0x27..0x30 reserved by ISO 14229	0x27..0x30	ISOSAERESRVD
	DCM_E_REQUESTOUTOFRANGE	0x31	ROOR
	value 0x32 reserved by ISO 14229	0x32	ISOSAERESRVD
	DCM_E_SECURITYACCESSDENIED	0x33	SAD
	value 0x34 reserved by ISO 14229	0x34	ISOSAERESRVD
	DCM_E_INVALIDKEY	0x35	IK
	DCM_E_EXCEEDNUMBEROFATTEMPTS	0x36	ENOA
	DCM_E_REQUIREDTIMEDELAYNOTEXPIRED	0x37	RTDNE
	range of values 0x38..0x4F reserved by ISO 15764	0x38..0x4F	RBEDLSD
	range of values 0x50..0x6F reserved by ISO 14229	0x50..0x6F	ISOSAERESRVD
	DCM_E_UPLOADDOWNLOADNOTACCEPTED	0x70	UDNA
	DCM_E_TRANSFERDATASUSPENDED	0x71	TDS
	DCM_E_GENERALPROGRAMMINGFAILURE	0x72	GPF
	DCM_E_WRONGBLOCKSEQUENCECOUNTER	0x73	WBSC
	range of values 0x74..0x77 reserved by ISO 14229	0x74..0x77	ISOSAERESRVD
	range of values 0x79..0x7D reserved by ISO 14229	0x79..0x7D	ISOSAERESRVD
	DCM_E_SUBFUNCTIONNOTSUPPORTEDINACTIVESESSION	0x7E	SFNSIAS
	DCM_E_SERVICENOTSUPPORTEDINACTIVESESSION	0x7F	SNSIAS
	value 0x80 reserved by ISO 14229	0x80	ISOSAERESRVD
	DCM_E_RPMTOOHIGH	0x81	RPMTH
	DCM_E_RPMTOOLOW	0x82	RPMTL
	DCM_E_ENGINEISRUNNING	0x83	EIR
	DCM_E_ENGINEISNOTRUNNING	0x84	EINR
	DCM_E_ENGINERUNTIMETOOLOW	0x85	ERTTL
	DCM_E_TEMPERATURETOOHIGH	0x86	TEMPTH
	DCM_E_TEMPERATURETOOLOW	0x87	TEMPTL
	DCM_E_VEHICLESPEEDTOOHIGH	0x88	VSTH

	DCM_E_VEHICLESPEEDTOOLOW	0x89	VSTL
	DCM_E_THROTTLE_PEDALTOOHIGH	0x8A	TPTH
	DCM_E_THROTTLE_PEDALTOOLOW	0x8B	TPTL
	DCM_E_TRANSMISSIONRANGENOTINNEUTRAL	0x8C	TRNIN
	DCM_E_TRANSMISSIONRANGENOTINGEAR	0x8D	TRNIG
	value 0x8E reserved by ISO 14229	0x8E	ISOSAERESRVD
	DCM_E_BRAKESWITCH_NOTCLOSED	0x8F	BSNC
	DCM_E_SHIFTERLEVERNOTINPARK	0x90	SLNIP
	DCM_E_TORQUECONVERTERCLUTCHLOCKED	0x91	TCCL
	DCM_E_VOLTAGETOOHIGH	0x92	VTH
	DCM_E_VOLTAGETOOLOW	0x93	VTL
	range of values 0x94..0xEF reserved by ISO 14229	0x94..0xEF	RFSCNC
	DCM_E_VMSCNC_0	0xF0	VMSCNC
	DCM_E_VMSCNC_1	0xF1	VMSCNC1
	DCM_E_VMSCNC_2	0xF2	VMSCNC2
	DCM_E_VMSCNC_3	0xF3	VMSCNC3
	DCM_E_VMSCNC_4	0xF4	VMSCNC4
	DCM_E_VMSCNC_5	0xF5	VMSCNC5
	DCM_E_VMSCNC_6	0xF6	VMSCNC6
	DCM_E_VMSCNC_7	0xF7	VMSCNC7
	DCM_E_VMSCNC_8	0xF8	VMSCNC8
	DCM_E_VMSCNC_9	0xF9	VMSCNC9
	DCM_E_VMSCNC_A	0xFA	VMSCNCA
	DCM_E_VMSCNC_B	0xFB	VMSCNCB
	DCM_E_VMSCNC_C	0xFC	VMSCNCC
	DCM_E_VMSCNC_D	0xFD	VMSCNCD
	DCM_E_VMSCNC_E	0xFE	VMSCNCE
	value 0xFF reserved by ISO 14229	0xFF	ISOSAERESRVD
Variation	--		

」()

[SWS_Dcm_01011] 「 ImplementationDataType DataArrayType_{Data}

Name	DataArrayType_{Data}		
Kind	Array	Element type	uint8
Size	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataSize)}+7)/8) || (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidDataSize)}+7/8)) Elements		
Description	--		
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataSize)} > 0) && ({ecuc(Dcm/DcmConfigSet/DcmDspDcmDspData/DcmDspDataType)} == UINT8)) || (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidDataSize)}> 0) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/ DcmDspPidData/DcmDspPidService01/DcmDspPidDataType)} == UINT8)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})		

」()

[SWS_Dcm_01012] 「 ImplementationDataType DcmDspDidRangeArrayType_{Range}

Name	DcmDspDidRangeArrayType_{Range}		
Kind	Array	Element type	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRangeMaxDataLength)} Elements		
Description	--		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRange.DcmDspDidRangeUsePort)} == TRUE Range = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRange.SHORT-NAME)}		

」()

[SWS_Dcm_01013] 「 ImplementationDataType InfoTypeServicesArrayType_{VehInfoData}

Name	InfoTypeServicesArrayType_{VehInfoData}		
Kind	Array	Element type	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspVehInfo/DcmDspVehInfoData. DcmDspVehInfoDataSize)} Elements		
Description	--		
Variation	VehInfoData = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspVehInfo/ DcmDspVehInfoData.SHORT-NAME)}		

」()

[SWS_Dcm_01014] 「 ImplementationDataType RequestControlServicesInArrayType_{Tid}

Name	RequestControlServicesInArrayType_{Tid}		
Kind	Array	Element type	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestControl. DcmDspRequestControlInBufferSize)} Elements		
Description	--		
Variation	Tid = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestControl.SHORT-NAME)}		

」()

[SWS_Dcm_01015] 「 ImplementationDataType RequestControlServicesOutArrayType_{Tid}

Name	RequestControlServicesOutArrayType_{Tid}		
Kind	Array	Element type	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestControl.DcmDspRequestControlOutBufferSize)} Elements		
Description	--		
Variation	Tid = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestControl.SHORT-NAME)}		

」()

[SWS_Dcm_01017] 「 ImplementationDataType ScalingInfoArrayType_{Data}

Name	ScalingInfoArrayType_{Data}		
Kind	Array	Element type	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDataInfo.DcmDspDataScalingInfoSize)} Elements		
Description	--		
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData->DcmDspDataInfoRef.DcmDspDataScalingInfoSize)} != NULL) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}		

」()

[SWS_Dcm_01018] 「 ImplementationDataType RequestDataOutType_{Routine}_{Signal}

Name	RequestDataOutType_{Routine}_{Signal}	
Kind	Type	
Derivedfrom	*BaseType*	*Variation*
	boolean	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineRequestResOut.DcmDspRoutineRequestResOutSignal.DcmDspRoutineSignalType)} == BOOLEAN
	sint16	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineRequestResOut.DcmDspRoutineRequestResOutSignal.DcmDspRoutineSignalType)} == SINT16
	sint32	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineRequestResOut.DcmDspRoutineRequestResOutSignal.DcmDspRoutineSignalType)} == SINT32
	sint8	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineRequestResOut.DcmDspRoutineRequestResOutSignal.DcmDspRoutineSignalType)} == SINT8
	uint16	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineRequestResOut.DcmDspRoutineRequestResOutSignal.DcmDspRoutineSignalType)} == UINT16

	uint32	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineRequestResOut.DcmDspRoutineRequestResOutSignal. DcmDspRoutineSignalType)} == UINT32
	uint8	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineRequestResOut.DcmDspRoutineRequestResOutSignal. DcmDspRoutineSignalType)} == UINT8
Description	--	
Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineRequestResOut.DcmDspRoutineRequestResOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

」()

[SWS_Dcm_01019] 「 ImplementationDataType RequestFlexibleOutArrayDataType_{Routine}_{Signal}

Name	RequestFlexibleOutArrayDataType_{Routine}_{Signal}		
Kind	Array	Element type	uint8
Size	{(ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRequestRoutineOut. DcmDspRequestRoutineOutSignal.DcmDspRoutineSignalLength)+7)/8} Elements		
Description	--		
Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineRequestResOut. DcmDspRoutineRequestResOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineRequestResOut.DcmDspRoutineRequestResOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		

」()

[SWS_Dcm_01020] 「 ImplementationDataType StartDataInType_{Routine}_{Signal}

Name	StartDataInType_{Routine}_{Signal}	
Kind	Type	
Derivedfrom	*BaseType*	*Variation*
	boolean	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineIn. DcmDspStartRoutineInSignal.DcmDspStartRoutineSignalType)} == BOOLEAN
	sint16	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineIn. DcmDspStartRoutineInSignal.DcmDspStartRoutineSignalType)} == SINT16
	sint32	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineIn. DcmDspStartRoutineInSignal.DcmDspStartRoutineSignalType)} == SINT32
	sint8	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineIn. DcmDspStartRoutineInSignal.DcmDspStartRoutineSignalType)} == SINT8

	uint16	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineIn.DcmDspStartRoutineInSignal.DcmDspStartRoutineSignalType)} == UINT16
	uint32	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineIn.DcmDspStartRoutineInSignal.DcmDspStartRoutineSignalType)} == UINT32
	uint8	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineIn.DcmDspStartRoutineInSignal.DcmDspStartRoutineSignalType)} == UINT8
Description	--	
Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineStartIn.DcmDspRoutineStartInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

」()

[SWS_Dcm_01021]「ImplementationDataType StartDataOutType_{Routine}_{Signal}

Name	StartDataOutType_{Routine}_{Signal}	
Kind	Type	
Derivedfrom	*BaseType*	*Variation*
	boolean	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineOut.DcmDspStartRoutineOutSignal.DcmDspStartRoutineSignalType)} == BOOLEAN
	sint16	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineOut.DcmDspStartRoutineOutSignal.DcmDspStartRoutineSignalType)} == SINT16
	sint32	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineOut.DcmDspStartRoutineOutSignal.DcmDspStartRoutineSignalType)} == SINT32
	sint8	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineOut.DcmDspStartRoutineOutSignal.DcmDspStartRoutineSignalType)} == SINT8
	uint16	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineOut.DcmDspStartRoutineOutSignal.DcmDspStartRoutineSignalType)} == UINT16
	uint32	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineOut.DcmDspStartRoutineOutSignal.DcmDspStartRoutineSignalType)} == UINT32
	uint8	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineOut.DcmDspStartRoutineOutSignal.DcmDspStartRoutineSignalType)} == UINT8
Description	--	
Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineStartOut.DcmDspRoutineStartOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

」()

[SWS_Dcm_01022] ⌈ ImplementationDataType StartFlexibleInArrayDataType

Name	StartFlexibleInArrayDataType_{Routine}_{Signal}		
Kind	Array	Element type	uint8
Size	{(ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineIn.DcmDspStartRoutineInSignal.DcmDspRoutineSignalLength)+7)/8} Elements		
Description	--		
Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineStartIn.DcmDspRoutineStartInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineStartIn.DcmDspRoutineStartIn.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		

⌋()

[SWS_Dcm_01023] ⌈ ImplementationDataType StartFlexibleOutArrayDataType_{Routine}_{Signal}

Name	StartFlexibleOutArrayDataType_{Routine}_{Signal}		
Kind	Array	Element type	uint8
Size	{(ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStartRoutineOut.DcmDspStartRoutineOutSignal.DcmDspRoutineSignalLength)+7)/8} Elements		
Description	--		
Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineStartOut.DcmDspRoutineStartOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineStartOut.DcmDspRoutineStartOut.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		

⌋()

[SWS_Dcm_01024] ⌈ ImplementationDataType StopDataInType_{Routine}_{Signal}

Name	StopDataInType_{Routine}_{Signal}	
Kind	Type	
Derivedfrom	*BaseType*	*Variation*
	boolean	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineIn.DcmDspStopRoutineInSignal.DcmDspStopRoutineSignalType)} == BOOLEAN
	sint16	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineIn.DcmDspStopRoutineInSignal.DcmDspStopRoutineSignalType)} == SINT16
	sint32	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineIn.DcmDspStopRoutineInSignal.DcmDspStopRoutineSignalType)} == SINT32
	sint8	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineIn.DcmDspStopRoutineInSignal.DcmDspStopRoutineSignalType)} == SINT8
	uint16	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineIn.DcmDspStopRoutineInSignal.DcmDspStopRoutineSignalType)} ==

		UINT16
	uint32	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineIn.DcmDspStopRoutineInSignal.DcmDspStopRoutineSignalType)} == UINT32
	uint8	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineIn.DcmDspStopRoutineInSignal.DcmDspStopRoutineSignalType)} == UINT8
Description	--	
Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineStopIn.DcmDspRoutineStopInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

」()

[SWS_Dcm_01025] 「 ImplementationDataType StopDataOutType_{Routine}_{Signal}

Name	StopDataOutType_{Routine}_{Signal}	
Kind	Type	
Derivedfrom	*BaseType*	*Variation*
	boolean	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineOut.DcmDspStopRoutineOutSignal.DcmDspStopRoutineSignalType)} == BOOLEAN
	sint16	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineOut.DcmDspStopRoutineOutSignal.DcmDspStopRoutineSignalType)} == SINT16
	sint32	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineOut.DcmDspStopRoutineOutSignal.DcmDspStopRoutineSignalType)} == SINT32
	sint8	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineOut.DcmDspStopRoutineOutSignal.DcmDspStopRoutineSignalType)} == SINT8
	uint16	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineOut.DcmDspStopRoutineOutSignal.DcmDspStopRoutineSignalType)} == UINT16
	uint32	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineOut.DcmDspStopRoutineOutSignal.DcmDspStopRoutineSignalType)} == UINT32
	uint8	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineOut.DcmDspStopRoutineOutSignal.DcmDspStopRoutineSignalType)} == UINT8
Description	--	
Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineStopOut.DcmDspRoutineStopOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

」()

[SWS_Dcm_01026] 「 ImplementationDataType StopFlexibleInArrayDataType_{Routine}_{Signal}

Name	StopFlexibleInArrayDataType_{Routine}_{Signal}		
Kind	Array	Element type	uint8
Size	{(ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineIn. DcmDspStopRoutineInSignal.DcmDspRoutineSignalLength)+7)/8} Elements		
Description	--		
Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineStopIn. DcmDspRoutineStopInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStopIn.DcmDspRoutineStoptIn.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		

」()

[SWS_Dcm_01027] 「 ImplementationDataType StopFlexibleOutArrayDataType_{Routine}_{Signal}

Name	StopFlexibleOutArrayDataType_{Routine}_{Signal}		
Kind	Array	Element type	uint8
Size	{(ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspStopRoutineOut. DcmDspStopRoutineOutSignal.DcmDspRoutineSignalLength)+7)/8} Elements		
Description	--		
Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineStopOut. DcmDspRoutineStopOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStopOut.DcmDspRoutineStoptOut.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		

」()

[SWS_Dcm_01028] 「 ImplementationDataType KeyArrayType_{SecurityLevel}

Name	KeyArrayType_{SecurityLevel}		
Kind	Array	Element type	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow. DcmDspSecurityKeySize)} Elements		
Description	--		
Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/ DcmDspSecurityRow.SHORT-NAME)}		

」()

[SWS_Dcm_01029] 「 ImplementationDataType SeedArrayType_{SecurityLevel}

Name	SeedArrayType_{SecurityLevel}		
Kind	Array	Element type	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow. DcmDspSecuritySeedSize)} Elements		
Description	--		
Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/		

	DcmDspSecurityRow.SHORT-NAME)}

」()

8.3 Type Definitions

The Dcm module shall ensure that implementation-specific types are not “visible” outside of Dcm. Otherwise, the complete architecture would be corrupted.」(BSW00353)

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

8.3.1 Dcm_StatusType

[SWS_Dcm_00976]「

Name:	Dcm_StatusType		
Type:	uint8		
Range:	DCM_E_OK	0x00	This value is representing a successful operation.
	DCM_E_ROE_NOT_ACCEPTED	0x06	ResponseOnOneEvent request is not accepted by DCM (e.g. old ResponseOnOneEvent is not finished) (used at API: Dcm_ResponseOnOneEvent())
	DCM_E_PERIODICID_NOT_ACCEPTED	0x07	Periodic transmission request is not accepted by DCM (e.g. old Periodic transmission is not finished) (used at API: Dcm_ResponseOnOneDataByPeriodicId())
Description:	Base item type to transport status information.		

」()

8.3.2 Dcm_SecLevelType

This type is defined in Rte_Dcm_Type.h header file, which is generated by the RTE generator.

[SWS_Dcm_00977]「

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		

」()

8.3.3 Dcm_SesCtrlType

This type is defined in Rte_Dcm_Type.h header file, which is generated by the RTE generator.

[SWS_Dcm_00978]「

Name:	Dcm_SesCtrlType		
Type:	uint8		
Range:	DCM_DEFAULT_SESSION	0x01	--
	DCM_PROGRAMMING_SESSION	0x02	--
	DCM_EXTENDED_DIAGNOSTIC_SESSION	0x03	--
	DCM_SAFETY_SYSTEM_DIAGNOSTIC_SESSION	0x04	--
	configuration dependent	0x40...0x7E	(according to "diagnosticSessionType" parameter of DiagnosticSessionControl request)
Description:	Session type definition. 0, 127 and all values above 127 are reserved by ISO.		

」()

[SWS_Dcm_00941]「 The type definition of Dcm_SesCtrlType in the standardized AUTOSAR interface shall include the required prefix "DCM_".」()

Note: In case the DCM generator creates the AUTOSAR interfaces, the prefix DCM_ needs to be added. This implies that in case the shortname of DcmDspSessionRow is already prefixed, the resulting enumeration is prefixed twice (shortname of DcmDspSessionRow "DCM_TEST" --> Dcm_SesCtrlType with "DCM_DCM_TEST")

8.3.4 Dcm_ProtocolType

This type is defined in Rte_Dcm_Type.h header file, which is generated by the RTE generator.

[SWS_Dcm_00979]「

Name:	Dcm_ProtocolType		
Type:	uint8		
Range:	DCM_OBD_ON_CAN	0x00	OBD on CAN (ISO15765-4; ISO15031-5)
	DCM_OBD_ON_FLEXRAY	0x01	(OBD on Flexray (Manufacturer specific; ISO15031-5))

	DCM_OBD_ON_IP	0x02	(OBD on Internet Protocol (Manufacturer specific; ISO15031-5))
	DCM_UDS_ON_CAN	0x03	UDS on CAN (ISO15765-3; ISO14229-1)
	DCM_UDS_ON_FLEXRAY	0x04	UDS on FlexRay (Manufacturer specific; ISO14229-1)
	DCM_UDS_ON_IP	0x05	(UDS on Internet Protocol (Manufacturer specific; ISO14229-1))
	DCM_ROE_ON_CAN	0x06	Response On Event on CAN
	DCM_ROE_ON_FLEXRAY	0x07	Response On Event on FlexRay
	DCM_ROE_ON_IP	0x08	(Response on Event on Internet Protocol)
	DCM_PERIODICTRANS_ON_CAN	0x09	Periodic Transmission on CAN
	DCM_PERIODICTRANS_ON_FLEXRAY	0x0A	Periodic Transmission on FlexRay
	DCM_PERIODICTRANS_ON_IP	0x0B	(Periodic Transmission on Internet Protocol)
	DCM_NO_ACTIVE_PROTOCOL	0x0C	No protocol has been started
	Reserved for further AUTOSAR implementation	0x0D..0xEF	--
	DCM_SUPPLIER_1	0xF0	Reserved for SW supplier specific.
	DCM_SUPPLIER_2	0xF1	Reserved for SW supplier specific.
	DCM_SUPPLIER_3	0xF2	Reserved for SW supplier specific.
	DCM_SUPPLIER_4	0xF3	Reserved for SW supplier specific.
	DCM_SUPPLIER_5	0xF4	Reserved for SW supplier specific.
	DCM_SUPPLIER_6	0xF5	Reserved for SW supplier specific.
	DCM_SUPPLIER_7	0xF6	Reserved for SW supplier specific.
	DCM_SUPPLIER_8	0xF7	Reserved for SW supplier specific.
	DCM_SUPPLIER_9	0xF8	Reserved for SW supplier specific.
	DCM_SUPPLIER_10	0xF9	Reserved for SW supplier specific.
	DCM_SUPPLIER_11	0xFA	Reserved for SW supplier specific.
	DCM_SUPPLIER_12	0xFB	Reserved for SW supplier specific.
	DCM_SUPPLIER_13	0xFC	Reserved for SW supplier specific.
	DCM_SUPPLIER_14	0xFD	Reserved for SW supplier specific.
	DCM_SUPPLIER_15	0xFE	Reserved for SW supplier specific.
Description:	Protocol type definition		

8.3.5 Dcm_NegativeResponseCodeType

This type is defined in Rte_Dcm_Type.h header file, which is generated by the RTE generator.

[SWS_Dcm_00980]⌈

Name:	Dcm_NegativeResponseCodeType		
Type:	uint8		
Range:	range of values 0x01..0x0F reserved by ISO 14229	0x01..0x0F	ISOSAERESRVD
	DCM_E_GENERALREJECT	0x10	GR
	DCM_E_SERVICENOTSUPPORTED	0x11	SNS
	DCM_E_SUBFUNCTIONNOTSUPPORTED	0x12	SFNS
	DCM_E_INCORRECTMESSAGELENGTHORINVALIDFORMAT	0x13	IMLOIF
	DCM_E_RESPONSETOOLONG	0x14	RTL
	range of values 0x15..0x20 reserved by ISO 14229	0x15..0x20	ISOSAERESRVD
	DCM_E_BUSYREPEATREQUEST	0x21	BRR
	DCM_E_CONDITIONSNOTCORRECT	0x22	CNC
	value 0x23 reserved by ISO 14229	0x23	ISOSAERESRVD
	DCM_E_REQUESTSEQUENCEERROR	0x24	RSE
	DCM_E_NORESPONSEFROMSUBNETCOMPONENT	0x25	NRFSC
	DCM_E_FAILUREPREVENTSEXECUTIONOFREQUESTEDACTION	0x26	FPEORA
	range of values 0x27..0x30 reserved by ISO 14229	0x27..0x30	ISOSAERESRVD
	DCM_E_REQUESTOUTOFRANGE	0x31	ROOR
	value 0x32 reserved by ISO 14229	0x32	ISOSAERESRVD
	DCM_E_SECURITYACCESSDENIED	0x33	SAD
	value 0x34 reserved by ISO 14229	0x34	ISOSAERESRVD
	DCM_E_INVALIDKEY	0x35	IK
	DCM_E_EXCEEDNUMBEROFATTEMPTS	0x36	ENOA
	DCM_E_REQUIREDTIMEDELAYNOTEXPIRED	0x37	RTDNE
	range of values 0x38..0x4F reserved by ISO 15764	0x38..0x4F	RBEDLSD
	range of values 0x50..0x6F reserved by ISO 14229	0x50..0x6F	ISOSAERESRVD
	DCM_E_UPLOADDOWNLOADNOTACCEPTED	0x70	UDNA
	DCM_E_TRANSFERDATASUSPENDED	0x71	TDS
	DCM_E_GENERALPROGRAMMINGFAILURE	0x72	GPF
	DCM_E_WRONGBLOCKSEQUENCECOUNTER	0x73	WBSC
	range of values 0x74..0x77 reserved by ISO 14229	0x74..0x77	ISOSAERESRVD
	range of values 0x79..0x7D reserved by ISO 14229	0x79..0x7D	ISOSAERESRVD
	DCM_E_SUBFUNCTIONNOTSUPPORTEDINACTIVESESSION	0x7E	SFNSIAS
	DCM_E_SERVICENOTSUPPORTEDINACTIVESESSION	0x7F	SNSIAS
	value 0x80 reserved by ISO 14229	0x80	ISOSAERESRVD
	DCM_E_RPMTOOHIGH	0x81	RPMTH
	DCM_E_RPMTOOLOW	0x82	RPMTL
	DCM_E_ENGINEISRUNNING	0x83	EIR
	DCM_E_ENGINEISNOTRUNNING	0x84	EINR
	DCM_E_ENGINERUNTIMETOOLOW	0x85	ERTTL

	DCM_E_TEMPERATURETOOHIGH	0x86	TEMPTH
	DCM_E_TEMPERATURETOOLOW	0x87	TEMPTL
	DCM_E_VEHICLESPEEDTOOHIGH	0x88	VSTH
	DCM_E_VEHICLESPEEDTOOLOW	0x89	VSTL
	DCM_E_THROTTLE_PEDALTOOHIGH	0x8A	TPTH
	DCM_E_THROTTLE_PEDALTOOLOW	0x8B	TPTL
	DCM_E_TRANSMISSIONRANGENOTINNEUTRAL	0x8C	TRNIN
	DCM_E_TRANSMISSIONRANGENOTINGEAR	0x8D	TRNIG
	value 0x8E reserved by ISO 14229	0x8E	ISOSAERESRVD
	DCM_E_BRAKESWITCH_NOTCLOSED	0x8F	BSNC
	DCM_E_SHIFTERLEVERNOTINPARK	0x90	SLNIP
	DCM_E_TORQUECONVERTERCLUTCHLOCKED	0x91	TCCL
	DCM_E_VOLTAGETOOHIGH	0x92	VTH
	DCM_E_VOLTAGETOOLOW	0x93	VTL
	range of values 0x94..0xEF reserved by ISO 14229	0x94..0xEF	RFSCNC
	DCM_E_VMSCNC_0	0xF0	VMSCNC
	DCM_E_VMSCNC_1	0xF1	VMSCNC1
	DCM_E_VMSCNC_2	0xF2	VMSCNC2
	DCM_E_VMSCNC_3	0xF3	VMSCNC3
	DCM_E_VMSCNC_4	0xF4	VMSCNC4
	DCM_E_VMSCNC_5	0xF5	VMSCNC5
	DCM_E_VMSCNC_6	0xF6	VMSCNC6
	DCM_E_VMSCNC_7	0xF7	VMSCNC7
	DCM_E_VMSCNC_8	0xF8	VMSCNC8
	DCM_E_VMSCNC_9	0xF9	VMSCNC9
	DCM_E_VMSCNC_A	0xFA	VMSCNCA
	DCM_E_VMSCNC_B	0xFB	VMSCNCB
	DCM_E_VMSCNC_C	0xFC	VMSCNCC
	DCM_E_VMSCNC_D	0xFD	VMSCNCD
	DCM_E_VMSCNC_E	0xFE	VMSCNCE
	value 0xFF reserved by ISO 14229	0xFF	ISOSAERESRVD
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.

8.3.6 Dcm_CommunicationModeType

[SWS_Dcm_00981]「

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:	--		

⌋()

8.3.7 Dcm_ConfigType

[SWS_Dcm_00982]⌈

Name:	Dcm_ConfigType	
Type:	Structure	
Range:	Implementation specific	--
Description:	This type defines a data structure for the post build parameters of the DCM . At initialization the DCM gets a pointer to a structure of this type to get access to its configuration data, which is necessary for initialization.	

⌋()

8.3.8 Dcm_ConfirmationStatusType

[SWS_Dcm_00983]⌈

Name:	Dcm_ConfirmationStatusType		
Type:	uint8		
Range:	DCM_RES_POS_OK	0x00	--
	DCM_RES_POS_NOT_OK	0x01	--

	DCM_RES_NEG_OK	0x02	--
	DCM_RES_NEG_NOT_OK	0x03	--
Description:	--		

⌋()

8.3.9 Dcm_OpStatusType

[SWS_Dcm_00984]⌈

Name:	Dcm_OpStatusType		
Type:	uint8		
Range:	DCM_INITIAL	0x00	Indicates the initial call to the operation
	DCM_PENDING	0x01	Indicates that a pending return has been done on the previous call of the operation
	DCM_CANCEL	0x02	Indicates that the DCM requests to cancel the pending operation
	DCM_FORCE_RCRRP_OK	0x03	Confirm a response pending transmission
Description:	--		

⌋()

For the operation using the Dcm_OpStatusType, the DCM shall work as follow :

[SWS_Dcm_00527] ⌈ At first call of an operation using the Dcm_OpStatusType, the DCM call the operation with OpStatus = DCM_INITIAL ⌋()

[SWS_Dcm_00528] ⌈ If the value DCM_E_FORCE_RCRRP is returned from an operation using Dcm_OpStatusType, the DCM shall invoke the transmit request for RCR-RP (NRC 0x78 transmission) and the DCM shall not realize further invocation of the operation till RCR-RP is transmitted (call to Dcm_Confirmation). ⌋()

[SWS_Dcm_00529] ⌈ After transmit confirmation of a RCR-RP transmitted on the context of **[SWS_Dcm_00528**, the DCM calls, from Dcm_MainFunction (due to call context), the operation again with OpStatus = DCM_FORCE_RCRRP_OK. ⌋()

[SWS_Dcm_00530] ⌈ If a DCM_E_PENDING value is returned from an operation using the Dcm_OpStatusType, the DCM call the operation on each Dcm_MainFunction call with OpStatus = DCM_PENDING as long as DCM_E_PENDING is returned. ⌋()

8.3.10 Dcm_ReturnReadMemoryType

[SWS_Dcm_00985]⌜

Name:	Dcm_ReturnReadMemoryType		
Type:	uint8		
Range:	DCM_READ_OK	0x00	Reading has been done
	DCM_READ_PENDING	0x01	Reading is pending, another call is request to finalize the reading
	DCM_READ_FAILED	0x02	Reading has failed
	DCM_READ_FORCE_RCRRP	0x03	Reading is pending, the Response pending transmission starts immediately
Description:	Return values of Callout Dcm_ReadMemory		

⌟()

8.3.11 Dcm_ReturnWriteMemoryType

[SWS_Dcm_00986]⌜

Name:	Dcm_ReturnWriteMemoryType		
Type:	uint8		
Range:	DCM_WRITE_OK	0x00	Writing has been done
	DCM_WRITE_PENDING	0x01	Writing is pending, another called is requested
	DCM_WRITE_FAILED	0x02	The writing has failed
	DCM_WRITE_FORCE_RCRRP	0x03	Writing is pending, the Response pending transmission starts immediately
Description:	Return type of callout Dcm_WriteMemory		

⌟()

8.3.12 Dcm_EcuStartModeType

[SWS_Dcm_00987]⌜

Name:	Dcm_EcuStartModeType		
Type:	uint8		
Range:	DCM_COLD_START	0x00	The ECU starts normally
	DCM_WARM_START	0x01	The ECU starts from a bootloader jump
Description:	Allows the DCM to know if a diagnostic response shall be sent in the case of a jump from bootloader		

⌟()

8.3.13 Dcm_ProgConditionsType

[SWS_Dcm_00988]⌜

Name:	Dcm_ProgConditionsType
Type:	Structure

Element:	uint16	TesterSourceAddr	Tester source address configured per protocol
	uint8	ProtocolId	Id of the protocol on wich the request has been received
	uint8	Sid	Service identifier of the received request
	uint8	SubFncId	Identifier of the received subfonction
	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.		

⌋()

8.3.14 Dcm_MsgItemType

[SWS_Dcm_00989]⌈

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

⌋()

8.3.15 Dcm_MsgType

[SWS_Dcm_00990]⌈

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

⌋()

8.3.16 Dcm_MsgLenType

[SWS_Dcm_00991]⌈

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.

⌋()

8.3.17 Dcm_MsgAddInfoType

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

[SWS_Dcm_00992]⌈

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

⌋()

8.3.18 Dcm_IdContextType

[SWS_Dcm_00993]⌈

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

⌋()

8.3.19 Dcm_MsgContextType

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

[SWS_Dcm_00994]⌈

Name:	Dcm_MsgContextType		
Type:	Structure		
Element:	Dcm_MsgType	reqData	Request data, starting directly after service identifier (which is not part of this data)
	Dcm_MsgLenType	reqDataLen	Request 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	resDataLen	Positive response data length (excluding service identifier)
	Dcm_MsgAddInfoType	msgAddInfo	Additional information about service request and response (see: Dcm_MsgAddInfo)
	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_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.
	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
Description:	This data structure contains all information which is necessary to process a diagnostic message from request to response and response confirmation.		

⌋()

8.4 Function definitions

This section defines the functions provided for other modules.

8.4.1 Functions provided for other BSW components

8.4.1.1 Dcm_Init

[**SWS_Dcm_00037**] ⌈

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 set in Variant Post-Build.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Service for basic initialization of DCM module.	

⌋(BSW00438, BSW101, BSW00358, BSW00414)

[SWS_Dcm_00334] ⌈ `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

[**SWS_Dcm_00065**] ⌈

Service name:	Dcm_GetVersionInfo	

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

Service ID[hex]:	0x24	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versionInfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Returns the version information of this module	

⌋(BSW00407)

8.4.1.3 Dcm_DemTriggerOnDTCStatus

[SWS_Dcm_00614] ⌈

Service name:	Dcm_DemTriggerOnDTCStatus

Syntax:	`Std_ReturnType Dcm_DemTriggerOnDTCStatus(`
	`  uint32 DTC,`
	`  Dem_UdsStatusByteType DTCStatusOld,`
	`  Dem_UdsStatusByteType DTCStatusNew`
	`)`

Service ID[hex]:	0x2B
Sync/Async:	Synchronous

Reentrancy:	Non Reentrant	
Parameters (in):	DTC	This is the DTC the change trigger is assigned to.
	DTCStatusOld	DTC status before change
	DTCStatusNew	DTC status after change
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	Triggers on changes of the UDS DTC status byte. Allows to trigger on ROE Event for subservice OnDTCStatusChanged.	

」()

8.4.1.4 Dcm_GetVin

[SWS_Dcm_00950] 「

Service name:	Dcm_GetVin	
Syntax:	`Std_ReturnType Dcm_GetVin( uint8* Data )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	Data	Pointer to where to store the VIN
Return value:	Std_ReturnType	E_OK: The Data pointer has been filled with valid VIN E_NOT_OK: The default VIN will be used in the DoIP
Description:	Callbackfunction to get the VIN.	

」()

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.

[**SWS_Dcm_00698**] 「 ClientServerInterface DCMServices

Name	DCMServices	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK

	1	E_NOT_OK

Operations

GetActiveProtocol		
Comments	--	
Variation	--	
Parameters	ActiveProtocol	
	Comment	--
	Type	Dcm_ProtocolType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful

GetSecurityLevel		
Comments	--	
Variation	--	
Parameters	SecLevel	
	Comment	--
	Type	Dcm_SecLevelType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

GetSesCtrlType		
Comments	--	
Variation	--	
Parameters	SesCtrlType	
	Comment	--
	Type	Dcm_SesCtrlType
	Variation	--

	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
ResetToDefaultSession		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

⌋()

[**SWS_Dcm_00699**]⌈ ClientServerInterface Dcm_Roe
The RoeEventId shall be a Portdefined argument value.

Name	Dcm_Roe	
Comment	--	
IsService	true	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoe/DcmDspRoeEvent)}	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

TriggerOnEvent		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

⌋()

8.4.2.1 Dcm_GetSecurityLevel

[SWS_Dcm_00338] ⌈

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.	

」(BSW04011)

8.4.2.2 Dcm_GetSesCtrlType

[SWS_Dcm_00339] 「

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.	

」(BSW04011)

8.4.2.3 Dcm_GetActiveProtocol

[SWS_Dcm_00340] 「

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.	

」(BSW04011)

8.4.2.4 Dcm_ResetToDefaultSession

[SWS_Dcm_00520] ⌜

Service name:	Dcm_ResetToDefaultSession	
Syntax:	`Std_ReturnType Dcm_ResetToDefaultSession( void )`	
Service ID[hex]:	0x2a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	The call to this function allows the application to reset the current session to Default session. Example: Automatic termination of an extended diagnostic session upon exceeding of a speed limit.	

⌟()

8.4.2.5 Dcm_TriggerOnEvent

[SWS_Dcm_00521] ⌜

Service name:	Dcm_TriggerOnEvent	
Syntax:	`Std_ReturnType Dcm_TriggerOnEvent( uint8 RoeEventId )`	
Service ID[hex]:	0x2D	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	RoeEventId	Identifier of the event that is triggered
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: RoeEventId value is valid E_NOT_OK: RoeEventId value is not valid
Description:	The call to this function allows to trigger an event linked to a ResponseOnEvent request. On the function call, the DCM will execute the associated service if the corresponding Mode of the RoeEventId is 'ROE started'.	

⌟()

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

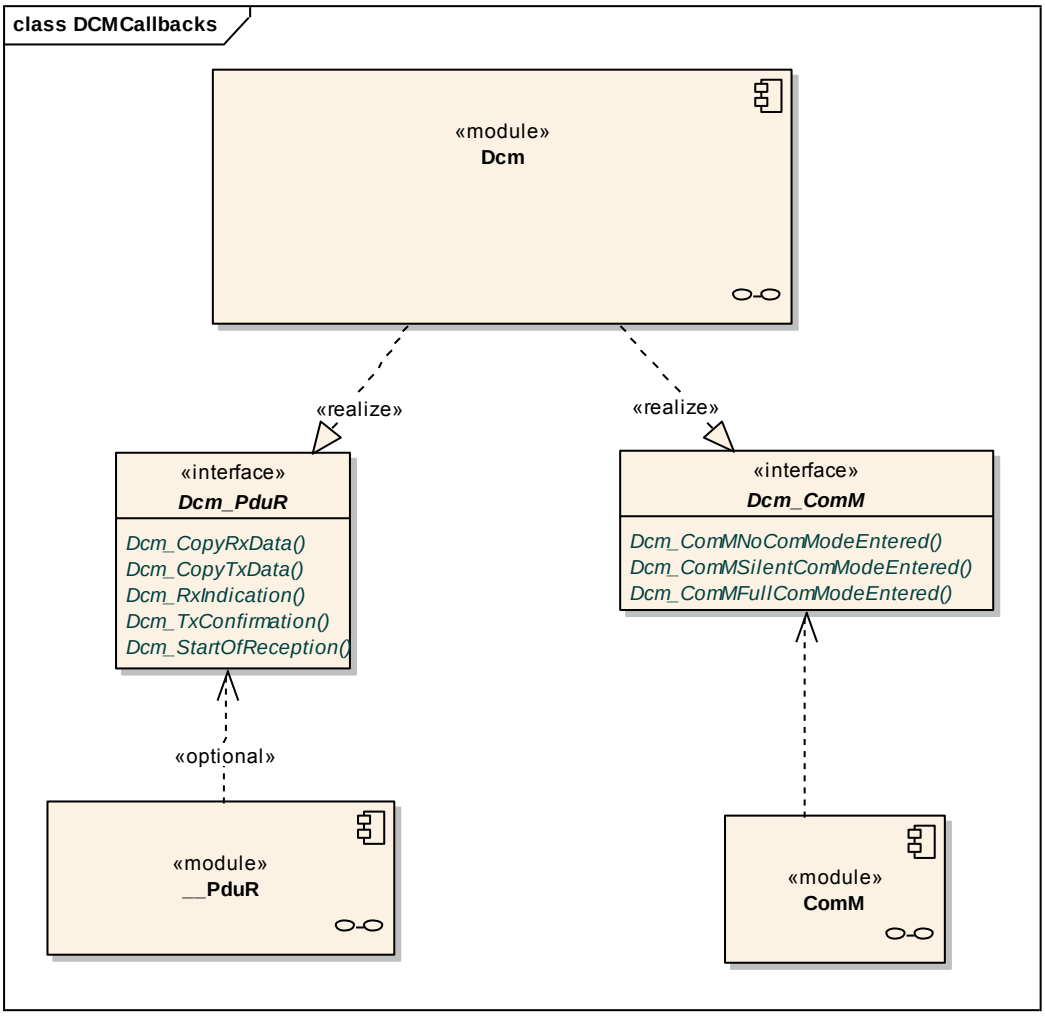


Figure 9: Overview of the callbacks provided by the DCM

8.5.1 Dcm_StartOfReception

[SWS_Dcm_00094] ⌜

Service name:	Dcm_StartOfReception	

Syntax:	`BufReq_ReturnType Dcm_StartOfReception(`
	`  PduIdType DcmRxPduId,`
	`  const PduInfoType* info,`
	`  PduLengthType TpSduLength,`
	`  PduLengthType* RxBufferSizePtr`
	`)`	

Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non 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.
	info	Pointer to a structure containing content and length of the first frame or single frame including MetaData.
	TpSduLength	This length identifies the overall number of bytes to be received.

