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

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

Document Change History

Date	Version	Changed by	Change Description
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
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

Document Change History

Date	Version	Changed by	Change Description
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
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 [15] 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 limited to the resolution of the MainFunction call.
- 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

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.

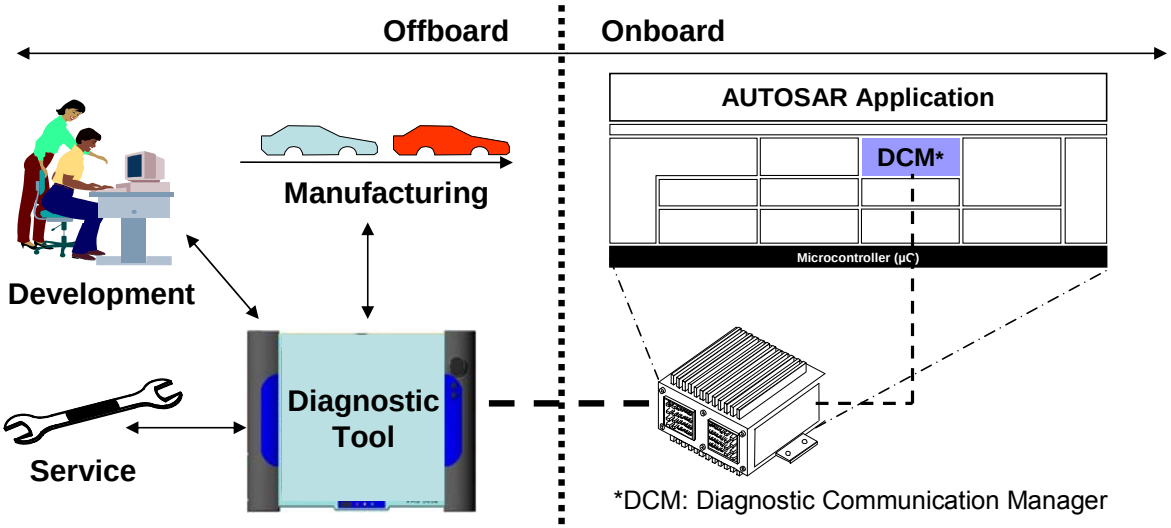


Figure 1 Overview of the communication between the external diagnostic tools and the onboard AUTOSAR Application

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 [14] services. In addition, the DCM module provides mechanisms to support the OBD services $01 - $0A defined in documents [19] and [15]. 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 [17]: Implementation of unified diagnostic services (UDS on CAN)
- ISO15765-4 [18]: 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).

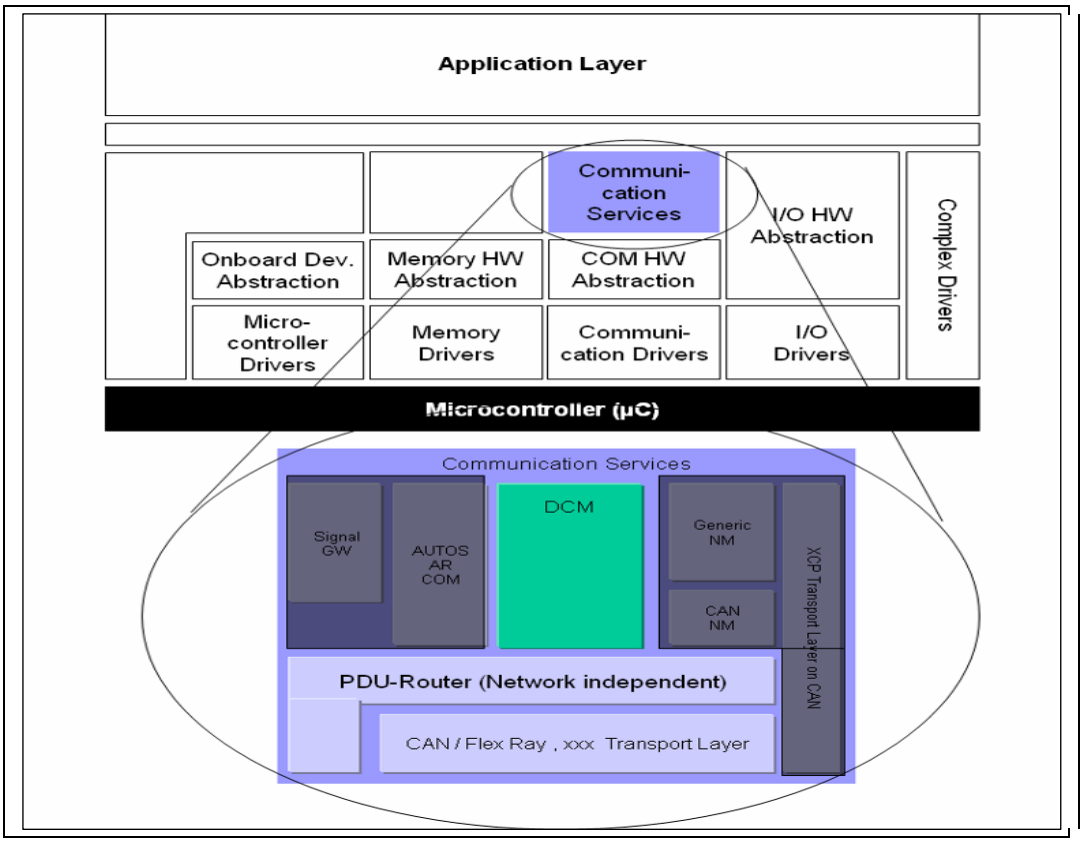


Figure 2 Position of the DCM module in AUTOSAR Architecture

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 [14])

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

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

3.2 Related standards and norms

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

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:

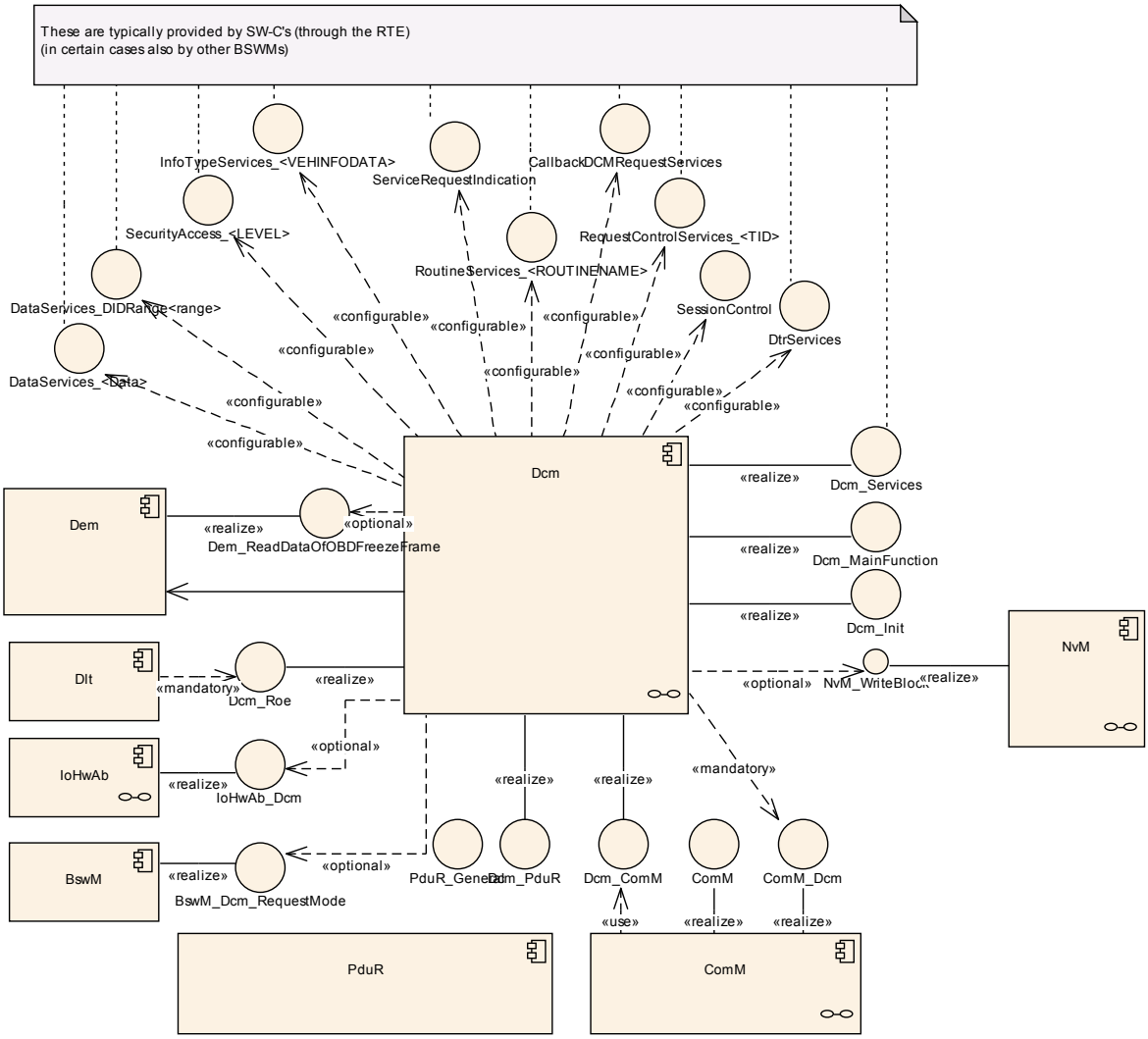


Figure 3 Interaction of the DCM with other modules

- 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 [14] 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 BswM allows the DCM to request a mode change, e.g. in case of reset or session change

5.1 File structure

5.1.1 Code file structure

[Dcm054]「 The code file structure shall not be defined within the DCM SWS completely. At this point it shall be pointed out that the code-file structure shall include the following files named:

- If single pre-compile const parameter file is implemented: Dcm_Cfg.c – for pre-compile const parameter
- If multiple pre-compile const parameter file is implemented: Dcm<*Implementation specific name*>_Cfg.c – for pre-compile const parameter
- If single link time configurable parameters file is implemented: Dcm_Lcfg.c – for link time configurable parameters
- If multiple link time configurable parameter file is implemented: Dcm<*Implementation specific name*>_Lcfg.c – for link time configurable parameters
- If single post build time configurable parameters file is implemented: Dcm_PBcfg.c – for post build time configurable parameters.
- If multiple post build time configurable parameter file is implemented: Dcm<*Implementation specific name*>_PBcfg.c – for post build time configurable parameters

These files shall contain all link time and post-build time configurable parameters.」(BSW00380, BSW00419, BSW00346, BSW158, BSW00370, BSW00301)

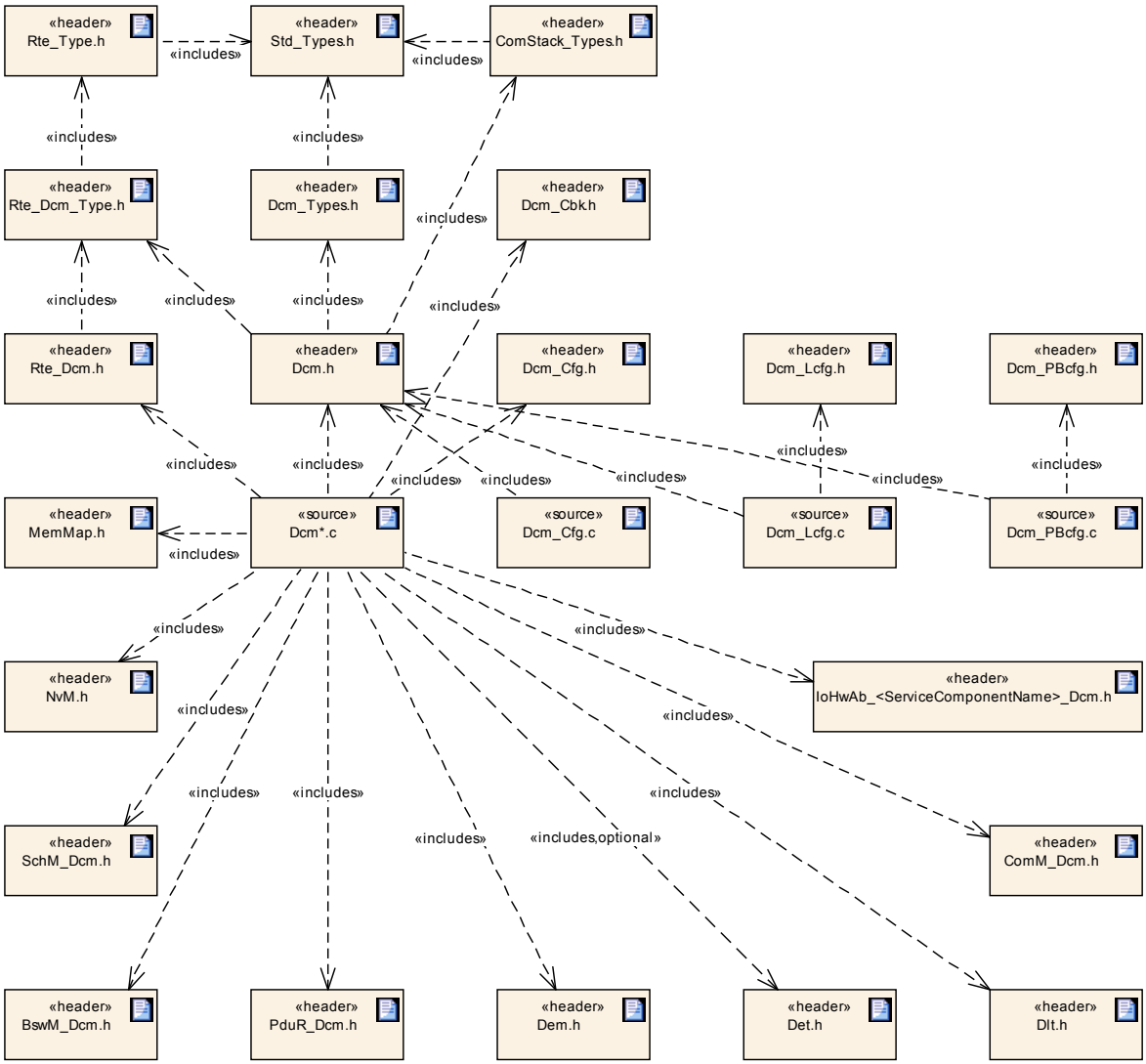


Figure 4: DCM module file structure

5.1.2 Header file structure

[Dcm055] ⌈ The DCM module shall use the header file structure shown in Figure 5 ⌋(BSW00381, BSW00412, BSW00435, BSW00436, BSW00302)

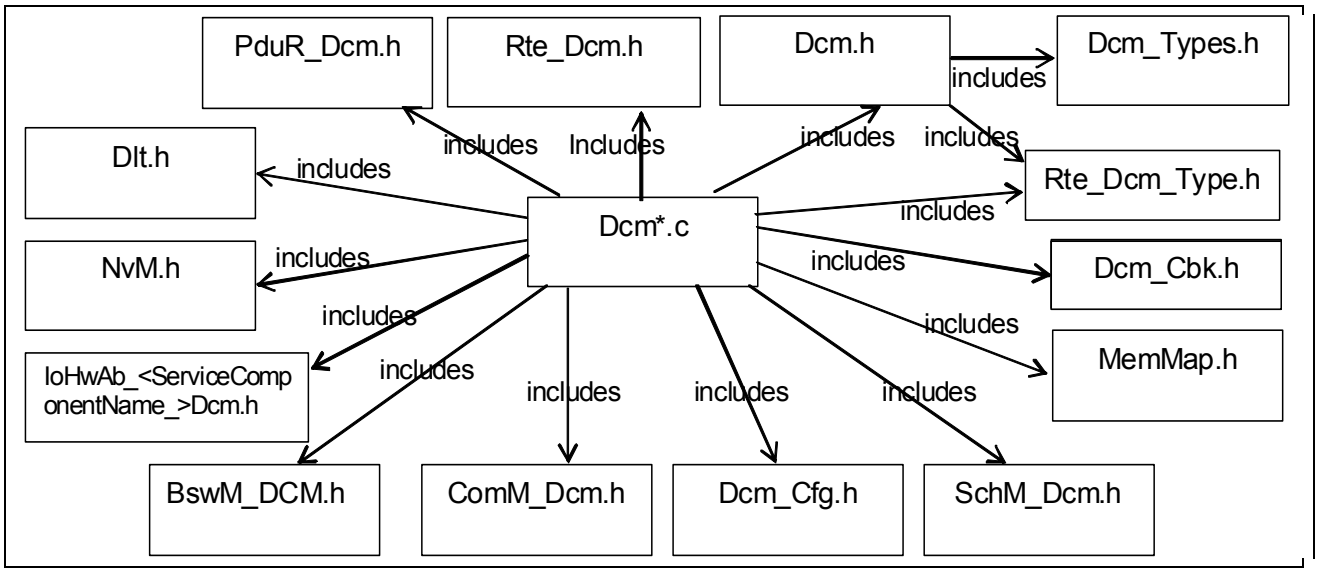


Figure 5: DCM module header file structure

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

[Dcm107]「 The implementation may be contained in one or multiple Dcm*.c files.」()

[**Dcm683**]「 The file Dcm_Types.h include all DCM type definition」()

[Dcm108]「 Any callbacks supplied by a SW-C shall be located in Rte_Dcm.h.」()

Dcm750「The file Dcm.h shall include Rte_Dcm_Type.h to include the types which are common used by BSW Modules and Software Components. This file shall only contain types, that are not already defined in Rte_Dcm_Type.h.」()

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

[Dcm109]「 Dcm.c shall include SchM_Dcm.h.」()

[Dcm332]「 The DCM module shall include the file Dem.h.」()

[Dcm367] ⌈ The DCM module shall perform Inter Module Checks to avoid the integration of incompatible files. The imported included files shall be checked by preprocessing directives.
The following version numbers shall be verified:
- `<MODULENAME>_AR_RELEASE_MAJOR_VERSION`
- `<MODULENAME>_AR_RELEASE_MINOR_VERSION`

If the values are not identical to the values expected, an error shall be reported.⌋(BSW004)

5.1.3 Design Rules

[Dcm550] ⌈ The DCM module's source (as long as it is written in C) shall conform to the HIS subset of the MISRA C Standard.⌋() **[Dcm551]** ⌈ The DCM module's source shall not use compiler and platform specific keywords ⌋(BSW00306)

[Dcm552] ⌈ The DCM module's source shall indicate all global data with read-only properties by explicitly assigning the keyword `const`.⌋(BSW00309)

[Dcm553] ⌈ The DCM module may use macros (instead of functions) where source code is used and runtime is critical.⌋(BSW00330)

[Dcm554] ⌈ The DCM module shall not define global data in header files (If global variables have to be used, the definition should take place in the C file)⌋(BSW00308)

[**Dcm555**] ⌈ The DCM module's source shall not be processor and compiler dependent.

⌋(BSW006)

6 Requirements traceability

Requirement	Satisfied by
-	Dcm235
-	Dcm107
-	Dcm597
-	Dcm093
-	Dcm435
-	Dcm470
-	Dcm407
-	Dcm520
-	Dcm240
-	Dcm231
-	Dcm550
-	Dcm568
-	Dcm518
-	Dcm211
-	Dcm646
-	Dcm253
-	Dcm655
-	Dcm356
-	Dcm752
-	Dcm618
-	Dcm574
-	Dcm692
-	Dcm196
-	Dcm379
-	Dcm398
-	Dcm118
-	Dcm706
-	Dcm114
-	Dcm654
-	Dcm707
-	Dcm145
-	Dcm272
-	Dcm420
-	Dcm159

-	Dcm690
-	Dcm161
-	Dcm354
-	Dcm362
-	Dcm249
-	Dcm753
-	Dcm222
-	Dcm442
-	Dcm468
-	Dcm287
-	Dcm325
-	Dcm595
-	Dcm197
-	Dcm699
-	Dcm198
-	Dcm110
-	Dcm403
-	Dcm438
-	Dcm371
-	Dcm324
-	Dcm217
-	Dcm241
-	Dcm439
-	Dcm495
-	Dcm603
-	Dcm167
-	Dcm587
-	Dcm157
-	Dcm394
-	Dcm221
-	Dcm614
-	Dcm352
-	Dcm473
-	Dcm164
-	Dcm544
-	Dcm271
-	Dcm715
-	Dcm202
-	Dcm332

-	Dcm008
-	Dcm567
-	Dcm160
-	Dcm610
-	Dcm133
-	Dcm094
-	Dcm129
-	Dcm566
-	Dcm647
-	Dcm440
-	Dcm121
-	Dcm640
-	Dcm488
-	Dcm483
-	Dcm688
-	Dcm656
-	Dcm606
-	Dcm612
-	Dcm653
-	Dcm192
-	Dcm321
-	Dcm085
-	Dcm590
-	Dcm123
-	Dcm541
-	Dcm527
-	Dcm641
-	Dcm681
-	Dcm718
-	Dcm151
-	Dcm578
-	Dcm437
-	Dcm153
-	Dcm137
-	Dcm084
-	Dcm620
-	Dcm399
-	Dcm651
-	Dcm701

-	Dcm602
-	Dcm256
-	Dcm558
-	Dcm345
-	Dcm260
-	Dcm684
-	Dcm255
-	Dcm436
-	Dcm716
-	Dcm336
-	Dcm049
-	Dcm463
-	Dcm539
-	Dcm149
-	Dcm679
-	Dcm594
-	Dcm469
-	Dcm627
-	Dcm148
-	Dcm632
-	Dcm611
-	Dcm397
-	Dcm511
-	Dcm601
-	Dcm622
-	Dcm218
-	Dcm490
-	Dcm642
-	Dcm720
-	Dcm698
-	Dcm236
-	Dcm556
-	Dcm494
-	Dcm607
-	Dcm257
-	Dcm608
-	Dcm154
-	Dcm195
-	Dcm048

-	Dcm275
-	Dcm418
-	Dcm125
-	Dcm269
-	Dcm134
-	Dcm323
-	Dcm193
-	Dcm422
-	Dcm386
-	Dcm703
-	Dcm562
-	Dcm360
-	Dcm139
-	Dcm135
-	Dcm672
-	Dcm712
-	Dcm695
-	Dcm580
-	Dcm617
-	Dcm224
-	Dcm351
-	Dcm401
-	Dcm112
-	Dcm769
-	Dcm334
-	Dcm052
-	Dcm163
-	Dcm561
-	Dcm128
-	Dcm604
-	Dcm626
-	Dcm346
-	Dcm645
-	Dcm395
-	Dcm570
-	Dcm659
-	Dcm660
-	Dcm113
-	Dcm344

-	Dcm358
-	Dcm302
-	Dcm443
-	Dcm117
-	Dcm702
-	Dcm512
-	Dcm166
-	Dcm387
-	Dcm259
-	Dcm311
-	Dcm708
-	Dcm433
-	Dcm680
-	Dcm120
-	Dcm729
-	Dcm156
-	Dcm521
-	Dcm540
-	Dcm529
-	Dcm489
-	Dcm165
-	Dcm719
-	Dcm131
-	Dcm530
-	Dcm251
-	Dcm467
-	Dcm691
-	Dcm115
-	Dcm416
-	Dcm481
-	Dcm589
-	Dcm132
-	Dcm517
-	Dcm492
-	Dcm671
-	Dcm392
-	Dcm119
-	Dcm258
-	Dcm697

-	Dcm670
-	Dcm342
-	Dcm609
-	Dcm710
-	Dcm624
-	Dcm337
-	Dcm673
-	Dcm682
-	Dcm588
-	Dcm109
-	Dcm687
-	Dcm652
-	Dcm169
-	Dcm405
-	Dcm201
-	Dcm402
-	Dcm152
-	Dcm396
-	Dcm677
-	Dcm621
-	Dcm372
-	Dcm569
-	Dcm689
-	Dcm232
-	Dcm605
-	Dcm162
-	Dcm563
-	Dcm581
-	Dcm571
-	Dcm644
-	Dcm127
-	Dcm462
-	Dcm147
-	Dcm146
-	Dcm705
-	Dcm683
-	Dcm377
-	Dcm579
-	Dcm297

-	Dcm557
-	Dcm223
-	Dcm700
-	Dcm254
-	Dcm419
-	Dcm491
-	Dcm560
-	Dcm713
-	Dcm150
-	Dcm108
-	Dcm170
-	Dcm373
-	Dcm619
-	Dcm639
-	Dcm565
-	Dcm203
-	Dcm409
-	Dcm704
-	Dcm678
-	Dcm237
-	Dcm474
-	Dcm178
-	Dcm349
-	Dcm300
-	Dcm623
-	Dcm204
-	Dcm126
-	Dcm404
-	Dcm694
-	Dcm685
-	Dcm543
-	Dcm350
-	Dcm613
-	Dcm696
-	Dcm333
-	Dcm228
-	Dcm616
-	Dcm168
-	Dcm528

-	Dcm564
-	Dcm599
-	Dcm039
-	Dcm493
-	Dcm669
-	Dcm628
-	Dcm674
-	Dcm711
-	Dcm686
-	Dcm122
-	Dcm111
-	Dcm081
-	Dcm238
-	Dcm482
-	Dcm415
-	Dcm400
-	Dcm714
-	Dcm638
-	Dcm600
-	Dcm668
-	Dcm273
-	Dcm444
-	Dcm516
-	Dcm353
-	Dcm643
-	Dcm155
-	Dcm434
-	Dcm423
-	Dcm598
-	Dcm136
-	Dcm709
-	Dcm225
-	Dcm092
-	Dcm307
-	Dcm751
-	Dcm408
BSW	Dcm999
BSW003	Dcm335
BSW00301	Dcm054