Parameters (inout):	None	
Parameters (out):	RxBufferSizePtr	Length of the available buffer
Return value:	BufReq_ReturnType	BUFREQ_OK: Connection has been accepted, RxBufferSizePtr indicates the available receive buffer. BUFREQ_E_NOT_OK: Connection has been rejected. BUFREQ_E_OVFL: No buffer of the required length can be provided, reception is aborted.
Description:	Called once to initialize the reception of a diagnostic request	

」()

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

[SWS_Dcm_00444] 「 If the requested size is large than the buffer available in the DCM, the function `Dcm_StartOfReception()` shall return BUFREQ_E_OVFL (see SWS_Dcm_00094).」()

[SWS_Dcm_00788] 「When processing a diagnostic request, the DCM module shall accept (`Dcm_StartOfReception()` shall return BUFREQ_OK) any new request using a different DcmDslConnection in case ***DcmDslDiagRespOnSecondDeclinedRequest*** is set to TRUE.」()

[SWS_Dcm_00789] 「In case **[SWS_Dcm_00788**, the Dcm respond with a NRC 0x21」()

[SWS_Dcm_00790] 「When processing a diagnostic request, the DCM module shall reject (`Dcm_StartOfReception()` shall return BUFREQ_E_NOT_OK) any new request using a different DcmDslConnection in case ***DcmDslDiagRespOnSecondDeclinedRequest*** is set to FALSE until the current diagnostic request processing is over.」()

[SWS_Dcm_00557] 「 When processing a diagnostic request, the DCM module shall reject (`Dcm_StartOfReception()` shall return BUFREQ_E_NOT_OK) any new diagnostic request with the same DcmDslConnection until the current diagnostic request processing is over (except for concurrent TesterPresent requests).」()

[SWS_Dcm_00642]「When the API Dcm_StartOfReception is invoked with TpSduLength equal to 0, the value BUFREQ_OK shall be returned and RxBufferSizePtr shall be set to the configured size of the allocated Rx buffer..」()

[SWS_Dcm_00655]「If the current session is a non-default session and a new diagnostic request with same or lower priority protocol than active one is detected, the DCM shall act according **[SWS_Dcm_00788**, **[SWS_Dcm_00789** and **[SWS_Dcm_00790**.」()

[SWS_Dcm_00656]「If the current session is the default session and a diagnostic request is in execution, for any new diagnostic request with same or lower priority protocol than active one, the DCM shall act according **[SWS_Dcm_00788**, **[SWS_Dcm_00789** and **[SWS_Dcm_00790**. 」()

[SWS_Dcm_00833]「Dcm_StartOfReception () shall be callable in interrupt context.」()

8.5.2 Dcm_CopyRxData

[SWS_Dcm_00556]「

Service name:	Dcm_CopyRxData	
Syntax:	`BufReq_ReturnType Dcm_CopyRxData( PduIdType DcmRxPduId, const PduInfoType* PduInfoPtr, PduLengthType* RxBufferSizePtr )`	
Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
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.
	PduInfoPtr	Pointer to a PduInfoType which indicates the number of bytes to be copied (SduLength) and the location of the source data (SduDataPtr). An SduLength of 0 is possible in order to poll the available receive buffer size. In this case no data are to be copied and PduInfoPtr might be invalid.
Parameters (inout):	None	
Parameters (out):	RxBufferSizePtr	Remaining free place in receive buffer after completion of this call.
Return value:	BufReq_ReturnType	BUFREQ_OK: Data has been copied to the receive buffer completely as requested.

		BUFREQ_E_NOT_OK: Data has not been copied. Request failed.
Description:	Called once upon reception of each segment. Within this call, the received data is copied from the receive TP buffer to the DCM receive buffer. The API should only be called with an SduLength greater 0 if the RxBufferSizePtr returned by the previous API call indicates sufficient receive buffer (SduLength ≤ RxBufferSizePtr). When the API Dcm_CopyRxData is invoked with SduLength from PduInfoPtr equal to 0, the value BUFREQ_OK shall be returned and RxBufferSizePtr shall be filled with the remaining size of the Rx buffer. The function must only be called if the connection has been accepted by an initial call to Dcm_StartOfReception.	

」()

[SWS_Dcm_00443]「 If `Dcm_StartOfReception()` returns BUFREQ_OK, the further call to `Dcm_CopyRxData()` shall copy the data from the buffer provided inPduInfoPointer parameter) to the DCM buffer and update the RxBufferSizePtr parameter with remaining free place in DCM receive buffer after completion of this call.」()

[SWS_Dcm_00996]「When the API `Dcm_CopyRxData` is invoked with SduLength from PduInfoPtr equal to 0, the value BUFREQ_OK shall be returned and RxBufferSizePtr shall be filled with the remaining size of the Rx buffer.」()

Note: The size of the Rx buffer is based on the buffer length, which is returned in the parameter RxBufferSizePtr of API `Dcm_StartOfReception()`.

[SWS_Dcm_00342]「 After starting to copy the received data (see SWS_Dcm_00443), the DCM module shall not access the receive buffer until it is notified by the service `Dcm_TpRxIndication()` about the successful completion or unsuccessful termination of the reception.」()

Note: Dcm_TpRxIndication is only expected when Dcm_StartOfReception succeeded

[SWS_Dcm_00831]「Dcm_CopyRxData() shall be callable in interrupt context.」()

8.5.3 Dcm_TpRxIndication

[SWS_Dcm_00093]「

Service name:	Dcm_TpRxIndication

Syntax:	`void Dcm_TpRxIndication(`
	`  PduIdType DcmRxPduId,`
	`  Std_ReturnType 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	E_OK: the complete N-PDU has been received and is stored in the receive buffer E_NOT_OK: the N_PDU has not been received properly, Dcm should prepare for a new reception.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This is called by the PduR to indicate the completion of a reception	

」()

[SWS_Dcm_00344] 「 If `Dcm_TpRxIndication()` is called with parameter Result different from E_OK, then the DCM module shall not evaluate the buffer assigned to the I-PDU, which is referenced in parameter DcmRxPduId. 」()

Rationale for SWS_Dcm_00344: It is undefined which part of the buffer contains valid data in this case

[SWS_Dcm_00345] 「 `Dcm_TpRxIndication()` shall be callable in interrupt context. 」()

8.5.4 Dcm_CopyTxData

[SWS_Dcm_00092] 「

Service name:	Dcm_CopyTxData	

Syntax:	`BufReq_ReturnType Dcm_CopyTxData(`
	`  PduIdType DcmTxPduId,`
	`  const PduInfoType* PduInfoPtr,`
	`  RetryInfoType* RetryInfoPtr,`
	`  PduLengthType* TxDataCntPtr`
	`)`	

Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	DcmTxPduId	Represents the TX PduId configured within the DCM (DcmDslProtocolTx-->DcmDslTxConfirmationPduId). Through this Id the DCM is able to determine the corresponding protocol for the communication. Range: 0..(maximum number of I-PDU IDs transmitted by DCM) – 1
	PduInfoPtr	Provides the destination buffer and the number of bytes to be copied. If not enough transmit data is available, no data is copied. The transport protocol module may retry. A copy size of 0 can be used to indicate state changes in the retry parameter or to query currently available data.
	RetryInfoPtr	If the transmitted TP I-PDU does not support the retry

		feature a NULL_PTR can be provided. This indicates that the copied transmit data can be removed from the buffer after it has been copied.
Parameters (inout):	None	
Parameters (out):	TxDataCntPtr	Remaining Tx data after completion of this call.
Return value:	BufReq_ReturnType	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_NOT_OK: Data has not been copied. Request failed. BUFREQ_E_BUSY: The amount of data requested by this call is currently not available.
Description:	At invocation of Dcm_CopyTxData the DCM module copies the requested transmit data with ID PduId from its internal transmit buffer to the location specified by the PduInfoPtr. The function Dcm_CopyTxData also calculates and sets the TxDataCntPtr to the amount of remaining bytes for the transmission of this data. If RetryInfoPtr is NULL_PTR or if TpDataState is not equal TP_DATARETRY, the Dcm shall always copy the next fragment of data to the SduDataPtr. If TpDataState equals TP_DATARETRY, the Dcm shall copy previously copied data again, beginning at the offset given in RetryInfoPtr->TxTpDataCnt from the current position.	

」()

If the copied data is smaller than the length requested to transmit within the service `PduR_DcmTransmit()` the DCM module will be requested by the service `Dcm_CopyTxData()` to provide another data when the current copied data have been transmitted.

[SWS_Dcm_00346]「 If the function `Dcm_CopyTxData()` is called and the DCM module successfully copied the data in the buffer provided in PduInfoPtr parameter, then the function shall return BUFREQ_OK.」()

[SWS_Dcm_00350]「 Caveats of `Dcm_CopyTxData()`:

- The value of parameter Length of function `Dcm_CopyTxData()` shall not exceed the number of Bytes still to be sent.
- If this service returns BUFREQ_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_TpTxConfirmation()`) is required to finish this service and to be able to start another transmission (call to `PduR_DcmTransmit()`). So it is up to the transport protocol to confirm the abort of transmission.
- The value of parameter DcmTxPduId in a call of `Dcm_CopyTxData()` has to be same as in the according service call `PduR_DcmTransmit()`.」()

[SWS_Dcm_00832] ⌈Dcm_CopyTxData() shall be callable in interrupt context.⌋()

8.5.5 Dcm_TpTxConfirmation

[SWS_Dcm_00351] ⌈

Service name:	Dcm_TpTxConfirmation	

Syntax:	`void Dcm_TpTxConfirmation(`
	`  PduIdType DcmTxPduId,`
	`  Std_ReturnType Result`
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	DcmTxPduId	Represents the TX PduId configured within the DCM (DcmDslProtocolTx-->DcmDslTxConfirmationPduId). Through this Id the DCM is able to determine the corresponding protocol for the communication. Range: 0..(maximum number of I-PDU IDs transmitted by DCM) – 1
	Result	E_OK: the complete N-PDU has been transmitted. E_NOT_OK: 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	

⌋()

[SWS_Dcm_00352] ⌈ If the function `Dcm_TpTxConfirmation()` is called, then the DCM module shall unlock the transmit buffer.⌋()

[SWS_Dcm_00353] ⌈ If the function `Dcm_TpTxConfirmation()` is called, then the DCM module shall stop error handling (Page buffer timeout, P2ServerMax/P2*ServerMax timeout).⌋()

[SWS_Dcm_00354] ⌈ `Dcm_TpTxConfirmation()` 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.

8.5.6 Dcm_ComM_NoComModeEntered

[SWS_Dcm_00356]⌈

Service name:	Dcm_ComM_NoComModeEntered	
Syntax:	`void Dcm_ComM_NoComModeEntered( uint8 NetworkId )`	
Service ID[hex]:	0x21	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	NetworkId	Identifier of the network concerned by the mode change
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.	

⌋()

[SWS_Dcm_00148]⌈ `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.⌋()

[SWS_Dcm_00149]⌈ `Dcm_ComM_NoComModeEntered()` shall disable the ResponseOnEvent transmissions.⌋()

[SWS_Dcm_00150]⌈ `Dcm_ComM_NoComModeEntered()` shall disable the periodicId transmissions (ReadDataByPeriodicIdentifier).⌋()

[SWS_Dcm_00151]⌈ `Dcm_ComM_NoComModeEntered()` shall disable normal transmissions.⌋()

[SWS_Dcm_00152]⌈ After `Dcm_ComM_NoComModeEntered()` has been called, the DCM module shall not call the function `PduR_DcmTransmit()`.⌋()

8.5.7 Dcm_ComM_SilentComModeEntered

[SWS_Dcm_00358]⌈

Service name:	Dcm_ComM_SilentComModeEntered
Syntax:	`void Dcm_ComM_SilentComModeEntered( uint8 NetworkId )`

Service ID[hex]:	0x22	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	NetworkId	Identifier of the network concerned by the mode change
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.	

」()

[SWS_Dcm_00153] 「`Dcm_ComM_SilentComModeEntered()` shall disable all transmission. This means that the message transmission shall be off.」()

[SWS_Dcm_00154] 「`Dcm_ComM_SilentComModeEntered()` shall disable the ResponseOnEvent transmissions.」()

[SWS_Dcm_00155] 「`Dcm_ComM_SilentComModeEntered()` shall disable the periodicId transmissions (ReadDataByPeriodicIdentifier) shall be disabled.」()

[SWS_Dcm_00156] 「`Dcm_ComM_SilentComModeEntered()` shall disable the normal transmissions.」()

8.5.8 Dcm_ComM_FullComModeEntered

[SWS_Dcm_00360] 「

Service name:	Dcm_ComM_FullComModeEntered	
Syntax:	`void Dcm_ComM_FullComModeEntered( uint8 NetworkId )`	
Service ID[hex]:	0x23	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	NetworkId	Identifier of the network concerned by the mode change
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.	

」()

[SWS_Dcm_00157] ⌈ `Dcm_ComM_FullComModeEntered()` shall enable all kind of communication. This means that the message reception and also the message transmission shall be on. ⌋()

[SWS_Dcm_00159] ⌈ `Dcm_ComM_FullComModeEntered()` shall enable the ResponseOnEvent transmissions. ⌋()

[SWS_Dcm_00160] ⌈ `Dcm_ComM_FullComModeEntered()` shall enable the periodicId transmissions (ReadDataByPeriodicIdentifier). ⌋()

[SWS_Dcm_00161] ⌈ `Dcm_ComM_FullComModeEntered()` shall enable the normal transmissions. ⌋()

[SWS_Dcm_00162] ⌈ After `Dcm_ComM_FullComModeEntered()` has been called, the DCM shall handle the functions `DslInternal_ResponseOnOneDataByPeriodicId()` or `DslInternal_ResponseOnOneEvent()` without restrictions. ⌋()

8.6 Callout Definitions

Callouts are pieces of code that have to be added to the DCM during ECU integration. The content of most callouts is hand-written code, for some callouts the DCM configuration tool shall generate a default implementation that is manually edited by the integrator. Conceptually, these callouts belong to the ECU Firmware.

Since callouts are no services of the DCM they do not have an assigned Service ID.
Note:
The Autosar architecture doesn't provide the possibility to access the ECU memory using a physical address. This realized using BlockId wich identified a memory block. According to that, the DCM is not able to fully support the implementation of ISO14229-1 services wich request a physical memory access.
Therefore, the DCM define callout to realize this kind of memory access.
This callout implementation could be simply realized by defining a mapping between the BlockId and the physical memory address.

8.6.1 Dcm_ReadMemory

[**SWS_Dcm_00539**]⌈

Service name:	Dcm_ReadMemory	
Syntax:	`Dcm_ReturnReadMemoryType Dcm_ReadMemory( Dcm_OpStatusType OpStatus, uint8 MemoryIdentifier, uint32 MemoryAddress, uint32 MemorySize, uint8* MemoryData )`	
Service ID[hex]:	0x26	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL: All In-parameters are valid DCM_PENDING: All In-parameters are set to 0x0 DCM_CANCEL: All In-parameters are set to 0x0 DCM_FORCE_RCRRP_OK: All In-parameters are set to 0x0
	MemoryIdentifier	Identifier of the Memory Block (e.g. used if memory section distinguishing is needed) Note: If it's not used this parameter shall be set to 0.
	MemoryAddress	Starting address of server memory from which data is to be retrieved.
	MemorySize	Number of bytes in the MemoryData
Parameters (inout):	None	
Parameters (out):	MemoryData	Data read (Points to the diagnostic buffer in DCM)
Return value:	Dcm_ReturnReadMemoryType	DCM_READ_OK: read was successful DCM_READ_FAILED: read was not successful DCM_READ_PENDING: read is not yet finished DCM_READ_FORCE_RCRRP: reading is pending, the Response pending transmission starts immediately
Description:	The Dcm_ReadMemory callout is used to request memory data identified by the parameter memoryAddress and memorySize from the UDS request message. This service is needed for the implementation of UDS services: - ReadMemoryByAdress - RequestUpload - ReadDataByIdentifier (in case of Dynamical DID defined by memory address)	

」()

[SWS_Dcm_00644]「 If the call to `Dcm_ReadMemory` returns DCM_READ_FAILED, the DCM module shall trigger a negative response with NRC 0x10 (GeneralReject).」()

[SWS_Dcm_00839]「 If the call to Dcm_ReadMemory returns DCM_READ_FORCE_RCRRP, the DCM shall invoke the transmit request for RCR-RP (NRC 0x78 transmission) and the DCM shall not realize further invocation of the operation till RCR-RP is transmitted (call to Dcm_Confirmation). 」()

[SWS_Dcm_00840]「 After transmit confirmation of a RCR-RP transmitted on the context of **SWS_Dcm_00839**, the DCM calls, from Dcm_MainFunction (due to call context), Dcm_ReadMemory again with OpStatus = DCM_FORCE_RCRRP_OK. 」()

8.6.2 Dcm_WriteMemory

[SWS_Dcm_00540]「

Service name:	Dcm_WriteMemory	

Syntax:	`Dcm_ReturnWriteMemoryType Dcm_WriteMemory(`
	`  Dcm_OpStatusType OpStatus,`
	`  uint8 MemoryIdentifier,`
	`  uint32 MemoryAddress,`
	`  uint32 MemorySize,`
	`  uint8* MemoryData`
	`)`	

Service ID[hex]:	0x27	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL: All In-parameters are valid DCM_PENDING: All In-parameters are set to 0x0 DCM_CANCEL: All In-parameters are set to 0x0 DCM_FORCE_RCRRP_OK: All In-parameters are set to 0x0
	MemoryIdentifier	Identifier of the Memory Block (e.g. used by WriteDataByIdentifier service). Note: If it's not used this parameter shall be set to 0.
	MemoryAddress	Starting address of server memory in which data is to be copied. Note: If it's not used (e.g. if the data is compressed) this parameter shall be set to 0.
	MemorySize	Number of bytes in MemoryData
	MemoryData	Data to write (Points to the diagnostic buffer in DCM)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dcm_ReturnWriteMemoryType	DCM_WRITE_OK: write was successful DCM_WRITE_FAILED: write was not successful DCM_WRITE_PENDING: write is not yet finished DCM_WRITE_FORCE_RCRRP: writing is pending, the Response pending transmission starts immediately
Description:	The Dcm_WriteMemory callout is used to write memory data identified by the parameter memoryAddress and memorySize. This service is needed for the implementation of UDS services : - WriteMemoryByAdress - RequestDownload	

」()

Note :
The callout implementation shall take care of the following points :

- When writing data in NVRAM, take care to keep the consistency with data in the mirror RAM
- When writing data in memory, take care that a SW-C won't overwrite the data. Maybe the SW-C should be informed of this writing

[SWS_Dcm_00643]⌈ If the call to `Dcm_WriteMemory` returns DCM_WRITE_FAILED, the DCM module shall trigger a negative response with NRC 0x72 (GeneralProgrammingFailure).⌋()

[SWS_Dcm_00837]⌈ If the call to Dcm_WriteMemory returns DCM_WRITE_FORCE_RCRRP, the DCM shall invoke the transmit request for RCR-RP (NRC 0x78 transmission) and the DCM shall not realize further invocation of the operation till RCR-RP is transmitted (call to Dcm_Confirmation). ⌋()

[SWS_Dcm_00838]⌈ After transmit confirmation of a RCR-RP transmitted on the context of **SWS_Dcm_00837**, the DCM calls, from Dcm_MainFunction (due to call context), Dcm_WriteMemory again with OpStatus = DCM_FORCE_RCRRP_OK. ⌋()

8.6.3 Dcm_Confirmation

[SWS_Dcm_00547] ⌈

Service name:	Dcm_Confirmation	

Syntax:	`void Dcm_Confirmation(`
	`  Dcm_IdContextType idContext,`
	`  PduIdType dcmRxPduId,`
	`  Dcm_ConfirmationStatusType status`
	`)`	

Service ID[hex]:	0x29	
Sync/Async:	Synchronous	
Reentrancy:	No	
Parameters (in):	idContext	Current context identifier which can be used to retrieve the relation between request and confirmation. Within the confirmation, the Dcm_MsgContext is no more available, so the idContext can be used to represent this relation. The idContext is also part of the Dcm_MsgContext
	dcmRxPduId	DcmRxPduId on which the request was received. The source of the request can have consequences for message processing.
	status	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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function confirms thesuccessful transmission or a transmission error of a	

	diagnostic service. The idContext and the dcmRxPduId are required to identify the message which was processed.

⌋(BSW04019)

8.6.4 Dcm_SetProgConditions

[SWS_Dcm_00543] ⌈

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 DCM_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.	

⌋()

8.6.5 Dcm_GetProgConditions

[SWS_Dcm_00544] ⌈

Service name:	Dcm_GetProgConditions	

Syntax:	`Dcm_EcuStartModeType 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:	Dcm_EcuStartModeType	--
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.	

⌋()

8.6.6 Dcm_ProcessRequestTransferExit

[SWS_Dcm_00755]「

Service name:	Dcm_ProcessRequestTransferExit	
Syntax:	`Std_ReturnType Dcm_ProcessRequestTransferExit( Dcm_OpStatusType OpStatus, uint8* ParameterRecord, uint32 ParameterRecordSize, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID[hex]:	0x32	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL: All In-parameters are valid DCM_PENDING: All In-parameters are set to 0x0 DCM_CANCEL: All In-parameters are set to 0x0 DCM_FORCE_RCRRP_OK: All In-parameters are set to 0x0
	ParameterRecord	(Optional) Pointer to vehicle-manufacturer-specific data
	ParameterRecordSize	(Optional) Length of ParameterRecord in bytes
Parameters (inout):	None	
Parameters (out):	ErrorCode	See below
Return value:	Std_ReturnType	E_OK: Transfer was successful E_NOT_OK: Transfer was not successful DCM_E_PENDING: Transfer is not yet finished
Description:	Callout function. DCM shall call this callout function to terminate a download or upload process. This service is needed for the implementation of UDS service RequestTransferExit.	

」()

[SWS_Dcm_00759]「If the operation `Dcm_ProcessRequestTransferExit` returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value. 」()

8.6.7 Dcm_ProcessRequestUpload

[SWS_Dcm_00756]「

Service name:	Dcm_ProcessRequestUpload
Syntax:	`Std_ReturnType Dcm_ProcessRequestUpload( Dcm_OpStatusType OpStatus, uint8 DataFormatIdentifier, uint32 MemoryAddress, uint32 MemorySize, uint32* BlockLength, Dcm_NegativeResponseCodeType* ErrorCode )`
Service ID[hex]:	0x31
Sync/Async:	Asynchronous
Reentrancy:	Non Reentrant

Parameters (in):	OpStatus	DCM_INITIAL: All In-parameters are valid DCM_PENDING: All In-parameters are set to 0x0 DCM_CANCEL: All In-parameters are set to 0x0 DCM_FORCE_RCRRP_OK: All In-parameters are set to 0x0
	DataFormatIdentifier	Bit 7 - 4: Compression Method - 0x0: not compressed - 0x1..F: vehicle-manufacturer-specific Bit 3 - 0: Encrypting method - 0x0: not encrypted - 0x1..F: vehicle-manufacturer-specific
	MemoryAddress	Starting address of server memory from which data are to be copied
	MemorySize	Uncompressed memory size in bytes
Parameters (inout):	None	
Parameters (out):	BlockLength	Max. Number of bytes for one Dcm_ReadMemory
	ErrorCode	If the operation Dcm_ProcessRequestUpload returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful E_NOT_OK: Request was not successful DCM_E_PENDING: Request is not yet finished
Description:	Callout function. DCM shall call this callout function to start an upload process. This service is needed for the implementation of UDS service RequestUpload.	

」()

[SWS_Dcm_00758] 「 If the operation `Dcm_ProcessRequestUpload` returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value. 」()

8.6.8 Dcm_ProcessRequestDownload

[SWS_Dcm_00754] 「

Service name:	Dcm_ProcessRequestDownload	

Syntax:	`Std_ReturnType Dcm_ProcessRequestDownload(`
	`  Dcm_OpStatusType OpStatus,`
	`  uint8 DataFormatIdentifier,`
	`  uint32 MemoryAddress,`
	`  uint32 MemorySize,`
	`  uint32* BlockLength,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x30	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL: All In-parameters are valid DCM_PENDING: All In-parameters are set to 0x0 DCM_CANCEL: All In-parameters are set to 0x0 DCM_FORCE_RCRRP_OK: All In-parameters are set to 0x0
	DataFormatIdentifier	Bit 7 - 4: Compression Method

		- 0x0: not compressed - 0x1..F: vehicle-manufacturer-specific Bit 3 - 0: Encrypting method - 0x0: not encrypted - 0x1..F: vehicle-manufacturer-specific
	MemoryAddress	Starting address of server memory to which data is to be written
	MemorySize	Uncompressed memory size in bytes
Parameters (inout):	None	
Parameters (out):	BlockLength	Max. Number of bytes for one Dcm_WriteMemory
	ErrorCode	If the operation Dcm_ProcessRequestDownload returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful E_NOT_OK: Request was not successful DCM_E_PENDING: Request is not yet finished
Description:	Callout function. DCM shall call this callout function to start a download process. This service is needed for the implementation of UDS service RequestDownload.	

⌋()

[SWS_Dcm_00757] ⌈ If the operation `Dcm_ProcessRequestDownload` returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value. ⌋()

8.7 Scheduled Functions

8.7.1 Dcm_MainFunction

[SWS_Dcm_00053] ⌈

Service name:	Dcm_MainFunction

Syntax:	`void Dcm_MainFunction(`
	`  void`
	`)`

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

⌋(BSW00424, BSW00373, BSW00376)

8.8 Expected Interfaces

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

8.8.1 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.
PduR_DcmTransmit	Requests transmission of an I-PDU.

8.8.2 Optional Interfaces

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

API function	*Description*
BswM_Dcm_ApplicationUpdated	This function is called by the DCM in order to report an updated application.
BswM_Dcm_CommunicationMode_CurrentState	Function called by DCM to inform the BswM about the current state of the communication mode.
Dem_DcmDisableDTCRecordUpdate	Disables the event memory update of a specific DTC (only one at one time).
Dem_DcmDisableDTCSetting	Disables the DTC setting for a DTC group.
Dem_DcmEnableDTCRecordUpdate	Enables the event memory update of the DTC disabled by Dem_DcmDisableDTCRecordUpdate() before.
Dem_DcmEnableDTCSetting	Enables the DTC setting for a DTC group. This API is intended for the Dcm. It can only be used through the RTE (due to work-around described below SWS_Dem_00035), and therefore no declaration is exported via Dem_Dcm.h.
Dem_DcmGetDTCByOccurrenceTime	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_DcmGetDTCOfOBDFreezeFrame	Gets DTC by freeze frame record number. API is needed in OBD-relevant ECUs only
Dem_DcmGetDTCStatusAvailabilityMask	Gets the DTC Status availability mask.
Dem_DcmGetExtendedDataRecordByDTC	Gets extended data by DTC. The function stores the data in the provided DestBuffer.
Dem_DcmGetFreezeFrameDataByDTC	Gets freeze frame data by DTC. The function stores the data in the provided DestBuffer.
Dem_DcmGetFunctionalUnitOfDTC	Gets the functional unit of the requested DTC.
Dem_DcmGetNextFilteredDTC	Gets the next filtered DTC matching the filter criteria. For UDS services, the interface has an asynchronous behavior, because a large number of DTCs has to be processed.
Dem_DcmGetNextFilteredDTCAndFDC	Gets the next filtered DTC and its associated Fault Detection Counter (FDC) matching the filter criteria. The interface has an asynchronous behavior, because a large number of DTCs has to be processed and the FDC might be received asynchronously from a SW-C, too.

Dem_DcmGetNextFilteredDTCAndSeverity	Gets the next filtered DTC and its associated Severity matching the filter criteria. The interface has an asynchronous behavior, because a large number of DTCs has to be processed.
Dem_DcmGetNextFilteredRecord	Gets the next freeze frame record number and its associated DTC stored in the event memory. The interface has an asynchronous behavior, because NvRAM access might be required.
Dem_DcmGetNumberOfFilteredDTC	Gets the number of a filtered DTC.
Dem_DcmGetOBDFreezeFrameData	Gets the most important DTC and its OBD freeze frame (which caused MIL on) for the output on UDS (service 0x19). The function stores the freeze frame data in the provided DestBuffer. API is needed in OBD-relevant ECUs only
Dem_DcmGetSeverityOfDTC	Gets the severity of the requested DTC. For large configurations and DTC-calibration, the interface behavior can be asynchronous (splitting the DTC-search into segments).
Dem_DcmGetSizeOfExtendedDataRecordByDTC	Gets the size of extended data by DTC.
Dem_DcmGetSizeOfFreezeFrameByDTC	Gets the size of freeze frame data by DTC.
Dem_DcmGetStatusOfDTC	Gets the status of a DTC. For large configurations and DTC-calibration, the interface behavior can be asynchronous (splitting the DTC-search into segments). The DTCs of OBD Events Suppression shall be reported as DEM_STATUS_WRONG_DTC.
Dem_DcmGetTranslationType	Gets the supported DTC formats of the ECU. The supported formats are configured via DemTypeOfDTCSupported.
Dem_DcmReadDataOfOBDFreezeFrame	Gets data element per PID and index of the most important freeze frame being selected for the output of service $02. The function stores the data in the provided DestBuffer. API is needed in OBD-relevant ECUs only
Dem_DcmSetDTCFilter	Sets the DTC Filter. The server shall perform a bit-wise logical AND-ing operation between the parameter DTCStatusMask and the current DTC status in 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]. The server shall process only the DTC Status bits that it is supporting. OBD Events Suppression shall be ignored for this computation. 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) != 0) && ((severity & DTCSeverityMask) != 0)) == TRUE
Dem_DcmSetFreezeFrameRecordFilter	Sets a freeze frame record filter.
Det_ReportError	Service to report development errors.
IoHwAb_Dcm_<EcuSignalName>	This function provides control access to a certain ECU Signal to the DCM module (<EcuSignalname> is the symbolic name of an ECU Signal). The ECU

	signal can be locked and unlocked by this function. Locking 'freezes' the ECU signal to the current value, the configured default value or a value given by the parameter 'signal'.
IoHwAb_Dcm_Read<EcuSignalName>	This function provides read access to a certain ECU Signal to the DCM module (<EcuSignalname> is the symbolic name of an ECU Signal).
NvM_ReadBlock	Service to copy the data of the NV block to its corresponding RAM block.
NvM_SetBlockLockStatus	Service for setting the lock status of a permanent RAM block or of the explicit synchronization of a NVRAM block.
NvM_SetRamBlockStatus	Service for setting the RAM block status of an NVRAM block.
NvM_WriteBlock	Service to copy the data of the RAM block to its corresponding NV block.
PduR_DcmCancelReceive	Requests cancellation of an ongoing reception of an I-PDU in a lower layer transport protocol module.
PduR_DcmCancelTransmit	Requests cancellation of an ongoing transmission of an I-PDU in a lower layer communication interface or transport protocol module.
PduR_DcmChangeParameter	Request to change a specific transport protocol parameter (e.g. block size).
SchM_ActMainFunction_Dcm	Invokes the SchM_ActMainFunction function to trigger the activation of a corresponding main processing function.

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

8.8.3 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).

8.8.3.1 SecurityAccess_{SecurityLevel}

Provides functions required for the UDS Service SecurityAccess (see SWS_Dcm_00323,**SWS_Dcm_00862** and **SWS_Dcm_00863**).

Note: The OpStatus parameter is only used for asynchronous operations (if DcmDspSecurityUsePort is set to USE_ASYNCH_CLIENT_SERVER or USE_ASYNCH_FNC). In case of synchronous operations (DcmDspSecurityUsePort is set to USE_SYNCH_CLIENT_SERVER or USE_SYNCH_FNC), the OpStatus parameter is not used.

Using the concepts of the SW-C template, the interface is defined as follows if ClientServer interface is used (DcmDspSecurityUsePort set to USE_SYNCH_CLIENT_SERVER or USE_ASYNCH_CLIENT_SERVER):

[SWS_Dcm_00685] [`ClientServerInterface SecurityAccess_{SecurityLevel}`

Name	SecurityAccess_{SecurityLevel}	
Comment	--	
IsService	true	
Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)} {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)} == (USE_SYNCH_CLIENT_SERVER || USE_ASYNCH_CLIENT_SERVER)	
Possible Errors	0	E_OK
	1	E_NOT_OK
	10	DCM_E_PENDING
	11	DCM_E_COMPARE_KEY_FAILED

Operations

CompareKey		
Comments	--	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)} == USE_ASYNCH_CLIENT_SERVER	
Parameters	Key	
	Comment	Key, which needs to be compared
	Type	KeyArrayType_{SecurityLevel}
	Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}
	Direction	IN
	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Request was successful.
	E_NOT_OK	Request was not successful.
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

	DCM_E_COMPARE_KEY_FAILED	Key did not match.

CompareKey		
Comments	--	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)} == USE_SYNCH_CLIENT_SERVER	
Parameters	Key	
	Comment	Key, which needs to be compared
	Type	KeyArrayType_{SecurityLevel}
	Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}
	Direction	IN
Possible Errors	E_OK	Request was successful.
	E_NOT_OK	Request was not successful.
	DCM_E_COMPARE_KEY_FAILED	Key did not match.

GetSeed		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityADRSize)}== NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)}== USE_ASYNCH_CLIENT_SERVER)	
Parameters	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	Seed	
	Comment	Pointer for provided seed
	Type	SeedArrayType_{SecurityLevel}

	Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspSecurity/DcmDspSecurityRow. SHORT-NAME)}
	Direction	OUT
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful.
	E_NOT_OK	Request was not successful.
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.
GetSeed		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow. DcmDspSecurityADRSize)}!= NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)}== USE_ASYNCH_CLIENT_SERVER)	
Parameters	SecurityAccessDataRecord	
	Comment	--
	Type	SecurityAccessDataRecordType_{SecurityLevel}
	Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspSecurity/DcmDspSecurityRow. SHORT-NAME)}
	Direction	IN
	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	Seed	

	Comment	Pointer for provided seed
	Type	SeedArrayType_{SecurityLevel}
	Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}
	Direction	OUT
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful.
	E_NOT_OK	Request was not successful.
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.
GetSeed		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityADRSize)}== NULL}) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)}== USE_SYNCH_CLIENT_SERVER)	
Parameters	Seed	
	Comment	Pointer for provided seed
	Type	SeedArrayType_{SecurityLevel}
	Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}
	Direction	OUT
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType

	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
GetSeed		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityADRSize)}!= NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)}== USE_SYNCH_CLIENT_SERVER)	
Parameters	SecurityAccessDataRecord	
	Comment	--
	Type	SecurityAccessDataRecordType_{SecurityLevel}
	Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}
	Direction	IN
	Seed	
	Comment	Pointer for provided seed
	Type	SeedArrayType_{SecurityLevel}
	Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}
	Direction	OUT
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful.
	E_NOT_OK	Request was not successful.

]()

[SWS_Dcm_00659]「 If the operation GetSeed() return value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to ErrorCode parameter value.」()

[SWS_Dcm_00660]「 If the operation CompareKey() return value DCM_E_COMPARE_KEY_FAILED or E_NOT_OK, the DCM module shall send a negative response with NRC 0x35 (InvalidKey) and shall not change the DCM internal security level.」()

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

If DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER, the following definition is used:

If DcmDspSecurityADRSize is present:

Service name:	Xxx_GetSeed	

Syntax:	`Std_ReturnType Xxx_GetSeed(`
	`  const uint8* SecurityAccessDataRecord,`
	`  uint8* Seed,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x42	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SecurityAccessDataRecord	This data record contains additional data to calculate the seed value; the size of this parameter is DcmDspSecurityADRSize which is at least "1".
Parameters (inout):	None	
Parameters (out):	Seed	Pointer for provided seed
	ErrorCode	If the operation Xxx_GetSeed returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	Request to application for synchronous provision of seed value	

If DcmDspSecurityADRSize is not present

Service name:	Xxx_GetSeed

Syntax:	`Std_ReturnType Xxx_GetSeed(`
	`  uint8* Seed,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`

Service ID[hex]:	0x43	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	Seed	Pointer for provided seed
	ErrorCode	If the operation Xxx_GetSeed returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	Request to application for synchronous provision of seed value	

Service name:	Xxx_CompareKey	

Syntax:	`Std_ReturnType Xxx_CompareKey(`
	`  const uint8* Key`
	`)`	

Service ID[hex]:	0x46	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Key	Key, which needs to be compared
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_COMPARE_KEY_FAILED: Key did not match.
Description:	Request to application for synchronous comparing key (DcmDspSecurityUsePort = USE_SYNCH_CLIENT_SERVER)	

if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER, the following definition is used:

If DcmDspSecurityADRSize is present

Service name:	Xxx_GetSeed	

Syntax:	`Std_ReturnType Xxx_GetSeed(`
	`  const uint8* SecurityAccessDataRecord,`
	`  Dcm_OpStatusType OpStatus,`
	`  uint8* Seed,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x44	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SecurityAccessDataRecord	This data record contains additional data to calculate the seed value; the size of this parameter is

		DcmDspSecurityADRSize which is at least "1".
	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	Seed	Pointer for provided seed
	ErrorCode	If the operation Xxx_GetSeed returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	Request to application for asynchronous provision of seed value	

If DcmDspSecurityADRSize is not present

Service name:	Xxx_GetSeed	

Syntax:	`Std_ReturnType Xxx_GetSeed(`
	`  Dcm_OpStatusType OpStatus,`
	`  uint8* Seed,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x45	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	Seed	Pointer for provided seed
	ErrorCode	If the operation Xxx_GetSeed returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	Request to application for asynchronous provision of seed value	

Service name:	Xxx_CompareKey	

Syntax:	`Std_ReturnType Xxx_CompareKey(`
	`  const uint8* Key,`
	`  Dcm_OpStatusType OpStatus`
	`)`	

Service ID[hex]:	0x47	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Key	Key, which needs to be compared
	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s)

		required to finish. DCM_E_COMPARE_KEY_FAILED: Key did not match.
Description:	Request to application for asynchronous comparing key (DcmDspSecurityUsePort = USE_ASYNCH_CLIENT_SERVER)	

8.8.3.2 DataServices_{Data}

One DataServices interface will be generated for each Data of each DID/PID, with following possible operations:

8.8.3.2.1 ReadData

ReadData allows requesting to the application a data value of a DID/PID. A ReadData interface is defined for every data of each DID/PID with read access. The Data specific type is an array of uint8 which represents either the fix length of this Data or the maximum possible length of this Data. If the length is variable, the operation ReadDataLength has to provide the current valid data length of this Data.
This interface is used for UDS Service ReadDataByIdentifier and for UDS Service ReadDataByPeriodicIdentifier (0x2A) and for UDS Service InputOutputControlByIdentifier (0x2F).
The ReadData interface can be defined as synchronous or asynchronous according to configuration parameter DcmDspDataUsePort.
The synchronous mechanism of the ReadData interface is compatible to the related DEM interface to allow the provider to use the same interface for both DCM and DEM.

8.8.3.2.2 WriteData

WriteData requests the application to write a data value of a DID. The Data specific type is an array of uint8 which represent either the fix length of this Data or the maximum possible length of this Data. A WriteData interface is defined for every data of each DID with write access
This interface is used for the UDS Service WriteDataByIdentifier (0x2E).

8.8.3.2.3 ReadDataLength

ReadDataLength requests the application to return the data length of a Data. A ReadDataLength interface is defined for every data of each DID with variable data length.
This interface is used for UDS Service ReadDataByIdentifier and for UDS Service ReadDataByPeriodicIdentifier (0x2A)

8.8.3.2.4 ConditionCheckRead

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

8.8.3.2.5 GetScalingInformation

Request to the application for the scaling information of a Data (see SWS_Dcm_00394)

8.8.3.2.6 ReturnControlToEcu

Request to the application to return control to ECU of an IOControl (see SWS_Dcm_00396)

8.8.3.2.7 ResetToDefault

Request to the application to reset an IOControl to default value (see SWS_Dcm_00397)

8.8.3.2.8 FreezeCurrentState

Request to the application to freeze the current state of an IOControl (see SWS_Dcm_00398)

8.8.3.2.9 ShortTermAdjustment

Request to the application to adjust the IO signal (see SWS_Dcm_00399).