BSW00302	Dcm055
BSW00306	Dcm551
BSW00307	Dcm999
BSW00308	Dcm554
BSW00309	Dcm552
BSW00310	Dcm548
BSW00314	Dcm999
BSW00318	Dcm335
BSW00321	Dcm999
BSW00323	Dcm043
BSW00326	Dcm999
BSW00327	Dcm364
BSW00328	Dcm999
BSW00330	Dcm553
BSW00331	Dcm364
BSW00334	Dcm999
BSW00336	Dcm999
BSW00337	Dcm041, Dcm012
BSW00338	Dcm042, Dcm040
BSW00339	Dcm999
BSW00341	Dcm999
BSW00342	Dcm999
BSW00346	Dcm054
BSW00347	Dcm999
BSW00350	Dcm042
BSW00353	Dcm549
BSW00358	Dcm037
BSW00361	Dcm999
BSW00369	Dcm044
BSW00370	Dcm054
BSW00373	Dcm053
BSW00374	Dcm335
BSW00375	Dcm999
BSW00376	Dcm053
BSW00378	Dcm999
BSW00379	Dcm335
BSW00380	Dcm054
BSW00381	Dcm055

BSW00383	Dcm999
BSW00385	Dcm999
BSW00386	Dcm999
BSW00387	Dcm999
BSW004	Dcm367
BSW00406	Dcm999
BSW00407	Dcm065
BSW00409	Dcm999
BSW00412	Dcm055
BSW00413	Dcm999
BSW00414	Dcm037
BSW00415	Dcm999
BSW00416	Dcm999
BSW00417	Dcm999
BSW00419	Dcm054
BSW00422	Dcm999
BSW00423	Dcm999
BSW00424	Dcm053
BSW00425	Dcm999
BSW00426	Dcm999
BSW00427	Dcm999
BSW00428	Dcm999
BSW00429	Dcm999
BSW00432	Dcm999
BSW00433	Dcm999
BSW00435	Dcm055
BSW00436	Dcm055
BSW00437	Dcm999
BSW00438	Dcm037
BSW00439	Dcm999
BSW00440	Dcm999
BSW00442	Dcm043, Dcm485, Dcm484, Dcm487, Dcm486, Dcm506, Dcm508, Dcm507, Dcm509
BSW00443	Dcm999
BSW00444	Dcm999

BSW00445	Dcm999
BSW00446	Dcm999
BSW00447	Dcm999
BSW00450	Dcm593
BSW00453	Dcm999
BSW00455	Dcm999
BSW005	Dcm999
BSW006	Dcm555
BSW010	Dcm999
BSW04001	Dcm421, Dcm243, Dcm246, Dcm244, Dcm245, Dcm414, Dcm417, Dcm411, Dcm410
BSW04003	Dcm030
BSW04005	Dcm020, Dcm252
BSW04006	Dcm022, Dcm250
BSW04010	Dcm299, Dcm298, Dcm295, Dcm296, Dcm293, Dcm393, Dcm383, Dcm384, Dcm385, Dcm388, Dcm389, Dcm289, Dcm284, Dcm286, Dcm279, Dcm330, Dcm441, Dcm304, Dcm466, Dcm465, Dcm464, Dcm376, Dcm378, Dcm476, Dcm475, Dcm478, Dcm380, Dcm381, Dcm382, Dcm519,

	Dcm248, Dcm406, Dcm007, Dcm004, Dcm005, Dcm413, Dcm412
BSW04011	Dcm338, Dcm339, Dcm340
BSW04015	Dcm144
BSW04016	Dcm024
BSW04017	Dcm028, Dcm038
BSW04019	Dcm547
BSW04020	Dcm001, Dcm200
BSW04033	Dcm496, Dcm499, Dcm505, Dcm502, Dcm504, Dcm503
BSW04058	Dcm077, Dcm295, Dcm296, Dcm293, Dcm393, Dcm383, Dcm384, Dcm385, Dcm388, Dcm389, Dcm279, Dcm465, Dcm378, Dcm476, Dcm475, Dcm382, Dcm004, Dcm005
BSW04065	Dcm004, Dcm005
BSW04067	Dcm293, Dcm378
BSW04079	Dcm441
BSW04082	Dcm421, Dcm243, Dcm246, Dcm244, Dcm245, Dcm414, Dcm417,

	Dcm411, Dcm410
BSW04098	Dcm592, Dcm533, Dcm532, Dcm531, Dcm537, Dcm536, Dcm535
BSW04100	Dcm582, Dcm522, Dcm524, Dcm523, Dcm526, Dcm525
BSW101	Dcm034, Dcm033, Dcm037, Dcm036
BSW158	Dcm054
BSW159	Dcm999
BSW161	Dcm999
BSW162	Dcm999
BSW164	Dcm999
BSW168	Dcm999
BSW170	Dcm999
BSW172	Dcm999

6.1 Document: General requirements on Basic Software modules:

Functional General Requirements	
Requirement	Satisfied by
Configuration	--
[BSW00344] Reference to link-time configuration	Fulfilled by chapter 10
[BSW00404] Reference to post build time configuration	Fulfilled by chapter 10
[BSW00405] Reference to multiple configuration sets	Fulfilled by chapter 10
[BSW00345] Pre-compile-time configuration	Fulfilled by chapter 10
[BSW159] Tool-based configuration	Not applicable
[BSW167] Static configuration checking	Requirement on configuration tool
[BSW171] Configurability of optional functionality	Fulfilled by chapter 10
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable
[BSW00380] Separate C-Files for configuration parameters	Dcm054

[BSW00419] Separate C-Files for pre-compile time configuration parameters	Dcm054
[BSW00381] Separate configuration header file for pre-compile time parameters	Dcm055
[BSW00412] Separate H-File for configuration parameters	Dcm055
[BSW00383] List dependencies of configuration files	Not applicable
[BSW00384] List dependencies to other modules	Fulfilled by chapter 5
[BSW00387] Specify the configuration class of callback function	Not applicable
[BSW00388] Introduce containers	Fulfilled by chapter 10
[BSW00389] Containers shall have names	Fulfilled by chapter 10
[BSW00390] Parameter content shall be unique within the module	Fulfilled by chapter 10
[BSW00391] Parameter shall have unique names	Fulfilled by chapter 10
[BSW00392] Parameters shall have a type	Fulfilled by chapter 10
[BSW00393] Parameters shall have a range	Fulfilled by chapter 10.2 to 10.5
[BSW00394] Specify the scope of the parameters	N/A
[BSW00395] List the required parameters (per parameter)	Fulfilled by chapter 10
[BSW00396] Configuration classes	Dcm171, Dcm172, Dcm173
[BSW00397] Pre-compile-time parameters	Fulfilled by chapter 10
[BSW00398] Link-time parameters	Fulfilled by chapter 10
[BSW00399] Loadable Post-build time parameters	Fulfilled by chapter 10
[BSW00400] Selectable Post-build time parameters	Fulfilled by chapter 10
[BSW00438] Post Build Configuration Data Structure	Dcm037
[BSW00402] Published information	Fulfilled by chapter 10
Wake-Up	--
[BSW00375] Notification of wake-up reason	Not applicable
Initialization	--
[BSW101] Initialization interface	Dcm033, Dcm034; Dcm035, Dcm036, Dcm037
[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	Dcm065
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable
[BSW00424] BSW main processing function task allocation	Dcm053
[BSW00425] Trigger conditions for schedulable objects	Not applicable
[BSW00426] Exclusive areas in BSW modules	Not applicable
[BSW00427] ISR description for BSW modules	Not applicable
[BSW00428] Execution order dependencies of main processing functions	Not applicable
[BSW00429] Restricted BSW OS functionality access	Not applicable
[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	Dcm593
[BSW00442] Debugging Support in Modules	Dcm043, Dcm484, Dcm485, Dcm486, Dcm487, Dcm506,

	Dcm507, Dcm508, Dcm509
Shutdown Operation	--
[BSW00336] Shutdown interface	Not applicable
Fault Operation and Error Detection	--
[BSW00337] Classification of errors	Dcm012, Dcm041
[BSW00338] Detection and Reporting of development errors	Dcm040, Dcm042
[BSW00369] Do not return development error codes via API	Dcm044
[BSW00339] Reporting of production relevant error status	Not applicable
[BSW00422] Debouncing of production relevant error status	Not applicable
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable
[BSW00323] API parameter checking	Dcm043
[BSW004] Version check	Dcm367
[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 chapter 8.2 Type Definitions
[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	Dcm548

[BSW00373] Main processing function naming convention	Dcm053
[BSW00327] Error values naming convention	Dcm364
[BSW00335] Status values naming convention	Fulfilled by API definitions in chapter 8
[BSW00350] Development error detection keyword	Dcm042
[BSW00408] Configuration parameter naming convention	Fulfilled by configuration chapter 0
[BSW00410] Compiler switches shall have defined values	Fulfilled by configuration chapter 0
[BSW00411] Get version info keyword	Fulfilled by configuration chapter 0

Module file structure

Requirement	***Satisfied by***
[BSW00346] Basic set of module files	Dcm054
[BSW158] Separation of configuration from implementation	Dcm054
[BSW00314] Separation of interrupt frames and service routines	Not applicable (this module does not provide any ISRs)
[BSW00370] Separation of callback interface from API	Dcm054
[BSW00435] Header File Structure for the Basic Software Scheduler	Dcm055
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	Dcm055
[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	Dcm549
[BSW00361] Compiler specific language extension header	Not applicable (requirement on implementation, not on specification)

Module Design

Requirement	***Satisfied by***
[BSW00301] Limit imported information	Dcm054
[BSW00302] Limit exported information	Dcm055
[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	Dcm555
[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	Fulfilled by chapter 8.7.3 1 Configurable Interfaces

Autosar Application Software shall return a Std_ReturnType	

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	Dcm551

Global data

Requirement	*Satisfied by*
[BSW00308] Definition of global data	Dcm554
[BSW00309] Global data with read-only constraint	Dcm552

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	Dcm037
[BSW00414] Parameter of init function	Dcm037
[BSW00376] Return type and parameters of main processing functions	Dcm053
[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	Dcm553
[BSW00331] Separation of error and status values	Dcm364

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	Dcm335
[BSW00379] Module identification	Dcm335
[BSW003] Version identification	Dcm335
[BSW00318] Format of module version	Dcm335
[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)

6.2 Document: AUTOSAR requirements on Basic Software, cluster Diagnostic

Requirements on Basic Software Module Diagnostic	
Requirement	Satisfied by
General	--
[BSW04010] Interface between Diagnostic service handling and Diagnostic event (error) management	Dcm007, Dcm247, Dcm005, Dcm248, Dcm376, Dcm293, Dcm378, Dcm380, Dcm381, Dcm295, Dcm296, Dcm478, Dcm475, Dcm476, Dcm479, Dcm382, Dcm480, Dcm298, Dcm299, Dcm383, Dcm384, Dcm385, Dcm441, Dcm429, Dcm430 Dcm431 Dcm388, Dcm389, , Dcm393, Dcm466, Dcm464, Dcm465, Dcm519, Dcm304, Dcm406, Dcm279, Dcm280, Dcm284, Dcm278, Dcm286,

	Dcm289, Dcm412, Dcm330, Dcm004, Dcm413
[BSW04082] Support of ISO15031-5 and SAE J1979	Dcm243, Dcm244, Dcm245, Dcm410, Dcm411, Dcm246, Dcm414, Dcm417, Dcm421
Diagnostic communication management (DCM)	--
[BSW04007] Provide Diagnostic service handling	Fulfilled by Section 7.4
[BSW04021] Switch diagnostic communication access	Dcm015
[BSW04032] Support of different diagnostic addresses	Dcm770_Conf, Dcm772_Conf
[BSW04058] Access to different event (fault) memories	Dcm077, Dcm005, Dcm293, Dcm378, Dcm295, Dcm296, Dcm475, Dcm476, Dcm382, Dcm383, Dcm384, Dcm385, Dcm429, Dcm388, Dcm389, Dcm390, Dcm393, Dcm465, Dcm279, Dcm004
[BSW04065] Clearing of events or event groups	Dcm005, Dcm004
[BSW04067] Counting and evaluation of events according to ISO 14229-1 DTCStatusMask	Dcm293, Dcm378
[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)	Dcm243, Dcm244, Dcm245, Dcm410, Dcm411, Dcm246, Dcm414, Dcm417, Dcm421
[BSW04005] SecurityAccess level handling is managed by DCM	Dcm252, Dcm020
[BSW04006] Session handling is managed by DCM	Dcm022, Dcm250
[BSW04016] Provision of Busy Handling	Dcm024
[BSW04019] Application callback after transmit confirmation	Dcm547
[BSW04020] Suppression of Responses	Dcm001, Dcm200
[BSW04033] Upload/Download services for data handling	Dcm496, Dcm499, , Dcm502, Dcm503, Dcm504, Dcm505
[BSW04036] Format checking of diagnostic services	Dcm272_Conf, Dcm273_Conf, Dcm071_Conf, Dcm074_Conf
BSW04098 Standard bootloader interaction	Dcm531, Dcm532, Dcm533, Dcm535, Dcm538, Dcm536, Dcm537 , Dcm592

BSW04100 DLT debug interface using diagnostic services	Fulfilled by ROE requirements: Dcm522, Dcm523, Dcm524, Dcm525, Dcm526 **Dcm582**
Timing Requirements	--
[BSW04015] Provision of timing handling according to ISO15765-3	Dcm027, Dcm143, Dcm144, Dcm311, Dcm750_Conf
Resource Usage	--
[BSW04017] Provide optimized buffer handling	Dcm028, Dcm038, Dcm775_Conf
Interface and API	--
[BSW04078] Interface to fault memory, fault status	Chapter 8
[BSW04011] Provide diagnostic state information	Dcm338, Dcm339, Dcm340
[BSW04003] Interface to PDU Router shall be network independent	Dcm030
[BSW04079] The size of a FreezeFrame shall be reported to the DCM by the DEM	Dcm441
Configuration	--
[BSW04059] Configuration of timing parameter	Dcm031_Conf
[BSW04024] Configurable size of transferred data	Dcm739_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.

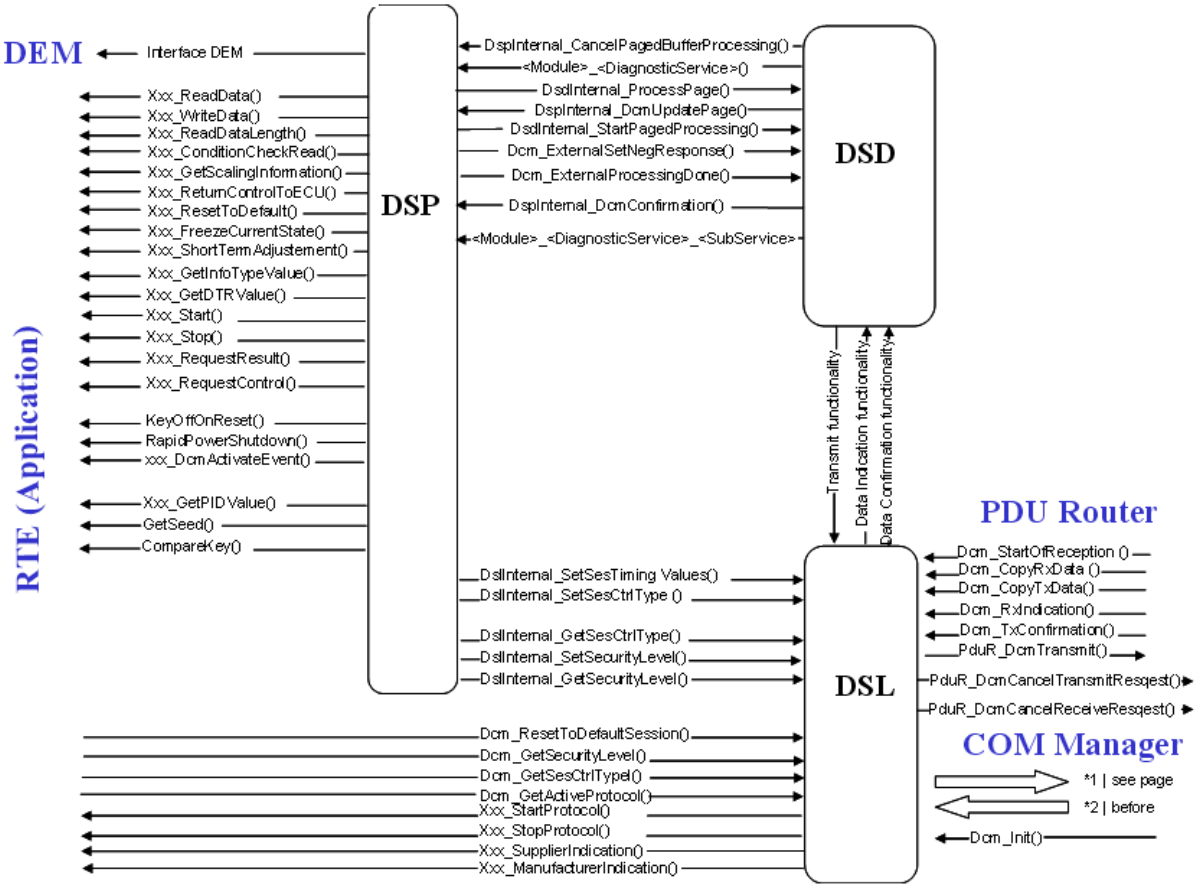


Figure 6 Possible interaction between the submodules in the DCM

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

7.1.2 Negative Response Code (NRC)

The standards defining the UDS Services and OBD Services define the 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`.

7.2 Diagnostic Session Layer (DSL)

7.2.1 Introduction

[Dcm030]「All functional areas of the DSL submodule shall be in conformance with the specifications ISO14229-1 [14] and the network-independent part of ISO15765-3 [17].」(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 [14] and ISO 15765-3 [17]),
- Application layer timing handling (as required by ISO14229-1 [14] and ISO 15765-3 [17]),
- Specific response behavior (as required by ISO14229-1 [14] and ISO 15765-3 [17]).

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 allows the DCM to reject the reception if the data size overflow its buffer size. 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 (successful or with errors) the PduR module will call `Dcm_TpRxIndication()` to give a receive indication to the DCM module.

[Dcm111] ⌈ The DSL submodule shall forward received data to the DSD submodule only after a call of `Dcm_TpRxIndication()` with parameter Result = NTFRSLT_OK (see Dcm093). ⌋()

[Dcm241] ⌈ As soon as a request message is received (after a call of `Dcm_TpRxIndication()` with parameter Result = NTFRSLT_OK (see Dcm093) and until a call to `Dcm_TpTxConfirmation()` (see Dcm351) 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 protocol type (e.g. an enhanced session can be ended by a OBD session) can be received, until the corresponding response message is sent and the DcmPduId is released again. ⌋()

More descriptions of the APIs (prototype, input/output parameter) can be found in the interface description of PduR module (see [2]).
It is allowed to have different DcmPduIds for different diagnostic communication applications. For example:

- OBD DcmRxPduId: for reception of OBD requests,
- OBD DcmTxPduId: for transmission of OBD responses,
- UDS phys DcmRxPduId: for reception of UDS physically addressed requests,
- 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 [14]. This functional “TesterPresent” will be received on a separate DcmRxPduId (UDS func DcmRxPduId) with a separate receive buffer. Due to that reason, the functional TesterPresent (and only functional TesterPresent without response) is handled in the following way:

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

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

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

7.2.4.4 Forward responses from the DSD submodule to the PduR module

[Dcm114] ⌈ The DSD submodule shall request the DSL submodule for transmission of responses.⌋()

[Dcm115] ⌈ 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 is given to the PduR module. After the DCM module has called `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.

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

[Dcm118] ⌈ In case of a failed transmission (failed `PduR_DcmTransmit()` request) or error confirmation (`Dcm_TpTxConfirmation()` with error), the DSD submodule shall not repeat the diagnostic response transmission.⌋()

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

7.2.4.5 Guarantee timing to tester by sending busy responses

[Dcm024] ⌈ If the Application (or the DSP submodule) is able to perform a requested diagnostic task, but needs additional time to finish the task and prepare the response, then the DSL submodule shall send a negative response with NRC 0x78 (Response pending) when reaching the response time (DcmTimStrP2ServerAdjust respectively DcmTimStrP2StarServerAdjust).⌋(BSW04016)

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

[Dcm119] ⌈ The DSL submodule shall send negative responses as required in Dcm024 from a separate buffer.⌋()

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

[Dcm120] ⌈ If the number of negative responses for a requested diagnostic tasks (see Dcm024) reaches the value defined in the configuration parameter ***DcmDslDiagRespMaxNumRespPend***, the DCM module shall stop processing the active diagnostic request, 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).⌋()

7.2.4.6 Support of periodic transmission

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

[Dcm121] ⌈ If a pending message for normal diagnostic responses (higher priority) exists, then the DSL submodule shall wait for the transmission confirmation (Call to `Dcm_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***

[Dcm122] ⌈ The DCM module shall send responses for periodic transmissions using a separate protocol and a separate buffer of configurable size. ⌋()

The ***DcmDslPeriodicTranmissionConRef*** configuration parameter allows linking the protocol used to receive the periodic transmission request / 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:

[Dcm123] ⌈ Periodic transmission communication shall only take place in Full Communication Mode. ⌋()

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

[Dcm125] ⌈ The DCM module shall discard periodic transmission events beside Full Communication Mode and shall not queue it for transmission. ⌋()

[Dcm126] ⌈ Periodic transmission events shall not activate the Full Communication Mode. ⌋()

7.2.4.7 Support of ROE transmission

With the UDS Service ResponseOnEvent (0x86), a tester requests an ECU to start or stop transmission of responses initiated by a specified event. Upon registering an event for transmission, the tester also specifies the corresponding service to respond to (e.g: UDS Service ReadDataByIdentifier (0x22)).

[Dcm595] ⌈ The ROE functionality is enabled only if the container DcmDslResponseOnEvent exists ⌋()

[Dcm597]「 If the configuration parameter *DcmDspRoeInitOnDSC* is set to TRUE: on reception of service DiagnosticSessionControl, if ROE is active (reception of startResponseOnEvent) the DCM module shall unactivate all ROE active event. For OnChangeOfDataIdentifier events configured as externally managed (presence of a DcmDspDidExtRoe container in the associated DcmDspDid container), the DCM shall call the configured Api or RTE operation xxx_ActivateEvent (see *DcmDspDidRoeActivateFnc*) with state value DCM_ROE_UNACTIVE and the configured RoeEventId (see DcmDspDidRoeEventId)」()

[Dcm618]「 If the configuration parameter *DcmDspRoeInitOnDSC* is set to TRUE: on S3Server timeout, if ROE is active (reception of startResponseOnEvent) the DCM module shall unactivate all ROE active event. For OnChangeOfDataIdentifier events configured as externally managed (presence of a DcmDspDidExtRoe container in the associated DcmDspDid container), the DCM shall call the configured Api or RTE operation xxx_ActivateEvent (see *DcmDspDidRoeActivateFnc*) with state value DCM_ROE_UNACTIVE and the configured RoeEventId (see DcmDspDidRoeEventId)」()

[Dcm709]「 The DCM shall implement the parameter *storageState* (Bit 6 of the eventType sub-function) which indicates whether the event shall be stored in non volatile memory in the server and re-activated upon the next power-up of the server or if it shall terminate once the server powers down.」()

[Dcm711]「The DCM shall implement the parameter *EventWindowTime* used to specify a window for the event logic to be active in the server.」()

Dcm747「If the parameter value of *EventWindowTime* is set to 0x02 then the response time is infinite.If the parameter value is different from 0x02, then this parameter is vehicleManufacturerSpecific.」()

Dcm748「If *EventWindowTime* parameter is vehicleManufacturerSpecific, the DCM shall call the configured Api *XXX_ROEInit(EventWindowTime)* (see configuration parameter DcmDspRoeInitFnc) to initialize the SWC/integration code. *Dcm_StopROE()* shall be called to stop the ROE event if the *EventWindowTime* timed out (see **[Dcm713**).*Dcm_RestartROE()* shall be called if the ROE needs to be restarted after a PowerDownReset (see **[Dcm714**).」()

[Dcm712] ⌈ If the ROE is stopped by a diagnostic request, the *EventWindowTime* needs to be stopped by calling the configured function *XXX_ROEStop()*.(see configuration parameter DcmDspRoeStopFnc)⌋()

[Dcm713] ⌈ On *Dcm_StopROE()* call, the DCM shall clear the ROE event from the transmission queue.⌋()

[Dcm714] ⌈ On *Dcm_RestartROE()* call, the DCM shall restart the ROE event.⌋()

7.2.4.7.1 TYPE1 = Responses are sent on the same DcmTxPduId that was used for the ROE service response.

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

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

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

7.2.4.7.2 TYPE2 = Responses are sent on a separate DcmTxPduId.

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

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

[Dcm132] ⌈ The content of the pMsgContext pointer (ROE message) shall be copied into the buffer.⌋()

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

[Dcm133] ⌈ ROE communication shall only be performed in Full Communication Mode.⌋()

[Dcm134] ⌈ ROE events shall be disabled in any other Communication Mode except for the Full Communication Mode.⌋()

[Dcm135] ⌈ ROE events beside Full Communication Mode shall be discarded and not queued for later transmission. If the storageState parameter is enabled (see **[Dcm709**), the ROE event shall be reactivated upon the next power-up of the server only after the Full Communication Mode activation.⌋()

[Dcm136] ⌈ ROE events requested by the Application shall not activate the Full Communication Mode.⌋()

[Dcm558] ⌈ If an ROE event occurs and the serviceToRespondTo is already in execution and is not re-entrant (e.g ReadDTCInformation), the DCM shall delay the event until the current request is finished.⌋()

7.2.4.7.3 **Please note the following limitations:**

[Dcm137] ⌈ While processing an ROE TYPE1 event, any additional requests (external and internal) shall be rejected with the same or lower priority of its Protocol Table. ⌋()

This limitation doesn't apply to ROE TYPE2 event.

7.2.4.7.4 **OnTimerInterrupt**

If the resolution of the Main function is sufficient, it can be implemented inside the DCM, in other case it would be a limitation of the DCM and the subservice OnTimerInterrupt would not be supported.

7.2.4.7.5 **OnChangeOfDataIdentifier**

The subservice OnChangeOfDataIdentifier can be fully implemented inside the DCM or partially outside.
In the first case, the DCM will poll periodically the DID value to compare it.
In the second case the DCM will activate the event in the module implementing this ROE event and wait a trigger of this module to execute the ROE response. It will be , e.g., the case of the DLT module.

[**Dcm522**] ⌈ The DCM shall implement the subservice OnChangeOfDataIdentifier ⌋(BSW04100)

[Dcm523] ⌈ If at least one ROE DID is configured as internally managed, the DCM shall provide a temporary buffer to store the DID value during the execution of a ROE process for subservice OnChangeOfDataIdentifier. The parameter *DcmDspRoeBufSize* is used to define the size of this buffer. ⌋(BSW04100)

Note: This buffer is used only in the case a ROE DID is managed internally. If all ROE DID are managed externally, this buffer is not required.

[Dcm524] ⌈ On reception of the subservice startResponseOnEvent, if the event has been set up (reception of ROE request with EventType equal to OnChangeOfDataIdentifier) and is configured as internally managed (absence of DcmDspDidExtRoe container in the associated DcmDspDid container), the DCM shall poll periodically (MainFunction period) the associated DID value and compare it to the previous value. If the value has changed the DCM shall execute the associated service. (The response message is transmitted directly or added in the ROE queue if configured) ⌋(BSW04100)

[Dcm525] ⌈ On reception of the subservice startResponseOnEvent, if the event has been set up (reception of ROE request with EventType equal to OnChangeOfDataIdentifier) and is configured as externally managed (presence of a DcmDspDidExtRoe container in the associated DcmDspDid container), the DCM shall call the configured Api or RTE operation xxx_ActivateEvent (see *DcmDspDidRoeActivateFnc*) with state value DCM_ROE_ACTIVE and the configured RoeEventId (see DcmDspDidRoeEventId) ⌋(BSW04100)

[Dcm526] ⌈ On reception of the subservice stopResponseOnEvent, if the event has been set up (reception of ROE request with EventType equal to OnChangeOfDataIdentifier) and is configured as externally managed (presence of a DcmDspDidExtRoe container in the associated DcmDspDid container), the DCM shall call the configured Api or RTE operation xxx_ActivateEvent (see *DcmDspDidRoeActivateFnc*) with state value DCM_ROE_UNACTIVE and the configured RoeEventId (see DcmDspDidRoeEventId)⌋(BSW04100)

[Dcm582] ⌈ The DCM shall process external ROE event only if no external diagnostic request is in process. If the event cannot be processed, the value E_NOT_OK shall be returned on call of Dcm_TriggerOnEvent().⌋(BSW04100)

Interaction with DLT :

For interaction with DLT module, the DCM shall be configured as follow:
- A DID with one data shall be configured for DLT purpose
- The data used for DLT purpose shall be set as available trough C function call (DcmDspDataUsePort=USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC)
- DcmDspDataReadFnc of DLT data shall be set with the name of the DLT Api : Dlt_ReadData
- DcmDspDataReadDataLengthFnc of DLT data shall be set with the name of the DLT Api : Dlt_ReadDataLength
- DcmDspDidRoeActivateFnc of DLT DID shall be set with the name of the DLT Api : Dlt_ActivateEvent

7.2.4.7.6 **OnDTCStatusChange**

[**Dcm598**] ⌈ The DCM shall implement the subservice OnDTCStatusChange⌋()

[Dcm599] ⌈ On reception of the subservice startResponseOnEvent, if the event has been set up (reception of ROE request with EventType equal to OnDTCStatusChange), the DCM shall activate the DTC status change supervision. ⌋()

[Dcm600] ⌈ On reception of the subservice stopResponseOnEvent, if the event has been set up (reception of ROE request with EventType equal to OnDTCStatusChange), the DCM shall unactivate the DTC status change supervision⌋()

[Dcm602] ⌈ On call to Dcm_DemTriggerOnDTCStatus() by DEM, the DCM shall check if the bits that have changed of state, correspond to the status mask provided in the the ROE request (with EventType equal to OnDTCStatusChange). If positive case, the DCM shall execute the associated service (The response message is transmitted directly or added in the ROE queue if enabled)⌋()

To prevent the network from being flooted with Roe-Messages when the fault memory provided by the DEM module is cleared by an external test tool, the DCM module suppresses the ROE messages of OnDtcStatusChange:

[Dcm603] ⌈ If the DCM module is requested by an external tester to clear all DTCs, the DCM module shall not send ROE message triggered by OnDtcStatusChange during the clearing. ⌋()

7.2.4.7.7 **Roe transmission queue**

The transmission queue of ROE messages is needed, if the DCM module cannot send out the message at the point in time when it was triggered (e.g. high bus-load on CAN). Therefore the DCM module provides a mechanism to shift the message transmission to a later point in time.

[Dcm604] ⌈ If the ROE functionality is enabled (See Dcm595), the DCM module shall provide a ROE transmission queue depending of value of configuration parameter DcmDspRoeQueueEnabled⌋()

Note: The provided transmission queue is used for all ROE events (e.g. OnDtcStatusChanged, OnChangeOfDataIdentifier, ...).

[Dcm605] ⌈ Due to limitation of resources the buffer for one ROE message is limited (refer to configuration Parameter DcmDspRoeMaxEventLength).⌋()

Note: If an ROE message on CAN consists of only one CAN frame (the parameter DcmRoeMaxEventLength is set to 8 byte), a flow control message is not needed.

[Dcm710] ⌈ If the transmission queue of ROE messages is full, the Dcm shall drop the oldest event from the queue to make space for the new event.⌋()

[Dcm606] ⌈ If the ROE queue mechanism is disabled for an event(see configuration parameter DcmDspDidRoeQueueEnabled), if the event occurs the response message is sended only if no transmission is in progress. The ROE queue mechanism can only be disabled event per event for subservice OnChangeOfDataIdendifier
⌋()

Note: There may some DIDs which will need a large message and therefore the queuing mechanism cannot be supported.

[Dcm607] ⌈ The ROE transmission queue of the DCM module shall implement the First-In-First-Out principle implying that the element which was put first in the queue will be transmitted first. ⌋()

[Dcm608] ⌈ The DCM module provides the configuration parameter DcmDspRoeMaxQueueLength defining the maximum number of queued messages.⌋()

Note: The defined length of one ROE message (refer to DcmRoeMaxEventLength) and the number of queued messages (refer to DcmDspRoeMaxQueueLength) defines the maximum memory resources, which have to be allocated/provided by the DCM module for the ROE transmission queue

[Dcm609] ⌈ The DCM module shall provide a retry-mechanism of a ROE-Message to send out the message at a later point in time.⌋()

[Dcm610] ⌈ The DCM module shall stop to retry the transmission of an ROE event after a configured number of time.(see configuration parameter DcmDspRoeMaxNumberOfRetry)⌋()

Note : if DcmDspRoeMaxNumberOfRetry is set to 0 no retry mechanism will be managed

[Dcm611] ⌈ If the maximum number of retries is reached (refer to DcmDspRoeMaxNumberOfRetry), the DCM module shall remove the specific event from ROE queue.⌋()

Note: If the first queued event of the ROE queue was removed, the DCM module continues the transmission with the next event (refer to First-In-First-Out principle).

[Dcm612]「 On reception of the subservice stopResponseOnEvent, the DCM shall clear the ROE transmission queue」()

[Dcm613]「 If the configuration parameter *DcmDspRoeInitOnDSC* is set to TRUE: on reception of service DiagnosticSessionControl, if ROE is active (reception of startResponseOnEvent) , the DCM shall clear the ROE transmission queue.」()

[Dcm619]「 If the configuration parameter *DcmDspRoeInitOnDSC* is set to TRUE: on S3Server timeout, if ROE is active (reception of startResponseOnEvent) , the DCM shall clear the ROE transmission queue.」()

7.2.4.7.8 **Roe transmission cycle**

[Dcm601]「 The DCM module shall respect a minimum time between two (2) consecutive Roe transmissions (see configuration parameter DcmDspRoeInterMessageTime)」()

Note: The inter message time is referred to a sending cycle. The configuration parameter DcmDspRoeInterMessageTime is used for the delay between two different consecutive Roe transmission and between two transmission of the retry mechanism.

7.2.4.8 Support of segmented response (paged-buffer)

[Dcm028]「 If enabled (***DcmPagedBufferEnabled***=TRUE), the DCM module shall provide a mechanism to send responses larger than the configured and allocated diagnostic buffer.」(BSW04017)

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

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

7.2.4.9 Support of ResponsePending response triggered by the Application

In some cases, e.g. in case of 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 E_FORCE_RCRRP, the DSL submodule will trigger the transmission of a negative response with NRC 0x78 (Response pending).
This response needs to be sent from a separate buffer, in order to avoid overwriting the ongoing processing of the request.

7.2.4.10 Manage security level

[Dcm020] ⌈ The DSL submodule shall save the level of the current active security level. ⌋(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()`

[Dcm033] ⌈ During DCM initialization the security level is set to the value 0x00. ⌋(BSW101)

By Dcm033, the ECU is locked.

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

Only one security level can be active at a time.

7.2.4.11 Manage session state

[Dcm022] ⌈ The DSL submodule shall save the state of the current active session. ⌋(BSW04006)

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

- set a new session: `DslInternal_SetSesCtrlType()`

[Dcm034]「 During DCM initialization, the session state is set to the value 0x01 (“DefaultSession”).」(BSW101)

7.2.4.12 Keep track of active non-default sessions

Dcm140「Whenever a non-default session is active and when the session timeout (S3Server) is reached without receiving any diagnostic request, the DSL submodule shall reset to the default session state (“DefaultSession”, 0x01) and invoke the the mode switch of the ModeDeclarationGroupPrototype DcmDiagnosticSessionControl by calling SchM_Switch_Dcm_<vendorApiInfix>_DcmDiagnosticSessionControl(SchM_MODE_DcmDiagnosticSessionControl_DcmConf_ DcmDspSessionRow_DEFAULT_SESSION) .」()

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

Dcm141「

Sub-sequent start	Completion of any final response message or an error indication (`Dcm_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.」()

7.2.4.13 Allow to modify timings

Dcm027「 The DCM module shall handle the following protocol timing parameters in compliance with [17]: P2ServerMin, P2ServerMax, P2*ServerMin, P2*ServerMax, S3Server」(BSW04015)

Dcm143「 P2min / P2*min and S3Server shall be set to defined values: P2min = 0ms, P2*min = 0ms, S3Server = 5s.」(BSW04015)

Every protocol (OBD, enhanced diagnosis) have their own protocol timing values (given by default session configuration). These timing values are set when the protocol is started.
These protocol timing parameters have influence on the session layer timing (no influence on Transport Layer timing). Some of these timing parameters can be modified while protocol is active with the following means:

- UDS Service DiagnosticSessionControl (0x10)
- UDS Service AccessTimingParameter (0x83)

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

- Provide the active timing parameters,
- Set the new timing parameters. Activation of new timing values is only allowed after sending the response.

7.2.4.14 Handling of different diagnostic protocols

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

Dcm018「 Different protocols shall be used for UDS Services versus OBD Services」()

Dcm018 is required because of

- different protocol settings
- different valid service tables
- prioritization of protocols
- preemption of protocols

7.2.4.14.1 **Different service tables**

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

Dcm035「 With every protocol initialization, the DSL submodule sets a link to the corresponding service table (see configuration parameter ***DcmDslProtocolSIDTable***).」(BSW101)

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

7.2.4.14.2 **Prioritization of protocol**

The configuration parameter ***DcmDslProtocolPriority*** makes it possible to give each protocol its own relative priority.
Possible use case: There are ECUs, communicating with a vehicle-internal diagnostic tester (running on enhanced diagnosis) and a vehicle-external OBD tester. The OBD communication must have a higher priority than the enhanced diagnosis.

Dcm015「A protocol with higher priority is allowed to preempt the already running protocol.」(BSW04021)

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

7.2.4.14.3 **Preemption of protocol**

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

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

Dcm460「When `PduR_DcmCancelTransmit ()` returns E_NOT_OK, the DCM module shall assume that the ongoing transmission cannot be cancelled and shall not retry to cancel the transmit request. The current protocol shall be stopped and the new one started.」()

The PduR later calls `Dcm_TpTxConfirmation()` to confirm the cancellation.

Dcm461「When `Dcm_TpTxConfirmation()` is called with NTFRSLT_E_CANCELATION_NOT_OK (meaning that an error occurred during the cancellation), the DCM module shall assume that the ongoing transmission cannot be cancelled and shall not retry to cancel the transmit request because this transmission has been confirmed. The current protocol shall be stopped and the new one started.」()

Dcm559「If a running diagnostic request is preempted by a higher prior request (of another protocol), the DCM shall cancel all pending operation related to this request : for DEM operation : call of Dem_DcmCancelOperation(), for external operation call the pending operation with Dcm_OpStatus set to DCM_CANCEL.」()

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

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

The PduR later calls `Dcm_TpRxIndication()` to confirm the cancellation.

Dcm577「When `Dcm_TpRxIndication()` is called with NTFRSLT_E_CANCELATION_NOT_OK (meaning that an error occurred during the cancellation), the DCM module shall assume that the ongoing reception cannot be cancelled and shall not retry to cancel the receive request because this reception has been indicated. The current protocol shall be stopped and the new one started.」()

Dcm625「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***).」()

Dcm728:「The handling of protocols with equal priority shall be possible.」

Dcm727「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 Dcm914_Conf). If this configuration parameter is TRUE, a negative response with NRC 0x21 (BusyRepeatRequest) shall be issued for the second request (see Dcm788 and Dcm789). If the configuration parameter is FALSE, no response shall be issued (see Dcm790).」()

[**Dcm729**] ⌈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.⌋()

7.2.4.14.4 **Detection of protocol start**

[Dcm036] ⌈ With first request of a diagnostic protocol, the DSL submodule shall call all configured `Xxx_StartProtocol()` functions (see configuration parameter ***DcmDslCallbackDCMRequestService***).⌋(BSW101)

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

[Dcm144] ⌈ After all `Xxx_StartProtocol()` functions have returned E_OK (meaning all components have allowed the start of the protocol), the default timing parameters are loaded from the default session configuration (see configuration parameter ***DcmDspSessionRow***). The DcmDspSession table used is the one linked to the started protocol (see configuration parameter ***DcmDslProtocolSessionRef***)⌋(BSW04015)

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

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

[Dcm147] ⌈ After all `Xxx_StartProtocol()` functions have returned E_OK (meaning all components have allowed the start of the protocol), the session state is reset to default session. Furthermore the DCM module shall invoke the the mode switch of the *ModeDeclarationGroupPrototype* DcmDiagnosticSessionControl by calling SchM_Switch_Dcm_<vendorApiInfix>_DcmDiagnosticSessionControl(SchM_MODE_DcmDiagnosticSessionControl_DcmConf_DcmDspSessionRow_DEFAULT_SESSION) .⌋()

7.2.4.14.5 **Protocol stop**

A protocol stop can appear only in case of protocol preemption (See chapter 7.2.4.14.3 Preemption of protocol)

[Dcm624] ⌈ 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.

7.2.4.14.6 **Parallel execution of diagnostic response**

[Dcm081] ⌈ The DCM module shall allow the parallel execution of protocol response in the case of ResponseOnEvent and PeriodicTransmission services.⌋()

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

7.2.4.15 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 Dcm028).

7.2.4.16 Communication Mode Handling

7.2.4.16.1 **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).

7.2.4.16.2 **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).

7.2.4.16.3 **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).

7.2.4.16.4 **Default Session:**

[Dcm163] ⌈ With every request in default session of a diagnostic protocol, the DCM shall call `ComM_DCM_ActiveDiagnostic(NetworkId)`,with the networkId associated to the received Pdu (see DcmDslProtocolRxChannelId), to inform the ComM module about the need to stay in Full Communication Mode.⌋()

[Dcm164] ⌈ 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 DcmDslProtocolRxChannelId), to inform the ComM module that Full Communication is not longer needed.⌋()

[Dcm165] ⌈ 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.⌋()

[Dcm166] ⌈ If a “suppressPosRspMsgIndicationBit” is indicated and the positive response will be suppressed, the DCM shall call `ComM_DCM_InactiveDiagnostic(NetworkId).`⌋()

[Dcm697] ⌈ If a negative response is suppressed in case of functional addressing (see Dcm001), the DCM shall call `ComM_DCM_InactiveDiagnostic(NetworkId).`⌋()

7.2.4.16.5 **Session Transitions:**

[Dcm167] ⌈ If the actual diagnostic session is changed into a session different than the default session (initiated by UDS Service DiagnosticSessionControl), the DCM shall call `ComM_DCM_ActiveDiagnostic(NetworkId)`,with the networkId associated to the received Pdu, to inform the ComM module about the need to stay in Full Communication Mode.⌋()

[Dcm168] ⌈ If the actual diagnostic session is changed from a session different than the default into the default session (initiated by UDS Service DiagnosticSessionControl or S3Server timeout 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.⌋()

7.2.4.16.6 **Non Default Session:**

[Dcm169] ⌈ As long as the server is in a session other than the default session, the DCM shall not call `ComM_DCM_ActiveDiagnostic(NetworkId)`,with the networkId associated to the received Pdu, when receiving a request from a client provided by the PduR module.⌋()

[Dcm170] ⌈ As long as the server is in a session other than the default session, the DCM shall not call `ComM_DCM_InactiveDiagnostic(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.

[Dcm178] ⌈ The DSD submodule shall only process valid requests and shall reject invalid ones. ⌋()

7.3.2 Use cases

The following use cases are relevant and are described in detail in the following:

- Receive a request message and transmit a positive response message
- Receive a request message and suppress a positive response
- Receive a request message and 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 Dcm241). During the processing of this request, no other request of the same protocol type (e.g. an enhanced session can be ended by a OBD session) can be received, until the corresponding response message is sent and the DcmPduId is released again.

7.3.2.2 Receive a request message and suppress a positive response

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

7.3.2.3 Receive a request message and suppress a negative response

In case of functional addressing the DSD submodule shall suppress the negative response for NRC 0x11, 0x12 and 0x31 (see Dcm001)

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. If all of these parameters reports an error, the DSD submodule shall transmit a negative response with the first NRC reported and no positive response shall be send for this request.

7.3.2.5 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 [14]).
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 2 Interaction between the DSD submodule and the DSL submodule

7.3.3.2 Interaction of the DSD with the DSP

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

Table 3 Interaction between the DSD submodule and the DSP submodule

7.3.4 Functional Description of the DSD

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

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

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

This range corresponds to the diagnostic response identifier.

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

[Dcm193] ⌈ During this check, the DSD submodule shall search the newly received diagnostic service identifier in the “Service Identifier Table”.⌋()

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

[Dcm195] ⌈ The DSL submodule shall provide the current “Service Identifier Table”⌋()

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

[Dcm196] ⌈For the check, the DSD submodule shall scan the active “Service Identifier Table” for a newly received diagnostic service identifier. If this service identifier is supported and if the configuration parameter ***DcmDsdSidTabFnc*** (see Dcm777_Conf) 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”.

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

[Dcm198] ⌈ The DSD submodule shall store the newly received diagnostic service identifier for later use.⌋()

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

7.3.4.2 Handling of „suppressPosRspMsgIndicationBit“

The “suppressPosRspMsgIndicationBit” is part of the subfunction parameter structure (Bit 7 based on second byte of the diagnostic message, see ISO14229-1 [14] Section 6.5: Server response implementation rules).

[Dcm200] ⌈ If the “suppressPosRspMsgIndicationBit” is TRUE, the DSD submodule shall NOT send a positive response message.⌋(BSW04020)

[Dcm201] ⌈ The DSD submodule shall remove the “suppressPosRspMsgIndicationBit” (by masking the Bit) from the diagnostic message.⌋()

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

[Dcm203] ⌈ In case of responsePending the DCM module shall clear the "suppressPosRspMsgIndicationBit."⌋()

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

[Dcm204] ⌈ The DCM module shall only perform the "suppressPosRspMsgIndicationBit" handling when the configuration parameter ***DcmDsdSidTabSubfuncAvail*** is set for the newly received service identifier⌋()

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

7.3.4.3 Verification functionality

The DSD submodule will only accept a service, if the following three verifications are passed:

1. Verification of the Diagnostic Session
2. Verification of the Service Security Access levels
3. Verification of the Application environment/permission

7.3.4.3.1 Verification of the Diagnostic Session

The UDS Service DiagnosticSessionControl (0x10) is used to enable different diagnostic sessions in the ECU (e.g. Default session, Extended session). A diagnostic session enables a specific set of diagnostic services and/or functionality in the ECU. It furthermore enables a protocol-depending data set of timing parameters applicable to the started session.
On receiving a service request, the DSD module will obtain the current Diagnostic Session with `Dcm_GetSesCtrlType()` and will verify whether the execution of the requested service (NOT the UDS Service DiagnosticSessionControl (0x10)) 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.

[Dcm211] ⌈ If the newly received diagnostic service is not allowed in the current Diagnostic Session (according to the configuration parameter ***DcmDsdSidTabSessionLevelRef***), the DSD submodule shall transmit a negative response with NRC 0x7F (serviceNotSupportedInActiveSession) to the DSL submodule.⌋()

[Dcm616] ⌈ If the newly received diagnostic service is allowed in the current Diagnostic Session (see Dcm211), 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()`.

[Dcm217] ⌈ 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.⌋()

[Dcm617] ⌈ If the newly received diagnostic service is allowed in the current Security level (see Dcm217), 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**

Dcm773 ⌈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.⌋()

Dcm774 ⌜If the newly received diagnostic service is allowed in the current mode condition (Dcm773), 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.

[Dcm273] ⌜ 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.⌟()

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

[Dcm696] ⌜ 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.

[Dcm218] ⌜ If configured (configuration parameter ***DcmRequestManufacturerNotificationEnabled***=TRUE), the DSD submodule shall call the operation `Xxx_Indication()` on all configured ServiceRequestIndication ports (see configuration parameter ***DcmDslServiceRequestManufacturerNotification***).⌟()

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

[Dcm463] ⌈ If at least a single `Xxx_Indication()` function called according to Dcm218 has returned E_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.

[Dcm516] ⌈ If configured (configuration parameter ***DcmRequestSupplierNotificationEnabled***=TRUE), the DSD submodule shall call the operation `Xxx_Indication()` on all configured ServiceRequestIndication ports (see configuration parameter ***DcmDslServiceRequestSupplierNotification***).⌋()

[Dcm517] ⌈ If at least a single `Xxx_Indication()` function called according to Dcm516 returns E_REQUEST_NOT_ACCEPTED, the DSD submodule shall give no response.⌋()

[Dcm518] ⌈ If at least a single `Xxx_Indication()` function called according to Dcm516 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

[Dcm221] ⌈ The DSD submodule shall search for the executable functionality of the DSP submodule for newly received diagnostic service identifier and shall call the corresponding DSP service interpreter.⌋()

7.3.4.6 Assemble positive or negative response

[Dcm222] ⌈ When the DSP submodule has finished the execution of the requested Diagnostic Service the DSD submodule shall assemble the response.⌋()

The execution of the DSP service interpreter can have the results:

- positive Result or
- negative Result.

Following possible Responses can be assembled:

- positive Response,
- negative Response, or
- no Response (in the case of suppression of responses).

7.3.4.6.1 **Positive Response**

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

[Dcm223] ⌈ The DSD submodule shall add the response service identifier and the response data stream (returned by the Application) in the parameter “Dcm_MsgContextType”.⌋()

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

[Dcm225] ⌈ The DSD submodule shall execute the “Initiate transmission” functionality in the next execution step⌋()

7.3.4.6.2 **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 [14] (see Section 4.2.4 Response code parameter definition Table 12) and ISO15031-5 [15]. The DSP and the Application have to take care of the correct use of NRC of the executed Service ID.

[Dcm228] ⌈ The DSD submodule shall handle all NRCs supported from the Application and defined in `Dcm_NegativeResponseCodeType`.⌋()

7.3.4.6.3 **Suppression of response**

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

[Dcm001] ⌈ In the case of a Negative Result of the execution and active Functional Addressing the DSD submodule shall activate the suppression of the following Negative Responses:
NRC 0x11 (Service not supported),
NRC 0x12 (SubFunction not supported),
NRC 0x31 (Request out of range).⌋(BSW04020)

7.3.4.7 Initiate transmission

[Dcm232] ⌈ The DSD submodule shall forward the diagnostic (response) message (positive or negative response) to the DSL submodule.⌋()

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

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

[Dcm235] ⌈ The DSL submodule shall forward the received confirmation from the PduR module to the DSD submodule.⌋()

[Dcm236] ⌈ The DSD submodule shall forward the confirmation via the internal function `DspInternal_DcmConfirmation()` to the DSP submodule.⌋()

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

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

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

Rationale for Dcm240: The DSP submodule is waiting for the execution of the `DspInternal_DcmConfirmation()` functionality. So it has to be sent, also when no Data Confirmation is provided.
Altogether this means that in any of the following cases:

- Positive Response,
- Negative Response,
- Suppressed Positive Response, and
- Suppressed Negative Response

the DSD submodule will finish by calling `DspInternal_DcmConfirmation()`.

Dcm741「The DSD submodule shall call the operation Xxx_Confirmation() on all ports using the ServiceRequestNotification interface (see configuration parameter DcmDslServiceRequestManufacturerNotification and DcmDslServiceRequestSupplierNotification) 」()

Dcm742「The call of Xxx_Confirmation() shall be done right after the call of `DspInternal_DcmConfirmation()`」()

Dcm770「The DSD submodule shall call the operation Xxx_Confirmation() on all ports using the ServiceRequestNotification interface (see configuration parameter DcmDslServiceRequestManufacturerNotification and DcmDslServiceRequestSupplierNotification).」()

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.

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

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

7.4.1.2 Assemble response

[Dcm039] ⌈ The DSP submodule shall assemble the response message excluding response service identifier and determine the response message length.⌋()

[Dcm038] ⌈ If the paged-buffer mechanism is used, the DSP submodule shall determine the overall response length before any data is passed to the DSD submodule or the DSL submodule respectively.⌋(BSW04017)

Requirement Dcm038 is needed because of segmented diagnostic data transmission on CAN using ISO15765-2 [16], which requires the provision of the overall length of the complete data stream in the very first CAN frame of the respective data transmission (please refer to Section 7.2.4.8 for details about the paged-buffer mechanism).