8.8.3.2.10 Client Server interface

Using the concepts of the SW-C template, the interface is defined as follows if ClientServer interface is used(DcmDspDataUsePort set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER):

[SWS_Dcm_00686] ⌈ `ClientServerInterface DataServices_{Data}`

Name	DataServices_{Data}	
Comment	--	
IsService	true	
Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)}) ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) ||({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidService01.DcmDspPidDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER)	
Possible Errors	0	E_OK
	1	E_NOT_OK
	10	DCM_E_PENDING

Operations

ConditionCheckRead		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspDidInfo/DcmDspDidAccess/DcmDspDidRead)} != NULL) && ({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataConditionCheckReadFncUsed)} == TRUE)	
Parameters	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN

	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

ConditionCheckRead		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidRead)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataConditionCheckReadFncUsed)} == TRUE)	
Parameters	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful

FreezeCurrentState	
Comments	--
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE)
Parameters	OpStatus

	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

FreezeCurrentState		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE)	
Parameters	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful

GetScalingInformation	
Comments	--

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData->DcmDspDataInfoRef.DcmDspDataScalingInfoSize)} != NULL)	
Parameters	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	ScalingInfo	
	Comment	--
	Type	ScalingInfoArrayType_{Data}
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
	Direction	OUT
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

GetScalingInformation		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData->DcmDspDataInfoRef.DcmDspDataScalingInfoSize)} != NULL)	
Parameters	ScalingInfo	
	Comment	--
	Type	ScalingInfoArrayType_{Data}

	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
	Direction	OUT
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful

ReadData		
Comments	The server is not allowed to return E_NOT_OK, but shall always provide a valid data value (e.g. a default/replacement value in an error-case) to Dcm/Dem nevertheless the signature of the operation includes E_NOT_OK to ensure compatibility between server runnable and RTE Call API, since the RTE may return negative Std_Return values in certain cases (e.g. partition of server stopped)	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidRead)} != NULL)	
Parameters	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	Data	
	Comment	--
	Type	DataArrayType_{Data}
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})
	Direction	OUT
Possible	E_OK	Request was successful

Errors	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.
ReadData		
Comments	The server is not allowed to return E_NOT_OK, but shall always provide a valid data value (e.g. a default/replacement value in an error-case) to Dcm/Dem nevertheless the signature of the operation includes E_NOT_OK to ensure compatibility between server runnable and RTE Call API, since the RTE may return negative Std_Return values in certain cases (e.g. partition of server stopped)	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidRead)} != NULL) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER))	
Parameters	Data	
	Comment	--
	Type	DataArrayType_{Data}
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful
ReadDataLength		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidRead)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataInfoRef->DcmDspDataFixedLength)} == FALSE)	
Parameters	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType

	Variation	--
	Direction	IN
	DataLength	
	Comment	--
	Type	uint16
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

ReadDataLength		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidRead)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataInfoRef->DcmDspDataFixedLength)} == FALSE)"	
Parameters	DataLength	
	Comment	--
	Type	uint16
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful

ResetToDefault	
Comments	--
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidControl/DcmDspDidResetToDefault)}

	== TRUE)	
Parameters	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

ResetToDefault		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidControl/DcmDspDidResetToDefault)} == TRUE)	
Parameters	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful

ReturnControlToECU		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidControl/DcmDspDidReturnControlToEcu)} == TRUE)	
Parameters	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

ReturnControlToECU		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidControl/DcmDspDidReturnControlToEcu)} == TRUE)	
Parameters	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT

Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful

ShortTermAdjustment		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDataInfo.DcmDspDataFixedLength)} == TRUE)	
Parameters	ControlStateInfo	
	Comment	--
	Type	DataArrayType_{Data}
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
	Direction	IN
	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

ShortTermAdjustment		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDataInfo.DcmDspDataFixedLength)} == FALSE)	
Parameters	ControlStateInfo	
	Comment	--
	Type	DataArrayType_{Data}
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
	Direction	IN
	DataLength	
	Comment	--
	Type	uint16
	Variation	--
	Direction	IN
	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

ShortTermAdjustment		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDataInfo.DcmDspDataFixedLength)} == TRUE)	
Parameters	ControlStateInfo	
	Comment	--
	Type	DataArrayType_{Data}
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
	Direction	IN
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful

ShortTermAdjustment		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDataInfo.DcmDspDataFixedLength)} == FALSE)	
Parameters	ControlStateInfo	
	Comment	--
	Type	DataArrayType_{Data}
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
	Direction	IN

	DataLength	
	Comment	--
	Type	uint16
	Variation	--
	Direction	IN
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful
WriteData		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDataInfo.DcmDspDataFixedLength)} == TRUE)	
Parameters	Data	
	Comment	--
	Type	DataArrayType_{Data}
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
	Direction	IN
	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	ErrorCode	
	Comment	--

	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

WriteData		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDataInfo.DcmDspDataFixedLength)} == FALSE)	
Parameters	Data	
	Comment	--
	Type	DataArrayType_{Data}
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
	Direction	IN
	DataLength	
	Comment	--
	Type	uint16
	Variation	--
	Direction	IN
	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	ErrorCode	
	Comment	--

	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

WriteData		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDataInfo.DcmDspDataFixedLength)} == TRUE)	
Parameters	Data	
	Comment	--
	Type	DataArrayType_{Data}
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
	Direction	IN
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful

WriteData	
Comments	--
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} ==

	USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDataInfo.DcmDspDataFixedLength)} == FALSE)	
Parameters	Data	
	Comment	--
	Type	DataArrayType_{Data}
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
	Direction	IN
	DataLength	
	Comment	--
	Type	uint16
	Variation	--
	Direction	IN
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Request was successful
	E_NOT_OK	Request was not successful

」()

8.8.3.2.11 Sender Receiver interface

Using the concepts of the SW-C template, the interface is defined as follows if SenderReceiver interface is used (DcmDspDataUsePort set to USE_DATA_SENDER_RECEIVER):

[**SWS_Dcm_00687**]「 `SenderReceiver DataServices_{Data}`

Name	DataServices_{Data}
Comment	--
IsService	false
Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME}) ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/

	DcmDspPidData/DcmDspPidService01/DcmDspPidDataUsePort} == USE_DATA_SENDER_RECEIVER)	
Data Elements	data	
	Type	boolean
	Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataSize)} == NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/ DcmDspDataType)} == BOOLEAN)) || (({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspPid/DcmDspPidData/DcmDspPidDataSize) == NULL}) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidService01/DcmDspPidDataType)} ==BOOLEAN))
	data	
	Type	sint8
	Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataSize)} == NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/ DcmDspDataType)} == SINT8)) ||(({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspPid/DcmDspPidData/DcmDspPidDataSize) == NULL}) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidService01/DcmDspPidDataType)} ==SINT8))
	data	
	Type	sint16
	Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataSize)} == NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/ DcmDspDataType)} == SINT16)) ||(({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspPid/DcmDspPidData/DcmDspPidDataSize) == NULL}) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidService01/DcmDspPidDataType)} ==SINT16))
	data	
	Type	sint32
	Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataSize)} == NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/ DcmDspDataType)} == SINT32)) ||(({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspPid/DcmDspPidData/DcmDspPidDataSize) == NULL}) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidService01/DcmDspPidDataType)} ==SINT32))
	data	
	Type	uint32
	Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataSize)} == NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDspDcmDspData/ DcmDspDataType)} == UINT32)) ||(({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspPid/DcmDspPidData/DcmDspPidDataSize) == NULL} && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidService01/DcmDspPidDataType)} ==UINT32))
	data	
	Type	uint8

	Variation	((1 < {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataSize)} <= 8) && ({ecuc(Dcm/DcmConfigSet/DcmDspDcmDspData/ DcmDspDataType)} == UINT8)) ||((1 < {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspPid/DcmDspPidData/DcmDspPidDataSize)} <= 8) && ({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/ DcmDspPidDataType)} == UINT8))
	data	
	Type	uint16
	Variation	((8 < {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataSize)} <= 16) && ({ecuc(Dcm/DcmConfigSet/DcmDspDcmDspData/ DcmDspDataType)}} == UINT16)) ||((8 < {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspPid/DcmDspPidData/DcmDspPidDataSize)} <= 16) && ({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/ DcmDspPidDataType)} == UINT16))
	data	
	Type	DataArrayType_{Data}
	Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataSize)}> 8) && ({ecuc(Dcm/DcmConfigSet/DcmDspDcmDspData/DcmDspDataType)} == UINT8) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/ DcmDspDataSize)} % 8 == 0)) || (({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspPid/DcmDspPidData/DcmDspPidDataSize)}> 8) && ({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/ DcmDspPidDataType)} == UINT8)&& ({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspPid/DcmDspPidData/DcmDspPidDataSize)} % 8 == 0)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})

⌋ ()

8.8.3.2.12 DataServices callout

From the point of view of the DCM, the operations have the following signatures:

Note : The OpStatus parameter shall only exist for asynchronous operations (if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC). In case of synchronous operations (DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC), the OpStatus parameter shall not exist.

1.1.1.1.1.14 ReadData

[SWS_Dcm_00793] ⌈

if DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used:

Service name:	Xxx_ReadData
Syntax:	`Std_ReturnType Xxx_ReadData( uint8* Data )`
Service ID[hex]:	0x34

Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	Data	Buffer where the requested data shall be copied to
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	This function requests to the application a data value of a DID/PID if DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER.	

if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC, the following definition is used:

Service name:	Xxx_ReadData	

Syntax:	`Std_ReturnType Xxx_ReadData(`
	`  Dcm_OpStatusType OpStatus,`
	`  uint8* Data`
	`)`	

Service ID[hex]:	0x3b	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	Data	Buffer where the requested data shall be copied to
Return value:	Std_ReturnType	E_OK: Request was successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application a data value of a DID/PID if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER.	

⌋()

1.1.1.1.1.15 WriteData

[SWS_Dcm_00794] ⌈

if DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used:

if DcmDspDataFixedLength is set to TRUE, the following definition is used:

Service name:	Xxx_WriteData	

Syntax:	`Std_ReturnType Xxx_WriteData(`
	`  uint8* Data,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x51	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Data	Buffer containing the data to be written
Parameters	None	

(inout):		
Parameters (out):	ErrorCode	If the operation Xxx_WriteData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests the application to write a data value of a DID.	

DcmDspDataFixedLength is set to FALSE, the following definition is used:

Service name:	Xxx_WriteData	

Syntax:	`Std_ReturnType Xxx_WriteData(`
	`  uint8* Data,`
	`  uint16 DataLength,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x52	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Data	Buffer containing the data to be written
	DataLength	Length in byte of the data to be written
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_WriteData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests the application to write a data value of a DID.	

if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC, the following definition is used:

DcmDspDataFixedLength is set to TRUE, the following definition is used:

Service name:	Xxx_WriteData	

Syntax:	`Std_ReturnType Xxx_WriteData(`
	`  uint8* Data,`
	`  Dcm_OpStatusType OpStatus,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x35	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Data	Buffer containing the data to be written
	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_WriteData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.

Description:	This function requests the application to write a data value of a DID.

if DcmDspDataFixedLength is set to FALSE, the following definition is used:

Service name:	Xxx_WriteData	
Syntax:	`Std_ReturnType Xxx_WriteData( uint8* Data, uint16 DataLength, Dcm_OpStatusType OpStatus, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID[hex]:	0x3e	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Data	Buffer containing the data to be written
	DataLength	Length in byte of the data to be written
	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_WriteData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests the application to write a data value of a DID.	

」()

1.1.1.1.1.16 ReadDataLength

[SWS_Dcm_00796]「

if DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used:

Service name:	Xxx_ReadDataLength	
Syntax:	`Std_ReturnType Xxx_ReadDataLength( uint16* DataLength )`	
Service ID[hex]:	0x36	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	DataLength	Length in byte of the data to be read
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	This function requests the application to return the data length in byte of a Data.	

if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC, the following definition is used:

Service name:	Xxx_ReadDataLength

Syntax:	`Std_ReturnType Xxx_ReadDataLength(`
	`  Dcm_OpStatusType OpStatus,`
	`  uint16* DataLength`
	`)`	

Service ID[hex]:	0x4c	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	DataLength	Length in byte of the data to be read
Return value:	Std_ReturnType	E_OK: this value is always returned. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests the application to return the data length in byte of a Data.	

」()

1.1.1.1.17 ConditionCheckRead

[SWS_Dcm_00797]「

if DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used:

Service name:	Xxx_ConditionCheckRead	

Syntax:	`Std_ReturnType Xxx_ConditionCheckRead(`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x49	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ConditionCheckRead returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests to the application if the conditions to read the Data are correct.	

if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC, the following definition is used:

Service name:	Xxx_ConditionCheckRead

Syntax:	`Std_ReturnType Xxx_ConditionCheckRead(`
	`  Dcm_OpStatusType OpStatus,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`

Service ID[hex]:	0x37
Sync/Async:	Asynchronous
Reentrancy:	Non Reentrant

Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ConditionCheckRead returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application if the conditions to read the Data are correct.	

」()

1.1.1.1.1.18 GetScalingInformation

[SWS_Dcm_00798] 「

if DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used:

Service name:	Xxx_GetScalingInformation	

Syntax:	`Std_ReturnType Xxx_GetScalingInformation(`
	`  uint8* ScalingInfo,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x4b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	ScalingInfo	Scaling information
	ErrorCode	If the operation Xxx_GetScalingInformation returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests to the application for the scaling information of a Data.	

if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC, the following definition is used:

Service name:	Xxx_GetScalingInformation	

Syntax:	`Std_ReturnType Xxx_GetScalingInformation(`
	`  Dcm_OpStatusType OpStatus,`
	`  uint8* ScalingInfo,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x38	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters	None	

(inout):		
Parameters (out):	ScalingInfo	Scaling information
	ErrorCode	If the operation Xxx_GetScalingInformation returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application for the scaling information of a Data.	

」()

1.1.1.1.1.19 ReturnControlToECU

[SWS_Dcm_00799]「if DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used:

Service name:	Xxx_ReturnControlToECU	

Syntax:	`Std_ReturnType Xxx_ReturnControlToECU(`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x4f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ReturnControlToECU returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests to the application to return control to ECU of an IOControl.	

if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC, the following definition is used:

Service name:	Xxx_ReturnControlToECU	

Syntax:	`Std_ReturnType Xxx_ReturnControlToECU(`
	`  Dcm_OpStatusType OpStatus,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x39	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ReturnControlToECU returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.

Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application to return control to ECU of an IOControl.	

」()

1.1.1.1.1.20 ResetToDefault

[SWS_Dcm_00800]
「if DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used:

Service name:	Xxx_ResetToDefault	

Syntax:	`Std_ReturnType Xxx_ResetToDefault(`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x4d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ResetToDefault returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests to the application to reset an IOControl to default value.	

if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC, the following definition is used:

Service name:	Xxx_ResetToDefault	

Syntax:	`Std_ReturnType Xxx_ResetToDefault(`
	`  Dcm_OpStatusType OpStatus,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x3c	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ResetToDefault returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application to reset an IOControl to default value.	

」()

1.1.1.1.1.21 FreezeCurrentState

[SWS_Dcm_00801]
⌈if DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used:

Service name:	Xxx_FreezeCurrentState	
Syntax:	`Std_ReturnType Xxx_FreezeCurrentState( Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID[hex]:	0x4a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_FreezeCurrentState returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests to the application to freeze the current state of an IOControl.	

if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC, the following definition is used:

Service name:	Xxx_FreezeCurrentState	
Syntax:	`Std_ReturnType Xxx_FreezeCurrentState( Dcm_OpStatusType OpStatus, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID[hex]:	0x3a	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_FreezeCurrentState returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application to freeze the current state of an IOControl.	

⌋()

1.1.1.1.1.22 ShortTermAdjustment

[SWS_Dcm_00802]

⌜ If DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC and if DcmDspDataFixedLength is set to TRUE, the following definition is used:

Service name:	Xxx_ShortTermAdjustment	
Syntax:	`Std_ReturnType Xxx_ShortTermAdjustment( uint8* ControlStateInfo, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID[hex]:	0x50	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControlStateInfo	ControlState information contained in the ControlOptionRecord parameter of the InputOutputControlByIdentifier diagnostic request
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ShortTermAdjustment returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests to the application to adjust the IO signal.	

If DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC and if DcmDspDataFixedLength is set to TRUE, the following definition is used:

Service name:	Xxx_ShortTermAdjustment	
Syntax:	`Std_ReturnType Xxx_ShortTermAdjustment( uint8* ControlStateInfo, Dcm_OpStatusType OpStatus, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID[hex]:	0x3d	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControlStateInfo	ControlState information contained in the ControlOptionRecord parameter of the InputOutputControlByIdentifier diagnostic request
	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ShortTermAdjustment returns value

		E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application to adjust the IO signal.	

If DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC and if DcmDspDataFixedLength is set to FALSE, the following definition is used:

Service name:	Xxx_ShortTermAdjustment	

Syntax:	`Std_ReturnType Xxx_ShortTermAdjustment(`
	`  uint8* ControlStateInfo,`
	`  uint16 DataLength,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x54	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControlStateInfo	ControlState information contained in the ControlOptionRecord parameter of the InputOutputControlByIdentifier diagnostic request
	DataLength	Length in byte of the data to be written
Parameters (inout):	None	
Parameters (out):	ErrorCode	NRC to be sent in the negative response in case of failure (E_NOT_OK)
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests to the application to adjust the IO signal.	

If DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC and if DcmDspDataFixedLength is set to FALSE, the following definition is used:

Service name:	Xxx_ShortTermAdjustment	

Syntax:	`Std_ReturnType Xxx_ShortTermAdjustment(`
	`  uint8* ControlStateInfo,`
	`  uint16 DataLength,`
	`  Dcm_OpStatusType OpStatus,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x55	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControlStateInfo	ControlState information contained in the ControlOptionRecord parameter of the InputOutputControlByIdentifier diagnostic request
	DataLength	Length in byte of the data to be written
	OpStatus	Status of the current operation
Parameters (inout):	None	

Parameters (out):	ErrorCode	NRC to be sent in the negative response in case of failure (E_NOT_OK)
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application to adjust the IO signal.	

」()

8.8.3.3 DataServices_DIDRange_{Range}
8.8.3.3.1 Client Server interface

The following interface defines an operation needed to get the DID range

Using the concepts of the SW-C template, the interface is defined as follows:
[**SWS_Dcm_00769**]「`ClientServerInterface DataServices_DIDRange_{Range}`

Name	DataServices_DIDRange_{Range}	
Comment	--	
IsService	true	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp/DcmDspDidRange. DcmDspDidRangeUsePort)} == TRUE Range = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp/DcmDspDidRange.SHORT-NAME)})	
Possible Errors	0	E_OK
	1	E_NOT_OK
	10	DCM_E_PENDING

Operations

IsDidAvailable		
Comments	--	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp/DcmDspDidRange. DcmDspDidRangeHasGaps)} == TRUE	
Parameters	DID	
	Comment	--
	Type	uint16
	Variation	--
	Direction	IN

	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	supported	
	Comment	--
	Type	uint8
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

ReadDidData		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRange->DcmDspDidRangeInfoRef/ DcmDspDidAccess/DcmDspDidRead)} != NULL)	
Parameters	DID	
	Comment	--
	Type	uint16
	Variation	--
	Direction	IN
	Data	
	Comment	--
	Type	DcmDspDidRangeArrayType_{Range}
	Variation	Range = ({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspDidRange.SHORT-NAME)})
	Direction	OUT
	OpStatus	
	Comment	--

	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	DataLength	
	Comment	--
	Type	uint16
	Variation	--
	Direction	OUT
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.
WriteDidData		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRange->DcmDspDidRangeInfoRef/ DcmDspDidAccess/DcmDspDidWrite)} != NULL)	
Parameters	DID	
	Comment	--
	Type	uint16
	Variation	--
	Direction	IN
	Data	
	Comment	--
	Type	DcmDspDidRangeArrayType_{Range}
	Variation	Range = ({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspDidRange.SHORT-NAME)})

	Direction	IN
	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	DataLength	
	Comment	--
	Type	uint16
	Variation	--
	Direction	IN
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

⌋()

8.8.3.3.2 DataServices callout

From the point of view of the DCM, the operations have the following signatures:

Note : The OpStatus parameter should only be used for asynchronous operations (if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC). In case of synchronous operations (DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC), the OpStauts parameter should not be used.

1.1.1.1.1.23 IsDidAvailable

[SWS_Dcm_00803]

⌈

Service name:	Xxx_IsDidAvailable

Syntax:	`Std_ReturnType Xxx_IsDidAvailable(`
	`  uint16 DID,`
	`  Dcm_OpStatusType OpStatus,`

	uint8* supported)	
Service ID[hex]:	0x53	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DID	DID value
	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	supported	Indicate if the DID is available within the range or not
Return value:	Std_ReturnType	E_OK: this value is always returned. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests if a specific DID is available within the range or not.	

]()

1.1.1.1.1.24 ReadDidData

[SWS_Dcm_00804]

⌜

Service name:	Xxx_ReadDidData	

Syntax:	`Std_ReturnType Xxx_ReadDidData(`
	`  uint16 DID,`
	`  uint8* Data,`
	`  Dcm_OpStatusType OpStatus,`
	`  uint16 DataLength,`
	`  Dcm_NegativeResponseCodeType ErrorCode`
	`)`	

Service ID[hex]:	0x40	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DID	Data ID value
	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	Data	Buffer where the requested data shall be copied to
	DataLength	Length of the data to be read
	ErrorCode	If the operation Xxx_ReadDidData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application a data value of a DID	

⌟()

1.1.1.1.1.25 WriteDidData

[SWS_Dcm_00805] ⌜

Service name:	Xxx_WriteDidData	

Syntax:	`Std_ReturnType Xxx_WriteDidData(`
	`  uint16 DID,`
	`  uint8* Data,`
	`  Dcm_OpStatusType OpStatus,`
	`  uint16 DataLength,`
	`  Dcm_NegativeResponseCodeType ErrorCode`
	`)`	

Service ID[hex]:	0x41	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DID	Data ID value
	Data	Buffer containing the data to be written
	OpStatus	Status of the current operation
	DataLength	Length of the data to be written
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_WriteDidData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.

		DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests the application to write a data value of a DID.	

」()

8.8.3.4 InfotypeServices_{VehInfoData}

The following interface defines an operation needed to get data from one or several SW-C in order to supply OBD Service $09 (see SWS_Dcm_00423).

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

[**SWS_Dcm_00688**]「`ClientServerInterface InfotypeServices_{VehInfoData}`

Name	InfotypeServices_{VehInfoData}	
Comment	--	
IsService	true	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspVehInfo/DcmDspVehInfoData/ DcmDspVehInfoDataUsePort)}==TRUE VehInfoData = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspVehInfo/ DcmDspVehInfoData.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	10	DCM_E_PENDING

Operations

GetInfotypeValueData		
Comments	--	
Variation	--	
Parameters	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	DataValueBuffer	
	Comment	--
	Type	InfoTypeServicesArrayType_{VehInfoData}
	Variation	VehInfoData = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspVehInfo/DcmDspVehInfoData.SHORT-NAME)}
	Direction	OUT

Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

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

```
Std_ReturnType Xxx_GetInfotypeValueData(Dcm_OpStatusType OpStatus,
      uint8* DataValueBuffer)
```

8.8.3.5 RoutineServices_{RoutineName}

The following interface defines operations needed for the UDS Service RoutineControl (0x31) (see SWS_Dcm_00400, SWS_Dcm_00401, SWS_Dcm_00402, SWS_Dcm_00403, SWS_Dcm_00404, SWS_Dcm_00405).

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

[**SWS_Dcm_00690**] `ClientServerInterface RoutineServices_{RoutineName}`

Name	RoutineServices_{RoutineName}	
Comment	--	
IsService	true	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRRoutine.DcmDspRoutineUsePort)} == TRUE RoutineName = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRRoutine.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	10	DCM_E_PENDING
	12	DCM_E_FORCE_RCRRP

Operations

RequestResults		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineFixedLength)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRequestResultsRoutineSupported)} == TRUE)	
Parameters	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--

	Direction	IN
	DataOut_{Signal}	
	Comment	--
	Type	RequestDataOutType_{Routine}_{Signal}
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineRequestResOut.DcmDspRoutineRequestResOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	Direction	OUT
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)
RequestResults		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineFixedLength)} == FALSE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRequestResultsRoutineSupported)} == TRUE)	
Parameters	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	DataOut_{Signal}	

	Comment	--
	Type	RequestDataOutType_{Routine}_{Signal}
	Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineRequestResOut. DcmDspRoutineRequestResOutSignal. DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineRequestResOut. DcmDspRoutineRequestResOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	OUT
	DataOut_{Signal}	
	Comment	--
	Type	RequestFlexibleOutArrayDataType_{Routine}_{Signal}
	Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineRequestResOut. DcmDspRoutineRequestResOutSignal. DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineRequestResOut. DcmDspRoutineRequestResOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	OUT
	currentDataLength	
	Comment	--
	Type	uint16
	Variation	--
	Direction	OUT
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Request was not successful

	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)
Start		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineFixedLength)} == TRUE)	
Parameters	DataIn_{Signal}	
	Comment	--
	Type	StartDataInType_{Routine}_{Signal}
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStartIn.DcmDspRoutineStartInSignal. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	IN
	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	DataOut_{Signal}	
	Comment	--
	Type	StartDataOutType_{Routine}_{Signal}
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStartOut. DcmDspRoutineStartOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	OUT
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType

	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)
Start		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineFixedLength)} == FALSE)	
Parameters	DataIn_{Signal}	
	Comment	--
	Type	StartDataInType_{Routine}_{Signal}
	Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStartIn.DcmDspRoutineStartInSignal. DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStartIn.DcmDspRoutineStartInSignal. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	IN
	DataIn_{Signal}	
	Comment	--
	Type	StartFlexibleInArrayDataType_{Routine}_{Signal}
	Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStartIn.DcmDspRoutineStartInSignal. DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStartIn.DcmDspRoutineStartIn.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	IN

	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	DataOut_{Signal}	
	Comment	--
	Type	StartDataOutType_{Routine}_{Signal}
	Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStartOut. DcmDspRoutineStartOutSignal. DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStartOut. DcmDspRoutineStartOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	OUT
	DataOut_{Signal}	
	Comment	--
	Type	StartFlexibleOutArrayDataType_{Routine}_{Signal}
	Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStartOut. DcmDspRoutineStartOutSignal. DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStartOut.DcmDspRoutineStartOut. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	OUT
	currentDataLength	
	Comment	--
	Type	uint16
	Variation	--
	Direction	INOUT
	ErrorCode	
	Comment	--

	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)
Stop		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineFixedLength)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspStopRoutineSupported)} == TRUE)	
Parameters	DataIn_{Signal}	
	Comment	--
	Type	StopDataInType_{Routine}_{Signal}
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineInfoRef.DcmDspRoutineStopIn.DcmDspRoutineStopInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	Direction	IN
	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	DataOut_{Signal}	
	Comment	--
	Type	StopDataOutType_{Routine}_{Signal}
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/

		DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStopOut.DcmDspRoutineStopOutSignal. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	OUT
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)
Stop		
Comments	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineFixedLength)} == FALSE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine. DcmDspStopRoutineSupported)} == TRUE)	
Parameters	DataIn_{Signal}	
	Comment	--
	Type	StopDataInType_{Routine}_{Signal}
	Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStopIn.DcmDspRoutineStopInSignal. DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStopIn.DcmDspRoutineStopInSignal. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	IN
	DataIn_{Signal}	

	Comment	--
	Type	StopFlexibleInArrayDataType_{Routine}_{Signal}
	Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStopIn.DcmDspRoutineStopInSignal. DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStopIn.DcmDspRoutineStopIn.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	IN
	OpStatus	
	Comment	--
	Type	Dcm_OpStatusType
	Variation	--
	Direction	IN
	DataOut_{Signal}	
	Comment	--
	Type	StopDataOutType_{Routine}_{Signal}
	Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStopOut.DcmDspRoutineStopOutSignal. DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStopOut.DcmDspRoutineStopOutSignal. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	OUT
	DataOut_{Signal}	
	Comment	--
	Type	StopFlexibleOutArrayDataType_{Routine}_{Signal}
	Variation	{ecuc(DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStopOut.DcmDspRoutineStopOutSignal. DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.DcmDspRoutineInfoRef. DcmDspRoutineStopOut.DcmDspRoutineStopOut. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine.SHORT-NAME)}
	Direction	OUT

	currentDataLength	
	Comment	--
	Type	uint16
	Variation	--
	Direction	INOUT
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)

」()

[SWS_Dcm_00668]「 If the operation Start() returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to ErrorCode parameter value.」()

[SWS_Dcm_00669]「 If the operation Start() returns value DCM_E_FORCE_RCRRP, the DCM module shall start the transmission of NRC 0x78.」()

[SWS_Dcm_00670]「 If the operation Stop() returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to ErrorCode parameter value.」()

[SWS_Dcm_00671] ⌈ If the operation Stop() returns value DCM_E_FORCE_RCRRP, the DCM module shall start the transmission of NRC 0x78.⌋()

[SWS_Dcm_00672] ⌈ If the operation RequestResults() returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to ErrorCode parameter value.⌋()

[SWS_Dcm_00673] ⌈ If the operation RequestResults () returns value DCM_E_FORCE_RCRRP, the DCM module shall start the transmission of NRC 0x78.⌋()

From the point of view of the DCM, the operations have the following signatures:

```
//If DcmDspRoutineFixedLength is set to FALSE, the following
//definition is used
Std_ReturnType Xxx_Start(DcmDspRoutineSignalType dataIn1,…,uint8*
dataInN,
                         Dcm_OpStatusType OpStatus,
                         DcmDspRoutineSignalType* dataOut1,…,uint8*
dataOutN,
                         uint16* currentDataLength,
                         Dcm_NegativeResponseCodeType* ErrorCode)
//If DcmDspRoutineFixedLength is set to TRUE, the following
//definition is used
Std_ReturnType Xxx_Start(DcmDspRoutineSignalType
dataIn1,…,DcmDspRoutineSignalType dataInN,
                         Dcm_OpStatusType OpStatus,
                         DcmDspRoutineSignalType* dataOut1,…,
                         DcmDspRoutineSignalType* dataOutN,
                         Dcm_NegativeResponseCodeType* ErrorCode)

//If DcmDspRoutineFixedLength is set to FALSE, the following
//definition is used
Std_ReturnType Xxx_Stop(DcmDspRoutineSignalType dataIn1,…,uint8*
dataInN,
                        Dcm_OpStatusType OpStatus,
                        DcmDspRoutineSignalType* dataOut1,…,uint8*
dataOutN
                        uint16* currentDataLength,
                        Dcm_NegativeResponseCodeType* ErrorCode)

//If DcmDspRoutineFixedLength is set to TRUE, the following
//definition is used
Std_ReturnType Xxx_Stop(DcmDspRoutineSignalType
dataIn1,…,DcmDspRoutineSignalType dataInN,
                        Dcm_OpStatusType OpStatus,
                        DcmDspRoutineSignalType* dataOut1,…,
                        DcmDspRoutineSignalType* dataOutN,
```

```
                        Dcm_NegativeResponseCodeType* ErrorCode)

//If DcmDspRoutineFixedLength is set to FALSE, the following
//definition is used
Std_ReturnType Xxx_RequestResults(Dcm_OpStatusType OpStatus,
                        DcmDspRoutineSignalType* dataOut1,…,uint8*
dataOutN
                        uint16* currentDataLength,
                        Dcm_NegativeResponseCodeType* ErrorCode)

//If DcmDspRoutineFixedLength is set to TRUE, the following
//definition is used
Std_ReturnType Xxx_RequestResults(Dcm_OpStatusType OpStatus,
                        DcmDspRoutineSignalType* dataOut1,…,
                        DcmDspRoutineSignalType* dataOutN,
                        Dcm_NegativeResponseCodeType* ErrorCode)
```

8.8.3.6 RequestControlServices_{Tid}

The interface RequestControlServices_{Name} allows the DCM to provide OBD Service $08 (see SWS_Dcm_00419).
Using the concepts of the SW-C template, the interface is defined as follows:

[**SWS_Dcm_00691**]⌈ `ClientServerInterface RequestControlServices_{Tid}`

Name	RequestControlServices_{Tid}	
Comment	--	
IsService	true	
Variation	Tid = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestControl.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

RequestControl		
Comments	--	
Variation	--	
Parameters	OutBuffer	
	Comment	--
	Type	RequestControlServicesOutArrayType_{Tid}
	Variation	Tid = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestControl.SHORT-NAME)}
	Direction	OUT
	InBuffer	
	Comment	--

	Type	RequestControlServicesInArrayType_{Tid}
	Variation	Tid = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestControl.SHORT-NAME)}
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

」()

From the point of view of the DCM, the operation has the following signature:
`Std_ReturnType Xxx_RequestControl(uint8* OutBuffer, uint8* InBuffer)`

8.8.3.7 CallbackDCMRequestServices

The interface CallbackDCMRequestServices provides information on the status of the protocol communication and allows the Application to disallow a protocol (see SWS_Dcm_00036, SWS_Dcm_00144, SWS_Dcm_00145, SWS_Dcm_00146; SWS_Dcm_00147, SWS_Dcm_00459).

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

[**SWS_Dcm_00692**]「`ClientServerInterface CallbackDCMRequestServices {`

Name	CallbackDCMRequestServices	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	5	E_PROTOCOL_NOT_ALLOWED

Operations

StartProtocol		
Comments	--	
Variation	--	
Parameters	ProtocolID	
	Comment	--
	Type	Dcm_ProtocolType
	Variation	--
	Direction	IN

Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	E_PROTOCOL_NOT_ALLOWED	conditions in application allows no further procession of protocol
StopProtocol		
Comments	--	
Variation	--	
Parameters	ProtocolID	
	Comment	--
	Type	Dcm_ProtocolType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

」()

[SWS_Dcm_00674]「 If the operation StartProtocol() returns value E_NOT_OK or E_PROTOCOL_NOT_ALLOWED, the DCM module shall send a negative response with NRC 0x22 (Conditions not correct).」()

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)
```

8.8.3.8 ServiceRequestNotification

The interface ServiceRequestNotification 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 SWS_Dcm_00218, SWS_Dcm_00462, SWS_Dcm_00463).

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

[**SWS_Dcm_00694**]⌈ `ClientServerInterface ServiceRequestNotification`

Name	ServiceRequestNotification	
Comment	--	
IsService	true	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsd/ DcmDsdRequestManufacturerNotificationEnabled)}==TRUE) || ({ecuc(Dcm/ DcmConfigSet/DcmDsd/DcmDsdRequestSupplierNotificationEnabled)}==TRUE)	
Possible Errors	0	E_OK
	1	E_NOT_OK
	8	E_REQUEST_NOT_ACCEPTED

Operations

Confirmation		
Comments	--	
Variation	--	
Parameters	SID	
	Comment	Value of service identifier
	Type	uint8
	Variation	--
	Direction	IN
	ReqType	
	Comment	Addressing type of the request(0=physical request, 1=functional request)
	Type	uint8
	Variation	--
	Direction	IN
	SourceAddress	

	Comment	Dcm client description
	Type	uint16
	Variation	--
	Direction	IN
	ConfirmationStatus	
	Comment	Confirmation of a successful transmission or a transmission error of a diagnostic service.
	Type	Dcm_ConfirmationStatusType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
Indication		
Comments	--	
Variation	--	
Parameters	SID	
	Comment	Value of service identifier
	Type	uint8
	Variation	--
	Direction	IN
	RequestData	
	Comment	This parameter contains the complete request data (diagnostic buffer), except the service ID
	Type	RequestDataArray
	Variation	--
	Direction	IN
	DataSize	
	Comment	This parameter defines how many bytes in the

		RequestData parameter are valid
	Type	uint16
	Variation	--
	Direction	IN
	ReqType	
	Comment	Addressing type of the request(0=physical request, 1=functional request
	Type	uint8
	Variation	--
	Direction	IN
	SourceAddress	
	Comment	Dcm client description
	Type	uint16
	Variation	--
	Direction	IN
	ErrorCode	
	Comment	--
	Type	Dcm_NegativeResponseCodeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	E_REQUEST_NOT_ACCEPTED	no response will be sent

」()

[SWS_Dcm_00677]「 If the operation Indication() returns value E_REQUEST_NOT_ACCEPTED, the DCM module shall not send any diagnostic response and shall end the current diagnostic request management.」()

[SWS_Dcm_00678] ⌈ If the operation Indication() returns value E_NOT_OK, the DCM module shall send a negative response with NRC value equal to ErrorCode parameter value. ⌋()

From the point of view of the DCM, the operations has the following signatures:

```
Std_ReturnType Xxx_Indication(uint8 SID, uint8* RequestData, uint16
DataSize, uint8 ReqType, uint16 SourceAddress,
Dcm_NegativeResponseCodeType* ErrorCode )

Std_ReturnType Xxx_Confirmation(uint8 SID, uint8 ReqType, uint16
SourceAddress, Dcm_ConfirmationStatusType ConfirmationStatus)
```

8.9 DCM as Service-Component

This section formally specifies the corresponding AUTOSAR Service using the concepts of the Software-Component-Template.
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.

Naming of the port : The prefix of the port name is fixed and defined herafter (e.g. DataServices_). The name behind the prefix corresponds to the name of the associated container in the ECU configuration and can be freely defined during the configuration step.
e.g. : for a *DcmDspData* container called *Speed* the port name would be *DataServices_Speed*

```
ServiceSwComponentType Dcm {

  //the presence and name of this port is configuration-independent
  ProvidePort DCMServices DCMServices;

  //see configuration parameter DcmDspSecurityRow
  RequirePort SecurityAccess_{SecurityLevel}
SecurityAccess_{SecurityLevel};
  …


  //see configuration parameter DcmDspData
RequirePort DataServices_{Data} DataServices_{Data};
ProvidePort DataServices_{Data} DataServices_{Data}; // Only if
the data can be written and DcmDspDataUsePort is set to
USE_DATA_SENDER_RECEIVER
  …
```

```
  //see configuration parameter DcmDspVehInfoData
  RequirePort InfotypeServices_{VehInfoData}
      InfotypeServices_{VehInfoData}
  …


  //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
DcmDsdServiceRequestManufacturerNotication
  RequirePort ServiceRequestNotification
    RequestManufacturerNotification_<SWC>;
  …

//see configuration parameter DcmDsdServiceRequestSupplierNotication
  RequirePort ServiceRequestNotification
    ServiceRequestSupplierNotification_<SWC>;
  …

//Interface containing the DEM operations DcmClearDTC,
//Dem_DisableDTCSetting and Dem_EnableDTCSetting
RequirePort Dem/DcmIf Dcm;

//see configuration parameter DcmDspDidRange
RequirePort DataServices_DIDRange_{Range}
DataServices_DIDRange_{Range};

//Note: When service 0x19 subfunctions 0x14 is used (call to
//Dem_DcmGetNextFilteredDTCAndFDC), the following is defined:
//Non-DEM-internal calculated fault detection counters are typically
//requested from SW-Cs through the RTE. To indicate an equivalent
call-tree //for these runables, a work-around is used: The DCM main
function //specifies a trigger to the DEM interface GeneralEvtInfo
(operation //GetFaultDetectionCounter), which triggers the according
 ehavior (refer to //RunnableEntity GetFaultDetectionCounter,
chapter "Service Interface //DiagnosticInfo & General" in DEM SWS)."
RequirePort Dem/CallbackGetFaultDetectCounter CBFaultDetectCtrDummy
(The client-server interface can be used from the DEM.)