[Dcm269] ⌈ The DSP submodule shall confirm the completion of the request processing with the function call `Dcm_ExternalProcessingDone()`.⌋()

7.4.1.3 Additional Negative Response Codes (NRCs)

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

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

7.4.1.4 Diagnostic mode declaration groups

Dcm775「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) 」()

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

Dcm776「The DCM SWS defines the mode of its ***ModeDeclarationGroups***.」()

Dcm778「For ModeDeclarationGroup Dcm_DiagnosticSessionControl, the mode declaration shall be identical to the symbolic names of the parameter DcmDspSessionLevel:

```
ModeDeclarationGroup DcmDiagnosticSessionControl {
                {
                <prefix>DefaultSession,
                <prefix>ProgrammingSession,
                <prefix>ExtendedSession,
                etc.
                }
                initialMode = <prefix>DefaultSession
};」()
```

Note: Refer [ecuc_sws_2108] defining the symbolic name prefix

Dcm806「The DCM shall define the ***ModeDeclarationGroupPrototype*** DcmDiagnosticSessionControl as provided-ModeGroup based on the

ModeDeclarationGroup DcmDiagnosticSessionControl. 」()

Dcm777「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

};」()

Dcm807「The DCM shall define the ***ModeDeclarationGroupPrototype*** DcmRapidPowerShutDown as provided-ModeGroup based on the following ***ModeDeclarationGroup***:
ModeDeclarationGroup DcmModeRapidPowerShutDown {
{
ENABLE_RAPIDPOWERSHUTDOWN,
DISABLE_RAPIDPOWERSHUTDOWN

}
initialMode = ENABLE_RAPIDPOWERSHUTDOWN

};」()

Dcm779「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」()

Dcm780「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>.」()

Dcm781「The DCM shall define the ***ModeDeclarationGroupPrototype*** DcmControlDTCSetting as provided-ModeGroup based on the following ***ModeDeclarationGroup***:

```
ModeDeclarationGroup DcmControlDTCSetting {
{
  ENABLEDTCSETTING,
  DISABLEDTCSETTING
}
        initialMode = ENABLEDTCSETTING
};」()
```

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 Dcm773 and Dcm774) 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**Dcm812**) is responded.

Dcm808「The DcmModeRule shall evaluate all referenced DcmModeConditions and/or nested DcmModeRules either by a logial 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.」()

Dcm809「Each DcmModeConditions shall either have a ***DcmSwcModeRef*** or a ***DcmBswModeRef***.」()

Dcm810「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 or by the API RTE_MODE.

Dcm811「In case multiple DcmModeConditions are referenced within a ***DcmModeRule*** they shall be evaluated in order of the order of the ***DcmArgumentRef*** references in the ECUC file. 」()

Note: This implies the priority of NRCs

Dcm782「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.」()

Dcm812「In case a nested DcmModeRule contains also a ***DcmModeRuleNrcValue*** parameter, this NRC shall be used prior the higher-level NRC.」()

Dcm813「In case DcmLogicalOperator is set to DCM_AND, the first failed DcmModeCondition shall be used to define the NRC for the response message. 」()

Dcm814「In case DcmLogicalOperator is set to DCM_OR, the last DcmModeCondition 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.

Dcm815「In case the complete evaluation result in no specific NRC the NRC 0x22 (ConditionsNotCorrect) shall be used.」()

7.4.2 UDS Services

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

SID	Service	Subfunction	Supported
0x10	DiagnosticSessionControl		supported
0x11	ECUReset		supported
0x14	ClearDiagnosticInformation		supported
0x19	ReadDTCInformation		supported
0x22	ReadDataByIdentifier		supported
0x23	ReadMemoryByAddress		supported (callout)
0x24	ReadScalingDataByIdentifier		supported
0x27	SecurityAccess		supported
0x28	CommunicationControl		supported
0x2A	ReadDataByPeriodicIdentifier		supported
0x2C	DynamicallyDefineDataIdentifier		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	0x00, 0x80	supported
0x3E	TesterPresent	Not 0x00 and not 0x80	NRC “SubFunctionNotSupported”
0x83	AccessTimingParameter		NRC “ServiceNotSupported”
0x84	SecuredDataTransmission		NRC “ServiceNotSupported”
0x85	ControlDTCSetting	On, off	supported
0x86	ResponseOnEvent	All excepted onComparision OfValues	supported
0x86	ResponseOnEvent	onComparision OfValues	NRC “SubFunctionNotSupported”
0x87	LinkControl		Supported (jump to OEM bootloader)

Table 4: Support of UDS Services」()

7.4.2.1 General behavior using DEM interfaces

[Dcm007]「 The DCM requirements use the term DTCStatusAvailabilityMask to indicate the value returned by the function `Dem_GetDTCStatusAvailabilityMask()`.」(BSW04010)

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

[Dcm371]「 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_DisableDTCRecordUpdate`() (which is an exclusive area function concerning to BSW00434) then call `Dem_GetSizeOfExtendedDataRecordByDTC()`, then `Dem_GetExtendedDataRecordByDTC(RecNum)` and finally shall re-enable updates by calling `Dem_EnableDTCRecordUpdate()`.」()

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

[Dcm702] ⌈ If function Dem_DisableDTCRecordUpdate() returns DEM_DISABLE_DTCRECUP_PENDING, the DCM shall retry to get the lock in the next Dcm_MainFunction().⌋()

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

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

Note: The parameter DTCFormat of the functions Dem_ClearDTC(), Dem_SetDTCFilter(), Dem_SetFreezeFrameRecordFilter() and Dem_GetNextFilteredDTCAndFDC() 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.

7.4.2.2 UDS Service 0x10 – Diagnostic Session Control

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

- diagnosticSessionType

[**Dcm250**] ⌈ The DCM module shall implement the UDS Service 0x10 ⌋(BSW04006)

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

If the requested subfunction value is configured, the following steps are processed even if the requested session type is equal to the already running session type (see ISO14229-1 [14] Section 8.2).

[Dcm311]「The sent 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_Dcm_<vendorApiInfix>_DcmDiagnosticSessionControl() with the new diagnostic session type (see **Dcm778**).」()BSW04015)

[Dcm085]「 The DSP submodule shall manage internally a read access for the dataIdentifier 0xF186 (ActiveDiagnosticSessionDataIdentifier) defined in ISO14229-1 [14].」()

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

[Dcm260]「 The DCM module shall implement the UDS Service ECUReset (0x11) 」()

[Dcm373]「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 he 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.

[**Dcm594**]「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_Dcm_<vendorApiInfix>_DcmEcuReset
(SchM_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.

Dcm818 ⌜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.

[Dcm589] ⌜ 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*.⌟()

7.4.2.4 UDS Service 0x14 - Clear Diagnostic Information

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

- groupOfDTC.[

Dcm247] ⌜ The DCM module shall implement UDS Service 0x14.⌟(BSW04010)

[Dcm005] ⌜ Upon reception of a UDS Service ClearDiagnosticInformation (0x14) request with parameter groupOfDTC, the DCM module shall call the operation `ClearDTC` with the following parameter values:
DTC: groupOfDTC from the service request
DTCFormat: DEM_DTC_FORMAT_UDS
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY⌟(BSW04010, BSW04058, BSW04065)

[Dcm705] ⌜ In case Dem_ClearDTC() returns DEM_CLEAR_OK, the DCM module shall send a positive response.⌟()

[Dcm706]「 In case Dem_ClearDTC() returns DEM_CLEAR_PENDING, the DCM shall invoke Dem_ClearDTC() on next Dcm_MainFunction call again. It is up to the DCM to send NRC 78 to respect the response behaviour」()

[Dcm707]「 In case Dem_ClearDTC() returns DEM_CLEAR_FAILED, the DCM shall send a negative response 0x22 – conditionsNotCorrect.」()

[Dcm708]「 In case Dem_ClearDTC() returns DEM_CLEAR_WRONG_DTC, the DCM shall send a negative response 0x31 – requestOutOfRange.」()

7.4.2.5 UDS Service 0x19 – Read DTC Information

[Dcm248]「 The DCM module shall implement the UDS Service 0x19.」(BSW04010)

Dcm739「If a DEM_STATUS_PENDING value is returned from Dem_GetStatusOfDTC, the DCM shall call the API (at least on each Dcm_MainFunction cycle) again as long as DEM_STATUS_PENDING is returned」()

Dcm740「If a DEM_FILTERED_PENDING value is returned from Dem_GetNextFilteredRecord or Dem_GetNextFilteredDTCAndFDC, the DCM shall call the API (at least on each Dcm_MainFunction cycle) again as long as DEM_FILTERED_PENDING is returned」()

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

[Dcm376] ⌈ When sending a positive response to UDS Service 0x19 with subfunction 0x01, 0x07, 0x11 or 0x12, the DCM module shall use the following data in the response message: ⌋(BSW04010)

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

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

	reportNumberOfDTCByStatusMask	**reportNumberOfDTCBySeverityMaskRecord**	**reportNumberOfMirrorMemoryDTCByStatusMask**	**reportNumberOfEmissionsRelatedOBDDTCByStatusMask**
	0x01	0x07	0x11	0x12
DTCStatusMask	DTCStatusMask from request (see Dcm700)	DTCStatusMask from request (see Dcm700)	DTCStatusMask from request (see Dcm700)	DTCStatusMask from request (see Dcm700)
DTCKind	DEM_DTC_GROUP_ALL_DTCS	DEM_DTC_GROUP_ALL_DTCS	DEM_DTC_GROUP_ALL_DTCS	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

⌋(BSW04010, 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.

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

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see Dcm007)
DTCAndStatusRecord	As defined in Dcm008 and Dcm378

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

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

	reportDTCByStatusMask	**reportSupportedDTCs**	**reportMirrorMemoryDTCByStatusMask**	**reportEmissionsRelatedOBDDTCByStatusMask**	**reportDTCWithPermanentStatus**
	0x02	0x0A	0x0F	0x13	0x15
DTCStatusMask	DTCStatusMask from request (see Dcm700)	0x00	DTCStatusMask from request (see Dcm700)	DTCStatusMask from request (see Dcm700)	0x00
DTCKind	DEM_DTC_GROUP_ALL_DTCS	DEM_DTC_GROUP_ALL_DTCS	DEM_DTC_GROUP_ALL_DTCS	EMISSION_REL_DTCS	DEM_DTC_GROUP_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

⌋(BSW04010, BSW04058, BSW04067)

Note:

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

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 :

[Dcm587] ⌈ 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. ⌋()

[Dcm588] ⌈ The DCM shall pad the response with the size calculated when sending the first page. If less DTC matche 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

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

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see Dcm007)
DTCAndSeverityRecord	As defined in Dcm380

⌋()

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

	reportDTCBySeverityMaskRecord
DTCStatusMask	DTCStatusMask from request (see Dcm700)
DTCKind	DEM_DTC_GROUP_ALL_DTCS
DTCFormat	DEM_DTC_FORMAT_UDS
DTCOrigin	PRIMARY_MEMORY
FilterWithSeverity	YES
DTCSeverityMask	DTCSeverityMask from request
FilterForFaultDetectionCounter	NO

」(BSW04010)

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

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

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

- DTCMaskRecord

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

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see Dcm007)

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

」(BSW04010)

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

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

- DTCMaskRecord
- DTCExtendedDataRecordNumber (equal to 0xFF)

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

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

⌋()

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

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

⌋(BSW04010, BSW04058)

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

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

⌋(BSW04010, BSW04058)

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

[Dcm478] ⌈ The DCM module shall obtain the size of the data returned by DEM in `Dem_GetExtendedDataRecordByDTC()` call by using `Dem_GetSizeOfExtendedDataRecordByDTC()`.⌋(BSW04010)

To get the size of all extended data, the DCM module call `Dem_GetSizeOfExtendedDataRecordByDTC()` 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)

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

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

⌋()

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

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

⌋(BSW04010, BSW04058)

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

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

⌋(BSW04010, BSW04058)

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

As required in Dcm478, the DCM module shall obtain the size of the extended data record by using `Dem_GetSizeOfExtendedDataRecordByDTC()`.

`To get the size of all OBD related extended data, the DCM module call Dem_GetSizeOfExtendedDataRecordByDTC() 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)

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

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

⌋()

[Dcm382] ⌈ When responding to UDS Service 0x19 with subfunction 0x06 or 0x10 and DTCExtendedDataRecordNumber different from 0xFF or 0xFE, the DCM module shall calculate the DTCExtendedDataRecord from `Dem_GetExtendedDataRecordByDTC()` with the following parameter values.⌋(BSW04010, 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 Dcm478, the DCM module shall obtain the size of the extended data record by using `Dem_GetSizeOfExtendedDataRecordByDTC()`.

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

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

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

Parameter name	**Value**
DTCRecord/ DTCSnapshotRecordNumber	As defined in Dcm299

⌋()

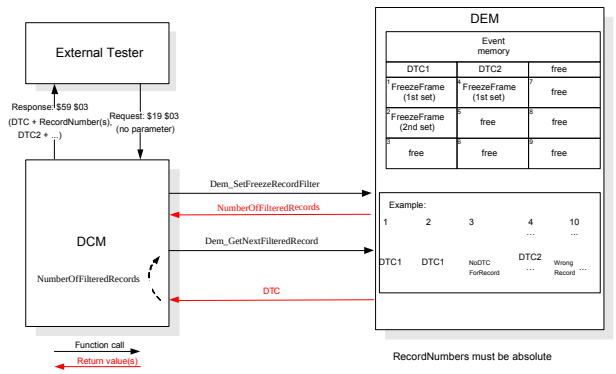


Figure 7: Request DTC Snapshot Record Identification

[Dcm298] ⌈ The DSP submodule shall call `Dem_SetFreezeFrameRecordFilter()` that returns the NumberOfFilteredRecords value with DTCFormat equal to DEM_DTC_FORMAT_UDS.⌋(BSW04010)

[Dcm299] ⌈ When responding to UDS Service 0x19 with subfunction 0x03, the DCM module shall obtain the consecutive DTCs and DTCSnapshotRecordNumbers by repeatedly calling `Dem_GetNextFilteredRecord()`.⌋(BSW04010)

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 Dcm587 and Dcm588 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

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

Parameter name	Value
DTCAndStatusRecord	DTC from the request, statusOfDTC according to Dcm383
DTCSnapshotRecordNumber	The DTCSnapshotRecordNumber is contained in the output buffer from the `Dem_GetFreezeFrameDataByDTC()` call. See Dcm384
DTCSnapshotRecordNumberOfIdentifiers/ DTCSnapshotRecord	As defined in Dcm384

⌋()

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

Parameter name	Value
DTCAndStatusRecord	DTC from the request, statusOfDTC according to Dcm383
DTCSnapshotRecordNumber	The DTCSnapshotRecordNumber is contained in the output buffer from the `Dem_GetFreezeFrameDataByDTC()` call. See Dcm385
DTCSnapshotRecordNumberOfIdentifiers/ DTCSnapshotRecord	As defined in Dcm385

」()

[Dcm383] 「 When responding to UDS Service 0x19 with subfunction 0x04, the DCM module shall obtain the status of the DTC by calling `Dem_GetStatusOfDTC()` with the following parameters:
DTC: DTC from the request
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY」(BSW04010, BSW04058)

[Dcm384] 「 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_GetFreezeFrameDataByDTC()` with the following parameter values:
DTC: DTCMaskRecord from the request in UDS format
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY
RecordNumber: DTCSnapshotRecordNumber from the request」(BSW04010, BSW04058)

[Dcm385] 「 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_GetFreezeFrameDataByDTC()` with the following parameter values:
DTC: DTCMaskRecord from the request in UDS format
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY
RecordNumber: value from 0x00 -> 0xFE」(BSW04010, BSW04058)

[Dcm441] 「 The DCM module shall obtain the size of the data returned by DEM in `Dem_GetFreezeFrameDataByDTC()` call by using `Dem_GetSizeOfFreezeFrameByDTC().`」(BSW04010, BSW04079)

To get the size of all FreezeFrame data, the DCM module call `Dem_GetSizeOfFreezeFrameByDTC()` 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.

[Dcm632] ⌈ 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).⌋()

[Dcm574] ⌈ 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 Dcm388, statusOfDTC according to Dcm389
DTCSnapshotRecordNumberOfIdentifiers / DTCSnapshotRecord	As defined in Dcm388

⌋()

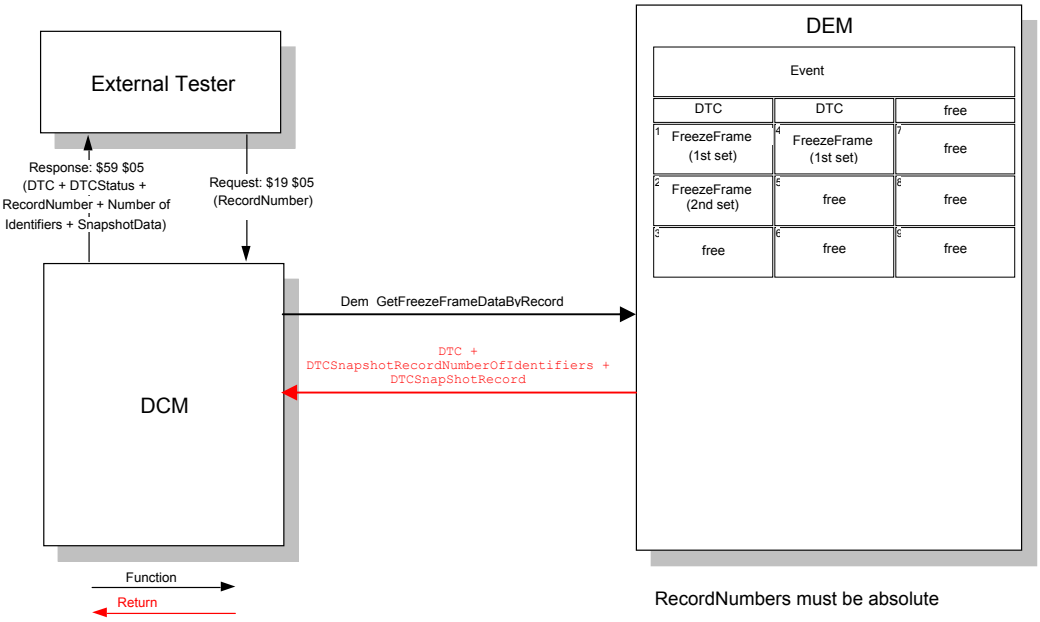


Figure 8: Request DTC Snapshot Record by Snapshot Record Number

[Dcm388] ⌈ 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_GetFreezeFrameDataByRecord()` with the following parameter values:
RecordNumber: DTCSnapshotRecordNumber from the request (0x00)
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 ⌋(BSW04010, BSW04058)

[Dcm389] ⌈ 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_GetStatusOfDTC() with the following parameters:
DTC: DTC as defined in Dcm388

DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY ⌋(BSW04010, BSW04058)

The function `Dem_GetFreezeFrameDataByRecord()` 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_GetFreezeFrameDataByRecord()` 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.

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

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

」()

[Dcm393]「 For the purpose of responding to UDS Service 0x19 with subfunctions 0x0B, 0x0C, 0x0D or 0x0E, the DCM module shall obtain the StatusOfDtc by calling `Dem_GetStatusOfDTC()` with the following parameter values:
DTC: the DTC value as defined in Dcm466
DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY」(BSW04010, BSW04058)

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

	reportFirstTestFailedDTC	**reportFirstConfirmedDTC**	**reportMostRecentTestFailedDTC**	**reportMostRecentConfirmedDTC**
	0x0B	**0x0C**	**0x0D**	**0x0E**
DTCRequest	FIRST_ FAILED_DTC	FIRST_ DET_CONFIRMED_DTC	MOST_RECENT_ FAILED_DTC	MOST_REC_ DET_CONFIRMED_DTC

」(BSW04010)

Dcm766「If the Dcm received DEM_OCCURR_NOT_AVAILABLE by calling Dem_GetDTCByOccurrenceTime 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.

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

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

」(BSW04010)

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

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

⌋(BSW04010, BSW04058)

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

[Dcm519] ⌈ The calls to Dem_SetDTCFilter() with parameter FilterForFaultDetectionCounter set to YES shall be done in the context of the Dcm_MainFunction()⌋(BSW04010)

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 Dcm587 and Dcm588 shall be considered for the implementation of this subservice.

7.4.2.6 UDS Service 0x22 - ReadDataByIdentifier

[Dcm253] ⌈ 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.

[Dcm438] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22)) with DID in the range 0x00FF to 0xF1FF or 0xF400 to 0xFEFF, 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).⌋()

[Dcm651] ⌈ 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).⌋()

[Dcm652] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22) with DID in the range 0xF200 to 0xF3FF, if verification has been successfully done (see Dcm651) and the dynamic DID has been defined with a DID source (See Dcm646), the DCM module shall use the configuration of this DID source to read the data.⌋()

[Dcm653] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22) with DID in the range 0xF200 to 0xF3FF, if verification has been successfully done (see Dcm651) and the dynamic DID has been defined with a memory address (See Dcm646), the DCM module shall use the callout Dcm_ReadMemory to read the data.⌋()

[Dcm561] ⌈ If a DID is set as unused (DcmDspDidUsed set to FALSE), the DCM shall consider the DID as not supported (according to Dcm438)⌋()

[Dcm433] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID the DCM module shall check if the DID has a Read access configured (see configuration parameter ***DcmDspDidRead*** in ***DcmDspDidAccess***). If none of the DID has a Read access, the DCM module shall send NRC 0x31 (Request out of range).⌋()

[Dcm434] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID the DCM module shall check if the DID can be read in the current session (see configuration parameter ***DcmDspDidReadSessionRef***). If none of the DID can be renden in the current session, the DCM module shall send a NRC 0x31 (Request out of Range).⌋()

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

Dcm819 ⌈ 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.⌋()

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

7.4.2.6.1 **Non-OBD DID**

[Dcm578] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID outside the OBD range (F400-F8FF), after all verification (see Dcm433, Dcm434 and Dcm435), If the data is configured as a “ECU signal” of the IoHwAb (parameter DcmDspDataUsePort), the DCM shall call the Api IoHwAb_Dcm_Read_<symbolic name of ECU signal (parameter DcmDspDataReadEcuSignal)>() to get the Data. In this case, the requirements Dcm439, Dcm436 and Dcm437 shall not apply.⌋()

[Dcm439] ⌈ 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.⌋()

[Dcm436] ⌈ 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 or call the associated *ReadDataLength* operation (if parameter ***DcmDspDataUsePort*** set to *USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER*). ⌋()

[Dcm437] ⌈ After all verification (see Dcm433, Dcm434, Dcm435 and Dcm436) 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*). ⌋()

[Dcm560] ⌈ 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*) ⌋()

[Dcm638] ⌈ 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 integer types for data transmitted in ReadDataByIdentifier response. (when sender/receiver interface DataServices_<Data> is used and reading from IoHwAb). ⌋()

7.4.2.6.2 **OBD DID**

[Dcm481] ⌈ 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 Dcm407 , Dcm408) ⌋()

[Dcm482] ⌈ 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 Dcm415 , Dcm416)⌋()

[Dcm483] ⌈ 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 Dcm422 , Dcm423)⌋()

7.4.2.7 UDS Service 0x24 - ReadScalingDataByIdentifier

[Dcm258] ⌈ 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***)

[Dcm394] ⌈ 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

[Dcm252] ⌈ 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.

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

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

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

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".

[Dcm325] ⌈ 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

[Dcm254] ⌈ The DSP submodule shall implement the UDS Service ReadDataByPeriodicIdentifier (0x2A)⌋()

Dcm721: ⌈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).⌋()

Dcm722:「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).」()

Dcm820「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***」()

[Dcm716]「 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 integer types for data transmitted in ReadDataByPeriodicIdentifier response (when sender/receiver interface DataServices_<Data> is used).」()

7.4.2.10 UDS Service 0x2C - DynamicallyDefineDataIdentifier

[Dcm259]「 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.

[Dcm646]「 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.」()

[Dcm647]「 On reception of service DynamicallyDefineDataIdentifier with subservice clearDynamicallyDefinedDataIdentifier, the DCM module shall remove the configuration of this DID.」()

Dcm723:「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 ***DcmDspDidWriteSessionRef***). If not, the DCM module shall send a NRC 0x31 (Request out of Range).」()

Dcm724:「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 ***DcmDspDidWriteSecurityLevelRef***). If not, the DCM module shall send NRC 0x33 (Security access denied). 」()

Dcm725:「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 session (see configuration parameter of referenced DID or memoryRange). If not, the DCM module shall send a NRC 0x31 (Request out of Range).」()

Dcm726:「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 or memoryRange). If not, the DCM module shall send a NRC 0x33 (Security access denied).」()

Dcm821「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 or memoryRange). 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.

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 or write the data.

7.4.2.11 UDS Service 0x2E - WriteDataByIdentifier

[Dcm255] ⌈ The DCM module shall implement the UDS Service WriteDataByIdentifier (0x2E) of the Unified Diagnostic Services.⌋()

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

[Dcm467] ⌈ On reception of the UDS Service WriteDataByIdentifier (0x2E) with DID in the range 0x00FF to 0xF1FF , 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).⌋()

[Dcm562] ⌈ If a DID is set as unused (DcmDspDidUsed set to FALSE), the DCM shall consider the DID as not supported (according to Dcm467)⌋()

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

[Dcm469] ⌈ On reception of the UDS Service WriteDataByIdentifier (0x2E), the DCM module shall check if the DID can be written in the current session (see configuration parameter ***DcmDspDidWriteSessionRef***). If not, the DCM module shall send a NRC 0x31 (Request Out of Range).⌋()

[Dcm470] ⌈ 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).⌋()

Dcm822 ⌜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***.

⌟()**[Dcm473]** ⌜ 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)⌟()

[Dcm395] ⌜ After all verification (see Dcm467, Dcm468, Dcm469, Dcm470, Dcm473) 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)⌟()

[Dcm541] ⌜ If the data is configured as a BlockId of the NvRam (parameter *DcmDspDataUsePort*), the DCM shall call the Api NvM_SetBlockLockStatus(), to lock the NvM Block and avoid its update by the application, and the Api NvM_WriteBlock() with the BlockId (parameter *DcmDspDataBlockIdRef*) ⌟()

[Dcm620] ⌜ 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

[Dcm639] ⌜ 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 integer types for data received in WriteDataByIdentifier response. (when sender/receiver interface DataServices_<Data> is used).⌟()

7.4.2.12 UDS Service 0x2F - InputOutputControlByIdentifier

[Dcm256] ⌈ 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***)

[Dcm579] ⌈ The DCM shall support the following InputOutputControlParameter definitions:

Hex	Description
00	returnControlToECU
01	resetToDefault
02	freezeCurrentState
03	shortTermAdjustment

⌋()

[Dcm563] ⌈ On reception of the UDS Service InputOutputControlByIdentifier (0x2F), 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).⌋()

[Dcm564] ⌈ If a DID is set as unused (DcmDspDidUsed set to FALSE), the DCM shall consider the DID as not supported (according to Dcm563) ⌋()

[Dcm565] ⌈ 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). ⌋()

[Dcm566] ⌈ 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). ⌋()

[Dcm567] ⌈ 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). ⌋()

Dcm823 ⌈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. ⌋()

[Dcm580] ⌈ On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) , if all verifications have been successfully done (see Dcm563, Dcm565, Dcm566, Dcm567) 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 Dcm396, Dcm397, Dcm398 and Dcm399 doesn’t apply. ⌋()

[Dcm581] ⌈ 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).

[Dcm680] ⌈ 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.

[Dcm396] ⌈ On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to returnControlToEcu, if all verifications have been successfully done (see Dcm563, Dcm565, Dcm566, Dcm567), the DCM module shall invoke all the configured function (if parameter ***DcmDspDataUsePort*** set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC; see configuration parameter ***DcmDspDataReturnControlToEcuFnc***) or 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.⌋()

[Dcm397] ⌈ On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to resetToDefault, if all verifications have been successfully done (see Dcm563, Dcm565, Dcm566, Dcm567), the DCM module shall invoke all the configured function (if parameter ***DcmDspDataUsePort*** set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC; see configuration parameter ***DcmDspDataResetToDefaultFnc***) or 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.⌋()

[Dcm398] ⌈ On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to freezeCurrentState, if all verifications have been successfully done (see Dcm563, Dcm565, Dcm566, Dcm567), the DCM module shall invoke all the configured function (if parameter ***DcmDspDataUsePort*** set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC; see configuration parameter ***DcmDspDataFreezeCurrentStateFnc***) or 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.⌋()

[Dcm399] ⌈ On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to shortTermAdjustment, if all verifications have been successfully done (see Dcm563, Dcm565, Dcm566, Dcm567), the DCM module shall invoke all the configured function (if parameter ***DcmDspDataUsePort*** set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC; see configuration parameter ***DcmDspDataShortTermAdjustmentFnc***) or 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.⌋()

[Dcm626] ⌈ On S3Server timeout, 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*)⌋()

[Dcm627] ⌈ On protocol preemption, 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*)⌋()

[Dcm628] ⌈ 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*)⌋()

[Dcm640] ⌈ 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 integer types for data received and transmitted in InputOutputControlByIdentifier service.(when reading from IoHwAb)⌋()

[Dcm682] ⌈ The ControlStatusRecord for the IoControl response shall be retrieved using the associated ReadData operations⌋()

7.4.2.13 UDS Service 0x31 - RoutineControl

[Dcm257] ⌈ 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***)

[Dcm568] ⌈ 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).⌋()

[Dcm569] ⌈ If a Routine is set as unused (DcmDspRoutineUsed set to FALSE), the DCM shall consider the Routine as not supported (according to Dcm568)⌋()

[Dcm570] ⌈ 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).⌋()

[Dcm571] ⌈ 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).⌋()

Dcm824 ⌈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.⌋()

[Dcm590] ⌈ When receiving a request for UDS Service RoutineControl (0x31) if all verifications have been successfully done (see Dcm568, Dcm570, Dcm571), 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***)⌋() **[Dcm400]** ⌈ When receiving a request for UDS Service RoutineControl (0x31) with subfunction startRoutine, if all verifications have been successfully done (see Dcm568, Dcm570, Dcm571), the DCM module shall call the configured `Xxx_Start()` function passing the dataIn, calculated from routineControlOptionRecord (see Dcm590), 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 bits in currentDataLength parameter. The datalength can be dynamic only on the last dataIn parameter.⌋()

[Dcm401] ⌈ Upon completing Dcm400, when `Xxx_Start()` returns no ErrorCode, the DCM module shall reply with a positive response with the data returned by `Xxx_Start()` in the 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 is provided in bits in currentDataLength parameter. The datalength can be dynamic only on the last dataOut parameter.⌋()

[Dcm402] ⌈ When receiving a request for UDS Service RoutineControl (0x31) with subfunction stopRoutine, if all verifications have been successfully done (see Dcm568, Dcm570, Dcm571), the DCM module shall call the configured `Xxx_Stop()` function passing the dataIn, calculated from routineControlOptionRecord (see Dcm590), 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 currentDataLength parameter. The datalength can be dynamic only on the last dataIn parameter.⌋()

[Dcm403] ⌈ Upon completing Dcm402, when `Xxx_Stop()` returns no ErrorCode, the DCM module shall reply with a positive response with the data returned by `Xxx_Stop()` in the 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 is provided in currentDataLength parameter.The datalength can be dynamic only on the last dataOut parameter.⌋()

[Dcm404] ⌈ When receiving a request for UDS Service RoutineControl (0x31) with subfunction requestRoutineResults, if all verifications have been successfully done (see Dcm568, Dcm570, Dcm571), 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***).⌋()

[Dcm405] ⌈ Upon completing Dcm404, when `Xxx_RequestResults()` returns no ErrorCode, the DCM module shall reply with a positive response with the data returned by `Xxx_RequestResults()` in 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 is provided in currentDataLength parameter.The datalength can be dynamic only on the last dataOut parameter.⌋()

[Dcm641] ⌈ 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 integer types for data received and transmitted in RoutineControl service.⌋()

[Dcm701] ⌈ 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 Dcm418 , Dcm419). ⌋()

7.4.2.14 UDS Service 0x3E - Tester Present

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

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

7.4.2.15 UDS Service 0x85 - ControlDTCSetting

[Dcm249] ⌈ The DCM module shall implement UDS Service ControlDTCSetting (0x85) for DTCSettingType on or off.⌋()

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

[Dcm304] ⌈On invocation of the sent confirmation function of the UDS Service 0x85 from DSD with DTCSettingType=on of the request, the DCM module shall call `Dem_EnableDTCSetting()`.⌋(BSW04010)

Dcm783 ⌈On invocation of the sent confirmation function of the UDS Service 0x85 from DSD with DTCSettingType=on of the request (see [Dcm304), the DCM shall invoke a mode switch of the *ModeDeclarationGroupPrototype* DcmControlDTCSetting by calling SchM_Switch_DCM_<vendorApiInfix>_ DcmControlDTCSetting (SCHM_MODE_ DcmControlDTCSetting _ENABLEDTCSETTING).⌋()

[Dcm406] ⌈On invocation of the sent confirmation function of the UDS Service 0x85 from DSD with DTCSettingType=off of the request, the DCM module shall call `Dem_DisableDTCSetting()`.⌋(BSW04010)

Dcm784 ⌈On invocation of the sent confirmation function of the UDS Service 0x85 from DSD with DTCSettingType=off of the request (see Dcm304), the DCM shall invoke a mode switch of the *ModeDeclarationGroupPrototype* DcmControlDTCSetting by calling SchM_Switch_DCM_<vendorApiInfix>_ DcmControlDTCSetting (SCHM_MODE_ DcmControlDTCSetting _DISABLEDTCSETTING).⌋()

[Dcm751] ⌈In case the DTCSetting is disabled, the DCM module shall call `Dem_EnableDTCSetting()` and doing the mode switch of the *ModeDeclarationGroupPrototype* DcmControlDTCSetting to ENABLEDTCSETTING while transitioning to default session or upon any diagnostic session change where the new session do not support UDS Service ControlDTCsetting anymore.⌋()

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

[Dcm752] ⌈Furthermore in case the DTCSetting is disabled, the DCM module shall call `Dem_EnableDTCSetting()` and doing the mode switch of the *ModeDeclarationGroupPrototype* DcmControlDTCSetting to ENABLEDTCSETTING in case at least one referenced arbitrary *ModeDeclarationGroupPrototypes* (see configuration parameter ***DcmDspControlDTCSettingReEnableModeRuleRef***) for service ControlDTCSetting (0x85) with DTCSettingType off (0x02) are not fulfilled anymore.⌋()

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)

7.4.2.16 UDS Service 0x3D – WriteMemoryByAddress

[Dcm488] ⌈ 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.

[Dcm489] ⌈ 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).⌋()

[Dcm490] ⌈ 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).⌋()

Dcm825 ⌈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.⌋()

[Dcm491] ⌈ On reception of the UDS Service WriteMemoryByAddress (0x3D), and after verification of the validity of the request (see **[Dcm489** and **[Dcm490**) the DCM module shall call the callout Dcm_WriteMemory().⌋()

7.4.2.17 UDS Service 0x23 – ReadMemoryByAddress

[Dcm492] ⌈ 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.

[Dcm493] ⌈ 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).⌋()

[Dcm494] ⌈ 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).⌋()

Dcm826 ⌈ 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.⌋()

[Dcm495] ⌈ On reception of the UDS Service ReadMemoryByAddress (0x23), and after verification of the validity of the request (see **[Dcm493** and **[Dcm494**) the DCM module shall call the callout Dcm_ReadMemory().⌋()

7.4.2.18 UDS Service 0x34 – RequestDownload

[Dcm496]「 The DCM module shall implement the RequestDownload (service 0x34) of the Unified Diagnostic Services.」(BSW04033)

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

7.4.2.19 UDS Service 0x35 – RequestUpload

[Dcm499]「 The DCM module shall implement the RequestUpload (service 0x35) of the Unified Diagnostic Services.」(BSW04033)

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

7.4.2.20 UDS Service 0x36 – TransferData

[Dcm502]「 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.

[Dcm503]「 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)

[Dcm504]「 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)

[Dcm645]「 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

[Dcm505] ⌈ 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

[Dcm511] ⌈ The DCM module shall implement the CommunicationControl (service 0x28) of the Unified Diagnostic Services. ⌋()

[Dcm512] ⌈ 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). ⌋()

Dcm785 ⌈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 ***DcmDslProtocolRxComMChannelRef***) 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) ⌋()

Dcm786 「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:

[Dcm753] 「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」()

7.4.2.23 UDS Service 0x87 – LinkControl

This service is used to gain bus bandwidth for diagnostic purposes

Dcm743 「The DCM module shall implement the LinkControl (service 0x87) of the Unified Diagnostic Services」()

Dcm744 「Upon receiving a request for UDS Service 0x87, the DCM shall jump to OEM bootloader (see Dcm533)」()

It is possible to disable the jump to bootloader upon the reception of a Linkcontrol request. To do so, a vendor specific service implementation callout should be configured to manage an external processing (see chapter 8.9 External diagnostic service processing)

7.4.3 OBD Services

7.4.3.1 Overview

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

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

Table 5: Support for OBD services in the DCM

7.4.3.2 General behavior

In many cases, the DCM protocol allows the bundling of several requests (for example several “PIDs”) and the corresponding bundling of the responses. The descriptions of the behavior for the individual services do not explicitly consider this. As the DCM needs to comply with OBD standard (as is defined through various

requirements below), the DCM might need to repeat the steps defined below to parse a request and assemble a valid response.

[Dcm077] ⌈ When calling the DEM module for OBD services, the DCM module shall use the following values for the parameter DTCOrigin:
Service $0A uses DEM_DTC_ORIGIN_PERMANENT_MEMORY
All other services use DEM_DTC_ORIGIN_PRIMARY_MEMORY ⌋(BSW04058)

7.4.3.3 OBD Service $01 – Request Current Powertrain diagnostic Data

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

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

To obtain the value for a PID, the DCM uses the configured `Xxx_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.

[Dcm408] ⌈ On reception of an OBD Service $01 request with one or more PIDs that are not "availability PIDs", the DCM shall obtain the current value of these PIDs by invoking the configured `Xxx_ReadData()` functions for every data of the PID and shall return these values as response to Service $01.⌋()

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.

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.

[Dcm621] ⌈ If a PID contains support information (presence of ***DcmDspPidDataSupportInfo*** container) the DCM shall add the support information in the diagnostic response.⌋()

[Dcm622] ⌈ 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.

[Dcm623] ⌈ When responding to OBD Service $01, the DCM shall add fill byte in the PID in order to fit to the PID size (see configuration parameter ***DcmDspPidSize***) ⌋()

[Dcm718] ⌈ 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 integer types for data received in Mode 1 response (when sender/receiver interface DataServices_<Data> is used). ⌋()

7.4.3.4 OBD Service $02 – Request Power Train FreezeFrame Data

[Dcm244] ⌈ The DCM shall implement OBD Service $02 (Request Power Train FreezeFrame Data) in compliance to all provisions of the OBD standard. ⌋(BSW04082, BSW04001)

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

[Dcm409] ⌈ The DCM shall ignore all requests regarding record-numbers that are not 0 ⌋()

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

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

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

[Dcm279] ⌈ On reception of a request for Service $02 with PID $02, the DCM shall call `Dem_GetDTCOfOBDFreezeFrame()` with FrameNumber set to 0x00 to get the DTC number ⌋(BSW04010, 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_GetDTCOfOBDFreezeFrame (). See DEM specification on how this is done.

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

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

[Dcm284] ⌈ On reception of a service $02 request with an “availability PID”, the DCM shall respond with the corresponding supported (=configured) PIDs encoded according to the OBD standard. ⌋(BSW04010)

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

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

[Dcm286] ⌈ On reception of a service $02 request with a PID that is not an “availability PID” and is not $02, the DCM shall call `Dem_ReadDataOfOBDFreezeFrame()` 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) ⌋(BSW04010)

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

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

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

[Dcm245] ⌈ The DCM module shall implement OBD Service $03 (Request emission-related diagnostic trouble codes) in compliance to all provisions of the OBD standard.⌋(BSW04082, BSW04001)

[Dcm410] ⌈ The DCM module shall implement OBD Service $07 (Request Emission-Related Diagnostic Trouble Codes Detected during Current or Last Completed Driving Cycle) in compliance to all provisions of the OBD standard.⌋(BSW04082, BSW04001)

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

[Dcm289] ⌈ When receiving a request for OBD Service $03, the DCM module shall obtain from the DEM all DTCs in primary memory and with a “confirmed” status using the functions `Dem_SetDTCFilter()` and `Dem_GetNextFilteredDTC()`.⌋(BSW04010)

[Dcm412] ⌈ When receiving a request for OBD Service $07, the DCM module shall obtain from the DEM module all DTCs in primary memory with a “pending” status using the functions `Dem_SetDTCFilter()` and `Dem_GetNextFilteredDTC()`.⌋(BSW04010)

[Dcm330] ⌈ When receiving a request for OBD Service $0A, the DCM module shall obtain from the DEM all DTCs stored in permanent memory using the functions `Dem_SetDTCFilter()` and `Dem_GetNextFilteredDTC()`.⌋(BSW04010)

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

Parameters to Dem_SetDTCFilter			
OBD Service	$03	$07	$0A

DTCStatusMask	0x08 (confirmed bit set)	0x04 (pending bit set)	0x00
DTCKind	DEM_DTC_KIND_ EMISSION_REL_DT CS	DEM_DTC_KIND_ EMISSION_REL_DT CS	DEM_DTC_KIND_ EMISSION_REL_DTC S
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 Dcm587 and Dcm588 shall be considered for the implementation of this service.

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

[Dcm246] ⌈ The DCM module shall implement OBD Service $04 (Clear/reset emission-related diagnostic information) in compliance to all provisions of the OBD standard.⌋(BSW04082, BSW04001)

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

[Dcm004] ⌈ When receiving a request for OBD Service $04, the DCM module shall call the operation `ClearDTC` with the following parameter values:
DTC = DEM_DTC_GROUP_ALL_DTCS
DTCFormat: DEM_DTC_FORMAT_OBD
DTCOrigin = DEM_DTC_ORIGIN_PRIMARY_MEMORY⌋(BSW04010, BSW04058, BSW04065)

[Dcm413] ⌈ In case Dem_ClearDTC() returns DEM_CLEAR_OK, the DCM module shall send a positive response.⌋(BSW04010)

[Dcm703] ⌈ In case Dem_ClearDTC() returns DEM_CLEAR_PENDING, the DCM shall invoke Dem_ClearDTC() on next Dcm_MainFunction call again. It is up to the DCM to send NRC 78 to respect the response behaviour ⌋()

[Dcm704] ⌈ In case Dem_ClearDTC() returns DEM_CLEAR_FAILED, the DCM shall send a negative response 0x22 – conditionsNotCorrect. ⌋()

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

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

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

- A list of TIDs (see configuration parameter ***DcmDspTestResultObdmidTids***)
- And for each TID:
  - The name of the port through which the DCM module has access to a DTRService interface (see configuration parameter DcmDspTestResultTid and DcmDspTestResultObdmidTid)
  - The Unit and Scaling ID to be returned with the service for the given OBDMID and TID (see configuration parameter ***DcmDspTestResultObdmidTidUaSid***)

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

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

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

Dcm416 implies that DTRs for which the function `Xxx_GetDTRValue()` returns a status of DCM_DTRSTATUS_INVISIBLE are ignored.
When using the `Xxx_GetDTRValue()` functions, the DCM module provides buffers in which the DTR can be stored. The actual implementation of the `Xxx_GetDTRValue()` function for a specific OBDMID is provided by a SW-C. The origin of the function is not known to the DCM but is part of the DCM configuration.

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

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

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

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

[Dcm420] ⌈ After the execution of Dcm419, the Dcm module shall respond to the service request using the data stored by the RequestControl function in the OutBuffer.⌋()

7.4.4.4 OBD Service $09 – Request Vehicle Information

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

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

[Dcm423] ⌈ On reception of an OBD Service $09 request for an InfoType that is not an "availability InfoType", the DCM module shall obtain the value of this InfoType by invoking all the configured `Xxx_GetInfotypeValueData()` function for every data of this InfoType and shall return the value as response to Service $09⌋()

[Dcm684] ⌈ 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 .⌋()

7.4.5 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.

7.4.5.1 Jump to Bootloader

[Dcm531] ⌈ A jump to bootloader is possible only with services DiagnosticSessionControl and LinkControl services⌋(BSW04098)

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

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

[Dcm533] ⌈ On reception of service LinkControl, the DCM shall prepare the jump to the OEM bootloader (see Dcm535 and Dcm538) by triggering the mode switch of *ModeDeclarationGroupPrototype* DcmEcuReset to JUMPTOBOOTLOADER⌋(BSW04098)

[Dcm654] ⌈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.

[Dcm719] ⌈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 [Dcm719, the exact response handling depends on the state of the ‘suppressPosRspMsgIndicationBit’ (TRUE or FALSE) in the request message.

[Dcm535] ⌈ If the jump to bootloader is requested (see [Dcm532, [Dcm592 and [Dcm533) and the configuration parameter DcmSendRespPendOnTransToBoot is set to TRUE (See [Dcm654), the DCM shall call Dcm_SetProgConditions() after confirmation of the 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.

[Dcm720] ⌈ If the jump to bootloader is requested (see [Dcm532, [Dcm592 and [Dcm533) and the configuration parameter DcmSendRespPendOnTransToBoot is set to FALSE (see [Dcm719), the DCM shall call Dcm_SetProgConditions() immediately.⌋()

[Dcm715] ⌈ If the jump to bootloader is requested (see [Dcm532, [Dcm592 and [Dcm533) and if the call to *Dcm_SetProgConditions()* returns E_NOT_OK (see [Dcm535), the DCM shall not request any reset, shall not perform the jump to bootloader, and shall answer negatively to the request with NRC 0x22 (Conditions not correct).⌋()

7.4.5.2 Jump from Bootloader

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

[Dcm537] If the initialization of the DCM is the consequence of a jump from the bootloader (see **[Dcm536**), the DCM shall call ComM_DCM_ActiveDiagnostic(`NetworkId`) to request the ComManager for the full communication mode.

Dcm767 ⌈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)

Dcm768 ⌈If the initialization of the DCM is the consequence of a jump from the bootloader (see **[Dcm536**) 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.⌋()

7.4.5.3 Flags management

On reception of a UDS Service 0x10 request (Diagnostic Session Control) with subfunction 0x02 (Start Programming Session) or UDS Service 0x87 (LinkControl), the DCM shall set the ReprogramingRequest flag and, if indicated for this service, the ResponseRequired flag.

After an ECU reset, when the flashloader is started, it will clear the ReprogramingRequest flag and - if required - send a response and clear the ResponseRequired flag.
After successful reprograming of the application software, the flashloader will set the ApplUpdated flag and, if configured, the ResponseRequired flag.

After an ECU reset, when the newly programmed application is started for the first time, the DCM shall clear the ApplUpdated flag and, if required, send a response and clear the ResponseRequired flag.

7.5 Error classification

This section describes how the DCM module has to treat the several error classes that may happen during the life cycle of the DCM module.
The general requirements document of AUTOSAR [1] specifies that all Basic Software modules shall distinguish (according to the product life cycle) two error types:
- Development errors: those errors shall be detected and fixed during the development phase. In most cases, those errors are software errors. The detection of errors that shall only occur during development can be switched off for production code (by static configuration namely preprocessor switches).
- Production errors: those errors are hardware errors and software exceptions that cannot be avoided and are expected to occur in production (i.e. series) code.

[**Dcm012**] ⌜ The DCM module shall provide no Production-Errors ⌟(BSW00337)

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

[Dcm044] ⌜ The used return values shall be the same for development and production. Only the values given by the DCM SWS shall be used. ⌟(BSW00369)

[Dcm364] ⌜ Development error values are of type uint8. ⌟(BSW00327, BSW00331)

[Dcm040] ⌜ 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 Dcm120)	Development	DCM_E_INTERFACE_TIMEOUT	0x01
Interface return-value is out of range	Development	DCM_E_INTERFACE_RETURN_VALUE	0x02
Interface: Boundary check of buffers provided by the Dcm failed during interaction with another module (application, Dem, PduR, etc.)	Development	DCM_E_INTERFACE_BUFFER_OVERFLOW	0x03
Internal: DCM not initialized	Development	DCM_E_UNINIT	0x05
DCM API function with invalid input parameter	Development	DCM_E_PARAM	0x06
DCM API service invoked with NULL POINTER as parameter	Development	DCM_E_PARAM_POINTER	0x07

[Dcm041] ⌈ Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the DCM module implementation specification. The classification and enumeration shall be compatible to the errors listed above ⌋(BSW00337)

7.6 Error detection

[Dcm042] ⌈ The detection of development errors is configurable (ON/OFF) at pre-compile time. The configuration parameter ***DcmDevErrorDetect*** shall activate or deactivate the detection of all development errors. ⌋(BSW00338, BSW00350)

[Dcm043] ⌈ If the development error detection is enabled for this module, for every request except Dcm_Init, it shall be ensured that the DCM module is already initialized. Detected errors shall be reported to the Development Error Tracer. ⌋(BSW00442, BSW00323)

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

7.7 Error notification

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

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

[Dcm052] ⌈ The header file of the DCM, DCM.h, shall provide a module ID called DCM_MODULE_ID set to the value 0x35.⌋()

7.8 Debugging

[Dcm484] ⌈ Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable.⌋(BSW00442)

[Dcm485] ⌈ All type definitions of variables which shall be debugged, shall beaccessible by the header file Dcm.h.⌋(BSW00442)

[Dcm486] ⌈ The declaration of variables in the header file shall be such, that it is possible to calculate the size of the variables by C-“sizeof”.⌋(BSW00442)

[Dcm487] ⌈ Variables available for debugging shall be described in the respective Basic Software Module Description⌋(BSW00442)

[Dcm506] ⌈ The current status of diagnostic activity (linked to ComM_DCM_InactiveDiagnostic(`NetworkId`) and ComM_DCM_ActiveDiagnostic(`NetworkId`) call) shall be available for debugging.⌋(BSW00442)

[Dcm507] ⌈ The current security level shall be available for debugging.⌋(BSW00442)

[Dcm508] ⌈ The current session state shall be available for debugging.⌋(BSW00442)

[Dcm509] ⌈ The current protocol shall be available for debugging.⌋(BSW00442)

7.9 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 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 Dcm686).
There is no correlation to the operation signature (i.e. existence of OpStatus parameter and E_PENDING return code) that demands AsynchronousServerCallPoint or SynchronousServerCallPoint usage.