RunnableEntity MainFunction
           symbol "Dcm_MainFunction"
           canbeInvokedConcurrently = FALSE
           SSCP = port CBFaultDetectCtrDummy,
GetFaultDetectionCounter
```

Connector from CBFaultDetectCtrDummy to Dem/GeneralEvtInfo

}

[SWS_Dcm_01030]⌈ProvidedPort DCMServices

Name	DCMServices		
Kind	ProvidedPort	Interface	DCMServices
Description	--		
Variation	--		

⌋()

[SWS_Dcm_01031]⌈ProvidedPort DataServices_{Data}

Name	DataServices_{Data}		
Kind	ProvidedPort	Interface	DataServices_{Data}
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == (USE_DATA_SENDER_RECEIVER)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidAccess/DcmDspDidWrite)} != NULL) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}		

⌋()

[SWS_Dcm_01032]⌈ProvidedPort DCM_Roe_{RoeName}

Name	DCM_Roe_{RoeName}		
Kind	ProvidedPort	Interface	Dcm_Roe
Description	--		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoe/DcmDspRoeEvent)} RoeName = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoe/DcmDspRoeEvent.SHORT-NAME)}		

⌋()

[SWS_Dcm_01033]⌈RequiredPort Dcm_CallbackDCMRequestServices_{Name}

Name	CallbackDCMRequestServices_{Name}		
Kind	RequiredPort	Interface	CallbackDCMRequestServices

Description	--
Variation	Name = {ecuc(Dcm/DcmConfigSet/DcmDsl/DcmDslCallbackDCMRequestService.SHORT-NAME)}

⌋()

[SWS_Dcm_01034] ⌈RequiredPort DataServices_DIDRange_{Range}

Name	DataServices_DIDRange_{Range}		
Kind	RequiredPort	Interface	DataServices_DIDRange_{Range}
Description	--		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp/DcmDspDidRange.DcmDspDidRangeUsePort)} == TRUE Range = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp/DcmDspDidRange.SHORT-NAME)})		

⌋()

[SWS_Dcm_01035] ⌈RequiredPort DataServices_{Data}

Name	DataServices_{Data}		
Kind	RequiredPort	Interface	DataServices_{Data}
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == (USE_DATA_SENDER_RECEIVER || USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_SYNCH_CLIENT_SERVER)) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataUsePort)} == (USE_DATA_SYNCH_CLIENT_SERVER || USE_DATA_SENDER_RECEIVER)) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)} || {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)}		

⌋()

[SWS_Dcm_01036] ⌈RequiredPort DataServices_{Data}

Name	DataServices_{Data}		
Kind	RequiredPort	Interface	DataServices_{Data}
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == (USE_DATA_SENDER_RECEIVER || USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_SYNCH_CLIENT_SERVER)) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER)		

	Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)} || {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)}

⌋()

[SWS_Dcm_01037] ⌈RequiredPort InfotypeServices_{VehInfoData}

Name	InfotypeServices_{VehInfoData}		
Kind	RequiredPort	Interface	InfotypeServices_{VehInfoData}
Description	--		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspVehInfo/DcmDspVehInfoData/ DcmDspVehInfoDataUsePort)}==TRUE VehInfoData = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspVehInfo/ DcmDspVehInfoData.SHORT-NAME)}		

⌋()

[SWS_Dcm_01038] ⌈RequiredPort RequestControlServices_{Tid}

Name	RequestControlServices_{Tid}		
Kind	RequiredPort	Interface	RequestControlServices_{Tid}
Description	--		
Variation	Tid = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestControl.SHORT-NAME)}		

⌋()

[SWS_Dcm_01039] ⌈RequiredPort RequestManufacturerNotification_{Name}

Name	RequestManufacturerNotification_{Name}		
Kind	RequiredPort	Interface	ServiceRequestNotification
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsd/ DcmDsdRequestManufacturerNotificationEnabled)}==TRUE) Name = {ecuc(Dcm/DcmConfigSet/DcmDsd/ DcmDsdServiceRequestManufacturerNotification.SHORT-NAME)}		

⌋()

[SWS_Dcm_01040] ⌈RequiredPort RoutineServices_{RoutineName}

Name	RoutineServices_{RoutineName}

Kind	RequiredPort	Interface	RoutineServices_{RoutineName}
Description	--		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRRoutine.DcmDspRoutineUsePort)} == TRUE RoutineName = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRRoutine.SHORT-NAME)}		

」()

[**SWS_Dcm_01041**]「RequiredPort SecurityAccess_{SecurityLevel}

Name	SecurityAccess_{SecurityLevel}		
Kind	RequiredPort	Interface	SecurityAccess_{SecurityLevel}
Description	--		
Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)} {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)} == (USE_SYNCH_CLIENT_SERVER || USE_ASYNCH_CLIENT_SERVER)		

」()

[**SWS_Dcm_01042**]「RequiredPort ServiceRequestSupplierNotification_{Name}

Name	ServiceRequestSupplierNotification_{Name}		
Kind	RequiredPort	Interface	ServiceRequestNotification
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsd/DcmDsdRequestSupplierNotificationEnabled)}==TRUE) Name = {ecuc(Dcm/DcmConfigSet/DcmDsd/DcmDsdServiceRequestSupplierNotification.SHORT-NAME)}		

」()

8.10 External diagnostic service processing

The following chapter applies only to external processed diagnostic services.

8.10.1 Dcm_ExternalSetNegResponse

[SWS_Dcm_00761] ⌜

Service name:	Dcm_ExternalSetNegResponse	
Syntax:	`void Dcm_ExternalSetNegResponse( Dcm_MsgContextType* pMsgContext, Dcm_NegativeResponseCodeType ErrorCode )`	
Service ID[hex]:	0x48	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	pMsgContext	Message-related information for one diagnostic protocol identifier
	ErrorCode	NRC to be sent in the negative response in case of failure.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Used by service interpreter outside of DCM to indicate that a the final response shall be a negative one. Dcm_ExternalSetNegResponse will not finalize the response processing.	

⌟()

8.10.2 Dcm_ExternalProcessingDone

[SWS_Dcm_00762] ⌜

Service name:	Dcm_ExternalProcessingDone	
Syntax:	`void Dcm_ExternalProcessingDone( Dcm_MsgContextType* pMsgContext )`	
Service ID[hex]:	0x31	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	pMsgContext	Message-related information for one diagnostic protocol identifier
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Used by service interpreter outside of DCM to indicate that a final response can be sent.	

⌟()

8.10.3 <Module>_<DiagnosticService>

[SWS_Dcm_00763] ⌜

Service name:	<Module>_<DiagnosticService>
Syntax:	`Std_ReturnType <Module>_<DiagnosticService>(`

	`Dcm_OpStatusType OpStatus,`
	`const Dcm_MsgContextType* pMsgContext`
	)	

Service ID[hex]:	0x32	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL: All In-parameters are valid DCM_CANCEL: All In-parameters are set to 0x0
	pMsgContext	Message-related information for one diagnostic protocol identifier The pointers in pMsgContext shall point behind the SID
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request was successful E_NOT_OK: Request was not successful DCM_E_PENDING: Request is not yet finished
Description:	Callout function. DCM shall call this callout function as soon as valid message is received on relevant DcmRxPduId on SID level . The usecase of multiple diagnostic protocols will be possible by using different arguments and the function shall be programmed in a way at it is reentrant. Caller is responsible for the lifetime of the argument pMsgContext. The name of the callout is defined within parameter DcmDsdSidTabFnc	

」()

[SWS_Dcm_00732] 「For the first call of <Module>_<DiagnosticService> the opStatus shall be set to DCM_INITIAL」()

[SWS_Dcm_00733] 「The DCM shall not accept further requests (on same or lower priority) until the application calls `Dcm_ExternalProcessingDone` to finalize the processing. Dcm-internal timeout handling (based on RCR-RP limitation) may lead to a cancellation of the external diagnostic service processing」()

[SWS_Dcm_00734] 「In case `Dcm_ExternalProcessingDone` is called in the context of <Module>_<DiagnosticService> the processing (including the start of the response transmission) shall be processed within the current `Dcm_MainFunction` cycle」()

[SWS_Dcm_00735] 「In case of cancellation the API <Module>_<DiagnosticService> is called again with the parameter opStatus set to DCM_CANCEL」()

[SWS_Dcm_00738] 「The DCM shall not use the opStatus parameter DCM_FORCE_RCRRP_OK in the API <Module>_<DiagnosticService>」()

[SWS_Dcm_00760] ⌈The return of DCM_E_PENDING shall do a re-triggering (e.g. in the next MainFunction cycle). Independent of any return value `Dcm_ExternalProcessingDone` is still necessary to finalize the processing⌋()

8.10.4 <Module>_<DiagnosticService>_<SubService>

[SWS_Dcm_00764] ⌈

Service name:	<Module>_<DiagnosticService>_<SubService>	
Syntax:	`Std_ReturnType <Module>_<DiagnosticService>_<SubService>( Dcm_OpStatusType OpStatus, const Dcm_MsgContextType* pMsgContext )`	
Service ID[hex]:	0x33	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL: All In-parameters are valid DCM_CANCEL: All In-parameters are set to 0x0
	pMsgContext	Message-related information for one diagnostic protocol identifier The pointer in pMsgContext shall point behind the SubFunction
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request was successful E_NOT_OK: Request was not successful DCM_E_PENDING: Request is not yet finished
Description:	Callout function. If a DcmDsdSubServiceFnc is configured for the received subservice, the DCM shall call this callout function as soon as this subservice is requested. The usecase of multiple diagnostic protocols will be possible by using different arguments and the function shall be programmed in a way that it is reentrant. Caller is responsible for the lifetime of the argument pMsgContext. The name of the callout is defined within parameter DcmDsdSubServiceFnc.	

⌋()

8.11 Internal interfaces (not normative)

The following interfaces are used in the DCM SWS in order to improve the understanding of the DCM module behavior. An implementation is not required to use these interfaces.

8.11.1 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()`

8.11.2 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_GetSesCtrlType()`

8.11.3 DspInternal_DcmConfirmation

```
void
DspInternal_DcmConfirmation(Dcm_IdContextType idContext,
                            uint16 SourceAddress
                           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. This API is also called if the response to a diagnostic service is suppressed.

8.11.4 DslInternal_ResponseOnOneEvent

```
Dcm_StatusType
DslInternal_ResponseOnOneEvent(const Dcm_MsgType MsgPtr,
                               Dcm_MsgLenType MsgLen,
                                uint16 SourceAddress
```

This API executes the processing of one event, requested internally in the DCM.

8.11.5 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).

8.11.6 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!

8.11.7 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.

8.11.8 DsdInternal_ProcessPage

```
void
DsdInternal_ProcessPage(Dcm_MsgLenType FilledPageLen)
```

DSP requests transmission of filled page

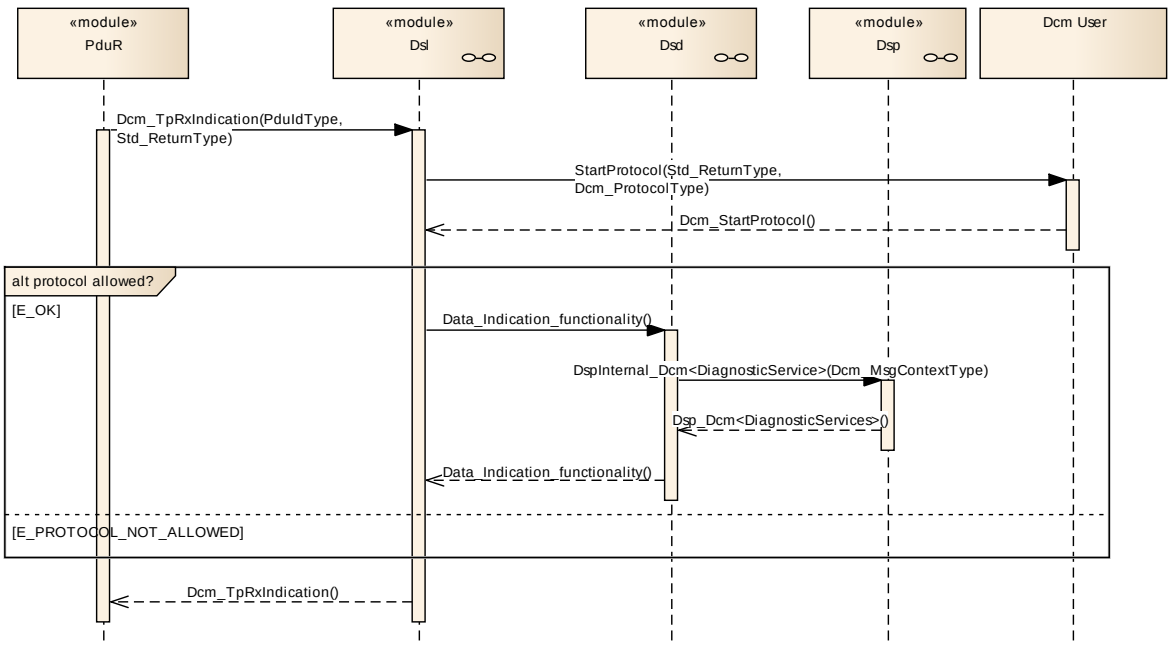
9 Sequence diagrams

9.1 Overview

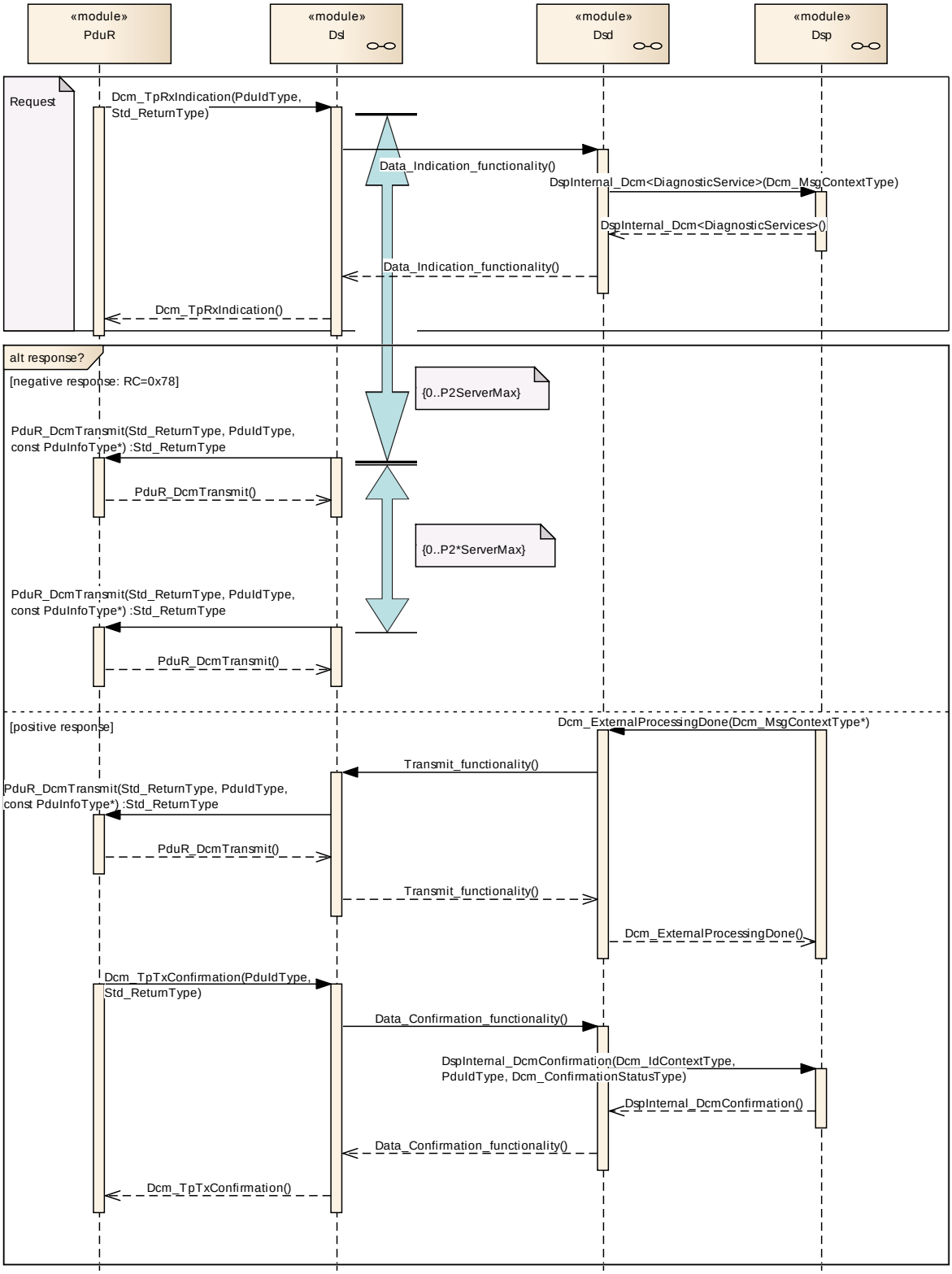
For clarification, the following sequence diagrams don't represent the full communication mechanism between the DCM module and the PduR module. This is to keep the sequence diagrams simple.
Before the `Dcm_TpRxIndication()` call, the PduR module will ask the DCM module for a buffer by calling `Dcm_StartOfReception()` and `Dcm_CopyRxData()`. 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_CopyTxData()` 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



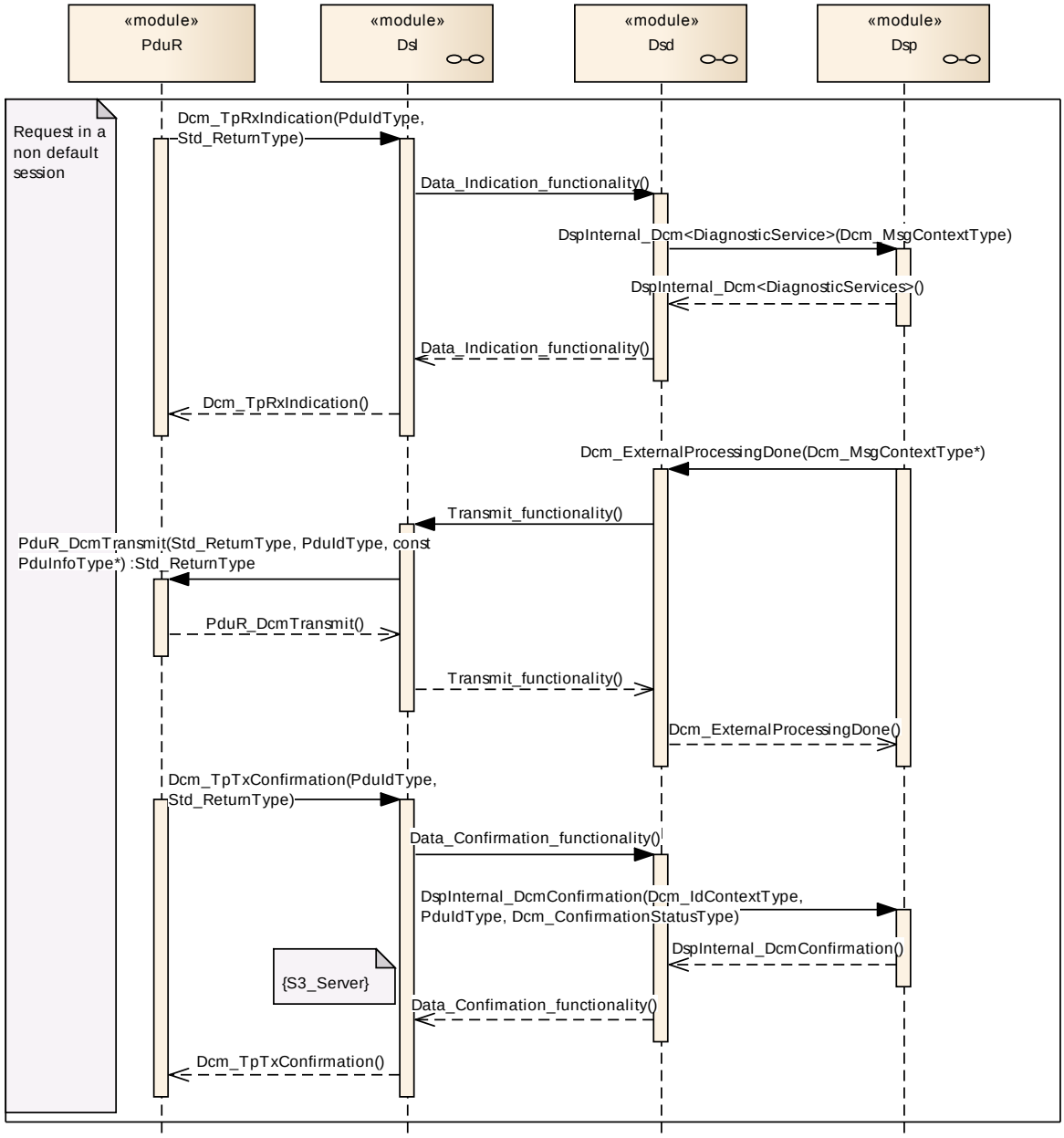
9.2.2 Process Busy behavior



Internally, the DSL submodule calculates the time to response the tester. In the case that the DSP submodule doesn't close the request with `Dcm_ExternalProcessingDone()` (in case of normal response handling) or `DsdInternal_ProcessPage()` (in case of paged-buffer handling) during the

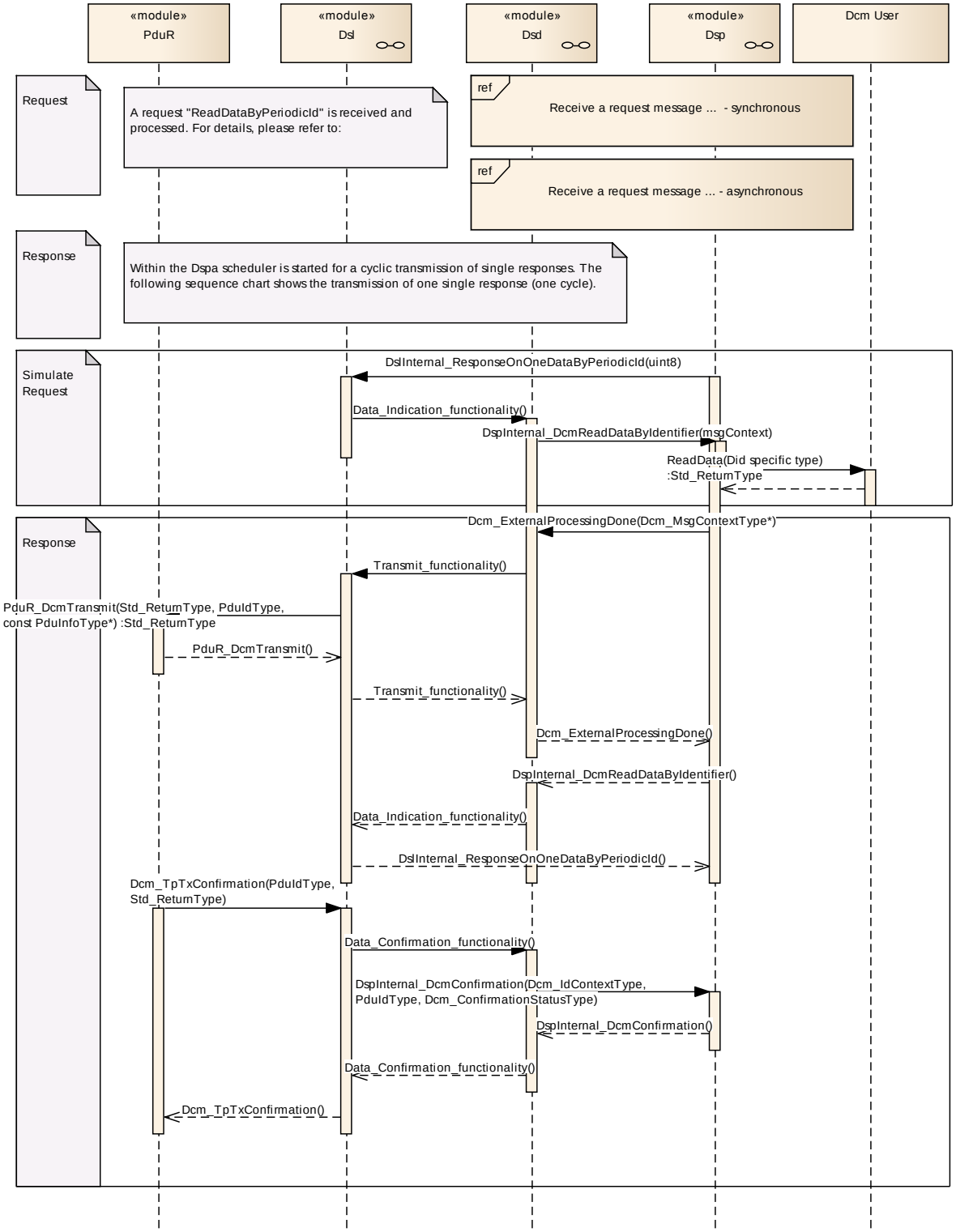
P2ServerMax and/or P2*ServerMax, the DSL submodule sends a negative response (requestCorectlyReceived-ResponsePending) independently.

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

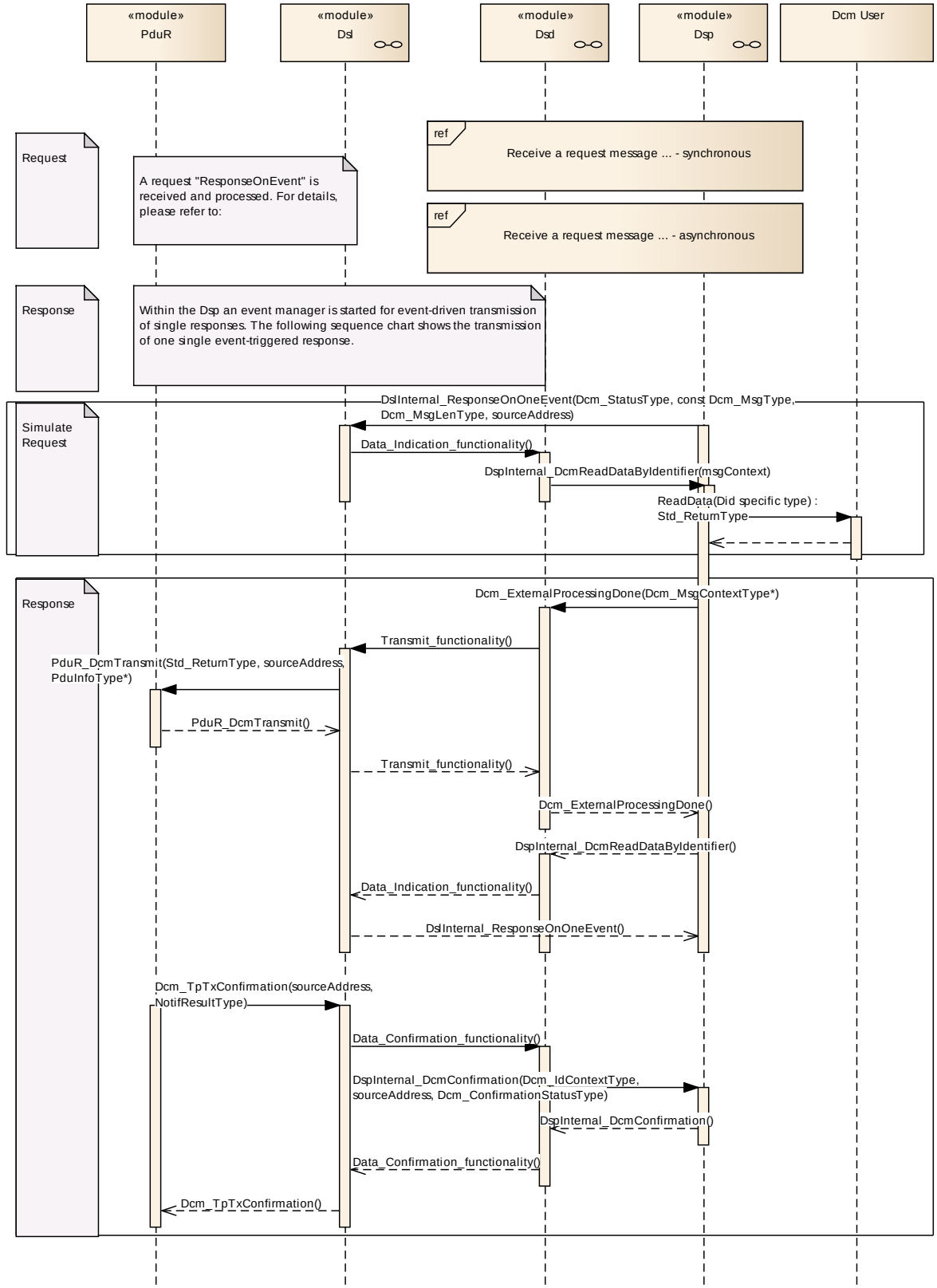


The DSP submodule requests sampling and transmission of Periodic Identifier data, when an event to Periodic Identifier occurs (i. e. a given time period is over). The DSP submodule initiates the sending of one periodic identifier calling the function ResponseOnOneDataByPeriodicId() provided by the DSL submodule.

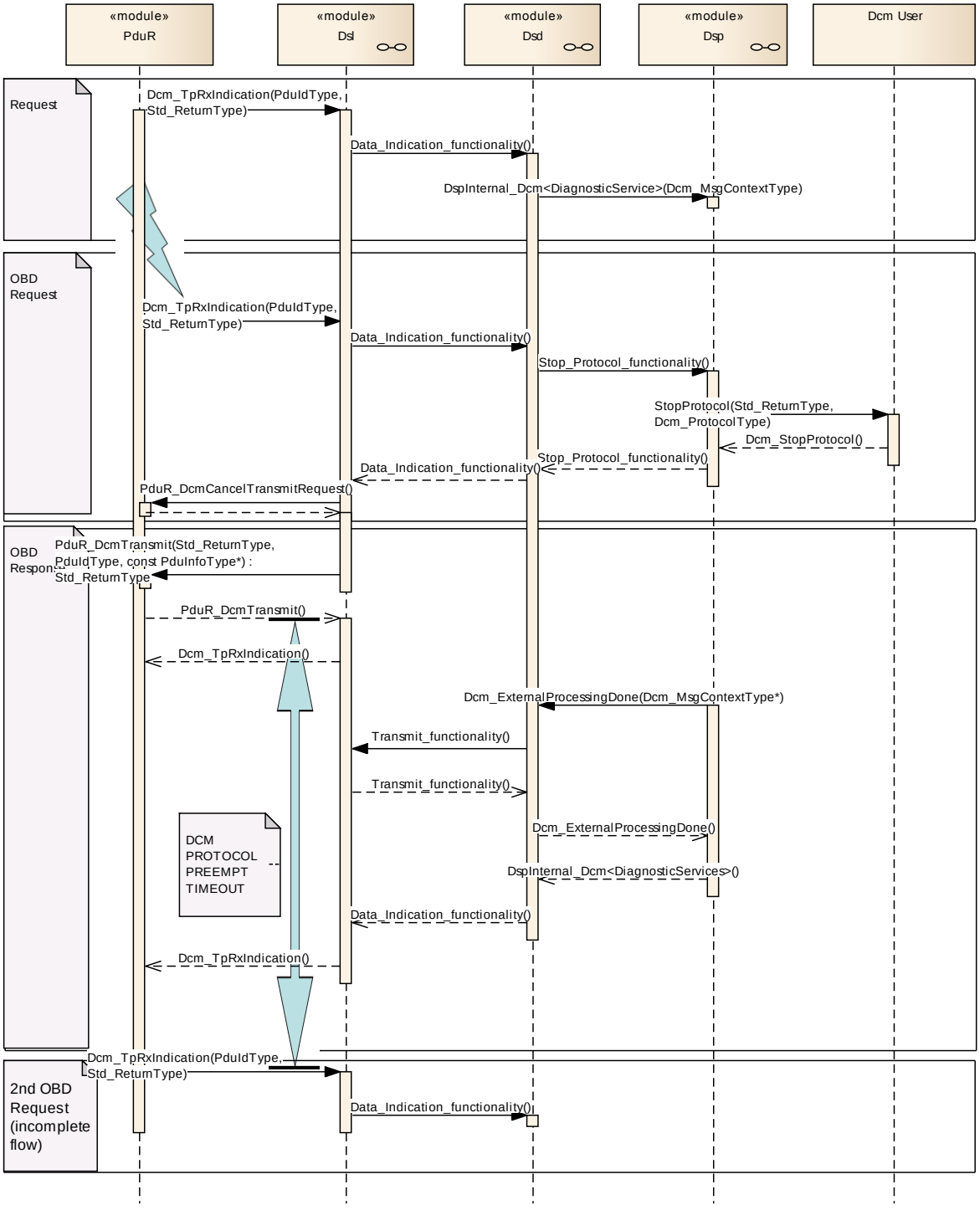
Within this function the DSL submodule simulates a “ReadDataByIdentifier” request for the given PeriodicId. The High byte of the DataIdentifier shall be set to 0xF2 as specified in [11]) and the low byte is set to value of the PeriodicId.
The ReadData interfaces of the corresponding Datas of the DID are called to get the DID value.
The DCM module is not able to receive for the same periodic identifier another event request from the DSP submodule, unless the confirmation of the current transmission is received.

9.2.5 Process single event-triggered response of ResponseOnEvent

This sequence diagram shows an example for ResponseOnEvent.
ResponseOnEvent is setup and started for onDTCStatusChange. Event changes are reported to the Dcm which will trigger a serviceToRespondTo.


«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
Dcm User
Request
A request "ResponseOnEvent" is received and processed. For details, please refer to:
ref
Receive a request message ... - synchronous
ref
Receive a request message ... - asynchronous
Response
Within the Dsp an event manager is started for event-driven transmission of single responses. The following sequence chart shows the transmission of one single event-triggered response.
Simulate Request
DslInternal_ResponseOnOneEvent(Dcm_StatusType, const Dcm_MsgType, Dcm_MsgLenType, sourceAddress)
Data_Indication_functionality()
DspInternal_DcmReadDataByIdentifier(msgContext)
ReadData(Did specific type) : Std_ReturnType
Response
Dcm_ExternalProcessingDone(Dcm_MsgContextType*)
Transmit_functionality()
PduR_DcmTransmit(Std_ReturnType, sourceAddress, PduInfoType*)
PduR_DcmTransmit()
Transmit_functionality()
Dcm_ExternalProcessingDone()
DspInternal_DcmReadDataByIdentifier()
Data_Indication_functionality()
DslInternal_ResponseOnOneEvent()
Dcm_TpTxConfirmation(sourceAddress, NotifResultType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, sourceAddress, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TpTxConfirmation()

9.2.6 Process concurrent requests

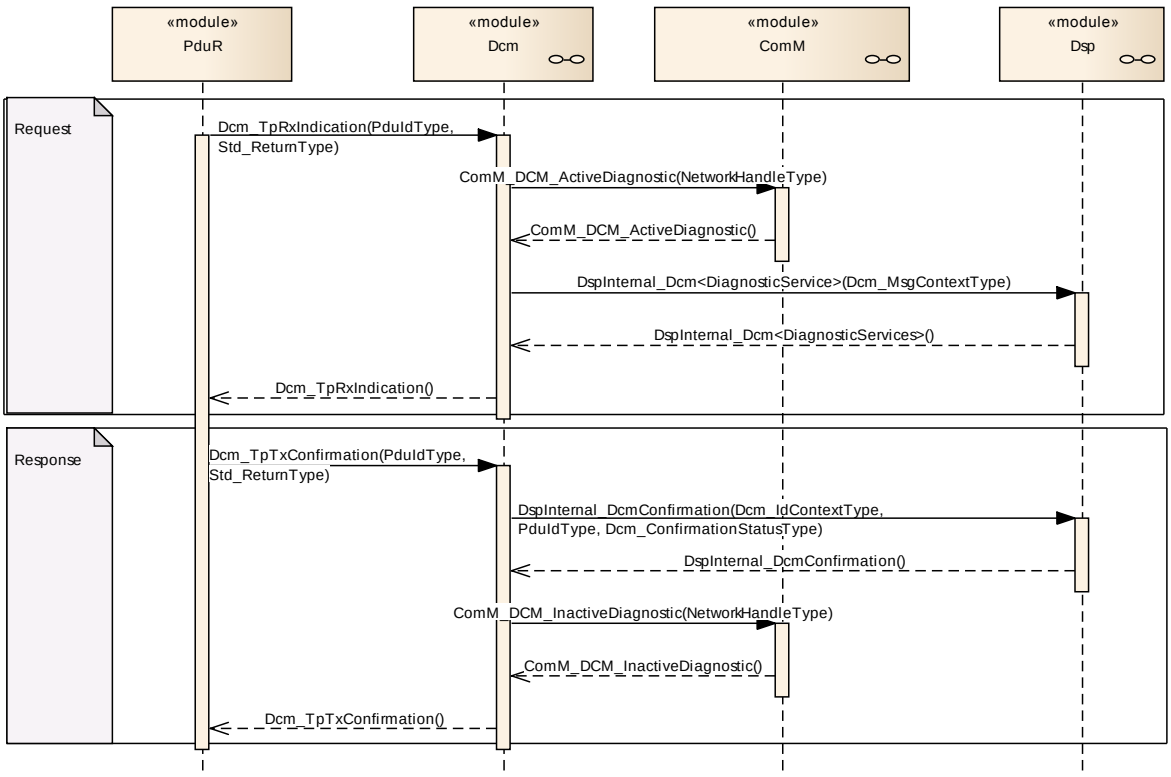


On reception of OBD request in parallel to processing of a normal diagnostic request (e.g enhanced diagnostic protocol, customer diagnostic protocol), running diagnostic request will be preempted. This is due to the configured higher priority of OBD protocol (see configuration parameter ***DcmDslProtocolPriority***).
The following is processed on reception of 1st OBD request:

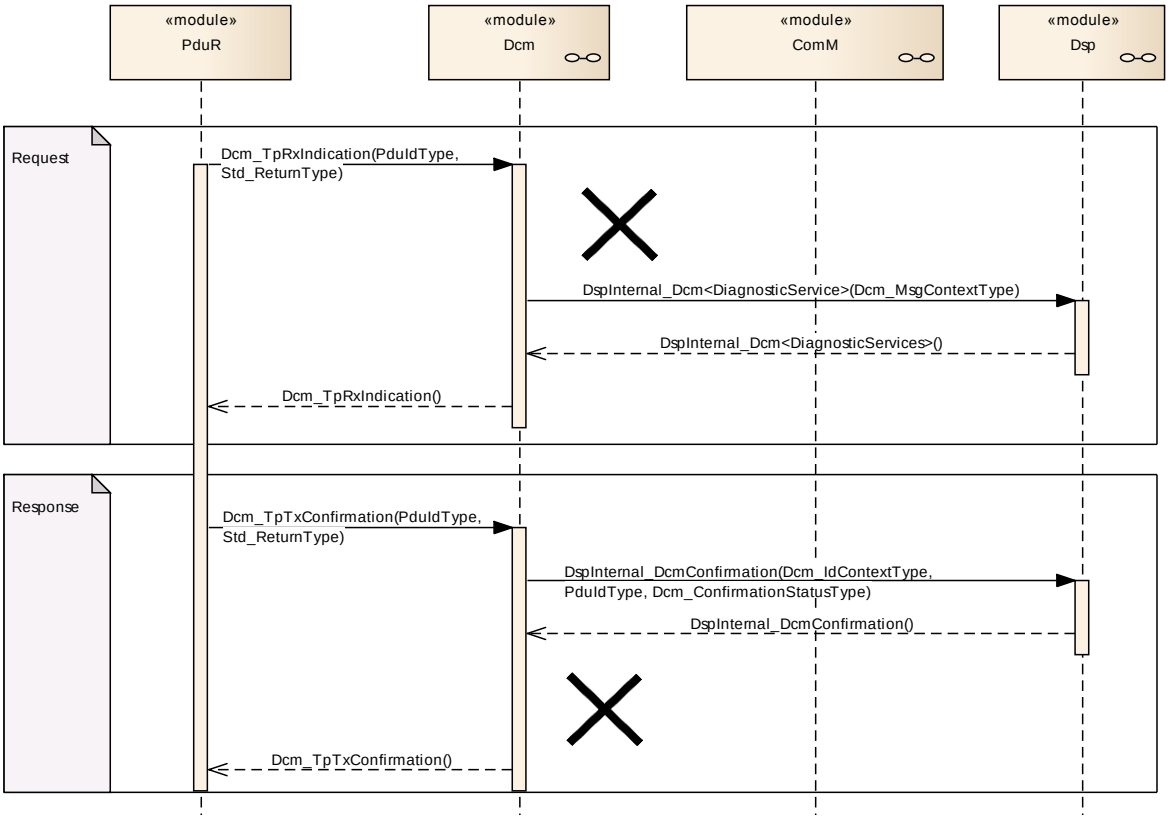
- The Application is informed of the protocol stop (done with `Xxx_StopProtocol()`) and resets to a stable state (e.g. switch of digital Ios,..).
- Lower Layer is requested to cancel ongoing transmission on the same N-PDU (done with `PduR_DcmCancelTransmitRequest()`).
- If the DCM is not able to switch fast enough from non OBD to OBD protocol, the DSL submodule responses with a negative response “BusyRepeatRequest” (NRC 0x21) to OBD tester.
It is in the responsibility of the system designer to ensure that the legislative timings are satisfied.
- 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 `Dcm_ExternalProcessingDone()`) or no timeout occurs, the DSL submodule responses with negative response “BusyRepeatRequest”.
With receiving `Dcm_ExternalProcessingDone()`, 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 `Dcm_ExternalProcessingDone()`, the DSL submodule runs into timeout and switches directly to further processing of preempting protocol.

9.2.7 Interface to ComManager

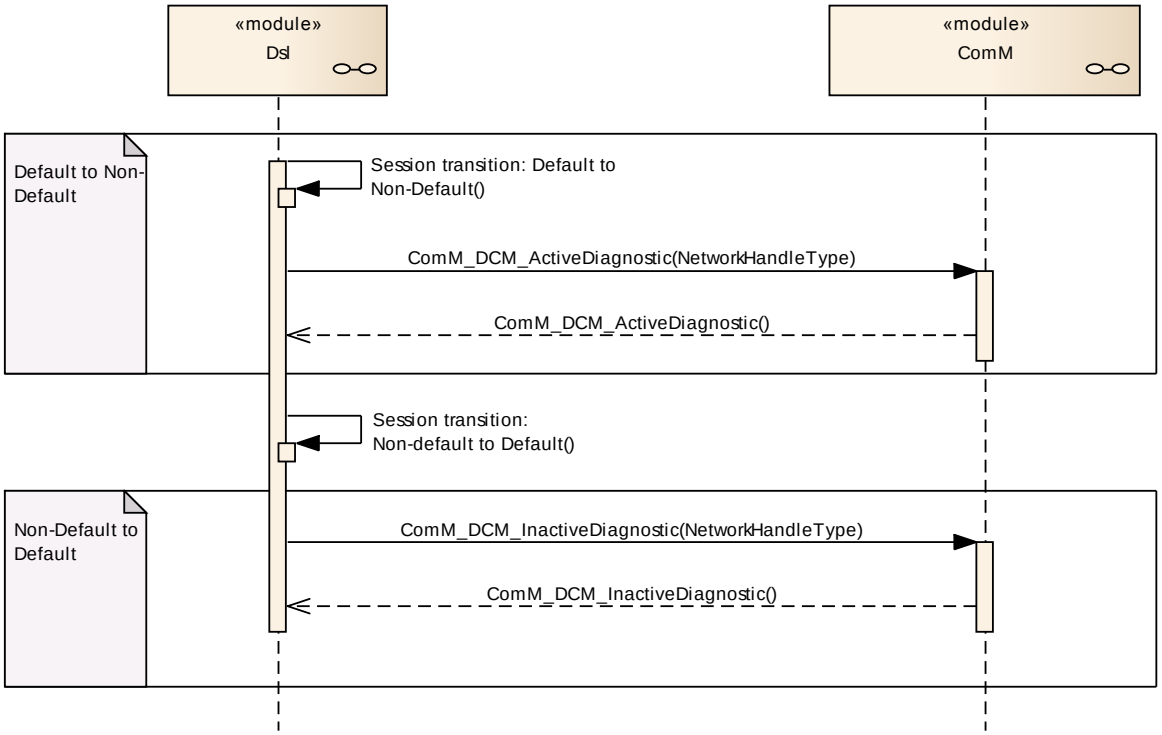
9.2.7.1 Handling in Default Session



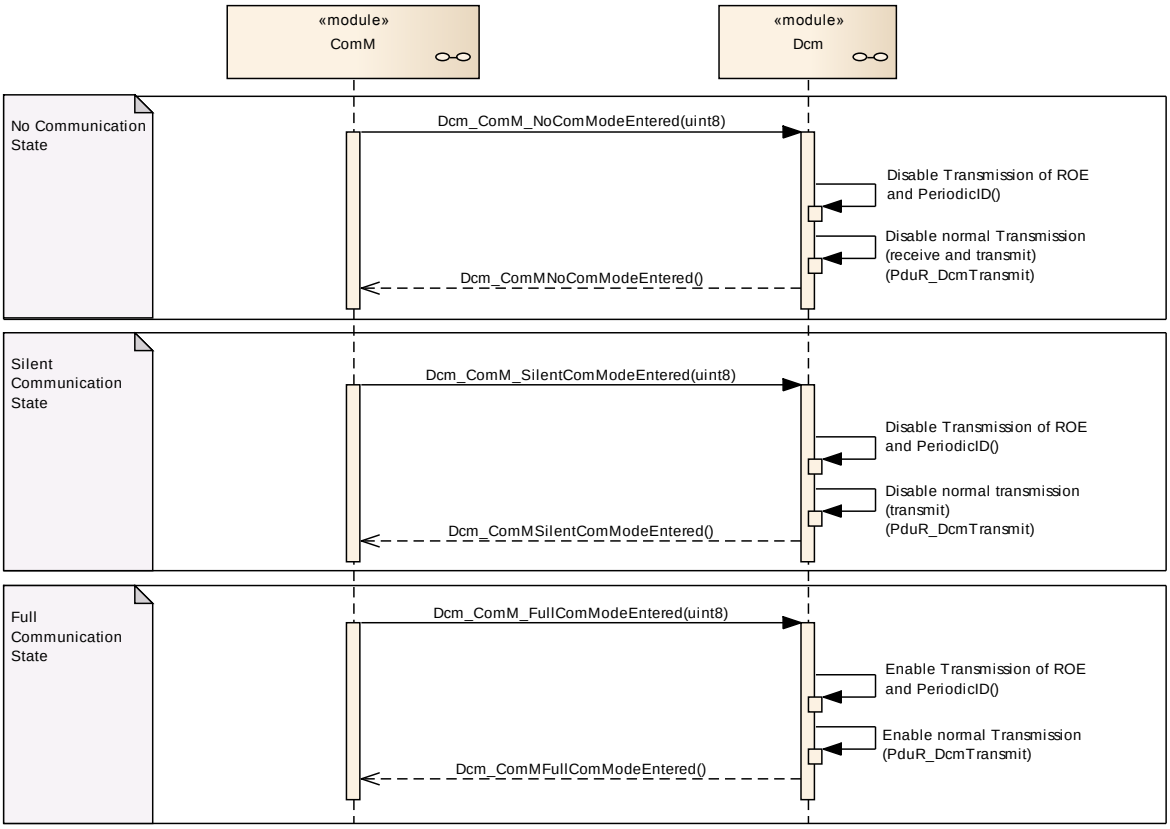
9.2.7.2 Handling in Non-Default Session



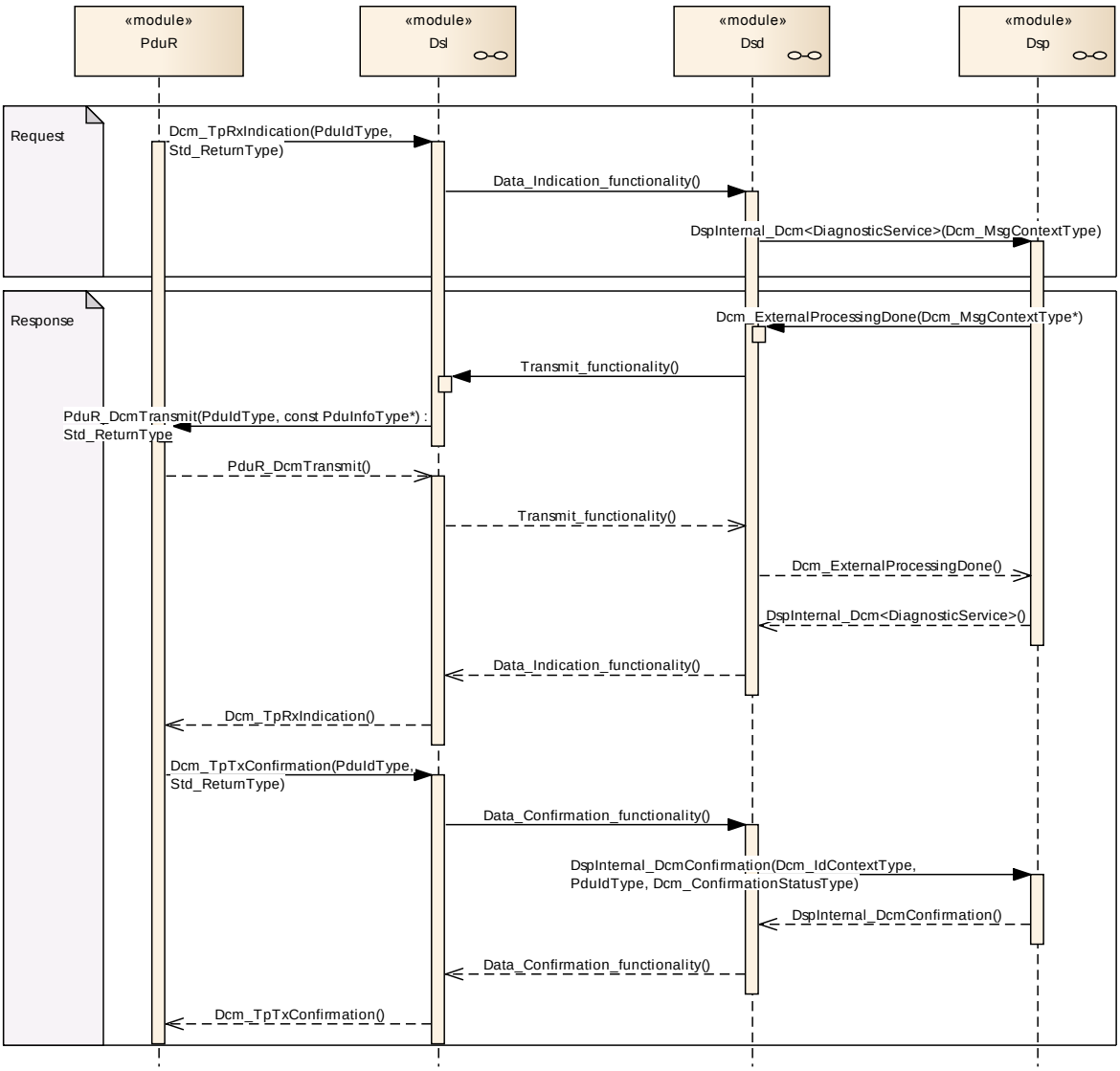
9.2.7.3 Session transitions



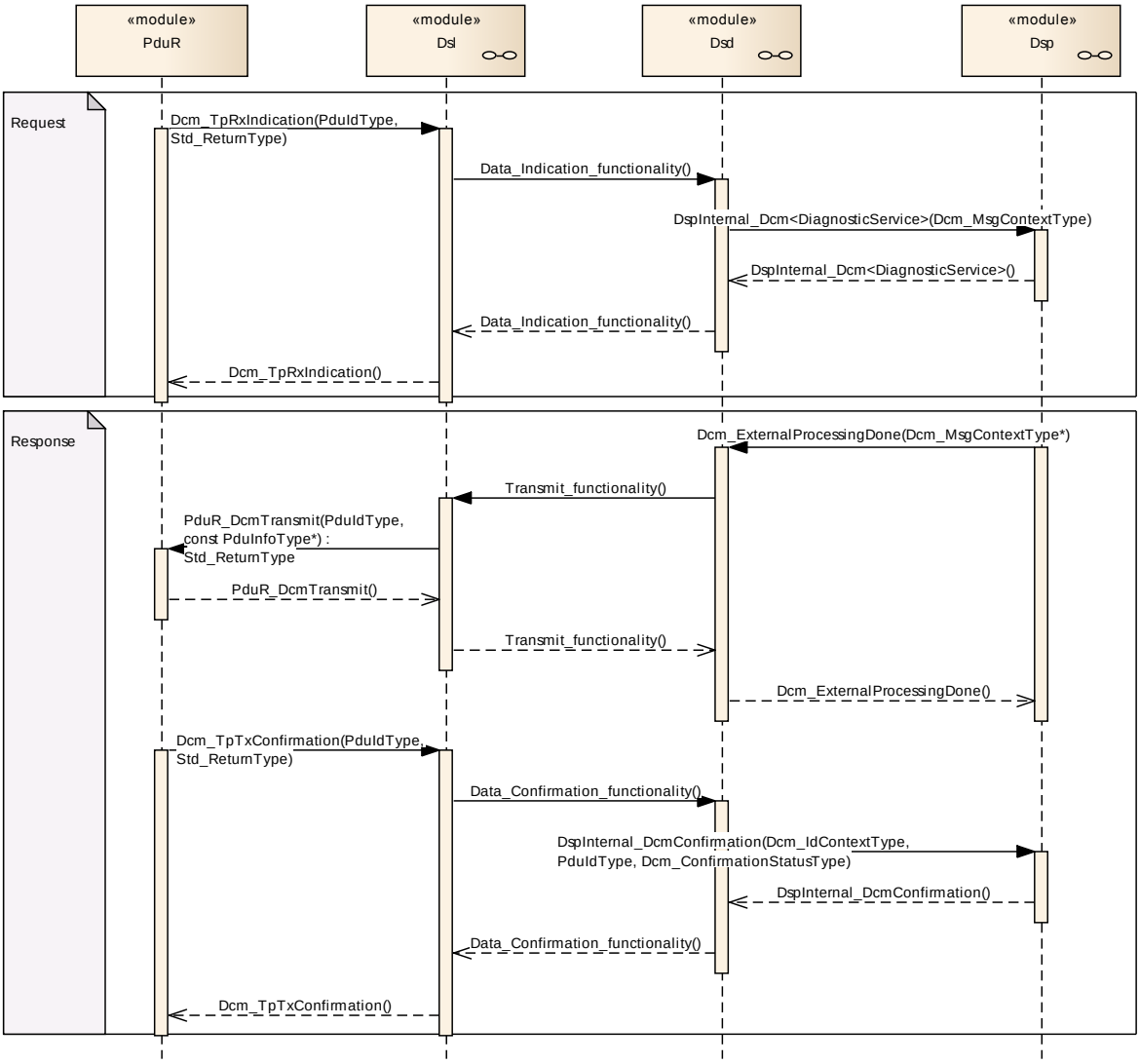
9.2.7.4 Communication States



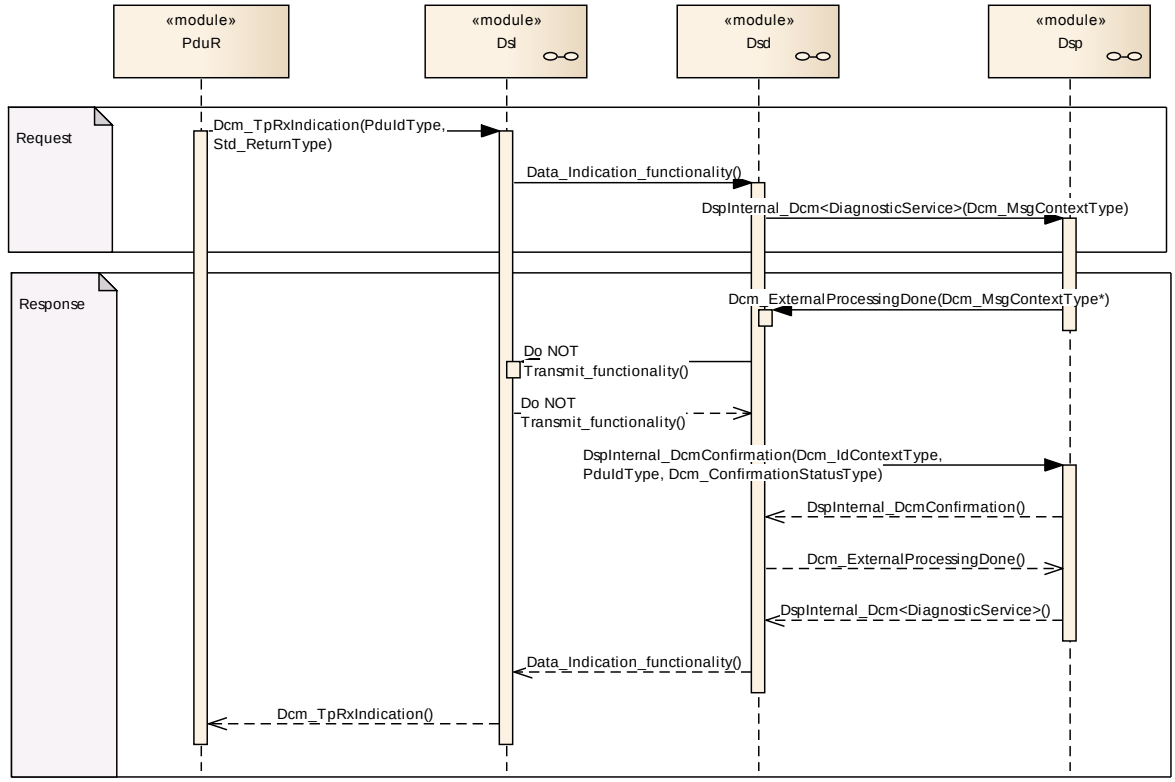
DSD (Diagnostic Service Dispatcher)
Receive a request message and transmit a positive response message – synchronous transmission



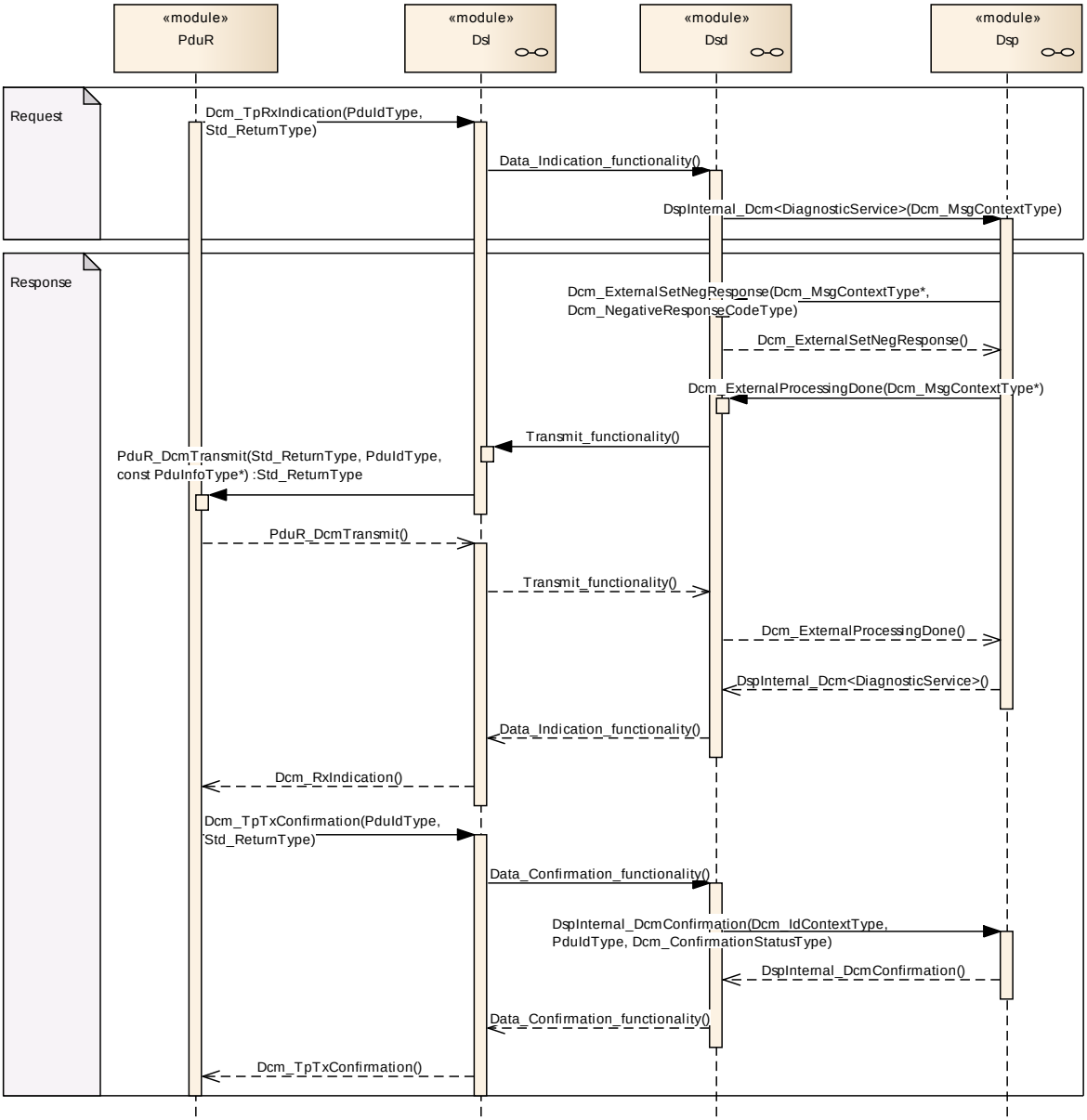
Receive a request message and transmit a positive response message – asynchronous transmission



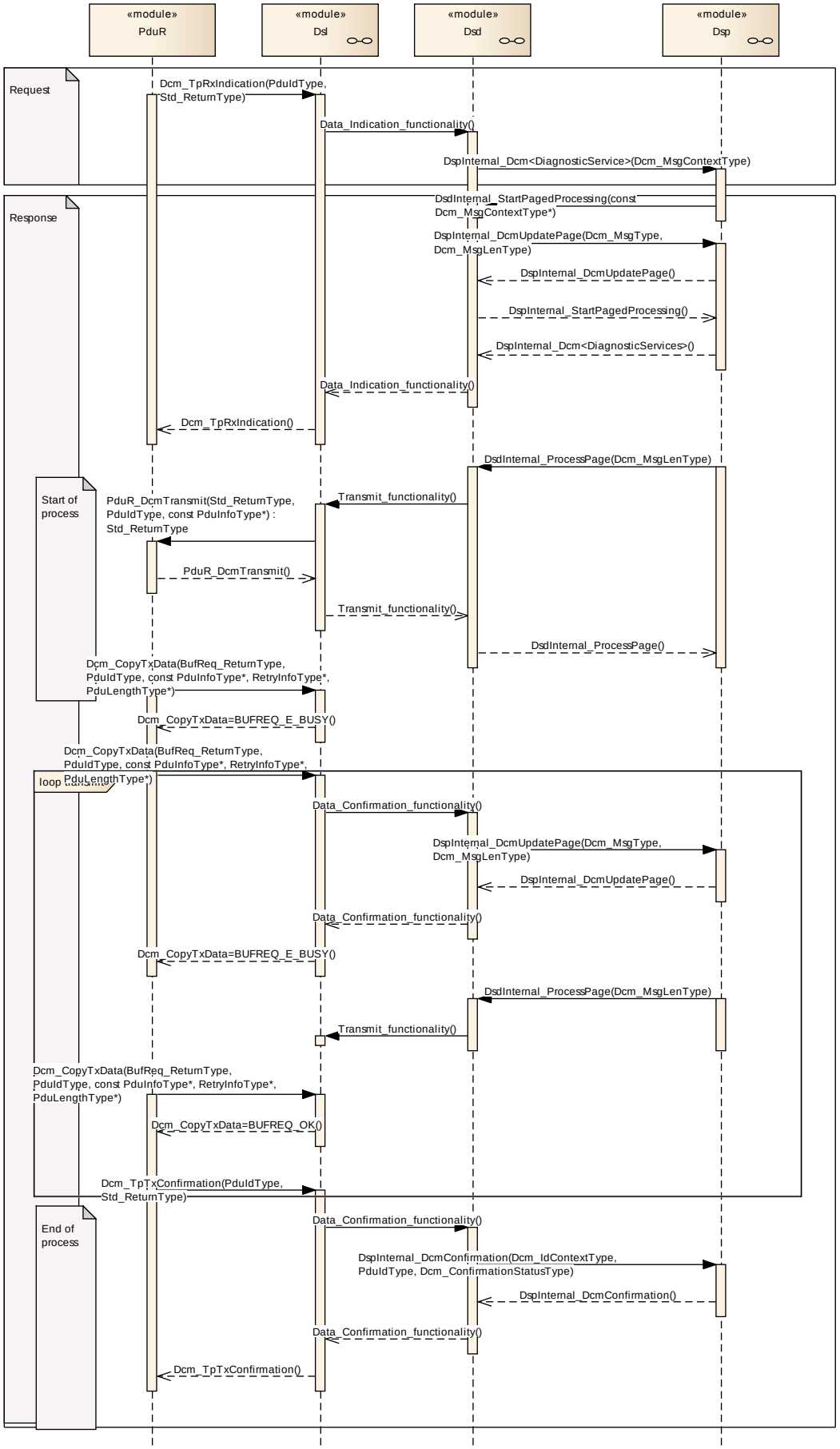
Receive a request message and suppress a positive response


«module»
PduR
«module»
DsI
«module»
Dsd
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType,
Std_ReturnType)
Data_Indication_functionality()
DspInternal_Dcm<DiagnosticService>(Dcm_MsgContextType)
Response
Dcm_ExternalProcessingDone(Dcm_MsgContextType*)
Do NOT
Transmit_functionality()
Do NOT
Transmit_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType,
PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Dcm_ExternalProcessingDone()
DspInternal_Dcm<DiagnosticService>()
Data_Indication_functionality()
Dcm_TpRxIndication()

9.2.8 Receive request message and transmit negative response message



9.2.9 Process Service Request with paged-buffer


«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
DspInternal_Dcm<DiagnosticService>(Dcm_MsgContextType)
Response
DsdInternal_StartPagedProcessing(const Dcm_MsgContextType*)
DspInternal_DcmUpdatePage(Dcm_MsgType, Dcm_MsgLenType)
DspInternal_DcmUpdatePage()
DspInternal_StartPagedProcessing()
DspInternal_Dcm<DiagnosticServices>()
Data_Indication_functionality()
Dcm_TpRxIndication()
DsdInternal_ProcessPage(Dcm_MsgLenType)
Start of process
PduR_DcmTransmit(Std_ReturnType, PduIdType, const PduInfoType*) : Std_ReturnType
Transmit_functionality()
PduR_DcmTransmit()
Transmit_functionality()
DsdInternal_ProcessPage()
Dcm_CopyTxData(BufReq_ReturnType, PduIdType, const PduInfoType*, RetryInfoType*, PduLengthType*)
Dcm_CopyTxData=BUFREQ_E_BUSY()
Dcm_CopyTxData(BufReq_ReturnType, PduIdType, const PduInfoType*, RetryInfoType*, PduLengthType*)
loop
Data_Confirmation_functionality()
DspInternal_DcmUpdatePage(Dcm_MsgType, Dcm_MsgLenType)
DspInternal_DcmUpdatePage()
Data_Confirmation_functionality()
Dcm_CopyTxData=BUFREQ_E_BUSY()
DsdInternal_ProcessPage(Dcm_MsgLenType)
Transmit_functionality()
Dcm_CopyTxData(BufReq_ReturnType, PduIdType, const PduInfoType*, RetryInfoType*, PduLengthType*)
Dcm_CopyTxData=BUFREQ_OK()
Dcm_TpTxConfirmation(PduIdType, Std_ReturnType)
End of process
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TpTxConfirmation()

The following flow is processed in case no error occurs on the Application side:

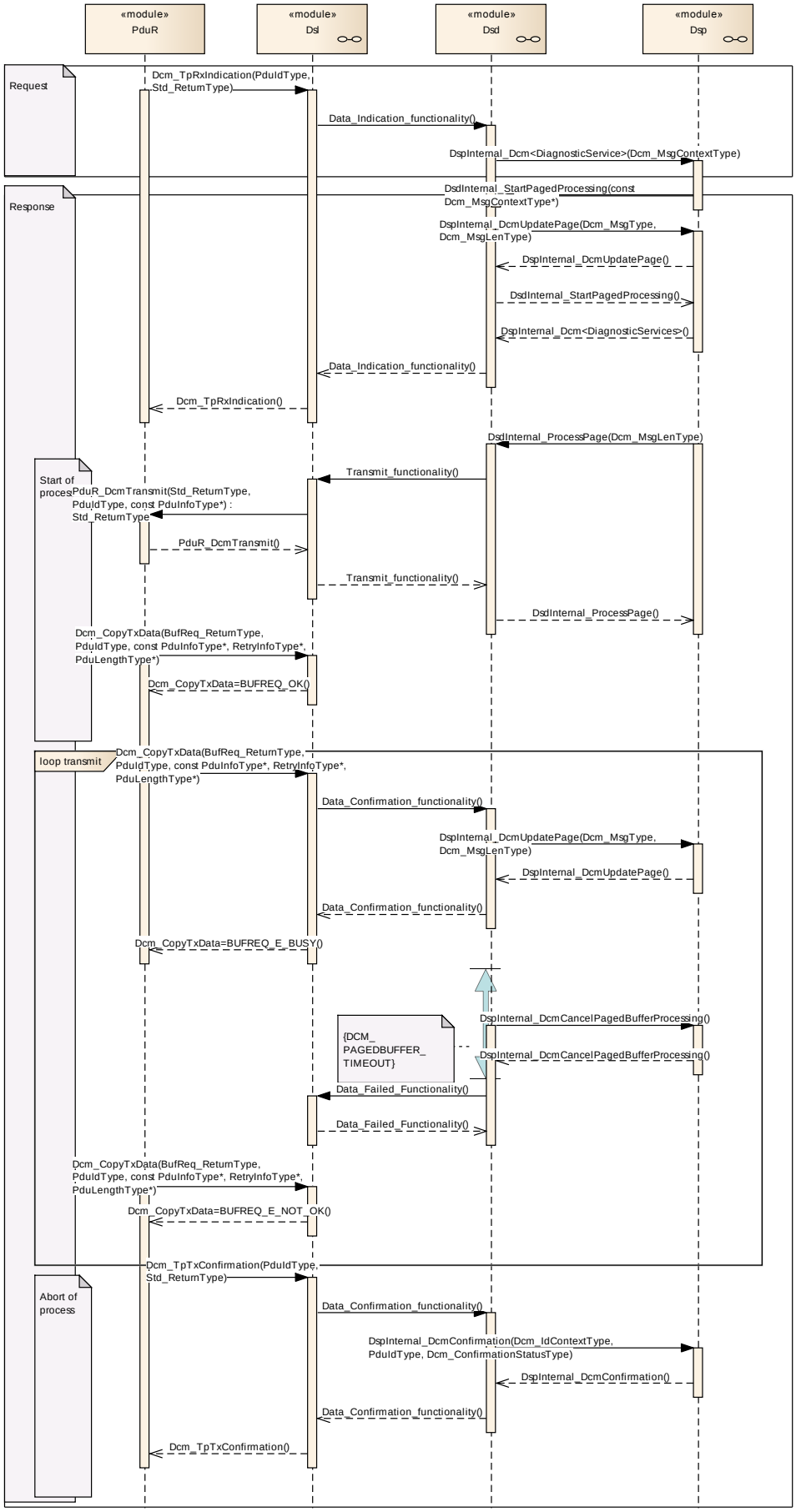
Start of process:
4) `DsdInternal_StartPagedProcessing()`: With this API, the DSP submodule gives the complete response length to the DCM module and starts paged-buffer handling. This API starts no transmission!
5) UpdatePage(): The DCM module requests data to be transmitted.
6) `DsdInternal_ProcessPage()`: 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_CopyTxData()`: The buffer is filled and the DCM module shall return "BUFREQ_OK"(10).

Start of the loop:
11) `Dcm_CopyTxData()`: 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_CopyTxData()` 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_TpTxConfirmation()` 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_CopyTxData()` request, item 17).


«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
DspInternal_Dcm<DiagnosticService>(Dcm_MsgContextType)
Response
DsdInternal_StartPagedProcessing(const Dcm_MsgContextType*)
DspInternal_DcmUpdatePage(Dcm_MsgType, Dcm_MsgLenType)
DspInternal_DcmUpdatePage()
DsdInternal_StartPagedProcessing()
DspInternal_Dcm<DiagnosticServices>()
Data_Indication_functionality()
Dcm_TpRxIndication()
DsdInternal_ProcessPage(Dcm_MsgLenType)
Transmit_functionality()
Start of process
PduR_DcmTransmit(Std_ReturnType, PduIdType, const PduInfoType*) : Std_ReturnType
PduR_DcmTransmit()
Transmit_functionality()
DsdInternal_ProcessPage()
Dcm_CopyTxData(BufReq_ReturnType, PduIdType, const PduInfoType*, RetryInfoType*, PduLengthType*)
Dcm_CopyTxData=BUFREQ_OK()
loop transmit
Dcm_CopyTxData(BufReq_ReturnType, PduIdType, const PduInfoType*, RetryInfoType*, PduLengthType*)
Data_Confirmation_functionality()
DspInternal_DcmUpdatePage(Dcm_MsgType, Dcm_MsgLenType)
DspInternal_DcmUpdatePage()
Data_Confirmation_functionality()
Dcm_CopyTxData=BUFREQ_E_BUSY()
DspInternal_DcmCancelPagedBufferProcessing()
{DCM_ PAGEDBUFFER_ TIMEOUT}
DspInternal_DcmCancelPagedBufferProcessing()
Data_Failed_Functionality()
Data_Failed_Functionality()
Dcm_CopyTxData(BufReq_ReturnType, PduIdType, const PduInfoType*, RetryInfoType*, PduLengthType*)
Dcm_CopyTxData=BUFREQ_E_NOT_OK()
Dcm_TpTxConfirmation(PduIdType, Std_ReturnType)
Abort of process
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TpTxConfirmation()

9.2.10 Process copy data in reception

Please refer to Figure 9 “CanTp I-PDU reception” in PduR SWS [2]

9.2.11 Process copy data in transmission

Please refer to Figure 14 “CanTp I-PDU transmission” in PduR SWS [2]

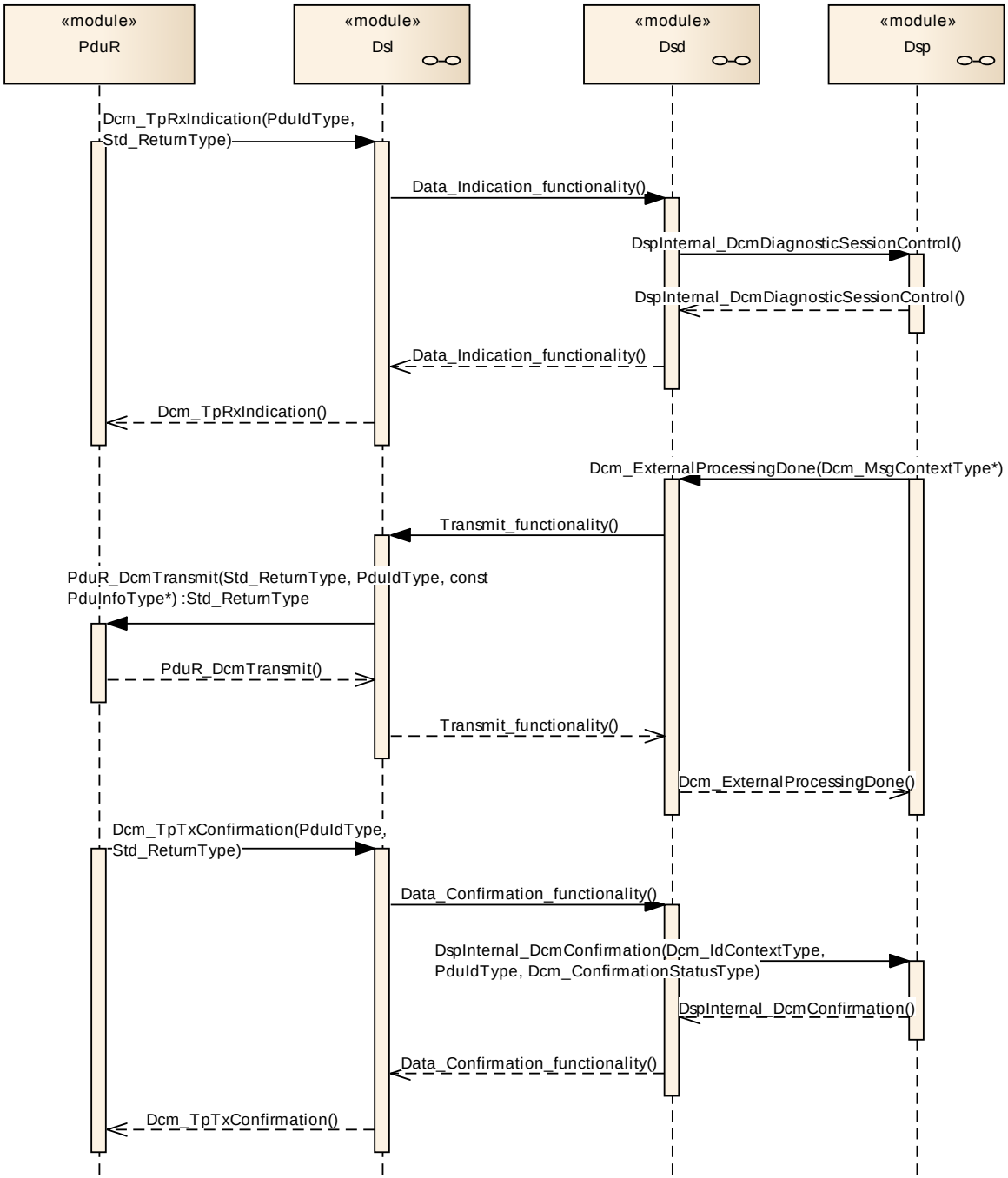
9.3 DSP (Diagnostic Service Processing)

9.3.1 Interface DSP – DEM (service 0x19, 0x14, 0x85)

Please refer to Section 9. In [7].

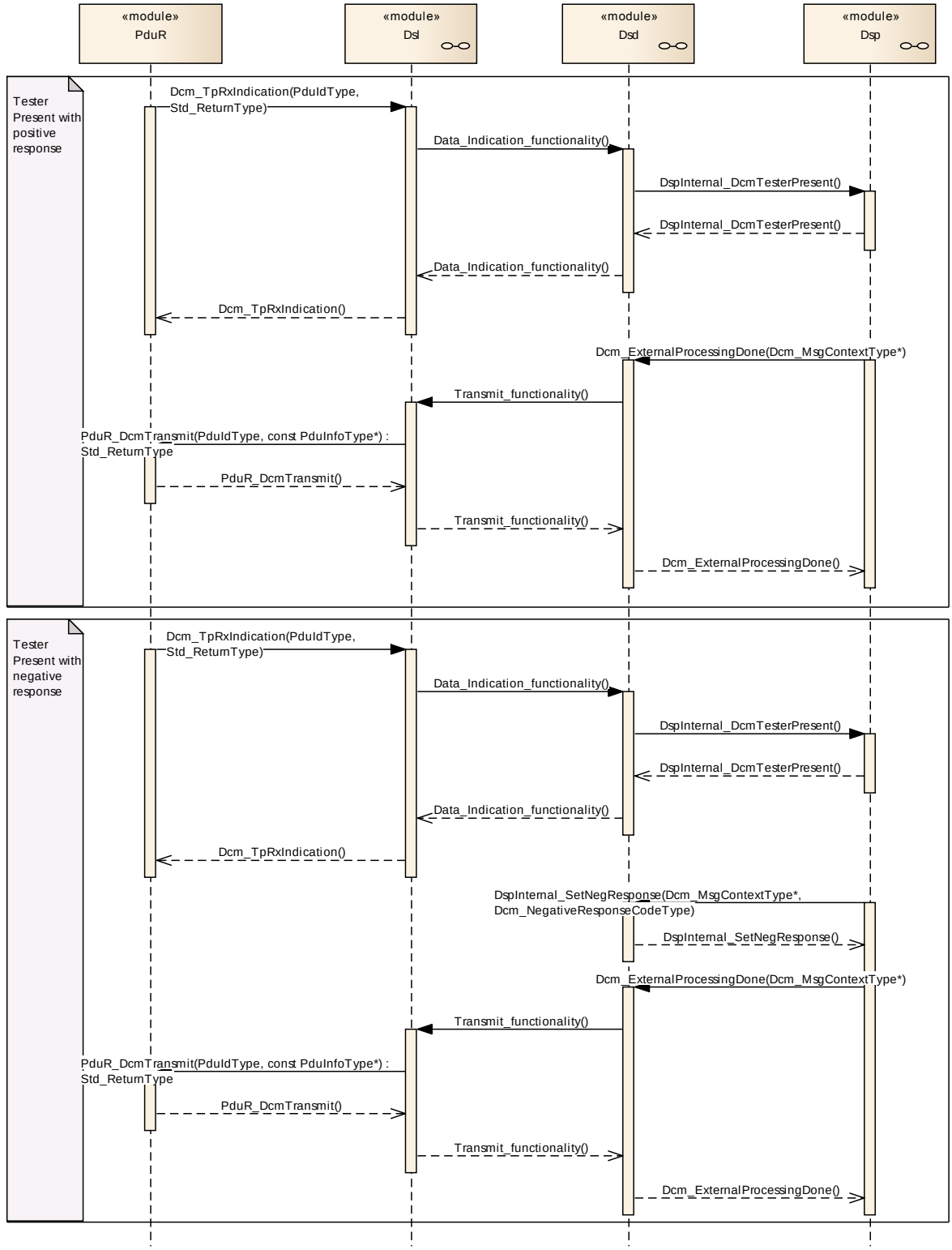
9.3.2 Interface special services

9.3.2.1 Process Diagnostic Session Control



Above sequence diagram shows processing of Diagnostic Session Control request from a tester.
Note that the new diagnostic session and timing parameters only apply after the transmission confirmation of the server positive response

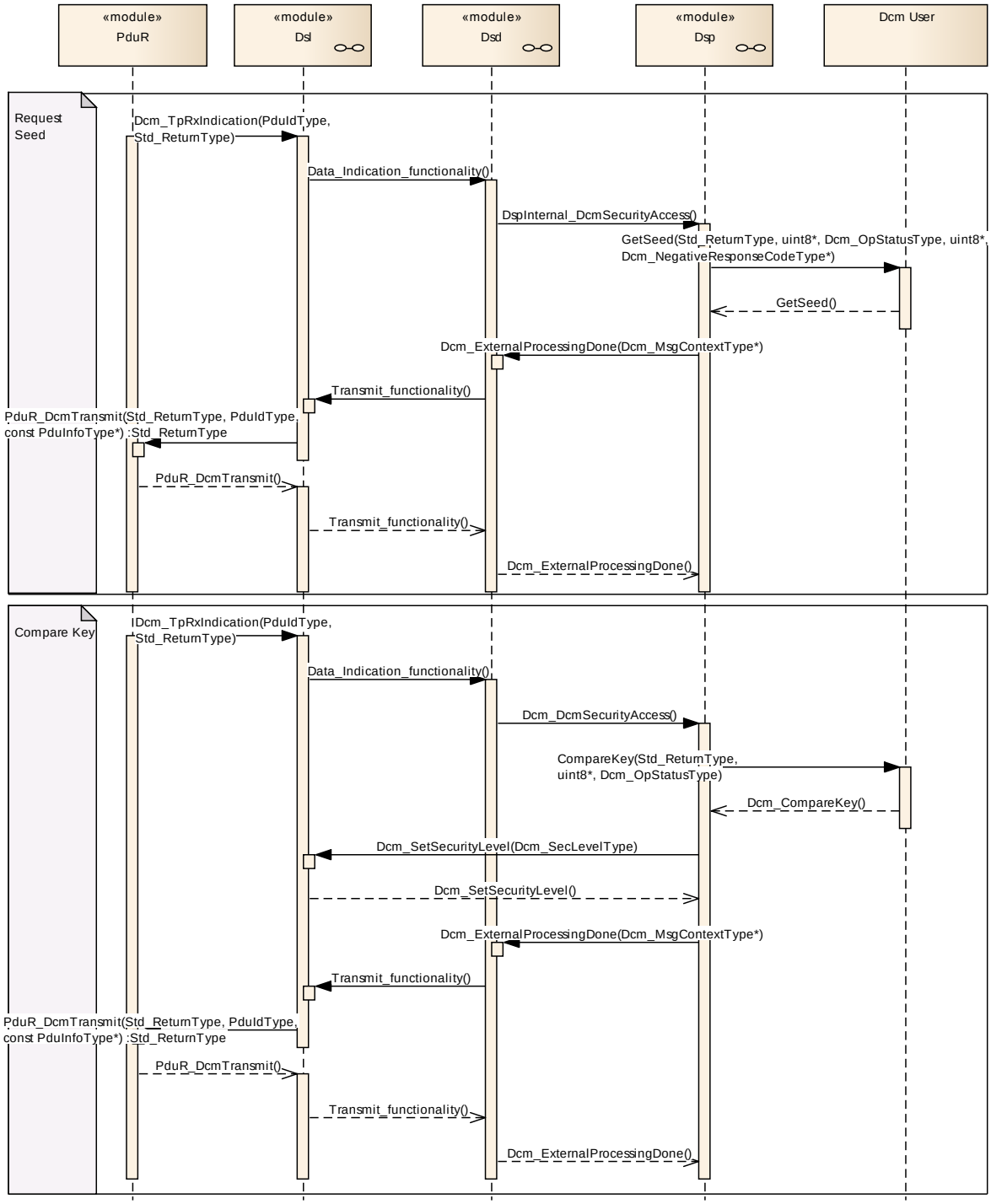
9.3.2.2 Process Tester Present



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.