7.10 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*** (see Dcm938_Conf **:**) and ***DcmDspDidRangeIdentifierUpperLimit*** (see Dcm939_Conf **:**) which delimited the range of the DIDs. ***DcmDspDidRangeMaxDataLength (***see Dcm940_Conf **:**) and ***DcmDspDidRangeHasGaps*** (see Dcm Dcm941_Conf **:**)

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.
[Dcm333]

⌈

Module	Imported Type
ComStack_Types	BufReq_ReturnType
	NetworkHandleType
	NotifResultType
	PduIdType
	PduInfoType
	PduLengthType
	RetryInfoType
	TPParameterType
Dem	Dem_DTCFormatType
	Dem_DTCGroupType
	Dem_DTCKindType
	Dem_DTCOriginType
	Dem_DTCRequestType
	Dem_DTCSeverityType
	Dem_DTCTranslationFormatType
	Dem_FilterForFDCType
	Dem_FilterWithSeverityType
	Dem_ReturnControlDTCSettingType
	Dem_ReturnDisableDTCRecordUpdateType
	Dem_ReturnGetDTCByOccurrenceTimeType
	Dem_ReturnGetExtendedDataRecordByDTCType
	Dem_ReturnGetFreezeFrameDataByDTCType
	Dem_ReturnGetFreezeFrameDataByRecordType
	Dem_ReturnGetFunctionalUnitOfDTCType
	Dem_ReturnGetNextFilteredDTCType
	Dem_ReturnGetNumberOfFilteredDTCType
	Dem_ReturnGetSeverityOfDTCType
	Dem_ReturnGetSizeOfExtendedDataRecordByDTCType
	Dem_ReturnGetSizeOfFreezeFrameByDTCType
	Dem_ReturnGetStatusOfDTCType
	Dem_ReturnSetFilterType
GENERIC TYPES	<EcuSignalDataType>
	<datatype>
NvM	NvM_BlockIdType

SchM	SchM_ReturnType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋()

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

```
ImplementationDataType SecLevelType {
      LOWER-LIMIT = 0
      UPPER-LIMIT = 255
0 -> DCM_SEC_LEV_LOCKED
1 -> DCM_SEC_LEV_L1
3 -> Configuration dependent (up to 127)
128 -> Reserved by document (up to 254)
255 -> DCM_SEC_LEV_ALL
};

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

ImplementationDataType ProtocolType {
      LOWER-LIMIT = 0
      UPPER-LIMIT = 255
0 -> DCM_OBD_ON_CAN
1 -> DCM_OBD_ON_FLEXRAY
2 -> DCM_OBD_ON_IP
3 -> DCM_UDS_ON_CAN
4 -> DCM_UDS_ON_FLEXRAY
5 -> DCM_UDS_ON_IP
6 -> DCM_ROE_ON_CAN
7 -> DCM_ROE_ON_FLEXRAY
8 -> DCM_ROE_ON_IP
9 -> DCM_PERIODICTRANS_ON_CAN
10 -> DCM_PERIODICTRANS_ON_FLEXRAY
11 -> DCM_PERIODICTRANS_ON_IP
12 -> Reserved for further AUTOSAR implementation (up to 239)
240 -> Reserved for software supplier specific implementation (up to
255)
};

ImplementationDataType NegativeResponseCodeType {
      LOWER-LIMIT = 0
      UPPER-LIMIT = 255
0x00  -> DCM_E_POSITIVERESPONSE
0x10  -> DCM_E_GENERALREJECT
0x11  -> DCM_E_SERVICENOTSUPPORTED
0x12  -> DCM_E_SUBFUNCTIONNOTSUPPORTED
0x13  -> DCM_E_INCORRECTMESSAGELENGTHORINVALIDFORMAT
```

```
0x14  -> DCM_E_RESPONSETOOLONG
0x21  -> DCM_E_BUSYREPEATREQUEST
0x22  -> DCM_E_CONDITIONSNOTCORRECT
0x24  -> DCM_E_REQUESTSEQUENCEERROR
0x25  -> DCM_E_NORESPONSEFROMSUBNETCOMPONENT
0x26  -> DCM_E_FAILUREPREVENTSEXECUTIONOFREQUESTEDACTION
0x31  -> DCM_REQUESTOUTOFRANGE
0x33  -> DCM_E_SECURITYACCESSDENIED
0x35  -> DCM_E_INVALIDKEY
0x36  -> DCM_E_EXCEEDNUMBEROFATTEMPTS
0x37  -> DCM_REQUIREDTIMEDELAYNOTEXPIRED
0x70  -> DCM_E_UPLOADDOWNLOADNOTACCEPTED
0x71  -> DCM_E_TRANSFERDATASUSPENDED
0x72  -> DCM_E_GENERALPROGRAMMINGFAILURE
0x73  -> DCM_E_WRONGBLOCKSEQUENCECOUNTER
0x78  -> DCM_E_REQUESTCORRECTLYRECEIVED-RESPONSEPENDING
0x7E  -> DCM_E_SUBFUNCTIONNOTSUPPORTEDINACTIVESESSION
0x7F  -> DCM_E_SERVICENOTSUPPORTEDINACTIVESESSION
0x81  -> DCM_E_RPMTOOHIGH
0x82  -> DCM_E_RPMTOOLOW
0x83  -> DCM_E_ENGINEISRUNNING
0x84  -> DCM_E_ENGINEISNOTRUNNING
0x85  -> DCM_E_ENGINERUNTIMETOOLOW
0x86  -> DCM_E_TEMPERATURETOOHIGH
0x87  -> DCM_E_TEMPERATURETOOLOW
0x88  -> DCM_E_VEHICLESPEEDTOOHIGH
0x89  -> DCM_E_VEHICLESPEEDTOOLOW
0x8A  -> DCM_E_THROTTLE_PEDALTOOHIGH
0x8B  -> DCM_E_THROTTLE_PEDALTOOLOW
0x8C  -> DCM_E_TRANSMISSIONRANGENOTINNEUTRAL
0x8D  -> DCM_E_TRANSMISSIONRANGENOTINGEAR
0x8F  -> DCM_E_BRAKESWITCH_NOTCLOSED
0x90  -> DCM_E_SHIFTERLEVERNOTINPARK
0x91  -> DCM_E_TORQUECONVERTERCLUTCHLOCKED
0x92  -> DCM_E_VOLTAGETOOHIGH
0x93  -> DCM_E_VOLTAGETOOLOW};
```

8.2 Type Definitions

[Dcm549] ⌜ If, for implementation reasons, some additional types have to be defined, the Dcm module shall label these types as follows: Dcm_<TypeName>Type, where <TypeName> is the name of this type adhering to the rules:

- No underscore usage
- First letter of each word upper case, consecutive letters lower case.

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.

8.2.1 Dcm_StatusType

Name:	Dcm_StatusType		
Type:	uint8		
Range:	DCM_E_OK	0x00	This value is representing a successful operation.
	DCM_E_COMPARE_KEY_FAILED	0x01	ECU compares by tester requested key with own calculated key. When comparison fails this definition is used. (used at API: RTE_DcmCompareKey() within processing of security access service)
	DCM_E_TI_PREPARE_LIMITS	0x02	New timing parameters are not ok, since requested values are not within the defined limits (used at API: Dcm_PrepareSesTimingValues()).
	DCM_E_TI_PREPARE_INCONSTENT	0x03	New timing parameters are not ok, since requested values are not consistent (e.g. P2min not smaller than P2max) (used at API: Dcm_PrepareSesTimingValues())
	DCM_E_SESSION_NOT_ALLOWED	0x04	Application does not allow start of requested session (used at API: RTE_DcmGetSesChgPermission())
	DCM_E_PROTOCOL_NOT_ALLOWED	0x05	Application does not allow start of requested protocol (used at API: RTE_DcmStartProtocol())
	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 ())
	DCM_E_REQUEST_NOT_ACCEPTED	0x08	Application rejects diagnostic request -> (used at API: RTE_DcmIndication())
	DCM_E_REQUEST_ENV_NOK	0x09	Diagnostic request is not allowed by application because of not fitting environmental conditions -> (used at API: RTE_DcmIndication())
Description:	Base item type to transport status information.		

8.2.2 Dcm_SecLevelType

This type is defined in Rte_Dcm_Type.h header file, which is generated by the RTE generator.

Name:	Dcm_SecLevelType		
Type:	uint8		
Range:	DCM_SEC_LEV_LOCKED	0x00	--
	DCM_SEC_LEV_L1	0x01	--
	configuration dependent	0x02...0x7F	--
	Reserved by Document	0x80...0xFE	--
	DCM_SEC_LEV_ALL	0xFF	--
Description:	Security Level type definition		

8.2.3 Dcm_SesCtrlType

This type is defined in Rte_Dcm_Type.h header file, which is generated by the RTE generator.

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)
	Reserved by Document	0x7F...0xFE	--
	DCM_ALL_SESSION_LEVEL	0xFF	--
Description:	Session type definition		

8.2.4 Dcm_ProtocolType

This type is defined in Rte_Dcm_Type.h header file, which is generated by the RTE generator.

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
	Reserved for further AUTOSAR implementation	0x07..0xEF	--
	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_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		

8.2.5 Dcm_NegativeResponseCodeType

This type is defined in Rte_Dcm_Type.h header file, which is generated by the RTE generator.

Name:	Dcm_NegativeResponseCodeType		
Type:	uint8		
Range:	DCM_E_POSITIVERESPONSE	0x00	PR
	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_FAILUREPREVENTSEXECUTIONOFREQUESTEDACTIO	0x26	FPEORA

	N		
	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
	DCM_E_REQUESTCORRECTLYRECEIVEDRESPONSEPENDING	0x78	RCRRP
	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..0xFE reserved by ISO 14229	0x94..0xFE	RFSCNC
	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.

8.2.6 Dcm_CommunicationModeType

Name:	Dcm_CommunicationModeType		
Type:	uint8		
Range:	DCM_ENABLE_RX_TX_NORM	0x00	Enable the Rx and Tx for normal communication
	DCM_ENABLE_RX_DISABLE_TX_NORM	0x01	Enable the Rx and disable the Tx for normal communication
	DCM_DISABLE_RX_ENABLE_TX_NORM	0x02	Disable the Rx and enable the Tx for normal communication
	DCM_DISABLE_RX_TX_NORMAL	0x03	Disable Rx and Tx for normal communication
	DCM_ENABLE_RX_TX_NM	0x04	Enable the Rx and Tx for network management communication

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

8.2.7 Dcm_ConfigType

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.	

8.2.8 Dcm_ConfirmationStatusType

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_O	0x03	--

	K		
Description:	--		

8.2.9 Dcm_OpStatusType

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 :

[Dcm527] ⌈ At first call of an operation using the Dcm_OpStatusType, the DCM call the operation with OpStatus = DCM_INITIAL ⌋()

[Dcm528] ⌈ If the value 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). ⌋()

[Dcm529] ⌈ After transmit confirmation of a RCR-RP transmitted on the context of [Dcm528, the DCM calls, from Dcm_MainFunction (due to call context), the operation again with OpStatus = DCM_FORCE_RCRRP_OK. ⌋()

[Dcm530] ⌈ If a 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 E_PENDING is returned. ⌋()

8.2.10 Dcm_ReturnReadMemoryType

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
Description:	Return values of Callout Dcm_ReadMemory		

8.2.11 Dcm_ReturnWriteMemoryType

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
Description:	Return type of callout Dcm_WriteMemory		

8.2.12 Dcm_RoeStateType

Name:	Dcm_RoeStateType		
Type:	uint8		
Range:	DCM_ROE_ACTIVE	0x00	--
	DCM_ROE_UNACTIVE	0x01	--
Description:	--		

8.2.13 Dcm_EcuStartModeType

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		

8.2.14 Dcm_ProgConditionsType

Name:	Dcm_ProgConditionsType		
Type:	Structure		
Element:	uint8	ProtocolId	Id of the protocol on wich the request has been received
	uint8	TesterSourceAddr	Tester source address configured per protocol
	uint8	Sid	Service identifier of the received request
	uint8	SubFncId	Identifier of the received 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.		

8.2.15 Dcm_MsgItemType

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

8.2.16 Dcm_MsgType

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

8.2.17 Dcm_MsgLenType

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

8.2.18 Dcm_MsgAddInfoType

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

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		

8.2.19 Dcm_IdContextType

Name:	Dcm_IdContextType

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

8.2.20 Dcm_MsgContextType

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

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.		

8.3 Function definitions

This section defines the functions provided for other modules.

[Dcm548] ⌈ The following provides the API Naming convention for the DCM services:
- The service name format is Dcm_<ServiceName>(…)
- <ServiceName>: is the name of the service primitive with first letter of each word upper case and consecutive letters lower case ⌋(BSW00310)

8.3.1 Functions provided for other BSW components

8.3.1.1 Dcm_Init

[**Dcm037**] ⌈

Service name:	Dcm_Init	
Syntax:	void Dcm_Init(Dcm_ConfigType * ConfigPtr)	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfigPtr	--
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Service for basic initialization of DCM module.	

⌋(BSW00438, BSW101, BSW00358, BSW00414)

[Dcm334] ⌈ `Dcm_Init()` shall initialize all DCM global variables with the values of the configuration ⌋()

The call of this service is mandatory before using the DCM module for further processing.

8.3.1.2 Dcm_GetVersionInfo

[Dcm065] ⌈

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)

[Dcm335] ⌈ `Dcm_GetVersionInfo()` shall return the version information of this module. The version information includes: Module Id, Vendor Id, Vendor specific version numbers (BSW00407).⌋(BSW00374, BSW00379, BSW003, BSW00318)

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

[Dcm337] ⌈ `Dcm_GetVersionInfo()` shall be pre compile time configurable On/Off by the configuration parameter ***DcmVersionInfoApi***.⌋()

8.3.1.3 Dcm_DemTriggerOnDTCStatus

[Dcm614] ⌈

Service name:	Dcm_DemTriggerOnDTCStatus	
Syntax:	Std_ReturnType Dcm_DemTriggerOnDTCStatus(uint32 DTC, uint8 DTCStatusOld, uint8 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	

」()

8.3.2 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.

[**Dcm698**]「 ClientServerInterface DCMServices

```
{
PossibleErrors {
                E_NOT_OK = 1};

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

」()

[**Dcm699**]「 ClientServerInterface DCM_Roe

```
{
PossibleErrors {
                E_NOT_OK = 1};

TriggerOnEvent(IN uint8 RoeEventId, ERR{E_NOT_OK});

}
```

」()

8.3.2.1 Dcm_GetSecurityLevel

[Dcm338]「

Service name:	Dcm_GetSecurityLevel	
Syntax:	Std_ReturnType Dcm_GetSecurityLevel(Dcm_SecLevelType* SecLevel)	
Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	SecLevel	Active Security Level value Conversion formula to calculate SecurityLevel out of tester requested SecurityAccessType parameter:

		SecurityLevel = (SecurityAccessType + 1) / 2 Content of SecurityAccessType is according to "securityAccessType" parameter of SecurityAccess request (see [11])
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	This function provides the active security level value.	

」(BSW04011)

8.3.2.2 Dcm_GetSesCtrlType

[Dcm339]「

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

」(BSW04011)

8.3.2.3 Dcm_GetActiveProtocol

[Dcm340]「

Service name:	Dcm_GetActiveProtocol	
Syntax:	Std_ReturnType Dcm_GetActiveProtocol(Dcm_ProtocolType* ActiveProtocol)	
Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	ActiveProtocol	Active protocol type value
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	This function returns the active protocol name.	

」(BSW04011)

8.3.2.4 Dcm_ResetToDefaultSession

[Dcm520] ⌈

Service name:	Dcm_ResetToDefaultSession	
Syntax:	Std_ReturnType Dcm_ResetToDefaultSession(void)	
Service ID[hex]:	0x2a	
Sync/Async:	Synchronous	
Reentrancy:	No	
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.	

⌋()

8.3.2.5 Dcm_TriggerOnEvent

[Dcm521] ⌈

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. This function shall be called only if the associated event has been activated through a xxx_ActivateEvent() call.	

⌋()

8.3.2.6 Dcm_StopROE

Dcm730⌈

Service name:	Dcm_StopROE
Syntax:	Std_ReturnType Dcm_StopROE(void

	)	
Service ID[hex]:	0x2e	
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:	This function stops ROE and clears the ROE event from the transmission queue.	

()

8.3.2.7 Dcm_RestartROE

Dcm731

Service name:	Dcm_RestartROE	
Syntax:	Std_ReturnType Dcm_RestartROE(void)	
Service ID[hex]:	0x2f	
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:	This function restarts the ROE event.	

()

8.4 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

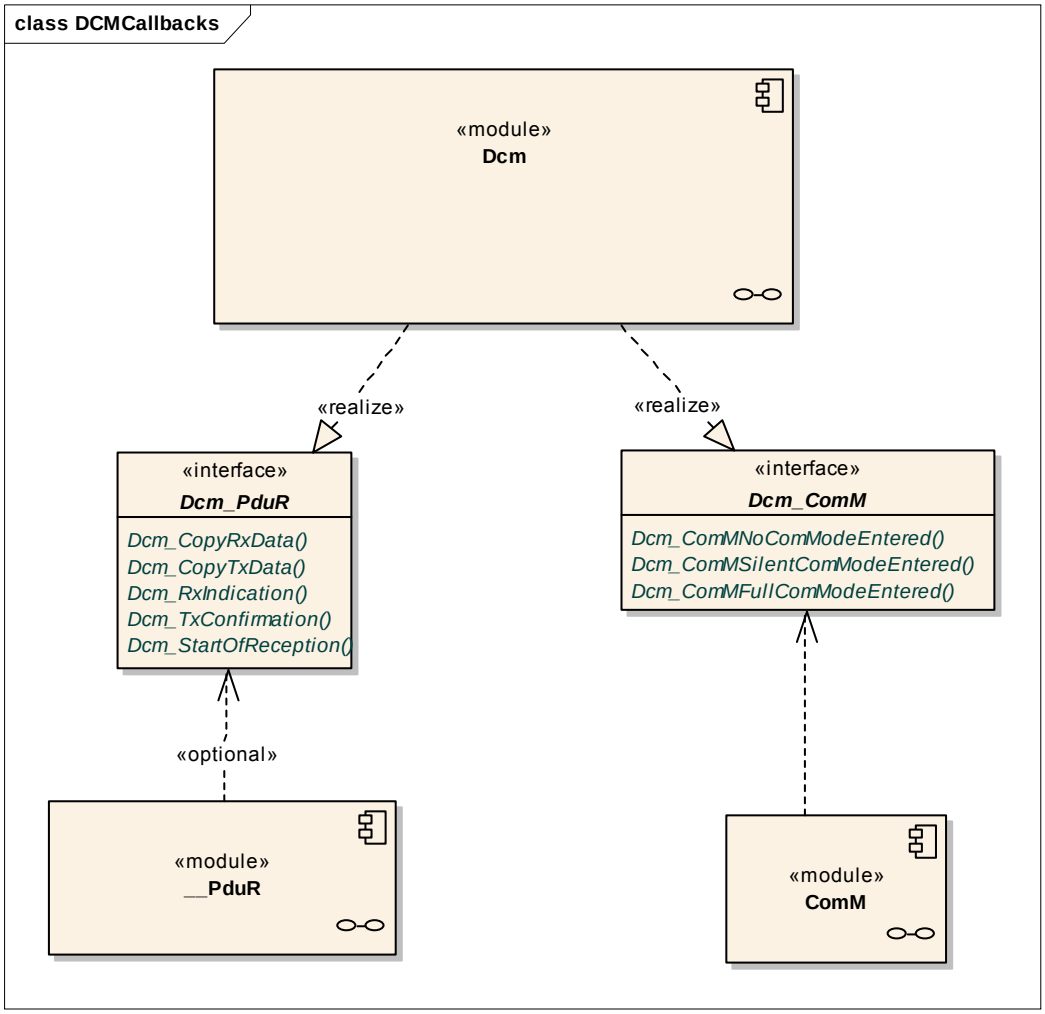


Figure 9: Overview of the callbacks provided by the DCM

8.4.1 Dcm_StartOfReception

[Dcm094] ⌈

Service name:	Dcm_StartOfReception	
Syntax:	BufReq_ReturnType Dcm_StartOfReception(PduIdType DcmRxPduId, 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.
	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	--
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.

[Dcm444] 「 If the requested size is large than the buffer available in the DCM, the function `Dcm_StartOfReception()` shall return BUFREQ_E_OVFL (see Dcm094).」()

Dcm788 「When processing a diagnostic request, the DCM module shall accept (`Dcm_StartOfReception()` shall return BUFREQ_E_OK) any new request using a different DcmRxPduId in case ***DcmDslDiagRespOnSecondDeclinedRequest*** is set to TRUE.」()

Dcm789 「In case Dcm788, the Dcm respond with a NRC 0x21」()

Dcm790 「When processing a diagnostic request, the DCM module shall reject (`Dcm_StartOfReception()` shall return BUFREQ_E_NOT_OK) any new request using a different DcmRxPduId in case ***DcmDslDiagRespOnSecondDeclinedRequest*** is set to FALSE until the current diagnostic request processing is over.」()

[Dcm557] 「 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 DcmRxPudId until the current diagnostic request processing is over.」()

[Dcm642] 「 When the API `Dcm_StartOfReception` (or `Dcm_CopyRxData`) is invoked with TpSduLength (or SduLength from PduInfoPtr) equal to 0, the value BUFREQ_OK shall be returned and RxBufferSizePtr shall be set to the size of the Rx buffer.」()

[Dcm655] ⌈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 Dcm788, Dcm789 and Dcm790.⌋()

[Dcm656] ⌈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 Dcm788, Dcm789 and Dcm790. ⌋()

8.4.2 Dcm_CopyRxData

[Dcm556] ⌈

Service name:	Dcm_CopyRxData	
Syntax:	BufReq_ReturnType Dcm_CopyRxData(PduIdType DcmRxPduId, 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 might only be called with an SduLength greater 0 if the RxBufferSizePtr returned by the previous API call indicates sufficient receive buffer (SduLength ≤ RxBufferSizePtr). The function must only be called if the connection has been accepted by an initial call to Dcm_StartOfReception.	

⌋()

[Dcm443] ⌈ 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.⌋()

[Dcm342] ⌈ After starting to copy the received data (see Dcm443), 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.⌋()

8.4.3 Dcm_TpRxIndication

[Dcm093] ⌈

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

⌋()

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

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

[Dcm345] ⌈ `Dcm_TpRxIndication()` shall be callable in interrupt context.⌋()

8.4.4 Dcm_CopyTxData

[Dcm092]「

Service name:	Dcm_CopyTxData	
Syntax:	BufReq_ReturnType Dcm_CopyTxData(PduIdType DcmTxPduId, 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	Identifies the DCM data to be sent. This information is used to derive the PCI information within the transport protocol. The value has to be same as in the according service call PduR_DcmTransmit().
	PduInfoPtr	Pointer to a PduInfoType, which indicates the number of bytes to be copied (SduLength) and the location where the data have to be copied to (SduDataPtr). An SduLength of 0 is possible in order to poll the available transmit data count. In this case no data are to be copied and SduDataPtr might be invalid.
	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, in case the corresponding I-PDU was stopped.
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 from the offset position given in RetryInfoPtr->TxTpDataCnt	

」()

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.

[Dcm346]「 If the function `Dcm_CopyTxData()` is called and the DCM module successfully copy the data in the buffer provided in PduInfoPtr parameter, then the the function shall return BUFREQ_OK.」()

[Dcm349] ⌈ After starting to copy the received data (see Dcm346), the DCM module shall not access this buffer unless it is requested to provide a new buffer with service `Dcm_CopyTxData()` for the same DcmTxPduId or it is notified about the successful transmission (Call of service `Dcm_TpTxConfirmation()`) or it is notified by an error indicating that the transmission was aborted (Call of service `Dcm_TpTxConfirmation()`).⌋()

[Dcm350] ⌈ 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()`.⌋()

8.4.5 Dcm_TpTxConfirmation

[Dcm351] ⌈

Service name:	Dcm_TpTxConfirmation	
Syntax:	void Dcm_TpTxConfirmation(PduIdType DcmTxPduId, NotifResultType Result)	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	DcmTxPduId	ID of DCM I-PDU that has been transmitted. Range: 0..(maximum number of I-PDU IDs transmitted by DCM) – 1
	Result	- NTFRSLT_OK if the complete N-PDU has been transmitted. - NTFRSLT_E_CANCELATION_OK if the N-PDU has been successfully cancelled - NTFRSLT_E_CANCELATION_NOT_OK if an error occurred when cancelling the N-PDU - any other value: an error occurred during transmission, the DCM can unlock the transmit buffer
Parameters (inout):	None	
Parameters (out):	None	

Return value:	None
Description:	This is called by the PduR to confirm a Transmit

⌋()

[Dcm352] ⌈ If the function `Dcm_TpTxConfirmation()` is called, then the DCM module shall unlock the transmit buffer.⌋()

[Dcm353] ⌈ If the function `Dcm_TpTxConfirmation()` is called, then the DCM module shall stop error handling (Page buffer timeout, P2ServerMax/P2*ServerMax timeout).⌋()

[Dcm354] ⌈ `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.

8.4.6 Dcm_ComM_NoComModeEntered

[Dcm356] ⌈

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.	

⌋()

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

[Dcm149] ⌈ `Dcm_ComM_NoComModeEntered()` shall disable the ResponseOnEvent transmissions.⌋()

[Dcm150] ⌈ `Dcm_ComM_NoComModeEntered()` shall disable the periodicId transmissions (ReadDataByPeriodicIdentifier).⌋()

[Dcm151] ⌈ `Dcm_ComM_NoComModeEntered()` shall disable normal transmissions.⌋()

[Dcm152] ⌈ After `Dcm_ComM_NoComModeEntered()` has been called, the DCM module shall not call the function `PduR_DcmTransmit().`⌋()

8.4.7 Dcm_ComM_SilentComModeEntered

[Dcm358] ⌈

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.	

⌋()

[Dcm153] ⌈ `Dcm_ComM_SilentComModeEntered()` shall disable all transmission. This means that the message transmission shall be off.⌋()

[Dcm154] ⌈ `Dcm_ComM_SilentComModeEntered()` shall disable the ResponseOnEvent transmissions.⌋()

[Dcm155] ⌈ `Dcm_ComM_SilentComModeEntered()` shall disable the periodicId transmissions (ReadDataByPeriodicIdentifier) shall be disabled.⌋()

[Dcm156] ⌈ `Dcm_ComM_SilentComModeEntered()` shall disable the normal transmissions. ⌋()

8.4.8 Dcm_ComM_FullComModeEntered

[Dcm360] ⌈

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.	

⌋()

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

[Dcm159] ⌈ `Dcm_ComM_FullComModeEntered()` shall enable the ResponseOnEvent transmissions. ⌋()

[Dcm160] ⌈ `Dcm_ComM_FullComModeEntered()` shall enable the periodicId transmissions (ReadDataByPeriodicIdentifier). ⌋()

[Dcm161] ⌈ `Dcm_ComM_FullComModeEntered()` shall enable the normal transmissions. ⌋()

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

8.5 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.

8.5.1 Dcm_ ReadMemory

[**Dcm539**]⌈

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
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)	

」()

[Dcm644] 「 If the call to `Dcm_ReadMemory` returns DCM_READ_FAILED, the DCM module shall trigger a negative response with NRC 0x72 (GeneralProgrammingFailure). 」()

8.5.2 Dcm_ WriteMemory

[Dcm540] 「

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

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 - WriteDataByIdentifier (in case of Dynamical DID defined by memory address)

」()

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

[Dcm643] 「 If the call to `Dcm_WriteMemory` returns DCM_WRITE_FAILED, the DCM module shall trigger a negative response with NRC 0x72 (GeneralProgrammingFailure). 」()

8.5.3 Dcm_Confirmation

[Dcm547] 「

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)

8.5.4 Dcm_SetProgConditions

[Dcm543] ⌈

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

⌋()

8.5.5 Dcm_GetProgConditions

[Dcm544] ⌈

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.	

⌋()

8.5.6 Dcm_ProcessRequestTransfertExit

Dcm755 ⌜

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 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.	

⌟()

Dcm759 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.

8.5.7 Dcm_PorcessRequestUpload

Dcm756 ⌜

Service name:	Dcm_ProcessRequestUpload
Syntax:	Std_ReturnType Dcm_ProcessRequestUpload(Dcm_OpStatusType OpStatus, uint8 DataFormatIdentifier, uint32 MemoryAddress, uint32 MemorySize, 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):	ErrorCode	See below
Return value:	Std_ReturnType	E_OK: Request was successful E_NOT_OK: Request was not successful 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.	

」()

Dcm758 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.

8.5.8 Dcm_ProcessRequestDownload

Dcm754「

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	See below
Return value:	Std_ReturnType	E_OK: Request was successful E_NOT_OK: Request was not successful 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.	

」()

Dcm757 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.

8.6 Scheduled Functions

8.6.1 Dcm_MainFunction

[Dcm053]「

Service name:	Dcm_MainFunction
Syntax:	void Dcm_MainFunction(void)
Service ID[hex]:	0x25
Timing:	FIXED_CYCLIC
Description:	This service is used for processing the tasks of the main loop.

」(BSW00424, BSW00373, BSW00376)

[Dcm362]「 `Dcm_MainFunction` shall integrate all scheduled functions (e.g. Diagnostic timer handling)」()

`Dcm_MainFunction()` must be called periodically.

[Dcm593]「 If the `Dcm_MainFunction` is called and the DCM module has not been initialized (Call to `Dcm_Init`), the function shall return immediately without performing any functionality and without raising any errors」(BSW00450)

8.7 Expected Interfaces

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

8.7.1 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.

8.7.2 Optional Interfaces

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

API function	Description
BswM_Dcm_CommunicationMode_CurrentState	Function called by DCM to inform the BswM about the current state of the communication mode.
Dem_DcmCancelOperation	Cancel pending operation started from Dcm.
Dem_DisableDTCRecordUpdate	Disables the event memory update of a specific DTC (only one at one time).
Dem_DisableDTCSetting	Disables the DTC setting for a DTC group.
Dem_EnableDTCRecordUpdate	Enables the event memory update of the DTC disabled by Dem_DisableDTCRecordUpdate() before.
Dem_EnableDTCSetting	Enables the DTC setting for a DTC group.
Dem_GetDTCByOccurrenceTime	Gets the DTC by occurrence time. There is no explicit parameter for the DTC-origin as the origin always is DEM_DTC_ORIGIN_PRIMARY_MEMORY.
Dem_GetDTCOfOBDFreezeFrame	Gets DTC by freeze frame record number.
Dem_GetDTCStatusAvailabilityMask	Gets the DTC Status availability mask.
Dem_GetExtendedDataRecordByDTC	Gets extended data by DTC. The function stores the data in the provided DestBuffer.
Dem_GetFreezeFrameDataByDTC	Gets freeze frame data by DTC. The function stores the data in the provided DestBuffer.
Dem_GetFreezeFrameDataByRecord	Gets DTC and its associated freeze frame record by absolute record number. The function stores the data in the provided DestBuffer.
Dem_GetFunctionalUnitOfDTC	Gets the functional unit of the requested DTC.
Dem_GetNextFilteredDTC	Gets the next filtered DTC.
Dem_GetNextFilteredDTCAndFDC	Gets the current DTC and its associated Fault Detection Counter (FDC) from the Dem. The interface has an asynchronous behavior, because the FDC might be received asynchronously from a SW-C, too.
Dem_GetNextFilteredDTCAndSeverity	Gets the current DTC and its Severity from the Dem.
Dem_GetNextFilteredRecord	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_GetNumberOfFilteredDTC	Gets the number of a filtered DTC.
Dem_GetSeverityOfDTC	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_GetSizeOfExtendedDataRecordByDTC	Gets the size of extended data by DTC.
Dem_GetSizeOfFreezeFrameByDTC	Gets the size of freeze frame data by DTC.
Dem_GetStatusOfDTC	Gets the status of a DTC. For large configurations and DTC-calibration, the interface behavior can be asynchronous (splitting the DTC-search into segments).
Dem_GetTranslationType	Gets the supported DTC formats of the ECU. The supported formats are configured via DemTypeOfDTCSupported.
Dem_ReadDataOfOBDFreezeFrame	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.
Dem_SetDTCFilter	The server shall perform a bit-wise logical AND-ing operation between the mask specified in the client's request and the actual status associated with each DTC supported by the server. In addition to the DTCStatusAvailabilityMask, the server shall return all DTCs for which the result of the AND-ing operation is non-zero [i.e. (statusOfDTC & DTCStatusMask) != 0]. If the client specifies a status mask that contains bits that the server does not support, then the server shall process the DTC information using only the bits that it does support. If no DTCs within the server match the masking criteria specified in the client's request, no DTC or status information shall be provided following the DTCStatusAvailabilityMask byte in the positive response message (statusOfDTC & DTCStatusMask) & (severity & DTCSeverityMask) != 0
Dem_SetFreezeFrameRecordFilter	Sets a freeze frame record filter.
Det_ReportError	Service to report development errors.
Dlt_ActivateEvent	This function is called by the Dcm if a specific ResponseOnEvent is enabled by a user. The RoeEventID shall be used by the Dlt to notify the Dcm about some actions to do for the ROE service. Normally for the Dlt module, only the ReadDataByIdendifer (0x22) should be used for ROE. In addition, when a specific ROE for the Dlt module is disabled/turned off by a user, the Dlt module is notified with this function.
Dlt_ConditionCheckRead	Is called from Dcm when UDS service ReadDataByIdendifer for Dlt DID is called to see if Dlt_ReadData can be called.
Dlt_ReadData	Is called from Dcm when UDS service ReadDataByIdendifer for Dlt DID is called.

Dlt_ReadDataLength	Is called from Dcm when UDS service ReadDataByIdendifer for Dlt DID is called.
Dlt_WriteData	Is called from Dcm when UDS service WriteDataByIdendifer for Dlt DID is called.
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 of an NVRAM block.
NvM_WriteBlock	Service to copy the data of the RAM block to its corresponding NV block.
PduR_DcmCancelReceive	Requests for cancellation of an ongoing reception of an I-Pdu in transport protocol.
PduR_DcmCancelTransmit	Request for cancellation of an ongoing transmission of an I-Pdu in transport protocol or communication interface.
PduR_DcmChangeParameter	Request to change a specific transport protocol parameter (e.g. block-size). The affected transport protocol module is selected using the I-PDU ID.
SchM_ActMainFunction_Dcm	Invokes the SchM_ActMainFunction function to trigger the activation of a corresponding main processing function.

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

8.7.3 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).

8.7.3.1 SecurityAccess_<LEVEL>

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

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

[Dcm685] ⌈
```
ClientServerInterface SecurityAccess_<LEVEL>
{
PossibleErrors {
  E_NOT_OK = 1,
  E_PENDING = 10, // application process not yet completed
```

```
  E_COMPARE_KEY_FAILED = 11 // compare failed
};

//Request to application for provision of seed value
//SecurityAccessDataRecord: This data record contains additional
//data to calculate the seed value; if size is 0 the parameter is
//still present but a "NULL"-pointer is passed
//Seed: Pointer for provided seed
//If the parameter DcmDspSecurityADRSize is present, the following
interface is used:
GetSeed(IN uint8 SecurityAccessDataRecord[<DcmDspSecurityADRSize>],
        IN Dcm_OpStatusType OpStatus,
        OUT uint8 Seed[<DcmDspSecuritySeedSize>],
        OUT NegativeResponseCodeType ErrorCode,
        ERR{E_NOT_OK, E_PENDING});

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

//Request to application for comparing key
//Key: Key, which needs to be compared
CompareKey(IN uint8 Key[<DcmDspSecurityKeySize>],
                IN Dcm_OpStatusType OpStatus,
           ERR{E_NOT_OK, E_PENDING, E_COMPARE_KEY_FAILED});
}
```

」()

[Dcm659]「 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.」()

[Dcm660]「 If the operation CompareKey() return value 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 (when DcmDspSecurityADRSize is present):

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

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

```
Std_ReturnType Xxx_GetSeed(Dcm_OpStatusType* OpStatus,
```

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

8.7.3.2 DataServices_<Data>

One DataServices interface will be generated for each Data of each DID/PID, with following possible operations:

8.7.3.2.1 **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.

8.7.3.2.2 **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).

8.7.3.2.3 **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)

8.7.3.2.4 **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

8.7.3.2.5 **GetScalingInformation**

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

8.7.3.2.6 **ReturnControlToEcu**

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

8.7.3.2.7 **ResetToDefault**

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

8.7.3.2.8 **FreezeCurrentState**

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

8.7.3.2.9 **ShortTermAdjustment**

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

8.7.3.2.10 **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):

[Dcm686] ⌈
```
ClientServerInterface DataServices_<Data>
{
PossibleErrors {
  E_NOT_OK = 1,
  E_PENDING = 10,
};

//the next operation is provided if the Data can be read and is
//used
// to get the data value of a DID or PID from a SW-C

// 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)

//if DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER, the
//following definition is used
ReadData(OUT uint8 Data[(<DcmDspDataSize>+7)/8],
         ERR{E_NOT_OK)
```

```
//if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER, the
//following definition is used
ReadData(IN Dcm_OpStatusType OpStatus,
         OUT uint8 Data[(<DcmDspDataSize>+7)/8],
         ERR{E_NOT_OK, E_PENDING)


//the next operation is provided if the Data can be written
//data : Data value
//dataLength: optional. Available, if the data length of
//the Data is variable. It defines the size of valid data in
//the parameter data
//if DcmDspDataFixedLength is set to FALSE, the following definition
is used
WriteData(IN uint8 Data[(<DcmDspDataSize>+7)/8],
          IN uint16 DataLength,
          IN Dcm_OpStatusType OpStatus,
          OUT NegativeResponseCodeType ErrorCode,
          ERR{E_NOT_OK, E_PENDING})

//if DcmDspDataFixedLength is set to TRUE, the following definition
is used
WriteData(IN uint8 Data[(<DcmDspDataSize>+7)/8],
          IN Dcm_OpStatusType OpStatus,
          OUT NegativeResponseCodeType ErrorCode,
          ERR{E_NOT_OK, E_PENDING})


//the next operation is provided optionally if the Data's length is
variable
//DataLength: Data length of the Data
//The module providing this operation shall ensure that the
//datalength provided within this operation is consistent with the
//data length in the associated ReadData operation


// 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)
ReadDataLength(OUT uint16 DataLength,
               ERR{E_NOT_OK })

//the next operation is always provided
ConditionCheckRead(IN Dcm_OpStatusType OpStatus,
                   OUT NegativeResponseCodeType ErrorCode,
                   ERR{E_NOT_OK, E_PENDING})


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

```
//the next operation is provided optionally
ReturnControlToECU(IN Dcm_OpStatusType OpStatus,
                   OUT NegativeResponseCodeType ErrorCode,
                   ERR{E_NOT_OK, E_PENDING})

//the next operation is provided optionally
ResetToDefault(IN Dcm_OpStatusType OpStatus,
               OUT NegativeResponseCodeType ErrorCode,
               ERR{E_NOT_OK, E_PENDING})

//the next operation is provided optionally
FreezeCurrentState(IN Dcm_OpStatusType OpStatus,
                   OUT NegativeResponseCodeType ErrorCode,
                   ERR{E_NOT_OK, E_PENDING})

//the next operation is provided optionally
ShortTermAdjustment(IN uint8
ControlOptionRecord[(<DcmDspDataSize>+7)/8],
                     IN Dcm_OpStatusType OpStatus,
                     OUT NegativeResponseCodeType ErrorCode,
                     ERR{E_NOT_OK, E_PENDING})


}」()
```

8.7.3.2.11 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):

[Dcm687] 「
```
SenderReceiver DataServices_<Data>
{
  <datatype> Data
}」()
```

8.7.3.2.12 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.

8.7.3.2.12.1 ReadData

if DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER, the following definition is used:

Dcm793Γ

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, 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. 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.	

J()

8.7.3.2.12.2 WriteData

if DcmDspDataFixedLength is set to TRUE, the following definition is used:

Dcm794Γ

Service name:	Xxx_WriteData
Syntax:	Std_ReturnType Xxx_WriteData(uint8* Data,

	Dcm_OpStatusType OpStatus, Dcm_NegativeResponseCodeType* ErrorCode)	
Service ID[hex]:	0x35	
Sync/Async:	Synchronous	
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	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. 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:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Data	Buffer containing the data to be written
	DataLength	Length of the data to be writen/read
	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. 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.	

⌋()

8.7.3.2.12.3 ReadDataLength

Dcm796⌈

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 of the data to be writen/read
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	This function requests the application to return the data length of a Data.	

⌋()

8.7.3.2.12.4 ConditionCheckRead

Dcm797⌈

Service name:	Xxx_ConditionCheckRead	
Syntax:	Std_ReturnType Xxx_ConditionCheckRead(Dcm_OpStatusType OpStatus, Dcm_NegativeResponseCodeType* ErrorCode)	
Service ID[hex]:	0x37	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	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. 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.	

⌋()

8.7.3.2.12.5 GetScalingInformation

Dcm798⌈

Service name:	Xxx_GetScalingInformation
Syntax:	Std_ReturnType Xxx_GetScalingInformation(Dcm_OpStatusType OpStatus, uint8* ScalingInfo, Dcm_NegativeResponseCodeType* ErrorCode

	)	
Service ID[hex]:	0x38	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	ScalingInfo	Scaling information
	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. 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.	

⌋()

8.7.3.2.12.6 ReturnControlToECU

Dcm799

⌈

Service name:	Xxx_ReturnControlToECU	
Syntax:	Std_ReturnType Xxx_ReturnControlToECU(Dcm_OpStatusType OpStatus, Dcm_NegativeResponseCodeType* ErrorCode)	
Service ID[hex]:	0x39	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	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. 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.	

⌋()

8.7.3.2.12.7 ResetToDefault

Dcm800

⌈

Service name:	Xxx_ResetToDefault
Syntax:	Std_ReturnType Xxx_ResetToDefault(Dcm_OpStatusType OpStatus,

	Dcm_NegativeResponseCodeType* ErrorCode)	
Service ID[hex]:	0x3c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	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. 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.	

⌋()

8.7.3.2.12.8 FreezeCurrentState

Dcm801

⌈

Service name:	Xxx_FreezeCurrentState	
Syntax:	Std_ReturnType Xxx_FreezeCurrentState(Dcm_OpStatusType OpStatus, Dcm_NegativeResponseCodeType* ErrorCode)	
Service ID[hex]:	0x3a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	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. 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.	

⌋()

8.7.3.2.12.9 ShortTermAdjustment

Dcm802

⌜

Service name:	Xxx_ShortTermAdjustment	
Syntax:	Std_ReturnType Xxx_ShortTermAdjustment(uint8* ControlOptionRecord, Dcm_OpStatusType OpStatus, Dcm_NegativeResponseCodeType* ErrorCode)	
Service ID[hex]:	0x3d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControlOptionRecord	Control option parameter for the adjustment request
	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. 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.	

⌟()

8.7.3.3 DataServices_DIDRange_<Range>

8.7.3.3.1 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:

[Dcm769] ⌜
```
ClientServerInterface DataServices_DIDRange_<Range>
{
PossibleErrors {
  E_NOT_OK = 1,
  E_PENDING = 10
```

```
};

//the next operation is provided optionally if there are no gaps in
the DID-ranges
//this operation shall return if a specific DID is available within
the range or not.

IsDidAvailable(IN uint16 DID,
               OUT uint8 supported,
               ERR{E_NOT_OK}) // in case there are no gaps in the
range this operation shall be optional

ReadDidData(IN uint16 DID,
         OUT uint8 Data[<DcmDspDidRangeMaxLength>],
         IN Dcm_OpStatusType OpStatus,
         OUT uint16 DataLength,
         OUT NegativeResponseCodeType ErrorCode,
         ERR{E_NOT_OK, E_PENDING})

WriteDidData(IN uint16 DID,
          IN uint8 Data[<DcmDspDidRangeMaxLength>],
          IN Dcm_OpStatusType OpStatus,
          IN uint16 DataLength,
          OUT NegativeResponseCodeType ErrorCode,
          ERR{E_NOT_OK, E_PENDING})
}
```

⌋()

8.7.3.3.2 **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.

8.7.3.3.2.1 IsDidAvailable

Dcm803

⌈

Service name:	Xxx_IsDidAvailable	
Syntax:	Std_ReturnType Xxx_IsDidAvailable(uint16 DID, uint8* supported)	
Service ID[hex]:	0x3F	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DID	DID value
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.
Description:	This function requests if a specific DID is available within the range or not.	

⌋()

8.7.3.3.2.2 ReadDidData

Dcm804

⌜

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:	Synchronous	
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	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. 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	

⌟()

8.7.3.3.2.3 WriteDidData

Dcm805⌜

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:	Synchronous	
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	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. 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.	

⌋()

8.7.3.4 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 Dcm423).

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

[Dcm688] ⌈
```
ClientServerInterface InfotypeServices_<VEHINFODATA>
{
PossibleErrors {
  E_NOT_OK = 1,
  E_PENDING = 10
};

// used to get Infotype data from SW-C
// the return value E_PENDING is not allowed for data of InfoType
0x02,
// 0x04, 0x08, 0x0A and 0x0B
//return value E_NOT_OK is not allowed as a data value has always
// to be provided to DCM
GetInfotypeValueData(IN Dcm_OpStatusType OpStatus,
  OUT uint8 DataValueBuffer[<DcmDspVehInfoDataSize>],
  ERR{E_NOT_OK, E_PENDING})
}
```

From the point of view of the DCM, the operation has the following signature:
```
Std_ReturnType Xxx_GetInfotypeValueData(Dcm_OpStatusType OpStatus,
       uint8* DataValueBuffer)
```
⌋()

8.7.3.5 DTRServices

The following interface defines an operation needed to supply OBD Service $06 (see Dcm416).
Using the concepts of the SW-C template, the interface is defined as follows:

[Dcm689] ⌈
```
ClientServerInterface DTRServices
{
PossibleErrors {
  E_NOT_OK = 1,
};

// used to get DTR data from SW-C for service 6
//return value E_NOT_OK is not allowed as a data value has always
```

```
// to be provided to DCM
GetDTRValue(IN Dcm_OpStatusType OpStatus,
  OUT uint16 Testval,
  OUT uint16 Minlimit,
  OUT uint16 Maxlimit,
  OUT DTRStatusType Status,
  ERR{E_NOT_OK });
}

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

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

```
Std_ReturnType Xxx_GetDTRValue(Dcm_OpStatusType OpStatus,
                               uint16* Testval,
                               uint16* Minlimit,
                               uint16* Maxlimit,
                               uint8* Status)」()
```

8.7.3.6 RoutineServices_<ROUTINENAME>

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

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

[Dcm690]「

```
ClientServerInterface RoutineServices_<RoutineName>
{
PossibleErrors {
      E_NOT_OK = 1,
      E_PENDING = 10,
      E_FORCE_RCRRP = 12,//application request the transmission of
                         //a response Response Pending (NRC 0x78)
      };


//the next operation is always present
//If DcmDspRoutineFixedLength is set to FALSE, the following
definition is used

Start(IN <datatype> dataIn1,…, IN uint8
dataInN[(<DcmDspRoutineSignalLength of
DcmDspStartRoutineInSignal> +7)/8],
      IN Dcm_OpStatusType OpStatus,
      OUT <datatype> dataOut1,…, OUT uint8
dataOutN[(<DcmDspRoutineSignalLength of
DcmDspStartRoutineOutSignal> +7)/8],
      INOUT uint16 currentDataLength,
      OUT NegativeResponseCodeType ErrorCode,
      ERR{E_NOT_OK, E_PENDING, E_FORCE_RCRRP })
```

```
//If DcmDspRoutineFixedLength is set to TRUE, the following
definition is used

Start(IN <datatype> dataIn1,…, IN <datatype> dataInN,
      IN Dcm_OpStatusType OpStatus,
      OUT <datatype> dataOut1,…, OUT <datatype> dataOutN,
      OUT NegativeResponseCodeType ErrorCode,
      ERR{E_NOT_OK, E_PENDING, E_FORCE_RCRRP })



//the next operation is present if the routine can be stopped
//If DcmDspRoutineFixedLength is set to FALSE, the following
definition is used

Stop(IN <datatype> dataIn1,…, IN uint8
dataInN[(<DcmDspRoutineSignalLength of
DcmDspRoutineStopInSignal> +7)/8],
     IN Dcm_OpStatusType OpStatus,
     OUT <datatype> dataOut1,…, OUT uint8
dataOut[(<DcmDspRoutineSignalLength of
DcmDspRoutineStopOutSignal> +7)/8],
     INOUT uint16 currentDataLength,
     OUT NegativeResponseCodeType ErrorCode,
     ERR{ E_NOT_OK, E_PENDING, E_FORCE_RCRRP })

//If DcmDspRoutineFixedLength is set to TRUE, the following
definition is used

Stop(IN <datatype> dataIn1,…, IN <datatype> dataInN,
     IN Dcm_OpStatusType OpStatus,
     OUT <datatype> dataOut1,…, OUT <datatype> dataOutN,
     OUT NegativeResponseCodeType ErrorCode,
     ERR{ E_NOT_OK, E_PENDING, E_FORCE_RCRRP })


//the next operation is present if the status of
//the routine can be requested
//If DcmDspRoutineFixedLength is set to FALSE, the following
definition is used

RequestResults(IN Dcm_OpStatusType OpStatus,
               OUT <datatype> dataOut1,…, OUT uint8
dataOutN[(<DcmDspRoutineSignalLength of
DcmDspRoutineRequestResOutSignal> +7)/8],
               OUT uint16 currentDataLength,
               OUT NegativeResponseCodeType ErrorCode,
               ERR{ E_NOT_OK, E_PENDING, E_FORCE_RCRRP })

//If DcmDspRoutineFixedLength is set to TRUE, the following
definition is used
RequestResults(IN Dcm_OpStatusType OpStatus,
               OUT <datatype> dataOut1,…, OUT <datatype> dataOutN,
               OUT NegativeResponseCodeType ErrorCode,
               ERR{ E_NOT_OK, E_PENDING, E_FORCE_RCRRP })

}┘()
```

[Dcm668] ⌈ 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.⌋()

[Dcm669] ⌈ If the operation Start() returns value E_FORCE_RCRRP, the DCM module shall start the transmission of NRC 0x78.⌋()

[Dcm670] ⌈ 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.⌋()

[Dcm671] ⌈ If the operation Stop() returns value E_FORCE_RCRRP, the DCM module shall start the transmission of NRC 0x78.⌋()

[Dcm672] ⌈ 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.⌋()

[Dcm673] ⌈ If the operation RequestResults () returns value 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(<datatype> dataIn1,…,uint8* dataInN,
                         Dcm_OpStatusType OpStatus,
                         <datatype> dataOut1,…,uint8* dataOutN,
                         uint16* currentDataLength,
                         Dcm_NegativeResponseCodeType* ErrorCode)
//If DcmDspRoutineFixedLength is set to TRUE, the following
definition is used
Std_ReturnType Xxx_Start(<datatype> dataIn1,…,<datatype> dataInN,
                         Dcm_OpStatusType OpStatus,
                         <datatype>* dataOut1,…,
                         <datatype>* dataOutN,
                         Dcm_NegativeResponseCodeType* ErrorCode)

//If DcmDspRoutineFixedLength is set to FALSE, the following
definition is used
Std_ReturnType Xxx_Stop(<datatype> dataIn1,…,uint8* dataInN,
                        Dcm_OpStatusType OpStatus,
                        <datatype> dataOut1,…,uint8* dataOutN
                        uint16* currentDataLength,
```

```
                        Dcm_NegativeResponseCodeType* ErrorCode)

//If DcmDspRoutineFixedLength is set to TRUE, the following
definition is used
Std_ReturnType Xxx_Stop(<datatype> dataIn1,…,<datatype> dataInN,
                        Dcm_OpStatusType OpStatus,
                        <datatype>* dataOut1,…,
                        <datatype>* dataOutN,
                        Dcm_NegativeResponseCodeType* ErrorCode)

//If DcmDspRoutineFixedLength is set to FALSE, the following
definition is used
Std_ReturnType Xxx_RequestResults(Dcm_OpStatusType OpStatus,
                        <datatype> 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,
                        <datatype>* dataOut1,…,
                        <datatype>* dataOutN,
                        Dcm_NegativeResponseCodeType* ErrorCode)
```

8.7.3.7 RequestControlServices_<TID>

The interface RequestControlServices_<TID> allows the DCM to provide OBD Service $08 (see Dcm419).
Using the concepts of the SW-C template, the interface is defined as follows:

[Dcm691]⌈
```
ClientServerInterface RequestControlServices_<TID> {
PossibleErrors {
  E_NOT_OK = 1,
  };
RequestControl(
  OUT OutBuffer uint8[<DcmDspRequestControlOutBufferSize>],
  IN InBuffer uint8[<DcmDspRequestControlInBufferSize>],
  ERR{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)
```

8.7.3.8 CallbackDCMRequestServices

The interface CallbackDCMRequestServices provides information on the status of the protocol communication and allows the Application to disallow a protocol (see Dcm036, Dcm144, Dcm145, Dcm146; Dcm147, Dcm459).