9.3.2.3 Process Security Access

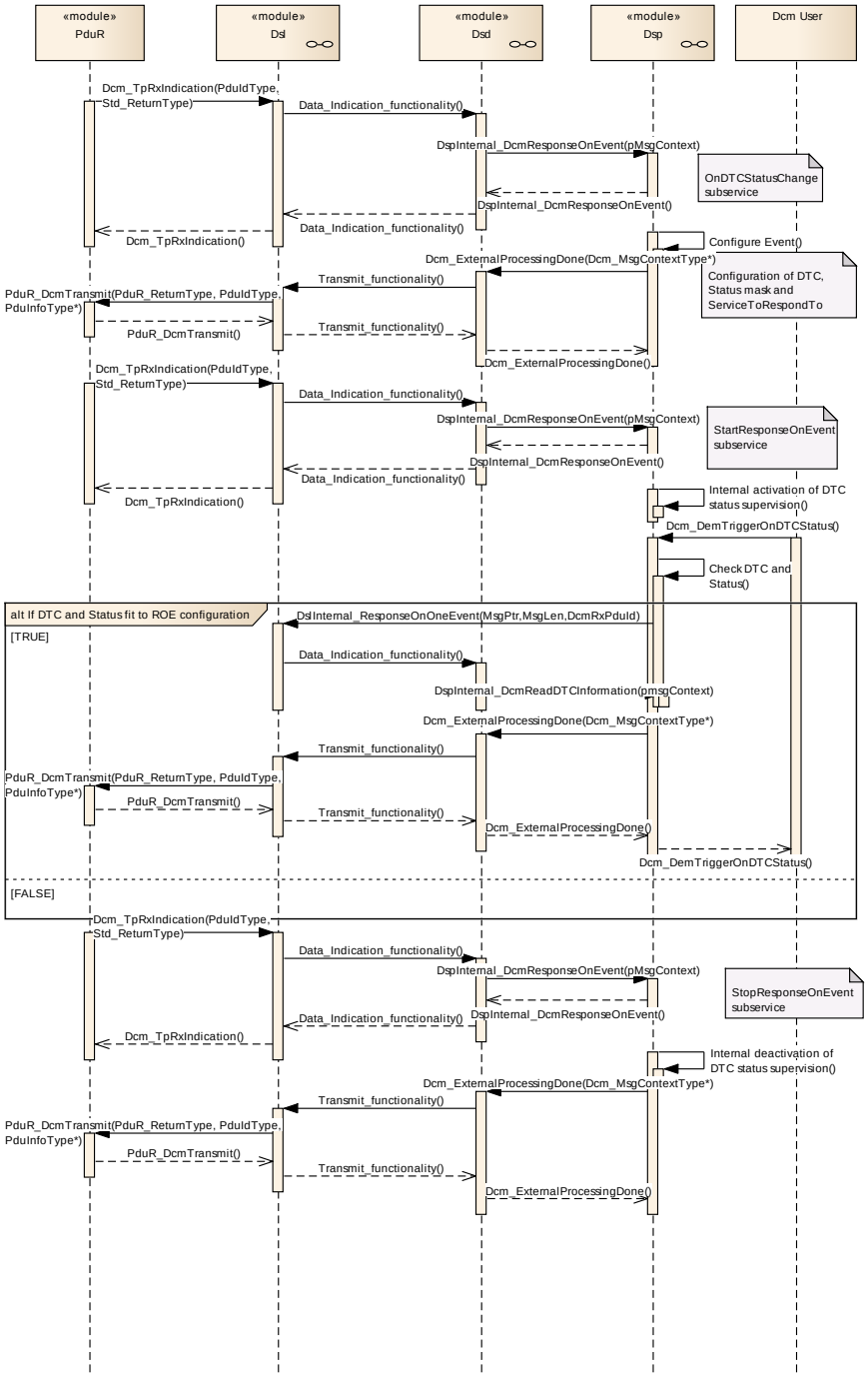


To get the security access, the DSD submodule has to call the DSP submodule to get the seed value from the application. If no error is detected, the seed value is sent in the positive response.

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.

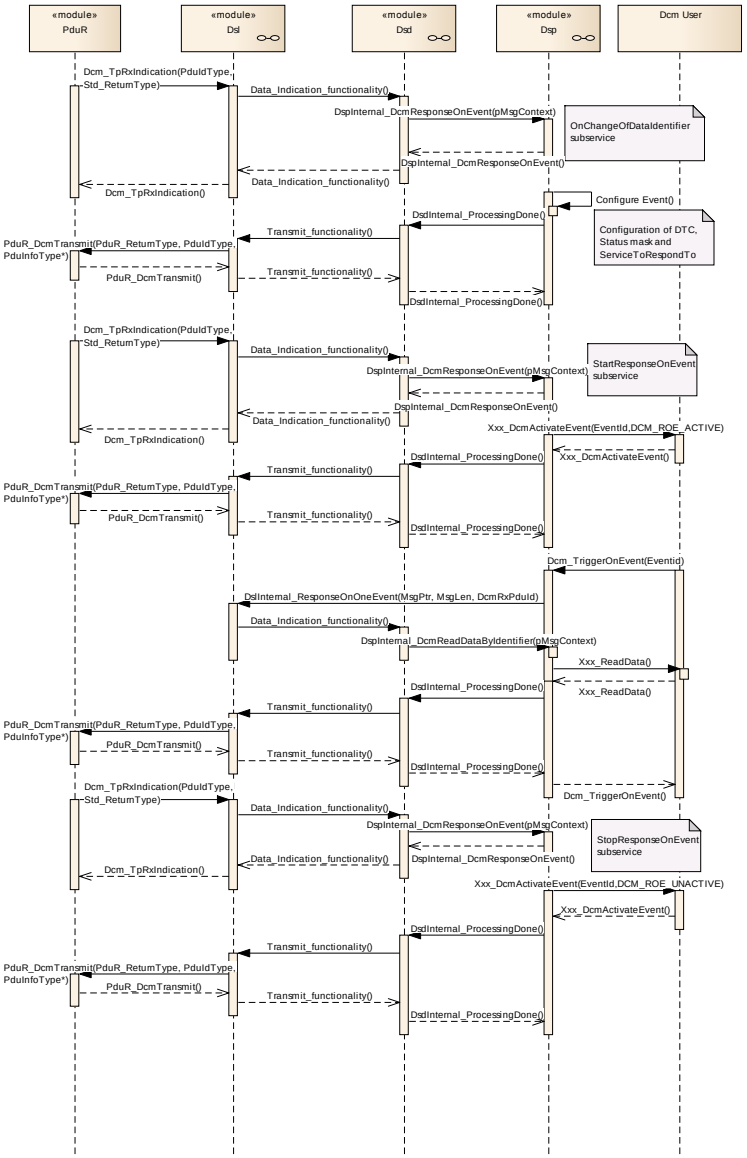
9.3.2.4 Process ResponseOnEvent OnDtcChange



Above sequence

diagram shows processing of ResponseOnEvent service for sub-service OnDtcChange.
After configuration and activation of the event by the service ResponseOnEvent, the DCM checks the status of the configured DTC on every call to interface Dcm_DemTriggerOnDTCStatus() in order to identify if the event shall be trigger. This interface is called by DEM for any DTC status change and independing of the activation/unactivation of ResponseOnEvent.

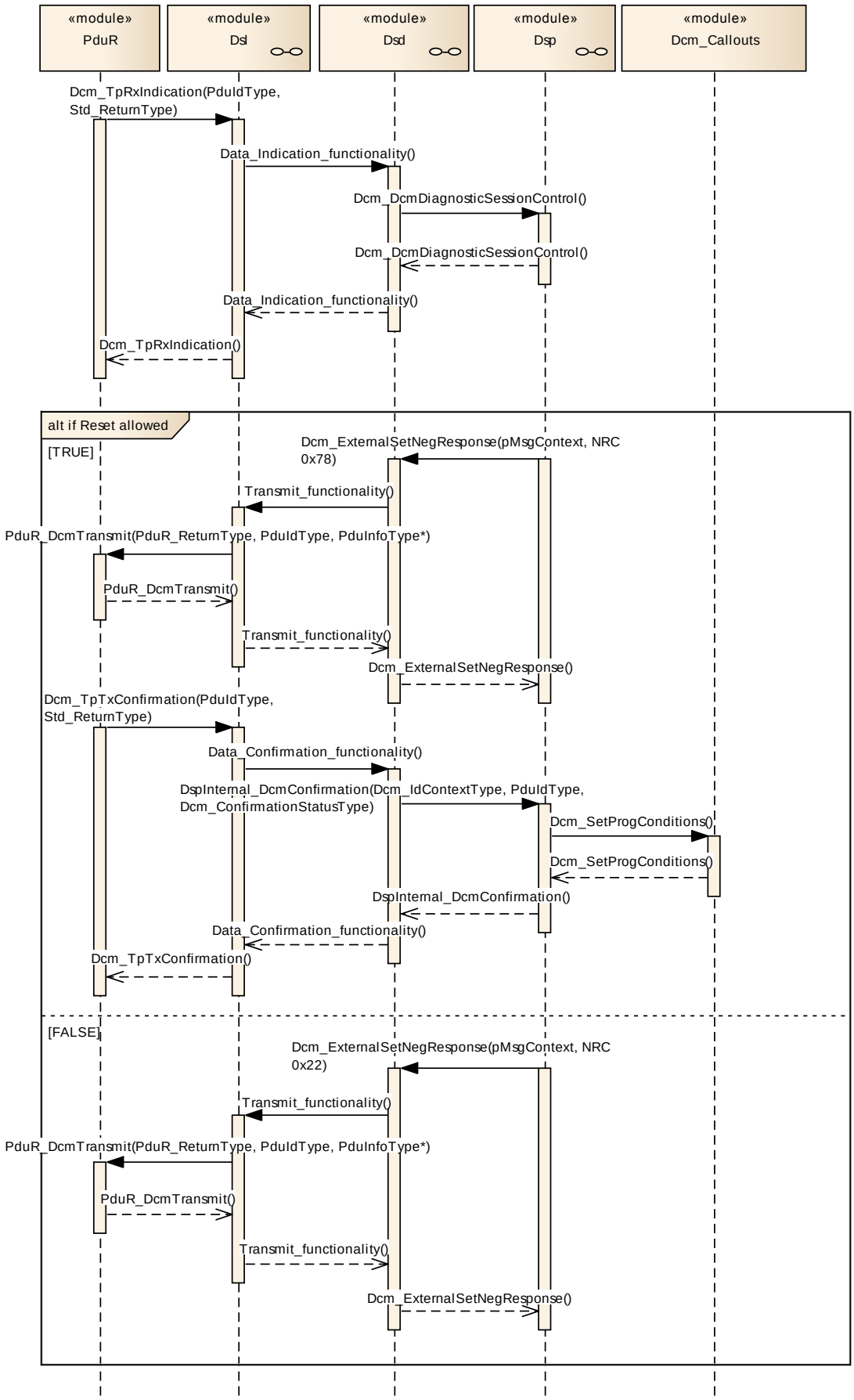
9.3.2.5 Process ResponseOnEvent OnChangeOfDataIdentifier



Above sequence diagram

shows processing of ResponseOnEvent service for sub-service OnChangeOfDataIdentifier in the case the event is externally managed (The event can be internally managed, but is not describe in this diagram).
After configuration and etxernal activation of the event by the service ResponseOnEvent, the DCM wait to be trigger by the external module managing this DID.

9.3.2.6 Process Jump to Bootloader


«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
«module»
Dcm_Callouts
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
Dcm_DcmDiagnosticSessionControl()
Dcm_DcmDiagnosticSessionControl()
Data_Indication_functionality()
Dcm_TpRxIndication()
alt if Reset allowed
[TRUE]
Dcm_ExternalSetNegResponse(pMsgContext, NRC 0x78)
Transmit_functionality()
PduR_DcmTransmit(PduR_ReturnType, PduIdType, PduInfoType*)
PduR_DcmTransmit()
Transmit_functionality()
Dcm_ExternalSetNegResponse()
Dcm_TpTxConfirmation(PduIdType, Std_ReturnType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
Dcm_SetProgConditions()
Dcm_SetProgConditions()
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TpTxConfirmation()
[FALSE]
Dcm_ExternalSetNegResponse(pMsgContext, NRC 0x22)
Transmit_functionality()
PduR_DcmTransmit(PduR_ReturnType, PduIdType, PduInfoType*)
PduR_DcmTransmit()
Transmit_functionality()
Dcm_ExternalSetNegResponse()

Above sequence diagram shows processing of a jump to bootloader on reception of DiagnosticSessionControl.
On reception of DiagnosticSessionControl, the DCM checks if the requested session is configured to trigger a jump to bootloader. In positive case, the DCM start the jump to bootloader process:

- Transmission of NRC 0x78 (ResponsePending)
- On confirmation of transmission of NRC 0x78, the DCM calls the callout DcmSetProgConditions to store all information needed for the bootloader

10 Configuration specification

10.1 How to read this section

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS_BSWGeneral*

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.

10.2.2 DcmConfigSet

SWS Item	**ECUC_Dcm_00819 :**
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***
DcmDsd	1	These parameters apply to Diagnostic Service Dispatcher. All parameters for all service dispatchers are included in this one configuration container. There must always be one service dispatcher in a DCM.
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	0..1	These parameters apply to Diagnostic Service Processing. There will always be one set of these parameters per 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
DcmProcessingConditions	0..1	This container contains the configuration (DSP parameter) for mode arbitration functionality of the Dcm

10.2.3 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. There must always be one service dispatcher in a DCM.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00783 :**		
Name	DcmDsdRequestManufacturerNotificationEnabled		
Description	Allows to enable or disable the requested notification mechanism for the Manufacturer.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00868 :**		
Name	DcmDsdRequestSupplierNotificationEnabled		
Description	Allows to enable or disable the requested notification mechanism for the Supplier.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDsdServiceRequestManufacturerNotification	0..*	The name of this container is used to define the name of the R-Port through which the DCM accesses the interface ServiceRequestNotification. The R-Port is named ServiceRequestManufacturerNotification_<SWC> where <SWC> is the name of the container DcmDsdServiceRequestManufacturerNotification. The lowerMultiplicity is 0: If DcmDsdRequestManufacturerNotificationEnabled = false the Indication API is not available.
DcmDsdServiceRequestSupplierNotification	0..*	The name of this container is used to define the

		name of the R-Port through which the DCM accesses the interface ServiceRequestNotification. The R-Port is named ServiceRequestSupplierNotification_<SWC> where <SWC> is the name of the container DcmDsdServiceRequestSupplierNotification. The lowerMultiplicity is 0: If DcmDsdRequestSupplierNotification = false the Indication API is not available.
DcmDsdServiceTable	1..256	This container contains the configuration (DSD parameters) for Service Identifier Table. Note: It is allowed to add OBD services to a DcmDsdServiceTable related to a UDS Protocol. But it is not allowed to add UDS services to a DcmDsdServiceTable related to a OBD Protocol.

10.2.4 DcmDsdServiceTable

SWS Item	**ECUC_Dcm_00732 :**
Container Name	DcmDsdServiceTable
Description	This container contains the configuration (DSD parameters) for Service Identifier Table. Note: It is allowed to add OBD services to a DcmDsdServiceTable related to a UDS Protocol. But it is not allowed to add UDS services to a DcmDsdServiceTable related to a OBD Protocol.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00736 :**		
Name	DcmDsdSidTabId		
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	EcucIntegerParamDef		
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.

10.2.5 DcmDsdService

SWS Item	**ECUC_Dcm_00689 :**
Container Name	DcmDsdService

Description	This container contains the configuration (DSD parameters) for Service.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00777 :**		
Name	DcmDsdSidTabFnc		
Description	Callback function of the ECU Supplier specific component for the particular DcmDsdSidTabServiceId. This parameter is related to the interface <Module>_<DiagnosticService>. If it is not configured the service is handled Dcm-internally.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
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		
Description	Id of the Service identifier. The possible Service identifier are predefined in the ISO 14229-1 and ISO 15031-5 and in Table 4 and Table 5.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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	DcmDsdSidTabSubfuncAvail		
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	EcucBooleanParamDef		
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: DcmDsdSidTabServiceId		

SWS Item	**ECUC_Dcm_00918 :**
Name	DcmDsdSidTabModeRuleRef
Description	Reference to DcmDspModeRule

	Mode rule which controls the execution of the DcmDsdService. If there is no reference, no check of the mode rule shall be done.		
Multiplicity	0..1		
Type	Reference to [DcmModeRule]		
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_00733 :**		
Name	DcmDsdSidTabSecurityLevelRef		
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. If there is no reference, no check of security level shall be done.		
Multiplicity	0..*		
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		
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. If there is no reference, no check of session level shall be done.		
Multiplicity	0..*		
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		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDsdSubService	0..*	This container contains the configuration (DSD parameters) for SubServices. This configuration is available only for services having Subfunction: this container exists only if parameter DcmDsdSidTabSubfuncAvail, of this service, is set to TRUE and the parameter DcmDsdSidTabFnc is not existing.

Note : The DCM internal interaction with the DSP is implementation specific and therefore not explicitly configured

10.2.6 DcmDsdSubService

SWS Item	**ECUC_Dcm_00802 :**		
Container Name	DcmDsdSubService		
Description	This container contains the configuration (DSD parameters) for SubServices. This configuration is available only for services having Subfunction: this container exists only if parameter DcmDsdSidTabSubfuncAvail, of this service, is set to TRUE and the parameter DcmDsdSidTabFnc is not existing.		
Configuration Parameters			

SWS Item	**ECUC_Dcm_00942 :**		
Name	DcmDsdSubServiceFnc		
Description	Callback function of the ECU Supplier specific component. This parameter is related to the interface <Module>_<DiagnosticService>_<SubService>. If it is not configured the sub-service is handled Dcm-internally.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dcm_00803 :**		
Name	DcmDsdSubServiceId		
Description	Id of the SubService identifier. The possible Subfunction parameter value are predefined in the ISO 14229-1 and ISO 15031-5.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00924 :**		
Name	DcmDsdSubServiceModeRuleRef		
Description	Reference to DcmDspModeRule Mode rule which controls execution of this the DcmDsdSubService. If there is no reference, no check of the mode rule shall be done.		
Multiplicity	0..1		
Type	Reference to [DcmModeRule]		
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_00812 :**

Name	DcmDsdSubServiceSecurityLevelRef		
Description	Link to the Security Level needed for execution of the DcmDsdSubService. 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 DcmDsdSubServiceSecurityLevelRef per DcmDsdSubService. If there is no reference, no check of security level shall be done.		
Multiplicity	0..*		
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_00804 :**		
Name	DcmDsdSubServiceSessionLevelRef		
Description	Link to the Session Control needed for execution of the DcmDsdSubService. 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 DcmDsdSubServiceSessionLevelRef per DcmDsdSubService. If there is no reference, no check of session level shall be done.		
Multiplicity	0..*		
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

10.2.7 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 used to define the name of the R-Port through which the DCM access the interface CallbackDCMRequestServices. The R-Port is named

		CallbackDCMRequestServices_<SWC> where _<SWC> is the name of the container DcmDslCallbackDCMRequestService.
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.

10.2.8 DcmDslBuffer

SWS Item	**ECUC_Dcm_00739 :**
Container Name	DcmDslBuffer
Description	This container contains the configuration (parameters) for the diagnostic buffer.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00738 :**		
Name	DcmDslBufferSize		
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	EcucIntegerParamDef		
Range	8 .. 4294967294		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.9 DcmDslCallbackDCMRequestService

SWS Item	**ECUC_Dcm_00679 :**
Container Name	DcmDslCallbackDCMRequestService
Description	The name of this container is used to define the name of the R-Port through which the DCM access the interface CallbackDCMRequestServices. The R-Port is named CallbackDCMRequestServices_<SWC> where _<SWC> is the name of the container DcmDslCallbackDCMRequestService.
Configuration Parameters	

No Included Containers

10.2.10 DcmDslDiagResp

SWS Item	**ECUC_Dcm_00691 :**
Container Name	DcmDslDiagResp
Description	This container contains the configuration (parameters) for the ResponsePending handling
Configuration Parameters	

SWS Item	**ECUC_Dcm_00693 :**		
Name	DcmDslDiagRespMaxNumRespPend		
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. Value 0xFF means that no limit number of NRC 0x78 response apply.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
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_00914 :**		
Name	DcmDslDiagRespOnSecondDeclinedRequest		
Description	Defines the reaction upon a second request (ClientB) that can not be processed (e.g. due to priority assessment). TRUE: when the second request (Client B) can not be processed, it shall be answered with NRC21 BusyRepeatRequest. FALSE: when the second request (Client B) can not be processed, it shall not be responded.		
Multiplicity	1		
Type	EcucBooleanParamDef		
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

10.2.11 DcmDslProtocol

SWS Item	**ECUC_Dcm_00694 :**
Container Name	DcmDslProtocol
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)

10.2.12 DcmDslProtocolRow

SWS Item	**ECUC_Dcm_00695 :**
Container Name	DcmDslProtocolRow
Description	Definition of a single Row of configuration for the protocol configuration (for each protocol)
Configuration Parameters	

SWS Item	**ECUC_Dcm_00886 : (Obsolete)**		
Name	DcmDslProtocolEndiannessConvEnabled		
Description	Enables /disables the endianness conversion, per diagnostic protocol. **Tags:** atp.Status=obsolete atp.StatusRevisionBegin=4.1.1		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
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: local		

SWS Item	**ECUC_Dcm_00696 :**	
Name	DcmDslProtocolID	
Description	The diagnostic protocol type for the DCM DSL protocol that is being configured. Implementation Type: Dcm_ProtocolType	
Multiplicity	1	
Type	EcucEnumerationParamDef (Symbolic Name generated for this parameter)	
Range	DCM_OBD_ON_CAN	OBD on CAN (ISO15765-4; ISO15031-5)
	DCM_OBD_ON_FLEXRAY	DCM_OBD_ON_FLEXRAY
	DCM_OBD_ON_IP	DCM_OBD_ON_IP
	DCM_PERIODICTRANS_ON_CAN	DCM_PERIODICTRANS_ON_CAN
	DCM_PERIODICTRANS_ON_FLEXRAY	DCM_PERIODICTRANS_ON_FLEXRAY
	DCM_PERIODICTRANS_ON_IP	DCM_PERIODICTRANS_ON_IP
	DCM_ROE_ON_CAN	DCM_ROE_ON_CAN
	DCM_ROE_ON_FLEXRAY	DCM_ROE_ON_FLEXRAY
	DCM_ROE_ON_IP	DCM_ROE_ON_IP
	DCM_SUPPLIER_1	Reserved for SW supplier specific
	DCM_SUPPLIER_10	Reserved for SW supplier specific
	DCM_SUPPLIER_11	Reserved for SW supplier specific
	DCM_SUPPLIER_12	Reserved for SW supplier specific
	DCM_SUPPLIER_13	Reserved for SW supplier specific
	DCM_SUPPLIER_14	Reserved for SW supplier specific

	DCM_SUPPLIER_15	Reserved for SW supplier specific	
	DCM_SUPPLIER_2	Reserved for SW supplier specific	
	DCM_SUPPLIER_3	Reserved for SW supplier specific	
	DCM_SUPPLIER_4	Reserved for SW supplier specific	
	DCM_SUPPLIER_5	Reserved for SW supplier specific	
	DCM_SUPPLIER_6	Reserved for SW supplier specific	
	DCM_SUPPLIER_7	Reserved for SW supplier specific	
	DCM_SUPPLIER_8	Reserved for SW supplier specific	
	DCM_SUPPLIER_9	Reserved for SW supplier specific	
	DCM_UDS_ON_CAN	UDS on CAN (ISO15765-3; ISO14229-1)	
	DCM_UDS_ON_FLEXRAY	DCM_UDS_ON_FLEXRAY UDS on FlexRay (Manufacturer specific; ISO14229-1)	
	DCM_UDS_ON_IP	DCM_UDS_ON_IP	
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 dependency: DcmDslProtocolRxAddrType		

SWS Item	**ECUC_Dcm_00697 : (Obsolete)**		
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). **Tags:** atp.Status=obsolete atp.StatusComment=DcmDslProtocolIsParallelExecutab is currently not used. Instead the information which protocol is allowed to be executed in parallel is defined by type of protocol (Type 1 – parallel execution is not allowed, Type 2 – parallel execution is allowed) atp.StatusRevisionBegin=4.1.3		
Multiplicity	1		
Type	EcucBooleanParamDef		
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: local		

SWS Item	**ECUC_Dcm_01020 :**		
Name	DcmDslProtocolMaximumResponseSize		
Description	This parameter is mandatory and defines the maximum length of the response message in case DcmPagedBufferEnabled == TRUE		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	4095		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmPagedBufferEnabled == TRUE		

SWS Item	**ECUC_Dcm_00698 :**

Name	DcmDslProtocolPreemptTimeout		
Description	This is the value for the timeout in seconds 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	EcucFloatParamDef		
Range	0 .. 1		
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 dependency: DcmDslProtocolID		

SWS Item	**ECUC_Dcm_00699 :**		
Name	DcmDslProtocolPriority		
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	EcucIntegerParamDef		
Range	0 .. 255		
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 dependency: DcmDslProtocolID		

SWS Item	**ECUC_Dcm_00700 :**	
Name	DcmDslProtocolTransType	
Description	Selects the transmission type for the Response On Event and PeriodicTransmission protocols. If the configuration parameter DcmDslProtocolId is not configure to Response on event or PeriodicTransmission, the parameter should not be used	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
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	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolRxPduId, DcmDslProtocol.DcmDslProtocolRow.DcmDslProtocolID		

SWS Item	**ECUC_Dcm_00910 :**		
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	EcucBooleanParamDef		
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: local		

SWS Item	**ECUC_Dcm_00729 :**		
Name	DcmTimStrP2ServerAdjust		
Description	This parameter is used to guarantee that the DCM response is available on the bus before reaching P2 by adjusting the current DcmDspSessionP2ServerMax. This parameter value in seconds has to be configured as a multiple of DcmTaskTime and is minimum in the timing handling of P2. 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	EcucFloatParamDef		
Range	0 .. 1		
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_00728 :**		
Name	DcmTimStrP2StarServerAdjust		
Description	This parameter is used to guarantee that the DCM response is available on the bus before reaching P2* by adjusting the current DcmDspSessionP2StarServerMax. This parameter value in seconds has to be configured as multiple of DcmTaskTime and is minimum in the timing handling of P2*. 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	EcucFloatParamDef		
Range	0 .. 5		
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_00701 :**		
Name	DcmDslProtocolRxBufferID		
Description	Link to buffer configuration (DcmDslBuffer) 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	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolRxPduRef, DcmDslBuffer		

SWS Item	**ECUC_Dcm_00702 :**		
Name	DcmDslProtocolSIDTable		
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	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolID, DcmDsdServiceIdTable.DcmDsdSidTabId		

SWS Item	**ECUC_Dcm_00704 :**		
Name	DcmDslProtocolTxBufferID		
Description	Link to buffer configuration (DcmDslBuffer) 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	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolTxPduRef, DcmDslBuffer		

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.

10.2.13 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	0..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 (DcmDslPeriodicTransmissionConRef) relevant for protocols with DcmDslProtocolTransType = TYPE2 or TYPE1.
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

10.2.14 DcmDslMainConnection

SWS Item	**ECUC_Dcm_00706 :**
Container Name	DcmDslMainConnection
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 (DcmDslPeriodicTransmissionConRef) relevant for protocols with DcmDslProtocolTransType = TYPE2 or TYPE1.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00826 :**		
Name	DcmDslProtocolRxTesterSourceAddr		
Description	Tester source address uniquely describes a client and will be used e.g within the jump to Bootloader interfaces. This parameter is not required for generic connections (DcmPdus with MetaDataLength ≥ 1).		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
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_00707 :**		
Name	DcmDslPeriodicTransmissionConRef		
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	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00952 :**		
Name	DcmDslProtocolComMChannelRef		
Description	Reference to the ComMChannel on which the DcmDslProtocolRxPdu is received and the DcmDslProtocolTxPdu is transmitted.		
Multiplicity	1		
Type	Symbolic name 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		

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	X	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.

10.2.15 DcmDslProtocolRx

SWS Item	**ECUC_Dcm_00709 :**
Container Name	DcmDslProtocolRx
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		
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	EcucEnumerationParamDef		
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	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolRxPduId, DcmDslProtocol.DcmDslProtocolRow.DcmDslProtocolID		

SWS Item	**ECUC_Dcm_00778 : (Obsolete)**		
Name	DcmDslProtocolRxChannelId		
Description	Channel Identifier associated to the received Pdu. **Tags:** atp.Status=obsolete atp.StatusRevisionBegin=4.1.1		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
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_00687 :**	
Name	DcmDslProtocolRxPduId	
Description	DcmRxPduId value for reception of requests.	
Multiplicity	1	
Type	EcucIntegerParamDef (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 dependency: DcmDslProtocolRxPduId		

SWS Item	**ECUC_Dcm_00906 : (Obsolete)**		
Name	DcmDslProtocolRxComMChannelRef		
Description	Reference to the ComMChannel on which the DcmDslProtocolRxPdu is received. **Tags:** atp.Status=obsolete atp.StatusRevisionBegin=4.1.1		
Multiplicity	0..1		
Type	Symbolic name 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		

SWS Item	**ECUC_Dcm_00770 :**		
Name	DcmDslProtocolRxPduRef		
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	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolRxAddrType, DcmDslProtocolRxPduRef		

No Included Containers

10.2.16 DcmDslProtocolTx

SWS Item	**ECUC_Dcm_00711 :**
Container Name	DcmDslProtocolTx
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_00864 :**	
Name	DcmDslTxConfirmationPduId	
Description	Handle Id to be used by the Lower Layer to confirm the transmission of the DcmDsProtocolTxPdu to the LowerLayer.	
Multiplicity	1	
Type	EcucIntegerParamDef (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_00772 :**		
Name	DcmDslProtocolTxPduRef		
Description	DcmTxPduId 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	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocol.DcmDslProtocolRow.DcmDslProtocolID		

No Included Containers

10.2.17 DcmDslPeriodicTransmission

SWS Item	**ECUC_Dcm_00741 :**
Container Name	DcmDslPeriodicTransmission
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	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDslPeriodicConnection	0..*	Holding the TxPduId configuration for PeriodicTransmission.

10.2.18 DcmDslPeriodicConnection

SWS Item	**ECUC_Dcm_00897 :**
Container Name	DcmDslPeriodicConnection
Description	Holding the TxPduId configuration for PeriodicTransmission.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00862 :**		
Name	DcmDslPeriodicTxConfirmationPduId		
Description	Handle Id to be used by the Lower Layer to confirm the transmission of the DcmDslPeriodicTxPdu to the LowerLayer.		
Multiplicity	1		
Type	EcucIntegerParamDef (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 dependency: DcmDslProtocolTransType		

SWS Item	**ECUC_Dcm_00742 :**		
Name	DcmDslPeriodicTxPduRef		
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 can 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	1		
Type	Reference to [Pdu]		
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 dependency: DcmDslProtocolTransType		

No Included Containers

10.2.19 DcmDslResponseOnEvent

SWS Item	**ECUC_Dcm_00744 :**		
Container Name	DcmDslResponseOnEvent		
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_00863 :**		
Name	DcmDslRoeTxConfirmationPduId		
Description	Handle Id to be used by the Lower Layer to confirm the transmission of the DcmDslRoeTxPdu to the LowerLayer.		
Multiplicity	0..1		
Type	EcucIntegerParamDef (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_00743 :**
Name	DcmDslRoeTxPduRef
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	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: DcmDslProtocolTransType		

No Included Containers

10.2.20 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	**ECUC_Dcm_00966 :**		
Name	DcmDspDDDIDcheckPerSourceDID		
Description	Defines the check the session, security and mode dependencies per source DIDs with a ReadDataByIdentifier (0x22) TRUE: DCM module shall check the session, security and mode dependencies per source DIDs with a ReadDataByIdentifier (0x22) with DID in the range 0xF200 to 0xF3FF FALSE:DCM module shall not check the session, security and mode dependencies per source DIDs with a ReadDataByIdentifier (0x22) with DID in the range 0xF200 to 0xF3FF		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
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: local		

SWS Item	**ECUC_Dcm_00638 :**		
Name	DcmDspMaxDidToRead		
Description	Indicates the maximum allowed DIDs in a single "ReadDataByIdentifier" request.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
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: ECU		

SWS Item	**ECUC_Dcm_00956 :**		
Name	DcmDspMaxPeriodicDidToRead		
Description	Indicates the maximum allowed periodicDIDs which can be read in a single "ReadDataByPeriodicIdentifier" request.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
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: local		

SWS Item	**ECUC_Dcm_00818 :**		
Name	DcmDspPowerDownTime		
Description	This parameter indicates to the client the minimum time of the stand-by sequence the server will remain in the power-down sequence. The resolution of this parameter is one second per count. The following values are valid: 00 - FE hex: 0 - 254 s powerDownTime; FF hex: indicates a failure or time not available. This value needs to be defined by the integrator according to the ECU capabilities. This parameter has to be available if the service EcuReset, sub-service enableRapidPowerShutDown is configured.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
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	--
DcmDspControlDTCSetting	0..1	Provide the configuration of the ControlDTCSetting mechanism.
DcmDspData	0..*	This container contains the configuration (parameters) of a Data belonging to a DID
DcmDspDataInfo	0..*	This container contains the configuration (parameters) of a Data
DcmDspDid	0..*	This container contains the configuration (parameters) of the DID.
DcmDspDidInfo	0..*	This container contains the configuration (parameters) of the DID's Info
DcmDspDidRange	0..*	This container defines the DID Range
DcmDspMemory	0..1	This container contains the configuration of the memory access.
DcmDspPeriodicDidTransmission	0..1	This container contains the configuration for the Periodic Did transmission. This container exists only if the UDS Service ReadDataByPeriodicIdentifier(0x2A) is configured.
DcmDspPeriodicTransmission	0..1	This container contains the configuration (parameters) for Periodic Transmission Processing.
DcmDspPid	0..*	This container defines the availability of a PID to the DCM.

DcmDspRequestControl	0..*	This container contains the configuration (parameters) of the "Request control of on-board system, test or component" service (Service $08). The DCM will request the control using an R-Port requiring a PortInteface RequestControlServices_{Tid}. The R-Port is named RequestControlServices_{Tid} where {Tid} is the name of the container DcmDspRequestControl.
DcmDspRoe	0..1	Provide the configuration of the ResponseOnEvent mechanism.
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.
DcmDspVehInfo	0..*	This container contains the configuration (parameters) of the "Request vehicle information service" (service $09).

10.2.21 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.
DcmDspComControlSetting	0..1	Provide the configuration of the Communication control.
DcmDspComControlSpecificChannel	0..*	Assigns subnet number to ComM channel which will be controlled.

10.2.22 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
Description	Reference to ComM channel.
Multiplicity	1
Type	Symbolic name 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

10.2.23 DcmDspComControlSetting

SWS Item	**ECUC_Dcm_00943 :**
Container Name	DcmDspComControlSetting
Description	Provide the configuration of the Communication control.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00944 :**		
Name	DcmDspComControlCommunicationReEnableModeRuleRef		
Description	Reference to DcmDspModeRule Mode rule which controls re-enabling of communication by DCM. The DCM module shall call BswM_Dcm_CommunicationMode_CurrentState(DCM_ENABLE_RX_TX_NORM_NM)) in case that the referenced mode rule is not fulfilled anymore.		
Multiplicity	0..1		
Type	Reference to [DcmModeRule]		
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

10.2.24 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	EcucIntegerParamDef		
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		
Description	Reference to ComM channel.		
Multiplicity	1		
Type	Symbolic name 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 DcmDspDid

SWS Item	**ECUC_Dcm_00601 :**
Container Name	DcmDspDid
Description	This container contains the configuration (parameters) of the DID.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00602 :**		
Name	DcmDspDidIdentifier		
Description	2 byte Identifier of the DID Within each DcmConfigSet all DcmDspDidIdentifier values shall be unique.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00859 :**		
Name	DcmDspDidRoeQueueEnabled		
Description	If set to TRUE, the ROE queue mechanism will be used in case of ROE OnChangeOfDataIdentifier on this DID. If set to FALSE, the ROE event will be sent without queuing.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
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_00805 :**
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	EcucBooleanParamDef		
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_00604 :**		
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	**ECUC_Dcm_00606 :**		
Name	DcmDspDidRef		
Description	Reference to DcmDspDid in case this DID refer to one or serveral other DID's **Attributes:** requiresIndex=true		
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***
DcmDspDidSignal	0..*	This container defines the reference to 1 DcmDspData container and position relevant for this DID.

10.2.26 DcmDspDidSignal

SWS Item	**ECUC_Dcm_00813 :**
Container Name	DcmDspDidSignal
Description	This container defines the reference to 1 DcmDspData container and position relevant for this DID.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00814 :**
Name	DcmDspDidDataPos
Description	Defines the position of the data defined by DcmDspDidDataRef reference to DcmDspData container in the DID.

	The position is defined in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00808 :**		
Name	DcmDspDidDataRef		
Description	Reference to 1 DcmDspData container relevant for this DID.		
Multiplicity	1		
Type	Reference to [DcmDspData]		
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

10.2.27 DcmDspDidRange

SWS Item	**ECUC_Dcm_00937 :**
Container Name	DcmDspDidRange
Description	This container defines the DID Range
Configuration Parameters	

SWS Item	**ECUC_Dcm_00941 :**		
Name	DcmDspDidRangeHasGaps		
Description	Parameter specifying if there are gaps in the DID range (parameter set to TRUE) or not (parameter set to FALSE)		
Multiplicity	1		
Type	EcucBooleanParamDef		
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_00938 :**		
Name	DcmDspDidRangeIdentifierLowerLimit		
Description	Lower limit of DID range.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00939 :**		
Name	DcmDspDidRangeIdentifierUpperLimit		
Description	Upper limit of DID range.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00946 :**		
Name	DcmDspDidRangeIsDidAvailableFnc		
Description	Function name to request from application if a specific DID is available within the range or not. Only relevant if DcmDspDidRangeUsePort is set to false. This parameter is related to the interface Xxx_IsDidAvailable.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00940 :**		
Name	DcmDspDidRangeMaxDataLength		
Description	Maximum data length in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00947 :**
Name	DcmDspDidRangeReadDidFnc
Description	Function name to request from application the data range value of a DID.(ReadData-function). Only relevant if DcmDspDidRangeUsePort is set to false. This parameter is related to the interface Xxx_ReadDidData.

Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00945 :**		
Name	DcmDspDidRangeUsePort		
Description	When the parameter DcmDspDidRangeUsePort is set to true the DCM will access the Data using an R-Port requiring a PortInteface DataServices_DIDRange. In that case, DcmDspDidRangeIsDidAvailableFnc, DcmDspDidRangeReadDidFnc and DcmDspDidRangeWriteDidFnc are ignored and the RTE APIs are used. When the parameter DcmDspDidRangeUsePort is false, the DCM calls the functions defined in DcmDspDidRangeIsDidAvailableFnc, DcmDspDidRangeReadDidFnc and DcmDspDidRangeWriteDidFnc.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00948 :**		
Name	DcmDspDidRangeWriteDidFnc		
Description	Function name to request application to write the data range value of a DID.(WriteData-function). Only relevant if DcmDspDidRangeUsePort is set to false. This parameter is related to the interface Xxx_WriteDidData.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00950 :**		
Name	DcmDspDidRangeInfoRef		
Description	Reference to DcmDspDidInfo containing information on this DID Range.		
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		

No Included Containers

10.2.28 DcmDspControlDTCSetting

SWS Item	**ECUC_Dcm_00935 :**
Container Name	DcmDspControlDTCSetting
Description	Provide the configuration of the ControlDTCSetting mechanism.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00965 :**		
Name	DcmSupportDTCSettingControlOptionRecord		
Description	This configuration switch defines if the DTCSettingControlOptionRecord is in general supported in the request message or not.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00936 :**		
Name	DcmDspControlDTCSettingReEnableModeRuleRef		
Description	Reference to DcmDspModeRule Mode rule which controls re-enabling of controlDTCsetting by DCM. The DCM module shall execute a ControlDTCSetting.Off (call Dem_DcmEnableDTCSetting()) in case that the referenced mode rule is not fulfilled anymore.		
Multiplicity	0..1		
Type	Reference to [DcmModeRule]		
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

10.2.29 DcmDspData

SWS Item	**ECUC_Dcm_00869 :**
Container Name	DcmDspData
Description	This container contains the configuration (parameters) of a Data belonging to a DID
Configuration Parameters	

SWS Item	**ECUC_Dcm_00677 :**

Name	DcmDspDataConditionCheckReadFnc		
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 DcmDspDataReadFnc. Only relevant if • DcmDspDataConditionCheckReadFncUsed is set to 'TRUE' and • o DcmDspDataUsePort=="USE_DATA_SYNCH_FNC or o DcmDspDataUsePort==USE_DATA_ASYNCH_FNC". This parameter is related to the interface Xxx_ConditionCheckRead.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDataReadFnc, DcmDspDataUsePort, DcmDspDataConditionCheckReadFncUsed		

SWS Item	**ECUC_Dcm_00955 :**		
Name	DcmDspDataConditionCheckReadFncUsed		
Description	This parameter determines if a condition check function is available or not. If the parameter is set to 'TRUE' and DcmDspDataUsePort is set to 'USE_DATA_ASYNCH_CLIENT_SERVER' or 'USE_DATA_SYNCH_CLIENT_SERVER', the DCM shall generate the according function call. If the parameter is set to 'TRUE' and DcmDspDataUsePort is set to 'USE_DATA_SYNCH_FNC' or 'USE_DATA_ASYNCH_FNC', the parameter 'DcmDspDataConditionCheckReadFnc shall contain a valid C-function.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: DcmDspDataConditionCheckReadFnc, DcmDspDataUsePort		

SWS Item	**ECUC_Dcm_00825 :**
Name	DcmDspDataEcuSignal
Description	Function name to control the access to a certain ECU Signal by the DCM. (IoHwAb_Dcm_<symbolic name of ECU signal>-function). Only relevant if DcmDspDataUsePort==USE_ECU_SIGNAL.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	--
maxLength	--

minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDataUsePort		

SWS Item	**ECUC_Dcm_00986 :**		
Name	DcmDspDataEndianness		
Description	Defines the endianness of the data belonging to a DID. If no DcmDspDataEndiness is defined the value of DcmDspDataDefaultEndianness is applicable.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN		Most significant byte shall come at the lowest address.
	LITTLE_ENDIAN		Most significant byte shall come highest address
	OPAQUE		opaque data endianness
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: local dependency: DcmDspDataDefaultEndianness		

SWS Item	**ECUC_Dcm_00674 :**		
Name	DcmDspDataFreezeCurrentStateFnc		
Description	Function name to request to application to freeze the current state of an IOControl. (FreezeCurrentState-function). Only relevant if DcmDspDataUsePort=="USE_DATA_SYNCH_FNC or DcmDspDataUsePort==USE_DATA_ASYNCH_FNC". This parameter is related to the interface Xxx_FreezeCurrentState.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDidFreezeCurrentState, DcmDspDataUsePort		

SWS Item	**ECUC_Dcm_00676 :**		
Name	DcmDspDataGetScalingInfoFnc		
Description	Function name to request to application the scaling information of the DID. (GetScalingInformation-function). Only relevant if DcmDspDataUsePort=="USE_DATA_SYNCH_FNC or DcmDspDataUsePort==USE_DATA_ASYNCH_FNC". This parameter is related to the interface Xxxx_GetScalingInformation.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		

maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDataScalingInfoSize, DcmDspDataUsePort		

SWS Item	**ECUC_Dcm_00671 :**		
Name	DcmDspDataReadDataLengthFnc		
Description	Function name to request from application the data length of a DID. (ReadDataLength-function). Only relevant if DcmDspDataUsePort=="USE_DATA_SYNCH_FNC or DcmDspDataUsePort==USE_DATA_ASYNCH_FNC". This parameter is related to the interface Xxx_ReadDataLength.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDataFixedLength, DcmDspDataUsePort		

SWS Item	**ECUC_Dcm_00824 :**		
Name	DcmDspDataReadEcuSignal		
Description	Function name for read access to a certain ECU Signal by the DCM. (IoHwAb_Dcm_Read<EcuSignalName>-function). Only relevant if DcmDspDataUsePort==USE_ECU_SIGNAL.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDataUsePort		

SWS Item	**ECUC_Dcm_00669 :**
Name	DcmDspDataReadFnc
Description	Function name to request from application the data value of a DID. (ReadData-function). Multiplicity shall be equal to parameter DcmDspDataConditionCheckReadFnc. Only relevant if • DcmDspDataConditionCheckReadFncUsed is set to 'TRUE' and •

	○ DcmDspDataUsePort=="USE_DATA_SYNCH_FNC or ○ DcmDspDataUsePort==USE_DATA_ASYNCH_FNC". This parameter is related to the interface Xxx_ReadData.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDataConditionCheckReadFnc, DcmDspDataUsePort, DcmDspDataConditionCheckReadFncUsed		

SWS Item	**ECUC_Dcm_00673 :**		
Name	DcmDspDataResetToDefaultFnc		
Description	Function name to request to application to reset an IOControl to default value. (ResetToDefault-function). Only relevant if DcmDspDataUsePort=="USE_DATA_SYNCH_FNC or DcmDspDataUsePort==USE_DATA_ASYNCH_FNC". This parameter is related to the interface Xxx_ResetToDefault.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDidResetToDefault, DcmDspDataUsePort		

SWS Item	**ECUC_Dcm_00672 :**		
Name	DcmDspDataReturnControlToEcuFnc		
Description	Function name to request to application to return control to ECU of an IOControl. (ReturnControlToECU-function). Only relevant if DcmDspDataUsePort=="USE_DATA_SYNCH_FNC or DcmDspDataUsePort==USE_DATA_ASYNCH_FNC". This parameter is related to the interface Xxx_ReturnControlToECU.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDidReturnControlToEcu, DcmDspDataUsePort		

SWS Item	**ECUC_Dcm_00675 :**		
Name	DcmDspDataShortTermAdjustmentFnc		
Description	Function name to request to application to adjuste the IO signal. (ShortTermAdjustment-function). Only relevant if DcmDspDataUsePort=="USE_DATA_SYNCH_FNC or DcmDspDataUsePort==USE_DATA_ASYNCH_FNC". This parameter is related to the interface Xxx_ShortTermAdjustment.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDidShortTermAdjustment, DcmDspDataUsePort		

SWS Item	**ECUC_Dcm_00605 :**		
Name	DcmDspDataSize		
Description	Length of data in bits associated to the Data. If Data has variable datalength, that corresponds to the maximum datalength.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
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_00985 :**		
Name	DcmDspDataType		
Description	Provide the implementation data type of data belonging to a DID.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BOOLEAN		type of the data is boolean.
	SINT16		type of the data is sint16.
	SINT32		type of the data is sint32.
	SINT8		type of the data is sint8.
	UINT16		type of the data is uint16.
	UINT32		type of the data is uint32.
	UINT8		type of the data is uint8.
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: DcmDspDataSize		

SWS Item	**ECUC_Dcm_00713 :**
Name	DcmDspDataUsePort
Description	Defines which interface shall be used to access the data.
Multiplicity	1

Type	EcucEnumerationParamDef		
Range	USE_BLOCK_ID	The DCM will access the Data using the NVRAM Apis with the BlockId defined in DcmDspDataBlockId	
	USE_DATA_ASYNCH_CLIENT_SERVER	The DCM will access the Data using an R-Port requiring a asynchronous ClientServertInterface DataServices_{Data}. The R-Port is named DataServices_{Data} where {Data} is the name of the container DcmDspData.	
	USE_DATA_ASYNCH_FNC	The DCM will access the Data using the functions that are defined in parameters of type EcucFunctionNameDef (but without DcmDspDataReadDataLengthFnc) in the DcmDspData container. DCM_E_PENDING return is allowed. OpStatus is existing as IN parameter.	
	USE_DATA_SENDER_RECEIVER	The DCM will access the Data using an Port requiring a SenderReceiverInteface DataServices_{Data}. The Port is namedDataServices_{Data} where {Data} is the name of the container DcmDspData.	
	USE_DATA_SYNCH_CLIENT_SERVER	The DCM will access the Data using an R-Port requiring a synchronous ClientServertInterface DataServices_{Data}. The R-Port is named DataServices_{Data} where {Data} is the name of the container DcmDspData.	
	USE_DATA_SYNCH_FNC	The DCM will access the Data using the functions that are defined in parameters of type EcucFunctionNameDef (but without DcmDspDataReadDataLengthFnc) in the DcmDspData container. DCM_E_PENDING return value is not allowed and OpStatus parameter is not existing in the prototype.	
	USE_ECU_SIGNAL	The DCM will access the Data using a direct access to IoHwAb	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00670 :**
Name	DcmDspDataWriteFnc
Description	Function name to request application to write the data value of a DID. (WriteData-function). Only relevant if DcmDspDataUsePort=="USE_DATA_SYNCH_FNC or DcmDspDataUsePort==USE_DATA_ASYNCH_FNC". This parameter is related to the interface Xxx_WriteData.
Multiplicity	0..1
Type	EcucFunctionNameDef

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

SWS Item	**ECUC_Dcm_00988 :**		
Name	DcmDspOdxDataDescription		
Description	Defines additional description for ODX documentation		
Multiplicity	0..1		
Type	EcucAddInfoParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dcm_00809 :**		
Name	DcmDspDataBlockIdRef		
Description	NRAM blockId to access the data. Only relevant if DcmDspDataUsePort==USE_BLOCK_ID.		
Multiplicity	0..1		
Type	Symbolic name reference to [NvMBlockDescriptor]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspDataUsePort		

SWS Item	**ECUC_Dcm_00811 :**		
Name	DcmDspDataInfoRef		
Description	Reference to 1 DcmDspDataInfo		
Multiplicity	1		
Type	Reference to [DcmDspDataInfo]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspDiagnosisScaling	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.
DcmDspExternalSRDataElementClass	0..1	This container defines the source of data in a provided port which shall be read respectively the target of data in a required port which shall be written. This container shall contain either one DcmSubElementInDataElementInstance OR DcmDataElementInstance OR

		DcmSubElementInImplDataElementInstance reference.

10.2.30 DcmDspDiagnosisScaling

SWS Item	**ECUC_Dcm_00993 :**
Choice container Name	DcmDspDiagnosisScaling
Description	This container contains the configuration (parameters) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspAlternativeDataInterface	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of a VariableDataPrototype in a DataInterface. Additionally a reference to PortInterfaceMapping can be defined which provide already the mapping rules between the VariableDataPrototype in a DataInterface used by the software component (DcmDspExternalSRDataElementClass) and the intended Diagnosis Representation defined by DcmDataElement.
DcmDspAlternativeDataProps	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of ECU configuration parameters. The physical unit of the alternative data representation is defined by the DataPrototype referenced by DcmDspExternalSRDataElementClass. Additionally the definition of a text table mapping can be a defined for DcmDspDataTypeCategory TEXTTABLE and SCALE_LINEAR_AND_TEXTTABLE
DcmDspAlternativeDataType	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of an ApplicationDataType. Additionally the definition of a text table mapping can be a defined for ApplicationDataTypes of category TEXTTABLE and SCALE_LINEAR_AND_TEXTTABLE.

10.2.31 DcmDspAlternativeDataProps

SWS Item	**ECUC_Dcm_01002 :**
Container Name	DcmDspAlternativeDataProps
Description	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of ECU configuration parameters. The physical unit of the alternative data representation is defined by the DataPrototype referenced by DcmDspExternalSRDataElementClass. Additionally the definition of a text table mapping can be a defined for DcmDspDataTypeCategory TEXTTABLE and SCALE_LINEAR_AND_TEXTTABLE
Configuration Parameters	

SWS Item	**ECUC_Dcm_01008 :**

Name	DcmDspDataTypeCategory		
Description	Data category of the alternative Diagnosis Representation.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	LINEAR	category is LINEAR	
	SCALE_LINEAR_AND_TEXTTABLE	category is SCALE_LINEAR_AND_TEXTTABLE	
	TEXTTABLE	category is TEXTTABLE	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspLinearScale	0..1	This container contains the configuration (parameters) of an linear scale of the alternative Diagnosis Representation.
DcmDspTextTableMapping	0..*	This container contains the configuration (parameters) of the mapping a DataPrototype typed by AutosarDataType that refer to a CompuMethods of category TEXTTABLE or SCALE_LINEAR_AND_TEXTTABLE. Each DcmDspTextTableMapping defines a value pair which is used to map the ECU internal value (DcmDspInternalDataValue) to the vale used in the diagnosis representation (DcmDspDiagnosisRepresentationDataValue) and vice versa. The set of all DcmDspTextTableMappings defines the whole mapping of an data.

10.2.32 DcmDspLinearScale

SWS Item	**ECUC_Dcm_01003 :**
Container Name	DcmDspLinearScale
Description	This container contains the configuration (parameters) of an linear scale of the alternative Diagnosis Representation.
Configuration Parameters	

SWS Item	**ECUC_Dcm_01007 :**		
Name	DcmDspDiagnosisRepresentationDataLowerRange		
Description	Lower Range for this scale of the data in the alternative Diagnosis Representation.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	-INF .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_01005 :**
Name	DcmDspDiagnosisRepresentationDataOffset

Description	Data offset of the alternative Diagnosis Representation for this scale.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	0		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_01004 :**		
Name	DcmDspDiagnosisRepresentationDataResolution		
Description	Data resolution of the alternative Diagnosis Representation for this scale.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_01006 :**		
Name	DcmDspDiagnosisRepresentationDataUpperRange		
Description	Upper Range for this scale of the data in the alternative Diagnosis Representation.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	-INF .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.33 DcmDspTextTableMapping

SWS Item	**ECUC_Dcm_00999 :**
Container Name	DcmDspTextTableMapping
Description	This container contains the configuration (parameters) of the mapping a DataPrototype typed by AutosarDataType that refer to a CompuMethods of category TEXTTABLE or SCALE_LINEAR_AND_TEXTTABLE. Each DcmDspTextTableMapping defines a value pair which is used to map the ECU internal value (DcmDspInternalDataValue) to the vale used in the diagnosis representation (DcmDspDiagnosisRepresentationDataValue) and vice versa. The set of all DcmDspTextTableMappings defines the whole mapping of an data.
Configuration Parameters	

SWS Item	**ECUC_Dcm_01001 :**		
Name	DcmDspDiagnosisRepresentationDataValue		
Description	The data value in the diagnosis representation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_01000 :**		
Name	DcmDspInternalDataValue		
Description	The ECU internal data value.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.34 DcmDspAlternativeDataInterface

SWS Item	**ECUC_Dcm_00994 :**		
Container Name	DcmDspAlternativeDataInterface		
Description	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of a VariableDataPrototoype in a DataInterface. Additionally a reference to PortInterfaceMapping can be defined which provide already the mapping rules between the VariableDataPrototoype in a DataInterface used by the software component (DcmDspExternalSRDataElementClass) and the intended Diagnosis Representation defined by DcmDataElement.		
Configuration Parameters			

SWS Item	**ECUC_Dcm_00995 :**		
Name	DcmDataElement		
Description	Alternative Diagnosis Representation for the data defined by the means of a VariableDataPrototoype in a DataInterface.		
Multiplicity	1		
Type	Foreign reference to [VARIABLE-DATA-PROTOTYPE]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**ECUC_Dcm_00996 :**		
Name	DcmPortInterfaceMapping		
Description	Optional reference to PortInterfaceMapping which defines the mapping rules.		
Multiplicity	0..1		
Type	Foreign reference to [PORT-INTERFACE-MAPPING]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.35 DcmDspAlternativeDataType

SWS Item	**ECUC_Dcm_00997 :**		
Container Name	DcmDspAlternativeDataType		
Description	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of an ApplicationDataType. Additionally the definition of a text table mapping can be a defined for ApplicationDataTypes of category TEXTTABLE and SCALE_LINEAR_AND_TEXTTABLE.		
Configuration Parameters			

SWS Item	**ECUC_Dcm_00998 :**		
Name	DcmApplicationDataType		
Description	Alternative Diagnosis Representation for the data defined by the means of a ApplicationPrimitiveDataType of category VALUE or BOOLEAN.		
Multiplicity	1		
Type	Foreign reference to [APPLICATION-PRIMITIVE-DATA-TYPE]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspTextTableMapping	0..*	This container contains the configuration (parameters) of the mapping a DataPrototype typed by AutosarDataType that refer to a CompuMethods of category TEXTTABLE or SCALE_LINEAR_AND_TEXTTABLE. Each DcmDspTextTableMapping defines a value pair which is used to map the ECU internal value (DcmDspInternalDataValue) to the vale used in the diagnosis representation (DcmDspDiagnosisRepresentationDataValue) and vice versa. The set of all DcmDspTextTableMappings defines the whole mapping of an data.

10.2.36 DcmDspExternalSRDataElementClass

SWS Item	**ECUC_Dcm_00989 :**
Choice container Name	DcmDspExternalSRDataElementClass
Description	This container defines the source of data in a provided port which shall be read respectively the target of data in a required port which shall be written. This container shall contain either one DcmSubElementInDataElementInstance OR DcmDataElementInstance OR DcmSubElementInImplDataElementInstance reference.

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDataElementInstance	0..1	Instance Reference to the primitive data in a port where the data element is typed with an ApplicationPrimitveDataType or an ImplementationDataType.
DcmSubElementInDataElementInstance	0..1	Instance Reference to the primitve sub-element (at any level) of composite data in a port where the data element is typed with an ApplicationCompositeDataType.
DcmSubElementInImplDataElementInstance	0..1	Instance Reference to the primitve sub-element (at any level) of composite data in a port where the data element is typed with an ImplementationDataType.

10.2.37 DcmDataElementInstance

SWS Item	**ECUC_Dcm_01010 :**
Container Name	DcmDataElementInstance
Description	Instance Reference to the primitive data in a port where the data element is typed with an ApplicationPrimitveDataType or an ImplementationDataType.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00991 :**		
Name	DcmDataElementInstanceRef		
Description	Instance Reference to the primitive data which shall be read or written. Supported are VariableDataPrototypes in SenderReceiverInterfaces and NvDataInterfaces and ParameterDataPrototypes in ParameterInterfaces (read only). This reference is applicable if the AutosarDataPrototype is typed with a ApplicationPrimitiveDataType of category VALUE or BOOLEAN or if the AutosarDataPrototype is typed with a ImplementationDataType of category VALUE or TYPE_REFERENCE that in turn boils down to VALUE		
Multiplicity	1		
Type	Instance reference to [AUTOSAR-DATA-PROTOTYPE context: ROOT-SW-COMPOSITION-PROTOTYPE SW-COMPONENT-PROTOTYPE PORT-PROTOTYPE]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.38 DcmSubElementInDataElementInstance

SWS Item	**ECUC_Dcm_01009 :**
Container Name	DcmSubElementInDataElementInstance
Description	Instance Reference to the primitve sub-element (at any level) of composite data in a port where the data element is typed with an ApplicationCompositeDataType.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00990 :**		
Name	DcmSubElementInDataElementInstanceRef		
Description	Instance Reference to the primitve sub-element (at any level) of composite data in a port which shall be read or written. Supported are VariableDataPrototypes in SenderReceiverInterfaces and NvDataInterfaces and ParameterDataPrototypes in ParameterInterfaces (read only). This reference is applicable if the AutosarDataPrototype is typed with a ApplicationCompositeDataType.		
Multiplicity	1		
Type	Instance reference to [AUTOSAR-DATA-PROTOTYPE context: ROOT-SW-COMPOSITION-PROTOTYPE SW-COMPONENT-PROTOTYPE PORT-PROTOTYPE AUTOSAR-DATA-PROTOTYPE APPLICATION-COMPOSITE-ELEMENT-DATA-PROTOTYPE*]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.39 DcmSubElementInImplDataElementInstance

SWS Item	**ECUC_Dcm_01011 :**
Container Name	DcmSubElementInImplDataElementInstance
Description	Instance Reference to the primitve sub-element (at any level) of composite data in a port where the data element is typed with an ImplementationDataType.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00992 :**
Name	DcmSubElementInImplDataElementInstanceRef
Description	Instance Reference to the primitve sub-element (at any level) of composite data in a port which shall be read or written. Supported are VariableDataPrototypes in SenderReceiverInterfaces and NvDataInterfaces and ParameterDataPrototypes in ParameterInterfaces (read only). This reference is applicable if the AutosarDataPrototype is typed with a ImplementationDataType of category STRUCTURE or ARRAY. Please note that in case of ARRAY the index attribute in the target reference has to be set to select a single array element.
Multiplicity	1
Type	Instance reference to [IMPLEMENTATION-DATA-TYPE-ELEMENT context: ROOT-SW-COMPOSITION-PROTOTYPE SW-COMPONENT-

	PROTOTYPE PORT-PROTOTYPE AUTOSAR-DATA-PROTOTYPE IMPLEMENTATION-DATA-TYPE-ELEMENT*]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.40 DcmDspDataInfo

SWS Item	**ECUC_Dcm_00810 :**
Container Name	DcmDspDataInfo
Description	This container contains the configuration (parameters) of a Data
Configuration Parameters	

SWS Item	**ECUC_Dcm_00608 :**		
Name	DcmDspDataFixedLength		
Description	Indicates if the datalength of the Data is fixed true = datalength of the Data is fixed false = datalength of the Data is variable		
Multiplicity	1		
Type	EcucBooleanParamDef		
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_00611 :**		
Name	DcmDspDataScalingInfoSize		
Description	If Scaling information service is available for this Data, it provides the size in bytes of the scaling information.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
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

10.2.41 DcmDspDidInfo

SWS Item	**ECUC_Dcm_00607 :**

Container Name	DcmDspDidInfo
Description	This container contains the configuration (parameters) of the DID's Info
Configuration Parameters	

SWS Item	**ECUC_Dcm_00612 :**		
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	EcucBooleanParamDef		
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

10.2.42 DcmDspDidAccess

SWS Item	**ECUC_Dcm_00609 :**
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.
DcmDspDidDefine	0..1	This container contains the configuration (parameters) of the DDDID 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.

10.2.43 DcmDspDidControl

SWS Item	**ECUC_Dcm_00619 :**
Container Name	DcmDspDidControl
Description	This container contains the configuration (parameters) of the DID control.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00624 :**
Name	DcmDspDidFreezeCurrentState

Description	**This indicates the presence of "FreezeCurrentState".**		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00623 :**		
Name	DcmDspDidResetToDefault		
Description	**This indicates the presence of "ResetToDefault".**		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00622 :**		
Name	DcmDspDidReturnControlToEcu		
Description	**This indicates the presence of "ReturnControlToEcu".**		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00625 :**		
Name	DcmDspDidShortTermAdjustment		
Description	This indicates the presence of "ShortTermAdjustment".		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00923 :**		
Name	DcmDspDidControlModeRuleRef		
Description	Reference to DcmDspModeRule Mode rule which controls this DID. If there is no reference, no check of the mode rule shall be done.		
Multiplicity	0..1		
Type	Reference to [DcmModeRule]		
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_00620 :**		
Name	DcmDspDidControlSecurityLevelRef		
Description	Reference to DcmDspSecurityRow Security levels allowed to control this DID. If there is no reference, no check of security level shall be done.		
Multiplicity	0..*		
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_00621 :**		
Name	DcmDspDidControlSessionRef		
Description	Reference to DcmDspSessionRow Sessions allowed to control this DID. If there is no reference, no check of session level shall be done.		
Multiplicity	0..*		
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

10.2.44 DcmDspDidDefine

SWS Item	**ECUC_Dcm_00972 :**
Container Name	DcmDspDidDefine
Description	This container contains the configuration (parameters) of the DDDID control.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00970 :**		
Name	DcmDspDDDIDMaxElements		
Description	Maximum number of source elements of a DDDID.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
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: local		

No Included Containers

10.2.45 DcmDspDidRead

SWS Item	**ECUC_Dcm_00613 :**		
Container Name	DcmDspDidRead		
Description	This container contains the configuration (parameters) of the DID read.		
Configuration Parameters			

SWS Item	**ECUC_Dcm_00917 :**		
Name	DcmDspDidReadModeRuleRef		
Description	Reference to DcmDspModeRule Mode rule which controls to read this DID. If there is no reference, no check of the mode rule shall be done.		
Multiplicity	0..1		
Type	Reference to [DcmModeRule]		
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_00614 :**		
Name	DcmDspDidReadSecurityLevelRef		
Description	Reference to DcmDspSecurityRow Security levels allowed to read this DID. If there is no reference, no check of security level shall be done.		
Multiplicity	0..*		
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_00615 :**		
Name	DcmDspDidReadSessionRef		
Description	Reference to DcmDspSessionRow Sessions allowed to read this DID. If there is no reference, no check of session level shall be done.		
Multiplicity	0..*		
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

10.2.46 DcmDspDidWrite

SWS Item	**ECUC_Dcm_00616 :**
Container Name	DcmDspDidWrite

Description	This container contains the configuration (parameters) of the DID write.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00922 :**		
Name	DcmDspDidWriteModeRuleRef		
Description	Reference to DcmDspModeRule Mode rule which controls to write this DID. If there is no reference, no check of the mode rule shall be done.		
Multiplicity	0..1		
Type	Reference to [DcmModeRule]		
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_00617 :**		
Name	DcmDspDidWriteSecurityLevelRef		
Description	Reference to DcmDspSecurityRow Security levels allowed to write this DID. If there is no reference, no check of security level shall be done.		
Multiplicity	0..*		
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_00618 :**		
Name	DcmDspDidWriteSessionRef		
Description	Reference to DcmDspSessionRow Sessions allowed to write this DID. If there is no reference, no check of session level shall be done.		
Multiplicity	0..*		
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

10.2.47 DcmDspMemory

SWS Item	**ECUC_Dcm_00784 :**
Container Name	DcmDspMemory
Description	This container contains the configuration of the memory access.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00912 : (Obsolete)**
Name	DcmDspUseMemoryId

Description	If this parameter is set to false, the DCM will not use MemoryIdentifier parameter. The Dcm_WriteMemory and Dcm_ReadMemory callouts shall be called without the MemoryIdentifier parameter. If this parameter is set to true, the DCM will use MemoryIdentifier parameter to select the memory device to use. The Dcm_WriteMemory and Dcm_ReadMemory callouts shall be called with the MemoryIdentifier parameter. Hint: This parameter is obsolete. Omitting the DcmDspMemoryIdValues has the identical semantic as setting DcmDspUseMemoryId == false. **Tags:** atp.Status=obsolete atp.StatusRevisionBegin=4.1.3		
Multiplicity	1		
Type	EcucBooleanParamDef		
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: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspAddressAndLengthFormatIdentifier	0..1	This container contains the configuration of the supported AddressAndLengthFormatIdentifiers for memory access.
DcmDspMemoryIdInfo	1..*	Provides the value of memory identifier used to select the desired memory device This container contains the configuration of the memory access requested through diagnostic services : ReadMemoryByAddress, WriteMemoryByAddress, RequestDownload, RequestUpload

10.2.48 DcmDspAddressAndLengthFormatIdentifier

SWS Item	**ECUC_Dcm_00963 :**
Container Name	DcmDspAddressAndLengthFormatIdentifier
Description	This container contains the configuration of the supported AddressAndLengthFormatIdentifiers for memory access.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00964 :**		
Name	DcmDspSupportedAddressAndLengthFormatIdentifier		
Description	This parameter defines the supported AddressAndLengthFormatIdentifier of the request message.		
Multiplicity	1..*		
Type	EcucIntegerParamDef		
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

10.2.49 DcmDspMemoryIdInfo

SWS Item	**ECUC_Dcm_00911 :**
Container Name	DcmDspMemoryIdInfo
Description	Provides the value of memory identifier used to select the desired memory device This container contains the configuration of the memory access requested through diagnostic services : ReadMemoryByAddress, WriteMemoryByAddress, RequestDownload, RequestUpload
Configuration Parameters	

SWS Item	**ECUC_Dcm_00913 :**
Name	DcmDspMemoryIdValue
Description	Value of the memory device identifier used. If this parameter is not configured, the DCM will not use MemoryIdentifier parameter. The Dcm_WriteMemory and Dcm_ReadMemory callouts shall be called without the MemoryIdentifier parameter. If this parameter is configured, the DCM will use MemoryIdentifier parameter to select the memory device to use. The Dcm_WriteMemory and Dcm_ReadMemory callouts shall be called with the MemoryIdentifier parameter. Every values configured in the configuration parameter DcmDspMemoryIdValue shall be unique. The MemoryIdValue is retrieved from the request messages (RMBA,WMBA,RD,RU,DDDI) according to ISO-14229-1.
Multiplicity	0..1
Type	EcucIntegerParamDef

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***
DcmDspReadMemoryRangeInfo	0..*	Provides the range of memory address allowed for reading
DcmDspWriteMemoryRangeInfo	0..*	Provides the range of memory address allowed for writting.

10.2.50 DcmDspReadMemoryRangeInfo

SWS Item	**ECUC_Dcm_00785 :**
Container Name	DcmDspReadMemoryRangeInfo
Description	Provides the range of memory address allowed for reading
Configuration Parameters	

SWS Item	**ECUC_Dcm_00787 :**		
Name	DcmDspReadMemoryRangeHigh		
Description	High memory address of a range allowed for reading		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967294		
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_00786 :**		
Name	DcmDspReadMemoryRangeLow		
Description	Low memory address of a range allowed for reading		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967294		
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_00920 :**
Name	DcmDspReadMemoryRangeModeRuleRef
Description	Reference to DcmDspModeRule Mode rule which controls read access on this memory address. If there is

	no reference, no check of the mode rule shall be done.		
Multiplicity	0..1		
Type	Reference to [DcmModeRule]		
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_00788 :**		
Name	DcmDspReadMemoryRangeSecurityLevelRef		
Description	Link to the Security Access Levels needed for read access on this memory address. If there is no reference, no check of security level shall be done.		
Multiplicity	0..*		
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

10.2.51 DcmDspWriteMemoryRangeInfo

SWS Item	**ECUC_Dcm_00789 :**
Container Name	DcmDspWriteMemoryRangeInfo
Description	Provides the range of memory address allowed for writting.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00791 :**		
Name	DcmDspWriteMemoryRangeHigh		
Description	High memory address of a range allowed for writting.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967294		
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_00790 :**		
Name	DcmDspWriteMemoryRangeLow		
Description	Low memory address of a range allowed for writting		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967294		
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_00916 :**		
Name	DcmDspWriteMemoryRangeModeRuleRef		
Description	Reference to DcmDspModeRule Mode rule which controls write access on this memory address. If there is no reference, no check of the mode rule shall be done.		
Multiplicity	0..1		
Type	Reference to [DcmModeRule]		
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_00793 :**		
Name	DcmDspWriteMemoryRangeSecurityLevelRef		
Description	Link to the Security Access Levels needed for write access on this memory address. If there is no reference, no check of security level shall be done.		
Multiplicity	0..*		
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

10.2.52 DcmDspPid

SWS Item	**ECUC_Dcm_00626 :**
Container Name	DcmDspPid
Description	This container defines the availability of a PID to the DCM.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00627 :**		
Name	DcmDspPidIdentifier		
Description	1 byte Identifier of the PID Within each DcmConfigSet all DcmDspPidIdentifier values shall be unique.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00893 :**		
Name	DcmDspPidService		
Description	Allow to indicate if this PID is used for service $01 or/and $02.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DCM_SERVICE_01	This PID is used for service $01 only.	
	DCM_SERVICE_01_02	This PID is used for service $01 and $02. Allowed with a PID configuration containing data elements on byte basis.	
	DCM_SERVICE_02	This PID is used for service $02 only. Allowed with a PID configuration containing data elements on byte basis.	
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_00870 :**		
Name	DcmDspPidSize		
Description	Length of the PID in byte.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00806 :**		
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	EcucBooleanParamDef		
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		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspPidData	1..*	This container defines the parameter for a Signal in the PID.
DcmDspPidSupportInfo	0..*	This container defines the support information (typically byte A) to declare the usability of the data bytes within the so-called bit-mapped PIDs (e.g. PID$68).

10.2.53 DcmDspPidSupportInfo

SWS Item	**ECUC_Dcm_00871 :**		
Container Name	DcmDspPidSupportInfo		
Description	This container defines the support information (typically byte A) to declare the usability of the data bytes within the so-called bit-mapped PIDs (e.g. PID$68).		
Configuration Parameters			

SWS Item	**ECUC_Dcm_00873 :**		
Name	DcmDspPidSupportInfoLen		
Description	Length of the supported information in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00872 :**		
Name	DcmDspPidSupportInfoPos		
Description	Position of the supported information in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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

10.2.54 DcmDspPidData

SWS Item	**ECUC_Dcm_00865 :**
Container Name	DcmDspPidData
Description	This container defines the parameter for a Signal in the PID.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00866 :**	
Name	DcmDspPidDataPos	
Description	This is the position in bit of the PID structure and will not start at position 0 in case a support information is availabe (for bit-mapped PIDs).	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 2040	
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_00628 :**		
Name	DcmDspPidDataSize		
Description	Length of data associated to the PID in bit.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 2040		
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***
DcmDspPidDataSupportInfo	0..1	This container defines the supported information.
DcmDspPidService01	0..1	Contains specific configuration parameter of PID for service $01. This container exists only if DcmDspPidService is set to DCM_SERVICE_01 or DCM_SERVICE_01_02.
DcmDspPidService02	0..1	Contains specific configuration parameter of PID for service $02. This container exists only if DcmDspPidService is set to DCM_SERVICE_02 or DCM_SERVICE_01_02.

10.2.55 DcmDspPidService01

SWS Item	**ECUC_Dcm_00894 :**
Container Name	DcmDspPidService01
Description	Contains specific configuration parameter of PID for service $01. This container exists only if DcmDspPidService is set to DCM_SERVICE_01 or DCM_SERVICE_01_02.
Configuration Parameters	

SWS Item	**ECUC_Dcm_01012 :**		
Name	DcmDspPidDataEndianness		
Description	Defines the endianness of the data belonging to a PID. If no DcmDspPidDataEndianness is defined the value of DcmDspDataDefaultEndianness is applicable.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN		Most significant byte shall come at the lowest address.
	LITTLE_ENDIAN		Most significant byte shall come highest address
	OPAQUE		opaque data endianness
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: ECU dependency: DcmDspDataDefaultEndianness

SWS Item	**ECUC_Dcm_00629 :**		
Name	DcmDspPidDataReadFnc		
Description	Function name for reading PID data value. This is only relevant if DcmDspPidDataUsePort==USE_DATA_SYNCH_FNC. This parameter is related to the interface Xxx_ReadData.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspPidDataUsePort		

SWS Item	**ECUC_Dcm_01018 :**		
Name	DcmDspPidDataType		
Description	Provide the implementation data type of data belonging to a PID.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BOOLEAN	type of the data is boolean.	
	SINT16	type of the data is sint16.	
	SINT32	type of the data is sint32.	
	SINT8	type of the data is sint8.	
	UINT16	type of the data is uint16.	
	UINT32	type of the data is uint32.	
	UINT8	type of the data is uint8.	
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: DcmDspDataSize		

SWS Item	**ECUC_Dcm_00720 :**		
Name	DcmDspPidDataUsePort		
Description	If this parameter is set to USE_DATA_SYNCH_FNC, the DCM will use the function defined in DcmDspPidDataReadFnc to get the PID data value. If this parameter is set to USE_DATA_SYNCH_CLIENT_SERVER, the DCM will have an R-Port requiring the interface DataServices_{Data}. If this parameter is set to USE_DATA_SENDER_RECEIVER, the DCM will have an R-Port requiring a SenderReceiverInterface The R-Port is named DataServices_{Data} where {Data} is the name of the container DcmDspPidData.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	USE_DATA_SENDER_RECEIVER	--	
	USE_DATA_SYNCH_CLIENT_SERVER	--	
	USE_DATA_SYNCH_FNC	--	
ConfigurationClass	***Pre-compile time***	X	All Variants

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

No Included Containers

10.2.56 DcmDspPidService02

SWS Item	**ECUC_Dcm_00895 :**
Container Name	DcmDspPidService02
Description	Contains specific configuration parameter of PID for service $02. This container exists only if DcmDspPidService is set to DCM_SERVICE_02 or DCM_SERVICE_01_02.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00887 :**		
Name	DcmDspPidDataDemRef		
Description	Reference to DemPidDataElement in DEM configuration. Allows to link the DCM PID and DEM PID configuration for Mode $02.		
Multiplicity	0..1		
Type	Reference to [DemPidDataElement]		
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

10.2.57 DcmDspPidDataSupportInfo

SWS Item	**ECUC_Dcm_00874 :**
Container Name	DcmDspPidDataSupportInfo
Description	This container defines the supported information.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00876 :**		
Name	DcmDspPidDataSupportInfoBit		
Description	referenced Bit of the SupportInfo		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00875 :**		
Name	DcmDspPidDataSupportInfoRef		
Description	Reference to DcmDspPidSupportInfo		
Multiplicity	1		
Type	Reference to [DcmDspPidSupportInfo]		
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

10.2.58 DcmDspRequestControl

SWS Item	**ECUC_Dcm_00637 :**
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 DCM will request the control using an R-Port requiring a PortInteface RequestControlServices_{Tid}. The R-Port is named RequestControlServices_{Tid} where {Tid} is the name of the container DcmDspRequestControl.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00722 :**		
Name	DcmDspRequestControlInBufferSize		
Description	Number of bytes to be provided in the input buffer of the interface RequestControlServices_{Tid} for OBD Service $08		
Multiplicity	1		
Type	EcucIntegerParamDef		
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	Number of bytes to be provided in the output buffer of the interface RequestControlServices_{Tid} for OBD Service $08		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00656 :**		
Name	DcmDspRequestControlTestId		
Description	Test Id for Service $08		
Multiplicity	1		
Type	EcucIntegerParamDef		
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

10.2.59 DcmDspRoe

SWS Item	**ECUC_Dcm_00858 :**
Container Name	DcmDspRoe
Description	Provide the configuration of the ResponseOnEvent mechanism.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00856 :**		
Name	DcmDspRoeInterMessageTime		
Description	Provide the minimum time in seconds between two transmissions of ROE event. It is used for the delay between two different consecutive Roe transmissions.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 5		
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: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspRoeEvent	1..*	This container contains a list of all supported Roe eventTypeRecords which are accepted by this ECU. At most one DcmDspRoeEvent container is allowed to define a DcmDspRoeEventProperties container with the choice DcmDspRoeOnDTCStatusChange.
DcmDspRoeEventWindowTime	1..*	This container configures the available EventWindowTime in this Ecu. This container contains a sub-set of EventWindowTimes supported by the Dcm, to limit the Ecu resources.

10.2.60 DcmDspRoeEvent

SWS Item	**ECUC_Dcm_00973 :**
Container Name	DcmDspRoeEvent
Description	This container contains a list of all supported Roe eventTypeRecords which are accepted by this ECU. At most one DcmDspRoeEvent container is allowed to define a DcmDspRoeEventProperties container with the choice DcmDspRoeOnDTCStatusChange.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00976 :**		
Name	DcmDspRoeEventId		
Description	EventId for a global identification of this ROE event it is used within APIs Dcm_TriggerOnEvent() and the ModeDeclarationGroup. The ratio Ids should be sequentially ordered beginning with 0 and no gaps in between.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dcm_00980 :**		
Name	DcmDspRoeInitialEventStatus		
Description	Initial Roe status of this RoeEvent		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DCM_ROE_CLEARED	-- (default)	
	DCM_ROE_STOPPED	--	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspRoeEventProperties	1	This container contains the properties of Roe eventTypeRecords. In one DcmDspRoeEventProperties container one DcmDspRoeOnDTCStatusChange or DcmDspRoeOnChangeOfDataIdentifier container shall be defined.

10.2.61 DcmDspRoeEventProperties

SWS Item	**ECUC_Dcm_00978 :**
Choice container Name	DcmDspRoeEventProperties
Description	This container contains the properties of Roe eventTypeRecords.

	In one DcmDspRoeEventProperties container one DcmDspRoeOnDTCStatusChange or DcmDspRoeOnChangeOfDataIdentifier container shall be defined.

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspRoeOnChangeOfDataIdentifier	0..1	This container contains configuration of a eventTypeRecord onChangeOfDataIdentifier accepted by this ECU.
DcmDspRoeOnDTCStatusChange	0..1	This container contains configuration of a eventTypeRecord onDTCStatusChange accepted by this ECU. Please note that currently are no additional parameters for DcmDspRoeOnDTCStatusChange are defined. Therefore the existence of the container denotes the choice.

10.2.62 DcmDspRoeOnChangeOfDataIdentifier

SWS Item	**ECUC_Dcm_00975 :**
Container Name	DcmDspRoeOnChangeOfDataIdentifier
Description	This container contains configuration of a eventTypeRecord onChangeOfDataIdentifier accepted by this ECU.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00979 :**		
Name	DcmDspRoeDidRef		
Description	Reference to a Did which is watched.		
Multiplicity	1		
Type	Reference to [DcmDspDid]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: DcmDslProtocolTransType		

No Included Containers

10.2.63 DcmDspRoeOnDTCStatusChange

SWS Item	**ECUC_Dcm_00974 :**
Container Name	DcmDspRoeOnDTCStatusChange
Description	This container contains configuration of a eventTypeRecord onDTCStatusChange accepted by this ECU. Please note that currently are no additional parameters for DcmDspRoeOnDTCStatusChange are defined. Therefore the existence of the container denotes the choice.
Configuration Parameters	

No Included Containers

10.2.64 DcmDspRoeEventWindowTime

SWS Item	**ECUC_Dcm_00981 :**
Container Name	DcmDspRoeEventWindowTime
Description	This container configures the available EventWindowTime in this Ecu. This container contains a sub-set of EventWindowTimes supported by the Dcm, to limit the Ecu resources.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00982 :**		
Name	DcmDspRoeEventWindowTime		
Description	Value of the EventWindowTime		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DCM_ROE_EVENT_WINDOW_CURRENT_AND_FOLLOWING_CYCLE	--	
	DCM_ROE_EVENT_WINDOW_CURRENT_CYCLE	--	
	DCM_ROE_EVENT_WINDOW_INFINITE	--	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dcm_00983 :**		
Name	DcmDspRoeStorageState		
Description	If this parameter is set to TRUE the StorageStateBit will be evaluated if this EventWindowTime is requested.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.65 DcmDspRoutine

SWS Item	**ECUC_Dcm_00640 :**
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) This parameter is related to the interface Xxx_RequestResults.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspRoutineUsePort		

SWS Item	**ECUC_Dcm_00899 :**		
Name	DcmDspRequestResultsRoutineSupported		
Description	Indicates if the optional requestRoutineResults in the RoutineControl is supported true = requestRoutineResults is supported false = requestRoutineResults is not supported		
Multiplicity	1		
Type	EcucBooleanParamDef		
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_00817 :**		
Name	DcmDspRoutineFixedLength		
Description	Indicates if the datalength of the optional record in the RoutineControl request and response is fixed. true = datalength of the optional record is fixed false = datalength of the optional record is variable In case DcmDspRoutineFixedLength is set to FALSE, the DcmDspRoutineSignalLength for the last signal is the maximum length (in bits) of the optional record.		
Multiplicity	1		
Type	EcucBooleanParamDef		
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_00641 :**	
Name	DcmDspRoutineIdentifier	
Description	2 bytes Identifier of the RID Within each DcmConfigSet all DcmDspRoutineIdentifier values shall be unique.	
Multiplicity	1	
Type	EcucIntegerParamDef	
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 parameter is set to true, the DCM uses a port requiring a PortInterface RoutineServices_{RoutineName}. The R-Port is named RoutineServices_{RoutineName} where {RoutineName} is the name of the container DcmDspRoutine 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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00807 :**		
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	EcucBooleanParamDef		
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_00664 :**		
Name	DcmDspStartRoutineFnc		
Description	Function name for request to application to start a routine. (Routine_Start-function) This parameter is related to the interface Xxx_Start.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspRoutineUsePort		

SWS Item	**ECUC_Dcm_00752 :**		
Name	DcmDspStopRoutineFnc		
Description	Function name for request to application to stop a routine. (Routine_Stop-function) This parameter is related to the interface Xxx_Stop.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: DcmDspRoutineUsePort		

SWS Item	**ECUC_Dcm_00898 :**		
Name	DcmDspStopRoutineSupported		
Description	Indicates if the optional stopRoutine in the RoutineControl is supported true = stopRoutine is supported false = stopRoutine is not supported		
Multiplicity	1		
Type	EcucBooleanParamDef		
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_00642 :**		
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

10.2.66 DcmDspRoutineInfo

SWS Item	**ECUC_Dcm_00643 :**
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.
DcmDspRoutineRequestResOut	0..1	Provide description of output parameter of RequestResult subservice for RoutineControl service.
DcmDspRoutineStopIn	0..1	Provide description of input parameter of Stop subservice for RoutineControl service.
DcmDspRoutineStopOut	0..1	Provide description of output parameter of Stop subservice for RoutineControl service.
DcmDspStartRoutineIn	0..1	Provide description of input parameter of Start subservice for RoutineControl service
DcmDspStartRoutineOut	0..1	Provide description of output parameter of Start subservice for RoutineControl service.

10.2.67 DcmDspRoutineAuthorization

SWS Item	**ECUC_Dcm_00644 :**
Container Name	DcmDspRoutineAuthorization
Description	This container contains the configuration (parameters) for the Routine Authorization.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00915 :**		
Name	DcmDspRoutineModeRuleRef		
Description	Reference to DcmDspModeRule Mode rule which controls this RID. If there is no reference, no check of the mode rule shall be done.		
Multiplicity	0..1		
Type	Reference to [DcmModeRule]		
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_00648 :**		
Name	DcmDspRoutineSecurityLevelRef		
Description	Reference to DcmDspSecurityRow Security levels allowed to control this RID. If there is no reference, no check of security level shall be done.		
Multiplicity	0..*		
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_00649 :**
Name	DcmDspRoutineSessionRef
Description	Reference to DcmDspSessionRow Sessions allowed to control this RID. If there is no reference, no check of session level shall be done.
Multiplicity	0..*

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

10.2.68 DcmDspRoutineRequestResOut

SWS Item	**ECUC_Dcm_00831 :**
Container Name	DcmDspRoutineRequestResOut
Description	Provide description of output parameter of RequestResult subservice for RoutineControl service.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspRoutineRequestResOutSignal	1..*	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataOutN elements in the XXX_RequestResult function call.

10.2.69 DcmDspRoutineRequestResOutSignal

SWS Item	**ECUC_Dcm_00836 :**
Container Name	DcmDspRoutineRequestResOutSignal
Description	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataOutN elements in the XXX_RequestResult function call.
Configuration Parameters	

SWS Item	**ECUC_Dcm_01013 :**		
Name	DcmDspRoutineSignalEndianness		
Description	Defines the endianness of the data belonging to a Request Routine Response Out Signal. If no DcmDspRoutineSignalEndianness is defined the value of DcmDspDataDefaultEndianness is applicable.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN		Most significant byte shall come at the lowest address.
	LITTLE_ENDIAN		Most significant byte shall come highest address
	OPAQUE		opaque data endianness
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: DcmDspDataDefaultEndianness		

SWS Item	**ECUC_Dcm_00838 :**		
Name	DcmDspRoutineSignalLength		
Description	Provide the length in bits of the signal in the RoutineControl request/response		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
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_00837 :**		
Name	DcmDspRoutineSignalPos		
Description	Provide the position of the signal in the RoutineControl request/response. The position is defined in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00881 :**		
Name	DcmDspRoutineSignalType		
Description	Provide the type of the signal in the RoutineControl request/response.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BOOLEAN	type of the signal is boolean.	
	SINT16	type of the signal is sint16.	
	SINT32	type of the signal is sint32.	
	SINT8	type of the signal is sint8.	
	UINT16	type of the signal is uint16.	
	UINT32	type of the signal is uint32.	
	UINT8	type of the signal is uint8.	
	VARIABLE_LENGTH	type of the signal is uint8[(DcmDspRoutineSignalLength+7)/8]. This is only valid for the last signal and when DcmDspRoutineFixedLength is set to FALSE.	
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: DcmDspRoutineSignalLength		

No Included Containers

10.2.70 DcmDspRoutineStopIn

SWS Item	**ECUC_Dcm_00832 :**
Container Name	DcmDspRoutineStopIn
Description	Provide description of input parameter of Stop subservice for RoutineControl service.
Configuration Parameters	

Included Containers		
Container Name	*Multiplicity*	*Scope / Dependency*
DcmDspRoutineStopInSignal	1..*	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataInN elements in the XXX_Stop function call.

10.2.71 DcmDspRoutineStopInSignal

SWS Item	**ECUC_Dcm_00839 :**
Container Name	DcmDspRoutineStopInSignal
Description	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataInN elements in the XXX_Stop function call.
Configuration Parameters	

SWS Item	**ECUC_Dcm_01014 :**		
Name	DcmDspRoutineSignalEndianness		
Description	Defines the endianness of the data belonging to a Stop Routine In Signal. If no DcmDspRoutineSignalEndianness is defined the value of DcmDspDataDefaultEndianness is applicable.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN		Most significant byte shall come at the lowest address.
	LITTLE_ENDIAN		Most significant byte shall come highest address
	OPAQUE		opaque data endianness
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: DcmDspDataDefaultEndianness		

SWS Item	**ECUC_Dcm_00841 :**
Name	DcmDspRoutineSignalLength

Description	Provide the length in bits of the signal in the RoutineControl request/response		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
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_00840 :**		
Name	DcmDspRoutineSignalPos		
Description	Provide the position of the signal in the RoutineControl request/response. The position is defined in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00882 :**		
Name	DcmDspRoutineSignalType		
Description	Provide the type of the signal in the RoutineControl request/response.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BOOLEAN	type of the signal is boolean.	
	SINT16	type of the signal is sint16.	
	SINT32	type of the signal is sint32.	
	SINT8	type of the signal is sint8.	
	UINT16	type of the signal is uint16.	
	UINT32	type of the signal is uint32.	
	UINT8	type of the signal is uint8.	
	VARIABLE_LENGTH	type of the signal is uint8[(DcmDspRoutineSignalLength+7)/8]. This is only valid for the last signal and when DcmDspRoutineFixedLength is set to FALSE.	
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: DcmDspRoutineSignalLength		

No Included Containers

10.2.72 DcmDspRoutineStopOut

SWS Item	**ECUC_Dcm_00833 :**
Container Name	DcmDspRoutineStopOut
Description	Provide description of output parameter of Stop subservice for RoutineControl service.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspRoutineStopOutSignal	1..*	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataOutN elements in the XXX_Stop function call.

10.2.73 DcmDspRoutineStopOutSignal

SWS Item	**ECUC_Dcm_00842 :**
Container Name	DcmDspRoutineStopOutSignal
Description	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataOutN elements in the XXX_Stop function call.
Configuration Parameters	

SWS Item	**ECUC_Dcm_01015 :**		
Name	DcmDspRoutineSignalEndianness		
Description	Defines the endianness of the data belonging to a Stop Routine Out Signal. If no DcmDspRoutineSignalEndianness is defined the value of DcmDspDataDefaultEndianness is applicable.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN	Most significant byte shall come at the lowest address.	
	LITTLE_ENDIAN	Most significant byte shall come highest address	
	OPAQUE	opaque data endianness	
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: DcmDspDataDefaultEndianness		

SWS Item	**ECUC_Dcm_00844 :**	
Name	DcmDspRoutineSignalLength	
Description	Provide the length in bits of the signal in the RoutineControl request/response	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
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_00843 :**		
Name	DcmDspRoutineSignalPos		
Description	Provide the position of the signal in the RoutineControl request/response. The position is defined in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00883 :**		
Name	DcmDspRoutineSignalType		
Description	Provide the type of the signal in the RoutineControl request/response.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BOOLEAN	type of the signal is boolean.	
	SINT16	type of the signal is sint16.	
	SINT32	type of the signal is sint32.	
	SINT8	type of the signal is sint8.	
	UINT16	type of the signal is uint16.	
	UINT32	type of the signal is uint32.	
	UINT8	type of the signal is uint8.	
	VARIABLE_LENGTH	type of the signal is uint8[(DcmDspRoutineSignalLength+7)/8]. This is only valid for the last signal and when DcmDspRoutineFixedLength is set to FALSE.	
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: DcmDspRoutineSignalLength		

No Included Containers

10.2.74 DcmDspStartRoutineIn

SWS Item	**ECUC_Dcm_00834 :**
Container Name	DcmDspStartRoutineIn
Description	Provide description of input parameter of Start subservice for RoutineControl service
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspStartRoutineInSignal	1..*	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataInN elements in the XXX_Start function call.

10.2.75 DcmDspStartRoutineInSignal

SWS Item	**ECUC_Dcm_00845 :**
Container Name	DcmDspStartRoutineInSignal
Description	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataInN elements in the XXX_Start function call.
Configuration Parameters	

SWS Item	**ECUC_Dcm_01016 :**		
Name	DcmDspRoutineSignalEndianness		
Description	Defines the endianness of the data belonging to a Start Routine In Signal. If no DcmDspRoutineSignalEndianness is defined the value of DcmDspDataDefaultEndianness is applicable.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN		Most significant byte shall come at the lowest address.
	LITTLE_ENDIAN		Most significant byte shall come highest address
	OPAQUE		opaque data endianness
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: DcmDspDataDefaultEndianness		

SWS Item	**ECUC_Dcm_00847 :**		
Name	DcmDspRoutineSignalLength		
Description	Provide the length in bits of the signal in the RoutineControl request/response		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
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_00846 :**
Name	DcmDspRoutineSignalPos

Description	Provide the position of the signal in the RoutineControl request/response. The position is defined in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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_00884 :**		
Name	DcmDspRoutineSignalType		
Description	Provide the type of the signal in the RoutineControl request/response.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BOOLEAN	type of the signal is boolean.	
	SINT16	type of the signal is sint16.	
	SINT32	type of the signal is sint32.	
	SINT8	type of the signal is sint8.	
	UINT16	type of the signal is uint16.	
	UINT32	type of the signal is uint32.	
	UINT8	type of the signal is uint8.	
	VARIABLE_LENGTH	type of the signal is uint8[(DcmDspRoutineSignalLength+7)/8]. This is only valid for the last signal and when DcmDspRoutineFixedLength is set to FALSE.	
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: DcmDspRoutineSignalLength, DcmDspRoutineFixedLength		

No Included Containers

10.2.76 DcmDspStartRoutineOut

SWS Item	**ECUC_Dcm_00835 :**
Container Name	DcmDspStartRoutineOut
Description	Provide description of output parameter of Start subservice for RoutineControl service.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmDspStartRoutineOutSignal	1..*	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataOutN elements in the XXX_Start function call.

10.2.77 DcmDspStartRoutineOutSignal

SWS Item	**ECUC_Dcm_00848 :**
Container Name	DcmDspStartRoutineOutSignal
Description	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataOutN elements in the XXX_Start function call.
Configuration Parameters	

SWS Item	**ECUC_Dcm_01017 :**		
Name	DcmDspRoutineSignalEndianness		
Description	Defines the endianness of the data belonging to a Start Routine Out Signal. If no DcmDspRoutineSignalEndianness is defined the value of DcmDspDataDefaultEndianness is applicable.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN		Most significant byte shall come at the lowest address.
	LITTLE_ENDIAN		Most significant byte shall come highest address
	OPAQUE		opaque data endianness
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: DcmDspDataDefaultEndianness		

SWS Item	**ECUC_Dcm_00850 :**		
Name	DcmDspRoutineSignalLength		
Description	Provide the length in bits of the signal in the RoutineControl request/response.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
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_00867 :**		
Name	DcmDspRoutineSignalPos		
Description	Provide the position of the signal in the RoutineControl request/response. The position is defined in bits.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
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_00885 :**		
Name	DcmDspRoutineSignalType		
Description	Provide the type of the signal in the RoutineControl request/response.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BOOLEAN	type of the signal is boolean.	
	SINT16	type of the signal is sint16.	
	SINT32	type of the signal is sint32.	
	SINT8	type of the signal is sint8.	
	UINT16	type of the signal is uint16.	
	UINT32	type of the signal is uint32.	
	UINT8	type of the signal is uint8.	
	VARIABLE_LENGTH	type of the signal is uint8[(DcmDspRoutineSignalLength+7)/8]. This is only valid for the last signal and when DcmDspRoutineFixedLength is set to FALSE.	
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: DcmDspRoutineSignalLength		

No Included Containers

10.2.78 DcmDspSecurity

SWS Item	**ECUC_Dcm_00764 :**
Container Name	DcmDspSecurity
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 configuration (per security level) The name of this container is used to define the name of the R-Port through which the DCM accesses the interface SecurityAccess_{SecurityLevel}. The R-Port is named SecurityAccess_{SecurityLevel} where {SecurityLevel} is the name of the container DcmDspSecurityRow. If there is no reference, no check of security level shall be done.

10.2.79 DcmDspSecurityRow

SWS Item	**ECUC_Dcm_00759 :**		
Container Name	DcmDspSecurityRow		
Description	Definition of a single Row of configuration for security level configuration (per security level) The name of this container is used to define the name of the R-Port through which the DCM accesses the interface SecurityAccess_{SecurityLevel}. The R-Port is named SecurityAccess_{SecurityLevel} where {SecurityLevel} is the name of the container DcmDspSecurityRow. If there is no reference, no check of security level shall be done.		
Configuration Parameters			

SWS Item	**ECUC_Dcm_00725 :**		
Name	DcmDspSecurityADRSize		
Description	Size in bytes of the AccessDataRecord used in GetSeed		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
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: local		

SWS Item	**ECUC_Dcm_00969 :**		
Name	DcmDspSecurityCompareKeyFnc		
Description	Function name to request the result of a key comparison. Parameter is only relevant if DcmDspSecurityUsePort=="USE_ SYNCH_FNCâ€• or DcmDspSecurityUsePort==â€• USE_ ASYNCH_FNC". This parameter is related to the interface Xxx_CompareKey.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: local dependency: DcmDspSecurityUsePort		

SWS Item	**ECUC_Dcm_00757 :**		
Name	DcmDspSecurityDelayTime		
Description	Delay time after failed security access in seconds. This is started after DcmDspSecurityNumAttDelay number of failed security accesses. min: A negative value is not allowed.		
Multiplicity	1		
Type	EcucFloatParamDef		
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: local		

SWS Item	**ECUC_Dcm_00726 :**		
Name	DcmDspSecurityDelayTimeOnBoot		
Description	Start delay timer on power on in seconds. This delay indicates the time at ECU boot power-on time where the Dcm remains in the default session and does not accept a security access. min: A negative value is not allowed.		
Multiplicity	1		
Type	EcucFloatParamDef		
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: local		

SWS Item	**ECUC_Dcm_00968 :**		
Name	DcmDspSecurityGetSeedFnc		
Description	Function name to request a seed. Parameter is only relevant if DcmDspSecurityUsePort=="USE_SYNCH_FNCâ€• or DcmDspSecurityUsePort==â€• USE_ASYNCH_FNC". This parameter is related to the interface Xxx_GetSeed.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: local dependency: DcmDspSecurityUsePort		

SWS Item	**ECUC_Dcm_00760 :**		
Name	DcmDspSecurityKeySize		
Description	size of the security key (in Bytes).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
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: local		

SWS Item	**ECUC_Dcm_00754 :**

Name	DcmDspSecurityLevel		
Description	Value of Security level. 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 Type: Dcm_SecLevelType		
Multiplicity	1		
Type	EcucIntegerParamDef (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: local		

SWS Item	**ECUC_Dcm_00762 :**		
Name	DcmDspSecurityNumAttDelay		
Description	Number of failed security accesses after which the delay time is activated		
Multiplicity	1		
Type	EcucIntegerParamDef		
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: local dependency: DcmDspSecurityDelayTime		

SWS Item	**ECUC_Dcm_00755 :**		
Name	DcmDspSecuritySeedSize		
Description	size of the security seed (in Bytes).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
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: local		

SWS Item	**ECUC_Dcm_00967 :**	
Name	DcmDspSecurityUsePort	
Description	Defines which kind of interface shall be used for security access.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	USE_ASYNCH_CLIENT_SERVER	The DCM will access the data using an R-Port requiring a asynchronous ClientServertInterface SecurityAccess_{SecurityLevel}. The R-Port is described in DcmDspSecurityRow description.

	USE_ASYNCH_FNC	The DCM will access the data using the functions that are defined in the parameters DcmDspSecurityGetSeedFnc and DcmDspSecurityCompareKeyFnc. DCM_E_PENDING return is allowed and OpStatus is existing as IN parameter.	
	USE_SYNCH_CLIENT_SERVER	The DCM will access the data using an R-Port requiring a synchronous ClientServertInterface SecurityAccess_{SecurityLevel}. The R-Port is described in DcmDspSecurityRow description.	
	USE_SYNCH_FNC	The DCM will access the data using the functions that are defined in the parameters DcmDspSecurityGetSeedFnc and DcmDspSecurityCompareKeyFnc. DCM_E_PENDING return value is not allowed and OpStatus parameter is not existing in the prototype.	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.80 DcmDspSession

SWS Item	**ECUC_Dcm_00769 :**
Container Name	DcmDspSession
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)

10.2.81 DcmDspSessionRow

SWS Item	**ECUC_Dcm_00767 :**
Container Name	DcmDspSessionRow
Description	Definition of a single Row of session control configuration (per session control)
Configuration Parameters	

SWS Item	**ECUC_Dcm_00815 :**

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	EcucEnumerationParamDef		
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: local		

SWS Item	**ECUC_Dcm_00765 :**		
Name	DcmDspSessionLevel		
Description	subFunction value of the DiagnosticSession. 0, 127 and all values above 127 are reserved by ISO		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	1 .. 126		
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: local		

SWS Item	**ECUC_Dcm_00766 :**		
Name	DcmDspSessionP2ServerMax		
Description	This is the session value for P2ServerMax in seconds (per Session Control). 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	EcucFloatParamDef		
Range	0 .. 1		
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: local		

SWS Item	**ECUC_Dcm_00768 :**
Name	DcmDspSessionP2StarServerMax
Description	This is the session value for P2*ServerMax in seconds (per Session Control). 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	EcucFloatParamDef		
Range	0 .. 100		
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: local		

No Included Containers

10.2.82 DcmDspVehInfo

SWS Item	**ECUC_Dcm_00630 :**
Container Name	DcmDspVehInfo
Description	This container contains the configuration (parameters) of the "Request vehicle information service" (service $09).
Configuration Parameters	

SWS Item	**ECUC_Dcm_00631 :**		
Name	DcmDspVehInfoInfoType		
Description	value of InfoType. Within each DcmConfigSet all DcmDspVehInfoInfoType values shall be unique.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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***
DcmDspVehInfoData	1..*	Data Item of an InfoType; post-fix of the port interface name.

10.2.83 DcmDspVehInfoData

SWS Item	**ECUC_Dcm_00888 :**
Container Name	DcmDspVehInfoData
Description	Data Item of an InfoType; post-fix of the port interface name.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00891 :**

Name	DcmDspVehInfoDataOrder		
Description	Defines the order of the data item in the InfoType; values: 0..255; first data item having the order number 0; the next 1 and so on. The configuration of order needs to be unique per InfoType.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00889 :**		
Name	DcmDspVehInfoDataReadFnc		
Description	Function name for reading InfoType data item.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00890 :**		
Name	DcmDspVehInfoDataSize		
Description	Size in bytes of the InfoType data item.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00727 :**		
Name	DcmDspVehInfoDataUsePort		
Description	When this parameter is set to true the DCM will access the Data using an R-Port requiring a PortInteface IInfotypeServices_{VehInfoData}. The R-Port is named InfotypeServices_{VehInfoData} where {VEHINFODATA} is the name of the container DcmDspVehInfoData. In that case, the DcmDspVehInfoDataReadFnc is ignored and the RTE APIs are used. When this parameter is set to false, the DCM calls the function defined in DcmDspVehInfoDataReadFnc.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.84 DcmDspPeriodicTransmission

SWS Item	**ECUC_Dcm_00957 :**
Container Name	DcmDspPeriodicTransmission
Description	This container contains the configuration (parameters) for Periodic Transmission Processing.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00960 :**		
Name	DcmDspPeriodicTransmissionFastRate		
Description	This parameter give the transmission rate of the requested periodicDataIdentifiers to be used if the parameter transmissionMode given in the ReadDataByPeriodicID request is equal to 0x03 ("sendAtFastRate"). This parameter value in seconds have to be configured as a multiple of DcmTaskTime. min: A negative value and zero is not allowed.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	1E-4 .. 0.255		
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: local		

SWS Item	**ECUC_Dcm_00959 :**		
Name	DcmDspPeriodicTransmissionMediumRate		
Description	This parameter give the transmission rate of the requested periodicDataIdentifiers to be used if the parameter transmissionMode given in the ReadDataByPeriodicID request is equal to 0x02 ("sendAtMediumRate"). This parameter value in seconds have to be configured as a multiple of DcmTaskTime. min: A negative value and zero is not allowed.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	1E-4 .. 0.255		
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: local		

SWS Item	**ECUC_Dcm_00958 :**
Name	DcmDspPeriodicTransmissionSlowRate
Description	This parameter give the transmission rate of the requested periodicDataIdentifiers to be used if the parameter transmissionMode given in the ReadDataByPeriodicID request is equal to 0x01 ("sendAtSlowRate"). This parameter value in seconds have to be configured as a multiple of DcmTaskTime. min: A negative value and zero is not allowed.

Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	1E-4 .. 0.255		
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: local		

No Included Containers

10.2.85 DcmDspPeriodicDidTransmission

SWS Item	**ECUC_Dcm_00961 :**
Container Name	DcmDspPeriodicDidTransmission
Description	This container contains the configuration for the Periodic Did transmission. This container exists only if the UDS Service ReadDataByPeriodicIdentifier(0x2A) is configured.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00962 :**		
Name	DcmDspMaxPeriodicDidScheduler		
Description	Defines the maximum number of periodicDataIdentifiers that can be scheduled concurrently.		
Multiplicity	1		
Type	EcucIntegerParamDef		
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: local		

No Included Containers

10.2.86 DcmGeneral

SWS Item	**ECUC_Dcm_00822 :**
Container Name	DcmGeneral
Description	This container contains the configuration (parameters) for Component wide parameters
Configuration Parameters	

SWS Item	**ECUC_Dcm_00971 :**
Name	DcmDDDIDStorage
Description	This configuration switch defines, whether DDDID definition is stored non-volatile or not. true: DDDID are stored non-volatile false: DDDID are only maintained

	volatile		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dcm_00823 :**		
Name	DcmDevErrorDetect		
Description	Preprocessor switch to enable or disable the Development Error Detection (DET) mechanism.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dcm_00987 :**		
Name	DcmDspDataDefaultEndianness		
Description	Defines the default endianness of the data belonging to a DID which is applicable if the DcmDspData does not define a endianness.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN		Most significant byte shall come at the lowest address.
	LITTLE_ENDIAN		Most significant byte shall come highest address
	OPAQUE		opaque data endianness
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dcm_01019 :**		
Name	DcmHeaderFileInclusion		
Description	Name of the header file(s) to be included by the Dcm module containing the used C-callback declarations.		
Multiplicity	0..*		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	[a-zA-Z0-9_]([a-zA-Z0-9\._])*		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00600 :**
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00820 :**		
Name	DcmTaskTime		
Description	Allow to configure the time for the periodic cyclic task. 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 and zero is not allowed. upperMultiplicity: Exactly one TaskTime must be specified per configuration. lowerMultiplicity: Exactly one TaskTime must be specified per configuration.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	1E-4 .. 0.1		
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: local		

SWS Item	**ECUC_Dcm_00821 :**		
Name	DcmVersionInfoApi		
Description	Preprocessor switch to enable or disable the output Version info of the functionality.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dcm_00984 :**		
Name	DcmVinRef		
Description	Reference to the Did containing the VIN Information. This parameter is needed for function Dcm_GetVin		
Multiplicity	0..1		
Type	Reference to [DcmDspDid]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

	dependency: Dcm_GetVin

No Included Containers

10.2.87 DcmPageBufferCfg

SWS Item	**ECUC_Dcm_00775 :**
Container Name	DcmPageBufferCfg
Description	This container contains the configuration (parameters) for Page Buffer handling
Configuration Parameters	

SWS Item	**ECUC_Dcm_00776 :**		
Name	DcmPagedBufferEnabled		
Description	Allow to enable or disable the Page buffer mechanism. true = Page buffer handling enabled false = Page Buffer handling disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00774 :**		
Name	DcmPagedBufferTimeout		
Description	Allow to configure the Timeout in seconds towards the application for filling the next page. This parameter is only relevant if the Page Buffer handling is enabled. (DcmPagedBufferEnabled = TRUE) 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	EcucFloatParamDef		
Range	0 .. 1		
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

10.2.88 DcmProcessingConditions

SWS Item	**ECUC_Dcm_00932 :**
Container Name	DcmProcessingConditions
Description	This container contains the configuration (DSP parameter) for mode arbitration functionality of the Dcm
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DcmModeCondition	1..*	This container contains the configuration of a mode condition which can be used as argument in DcmModeRules. One DcmModeCondition shall contain either one DcmSwcModeRef or one DcmBswModeRef. Please note that the Dcm acts as well as mode manager. Therefore the references DcmSwcModeRef or one DcmBswModeRef. might point to provided ModeDeclarationGroupPrototypes of the Dcm itself as well as to provided ModeDeclarationGroupPrototypes of other Bsw Modules or software components.
DcmModeRule	1..*	This container contains the configuration of a mode rule which represents a logical expression with DcmModeCondistions or other DcmModeRules as arguments. All arguments are processed with the operator defined by DcmLogicalOperator, for instance: Argument_A AND Argument_B AND Argument_C

10.2.89 DcmModeCondition

SWS Item	**ECUC_Dcm_00928 :**
Container Name	DcmModeCondition
Description	This container contains the configuration of a mode condition which can be used as argument in DcmModeRules. One DcmModeCondition shall contain either one DcmSwcModeRef or one DcmBswModeRef. Please note that the Dcm acts as well as mode manager. Therefore the references DcmSwcModeRef or one DcmBswModeRef. might point to provided ModeDeclarationGroupPrototypes of the Dcm itself as well as to provided ModeDeclarationGroupPrototypes of other Bsw Modules or software components.
Configuration Parameters	

SWS Item	**ECUC_Dcm_00929 :**		
Name	DcmConditionType		
Description	This parameter specifies what kind of comparison that is made for the evaluation of the mode condition.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DCM_EQUALS	--	
	DCM_EQUALS_NOT	--	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dcm_00931 :**		
Name	DcmBswModeRef		
Description	This parameter references a mode of a ModeDeclarationGroupPrototype provided by a Basic Software Module used for the condition. Please note that such ModeDeclarationGroupPrototype are owned by a Basic Software Module Description in the role providedModeGroup.		
Multiplicity	0..1		
Type	Instance reference to [MODE-DECLARATION context: MODE-DECLARATION-GROUP-PROTOTYPE]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dcm_00930 :**		
Name	DcmSwcModeRef		
Description	This parameter references a mode in a particular mode request port of a software component that is used for the condition.		
Multiplicity	0..1		
Type	Instance reference to [MODE-DECLARATION context: ROOT-SW-COMPOSITION-PROTOTYPE SW-COMPONENT-PROTOTYPE P-PORT-PROTOTYPE MODE-DECLARATION-GROUP-PROTOTYPE]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.90 DcmModeRule

SWS Item	**ECUC_Dcm_00925 :**
Container Name	DcmModeRule
Description	This container contains the configuration of a mode rule which represents a logical expression with DcmModeCondistions or other DcmModeRules as arguments. All arguments are processed with the operator defined by DcmLogicalOperator, for instance: Argument_A AND Argument_B AND Argument_C
Configuration Parameters	

SWS Item	**ECUC_Dcm_00926 :**		
Name	DcmLogicalOperator		
Description	This parameter specifies the logical operator to be used in the logical expression. If the expression only consists of a single condition this parameter shall not be used.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	DCM_AND	--	
	DCM_OR	--	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

SWS Item	**ECUC_Dcm_00949 :**		
Name	DcmModeRuleNrcValue		
Description	Optional parameter which defines the NRC to be sent in case the mode rule condition is not valid.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dcm_00927 :**		
Name	DcmArgumentRef		
Description	This is a choice reference either to a mode condition or a an other mode rule serving as sub-expression. **Attributes:** requiresIndex=true		
Multiplicity	1..*		
Type	Choice reference to [DcmModeCondition , DcmModeRule]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.3 Protocol Configuration Example

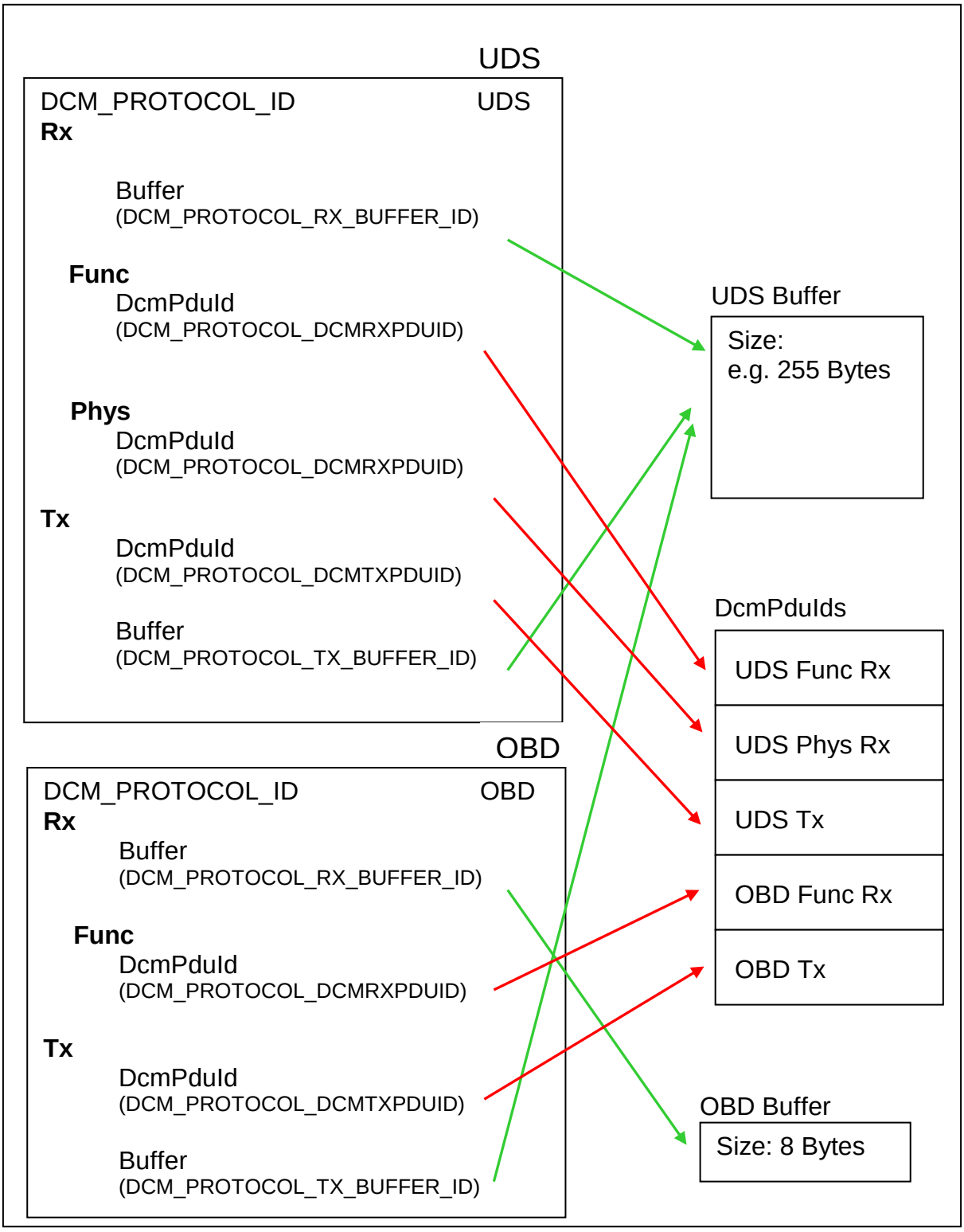


Figure 10 Examples of protocol configuration with focus on buffer / DcmPduId settings

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

For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*

10.5 Configuration constraints

[constr_6000] Harmonize the naming between interfaces and modes「The shortname of DcmDspSessionRow shall match names of Dcm_SesCtrlType and of the mode declarations of DcmDiagnosticSessionControl (excluding AR-defined prefixes).」()

[constr_6001] Provide standardized names for ISO standardized diagnostic sessions「The following values of ***DcmDspSessionLevel*** which represent ISO defined diagnostic sessions shall be used for the shortname of ***DcmDspSessionRow***:

1 DEFAULT_SESSION

2 PROGRAMMING_SESSION

3 EXTENDED_DIAGNOSTIC_SESSION

4 SAFETY_SYSTEM_DIAGNOSTIC_SESSION.」()

11 Not applicable requirements

[SWS_Dcm_00999] [These requirements are not applicable to this specification. **]** (BSW159, BSW170, BSW00383, BSW00387, BSW00375, BSW00416, BSW00406, BSW00437, BSW168, BSW00423, BSW00425, BSW00426, BSW00427, BSW00428, BSW00429, BSW00432, BSW00433, BSW00336, BSW00339, BSW00422, BSW00417, BSW00409, BSW00385, BSW00386, BSW00455, BSW161, BSW162, BSW005, BSW00415, BSW164, BSW00326, BSW00342, BSW00453, BSW00413, BSW00347, BSW00307, BSW00314, BSW00447, BSW00361, BSW00328, BSW00439, BSW00378, BSW00440, BSW00443, BSW00444, BSW00445, BSW00446, BSW172, BSW010, BSW00321, BSW00341, BSW00334, BSW)