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

[Dcm692]⌈
```
ClientServerInterface CallbackDCMRequestServices {
PossibleErrors {
  E_NOT_OK = 1,
```

```
  E_PROTOCOL_NOT_ALLOWED = 5, //conditions in application allows no
                              // further procession of protocol

};

//Indication of protocol start. Application is able to reject
//further processing of requested protocol due to failed
//conditions.
StartProtocol(IN ProtocolType ProtocolID,
              ERR{E_PROTOCOL_NOT_ALLOWED, E_NOT_OK });

//Indication of protocol stop. If a running diagnostic
//requested is preempted by a higher prior request (of an
//other protocol, e.g. OBD), application is requested to abort
//further processing of running request
//ProtocolID: Name of  the protocol(IDs configured within
//DCM_PROTOCOL_ID)
StopProtocol(IN ProtocolType ProtocolID,
             ERR{E_NOT_OK});
```

}」()

[Dcm674]「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)
```

8.7.3.9 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 Dcm218, Dcm462, Dcm463).

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

[Dcm694] ⌈
```
ClientServerInterface ServiceRequestNotification {
PossibleErrors {
  E_NOT_OK = 1,
  E_REQUEST_NOT_ACCEPTED = 8,

};

//Indication of the successful reception of a new request to
// application and it is called right before before the DSD
verification (SID, security access, diagnostic session). Within
// this function application can examine the permission of the
// diagnostic service / environment (e.g. ECU state afterrun).
// Return of application will lead to an acceptance of diagnostic
//request, reject of diagnostic request (E_REQUEST_NOT_ACCEPTED ->
// no response will be send) or to an NRC (E_NOT_OK) with
// value from ErrorCode parameter
//SID: Value of service identifier
//RequestData: This parameter contains the complete request
//data (diagnostic buffer), except the service ID.
//DataSize: This parameter defines how many bytes in the
//RequestData parameter are valid
//ReqType: Addressing type of the request(0=physical request
//1=functional request)
//SourceAddress: Dcm client description
Indication(IN uint8 SID,
           IN uint8 RequestData[<Data size>],
           IN uint16 DataSize,
           IN uint8 ReqType,
             IN uint16 SourceAddress,
,          OUT NegativeResponseCodeType* ErrorCode,
           ERR{E_REQUEST_NOT_ACCEPTED, E_NOT_OK });
//Confirmation of the successful reception of a new request to
// application
//SID: Value of service identifier
//ReqType: Addressing type of the request(0=physical request
//1=functional request)
//SourceAddress: Dcm client description
//SourceAddress: Dcm client description
//ConfirmationStatus: Confirmation of a successful transmission or a
transmission error of a diagnostic service.
Confirmation(
           IN uint8 SID,
           IN uint8 ReqType,
           IN uint16 SourceAddress,
           IN ConfirmationStatusType ConfirmationStatus,
           ERR{E_NOT_OK});
```

}」()

[Dcm677]「 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.」()

[Dcm678]「 If the operation Indication() or Confirmation() 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)
```

8.7.3.10 GeneralRequiredROEServices

The following interface defines an operation needed to supply ROE ISO Service.

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

Dcm765「

```
ClientServerInterface GeneralRequiredROEServices
{
PossibleErrors {
  E_NOT_OK = 1,
  };

// used to manage an ROE event with EventWindowTime
ROEInit (IN uint8 EventWindowTime,ERR{E_NOT_OK});

// used to stop the EventWindowTime when the ROE is stopped by an
//Diagnostic Request
ROEStop (void,ERR{E_NOT_OK});
```

```
}」()
```

8.7.3.11 GeneralProvidedROEServices

The following interface defines an operation needed to supply ROE ISO Service.

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

Dcm772「
```
ClientServerInterface GeneralRequiredROEServices
{
PossibleErrors {
  E_NOT_OK = 1,
  };

StopROE(ERR{E_NOT_OK});

RestartROE(ERR{E_NOT_OK});
}」()
```

From the point of view of the DCM, the operations has the following signatures:

```
Std_ReturnType Xxx_ROEInit (uint8 EventWindowTime)

Std_ReturnType Xxx_ROEStop (void)
```

8.7.3.12 ROEServices

The following interface defines an operation needed to supply ROE ISO Service.
Using the concepts of the SW-C template, the interface is defined as follows:

[Dcm695]「
```
ClientServerInterface ROEServices
{
PossibleErrors {
  E_NOT_OK = 1,
  };

// used to activate an ROE event
ActivateEvent(IN uint8 RoeEventId,
  IN Dcm_RoeStateType RoeState,
  ERR{E_NOT_OK});

}」()
```

[Dcm679] ⌈ If the operation ActivateEvent() returns value E_NOT_OK, the DCM module shall send a negative response with NRC 0x31 (Request out of range) ⌋()

From the point of view of the DCM, the operations has the following signatures:

```
Std_ReturnType Xxx_ActivateEvent(uint8 RoeEvenId,
                                 Dcm_RoeStateType RoeState)
```

8.8 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_<LEVEL> SecurityAccess_<LEVEL>;
  …


  //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 DcmDspTestResultTid for <TID> and
 //DcmDspTestResultObdmidTid for <OBDMID>
  RequirePort DTRServices DtrServices_<OBDMID>_<TID>;
  …
```

```
  //see configuration parameter DcmDspRoutine
  RequirePort RoutineServices_<ROUTINENAME>
    RoutineServices_<ROUTINENAME>;
  …

  //see configuration parameter DcmDspRequestControl
  RequirePort RequestControlServices_<TID>
    RequestControlServices_<TID>;
  …

  //see configuration parameter DcmDslCallbackDCMRequestService
  RequirePort CallbackDCMRequestServices
    CallbackDCMRequestServices_<SWC>;
  …

  //see configuration parameter
DcmDslServiceRequestManufacturerNotication
  RequirePort ServiceRequestNotification
    RequestManufacturerNotification_<SWC>;
  …

//see configuration parameter DcmDslServiceRequestSupplierNotication
  RequirePort ServiceRequestNotification
    ServiceRequestSupplierNotification_<SWC>;
  …

//see configuration parameter DcmDspDidExtRoe
  ProvidePort DCM_Roe DCM_Roe_<SWC>;…//see configuration parameter
DcmDspDidExtRoe
  RequirePort ROEServices ROEServices_<SWC>;
  …

//see configuration parameter DcmDspRoe
  RequirePort GeneralRequiredROEServices GeneralRequiredROEServices;
  ProvidePort GeneralProvidedROEServices GeneralProvidedROEServices;
  …

//Interface containing the DEM operation ClearDtc
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_GetNextFilteredDTCAndFDC), 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


}
```

8.9 External diagnostic service processing

The following chapter applies only to external processed diagnostic services.

8.9.1 Dcm_ExternalSetNegResponse

Dcm761「

Service name:	Dcm_ExternalSetNegResponse	
Syntax:	void Dcm_ExternalSetNegResponse(Dcm_MsgContextType* pMsgContext, Dcm_NegativeResponseCodeType ErrorCode)	
Service ID[hex]:	0x30	
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.	

」()

8.9.2 Dcm_ExternalProcessingDone

Dcm762「

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.	

」()

8.9.3 <Module>_<DiagnosticService>

Dcm763「

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 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	

」()

Dcm732「For the first call of <Module>_<DiagnosticService> the opStatus shall be set to DCM_INITIAL」()

Dcm733「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」()

Dcm734 「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」()

Dcm735 「In case of cancellation the API <Module>_<DiagnosticService> is called again with the parameter opStatus set to DCM_CANCEL」()

Dcm738 「The DCM shall not use the opStatus parameter DCM_FORCE_RCRRP_OK in the API <Module>_<DiagnosticService>」()

Dcm760 「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」()

8.9.4 <Module>_<DiagnosticService>_<SubService>

Dcm764 「

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 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.	

⌋()

8.10 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.

8.10.1 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()`

8.10.2 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()`

8.10.3 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.

8.10.4 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.

8.10.5 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).

8.10.6 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!

8.10.7 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.

8.10.8 DsdInternal_ProcessPage

```
void
DsdInternal_ProcessPage(Dcm_MsgLenType FilledPageLen)
```

DSP requests transmission of filled page

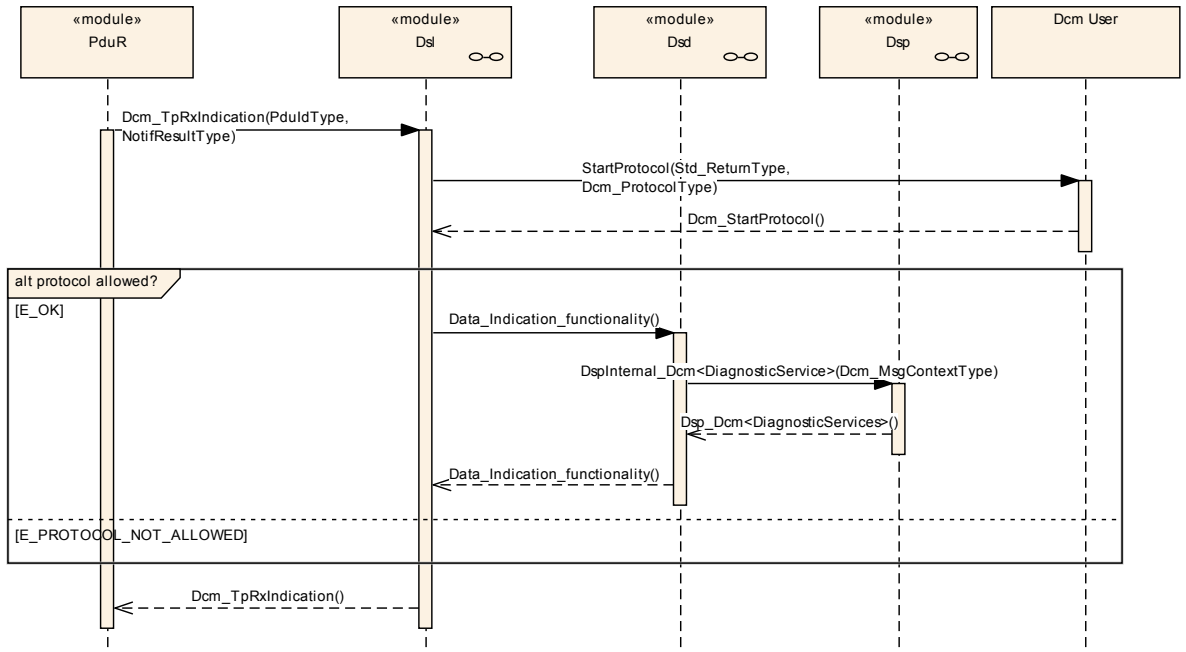
9 Sequence diagrams

9.1 Overview

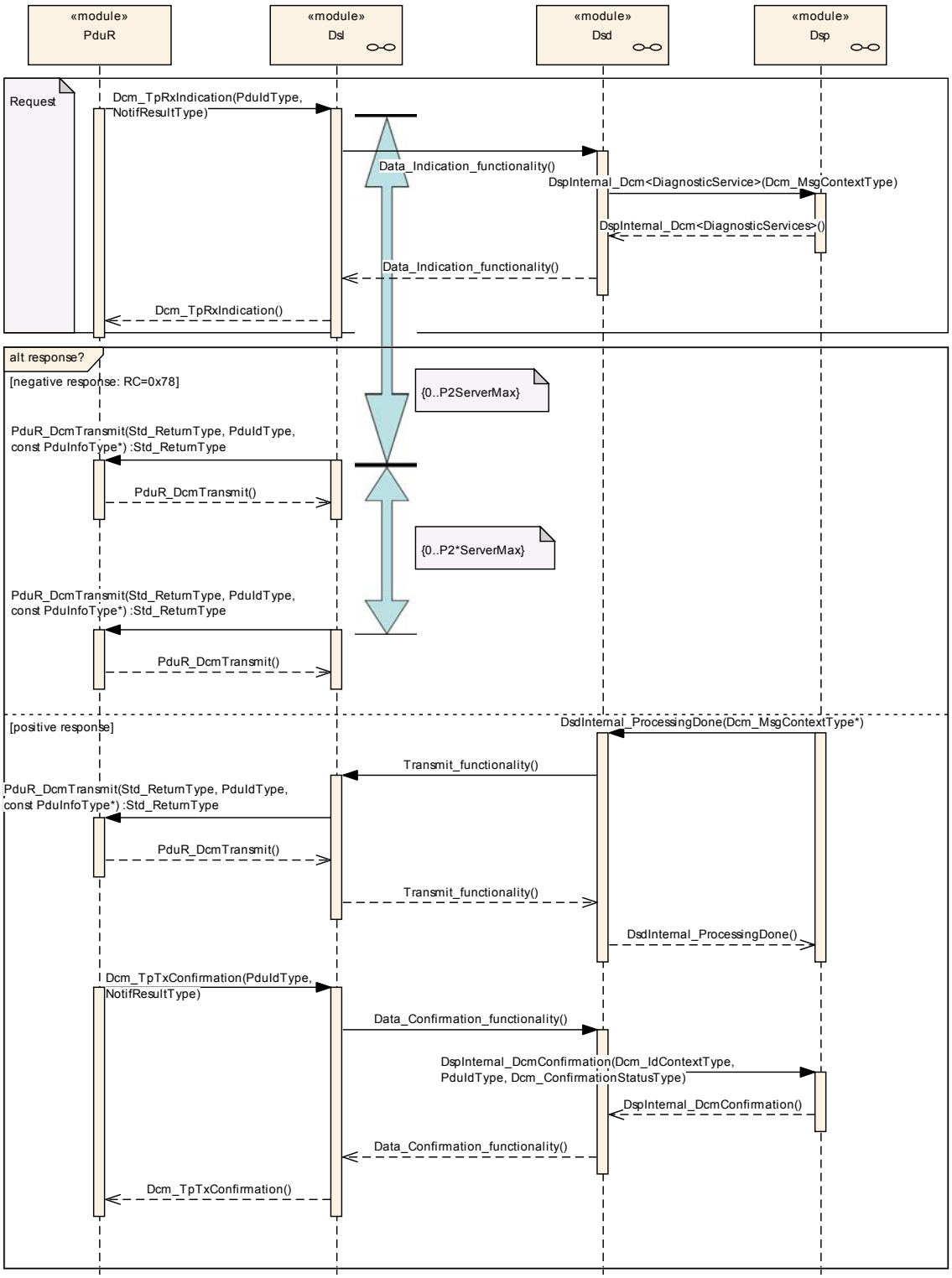
For clarification, the following sequence diagrams don't represent the full communication mechanism between the DCM module and the PduR module. This is to keep the sequence diagrams simple.
Before the `Dcm_TpRxIndication()` call, the PduR module will ask the DCM module for a buffer by calling `Dcm_StartOfReception()` 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



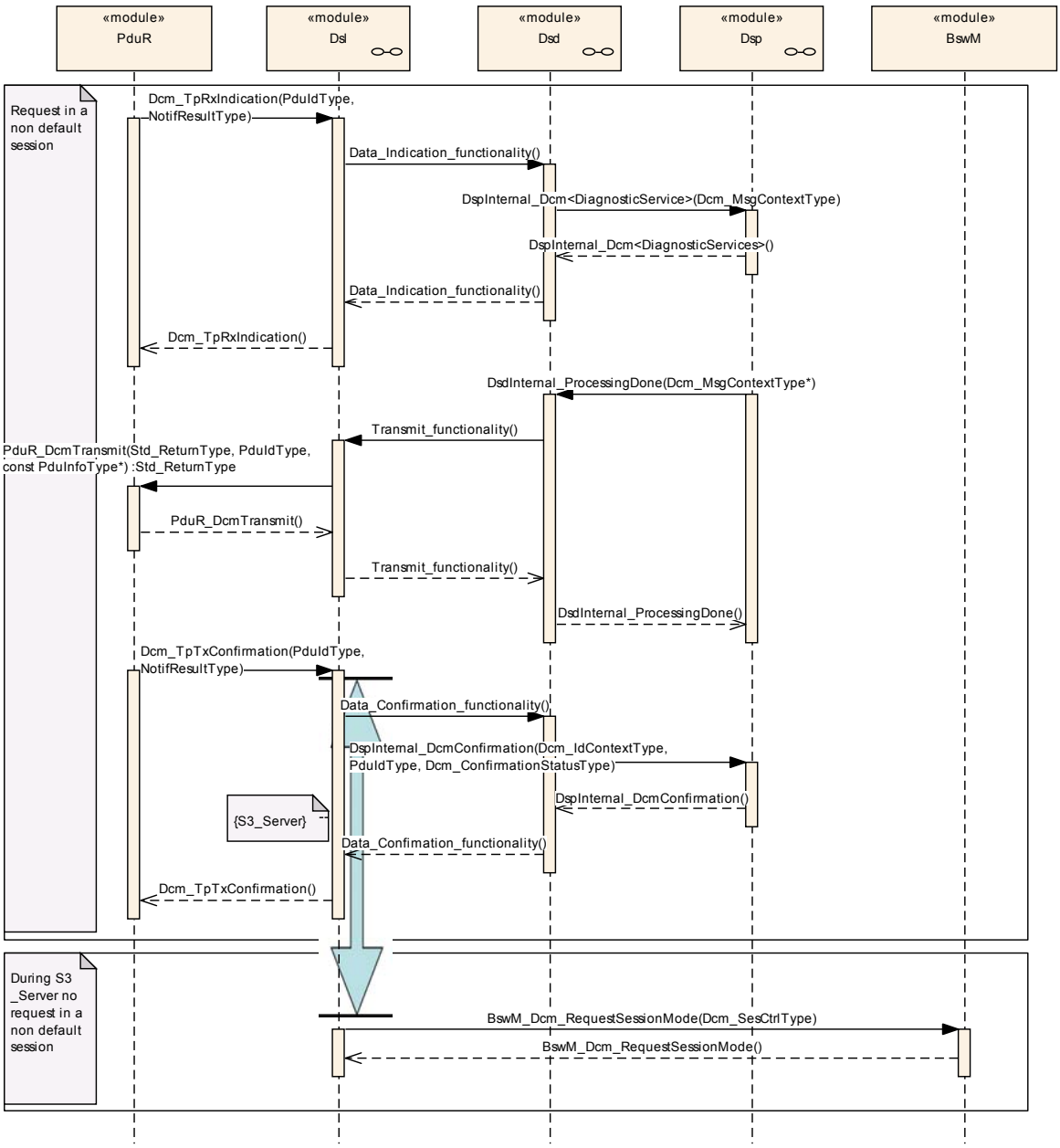
9.2.2 Process Busy behavior



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

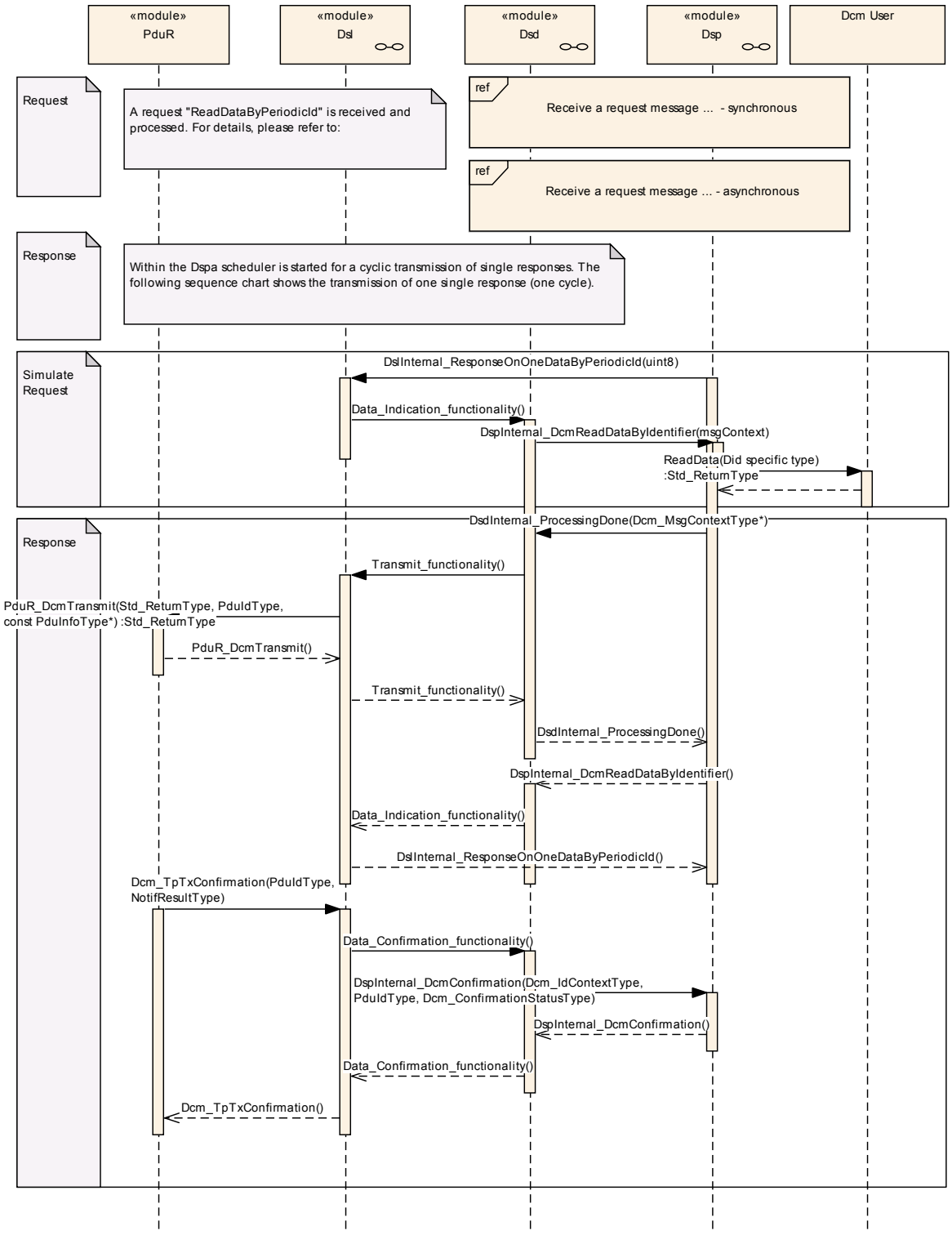
P2ServerMax and/or P2*ServerMax, the DSL submodule sends a negative response (requestCorectlyReceived-ResponsePending) independently.

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.

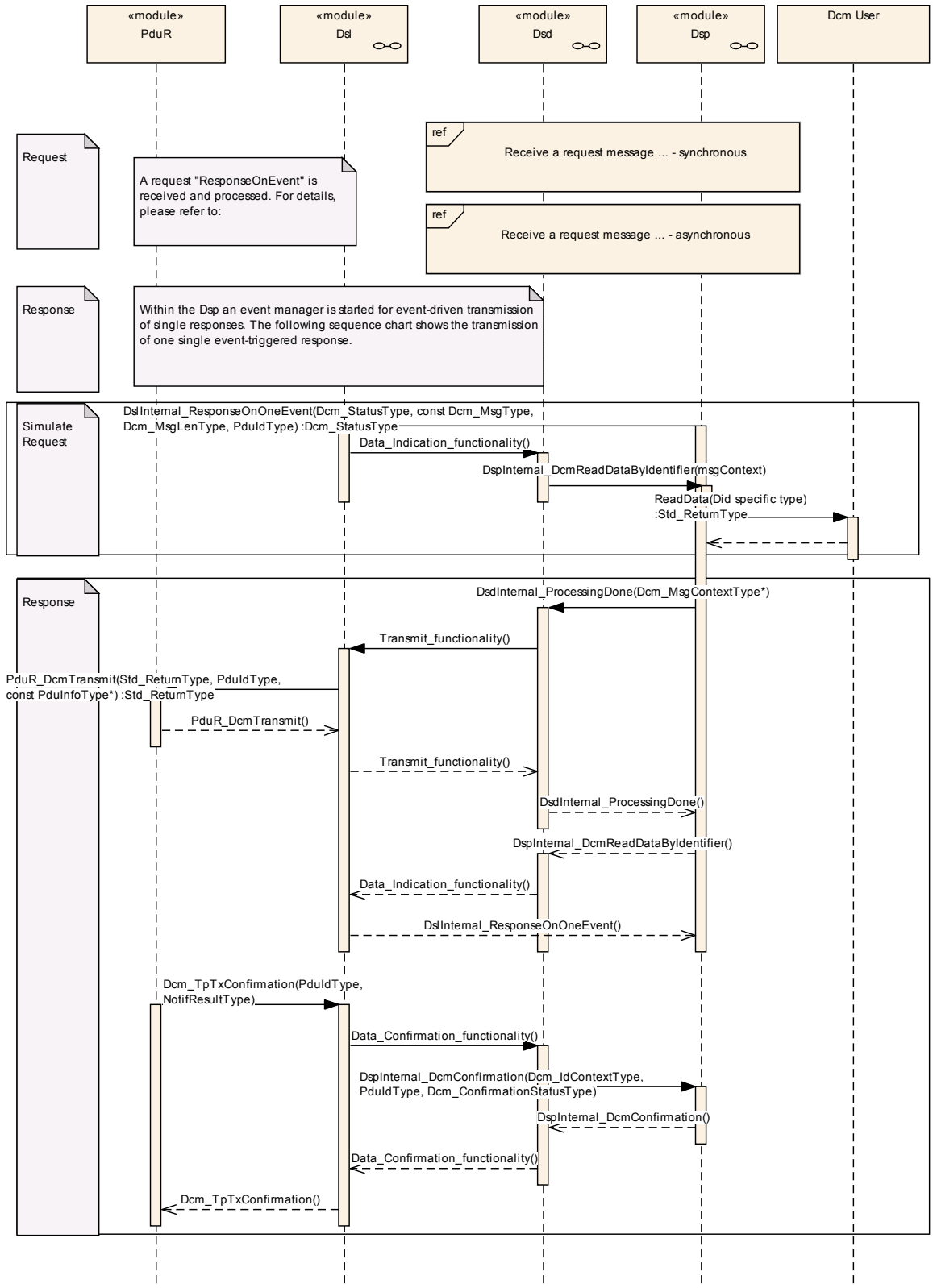
9.2.4 Process single response of ReadDataByPeriodicIdentifier



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.

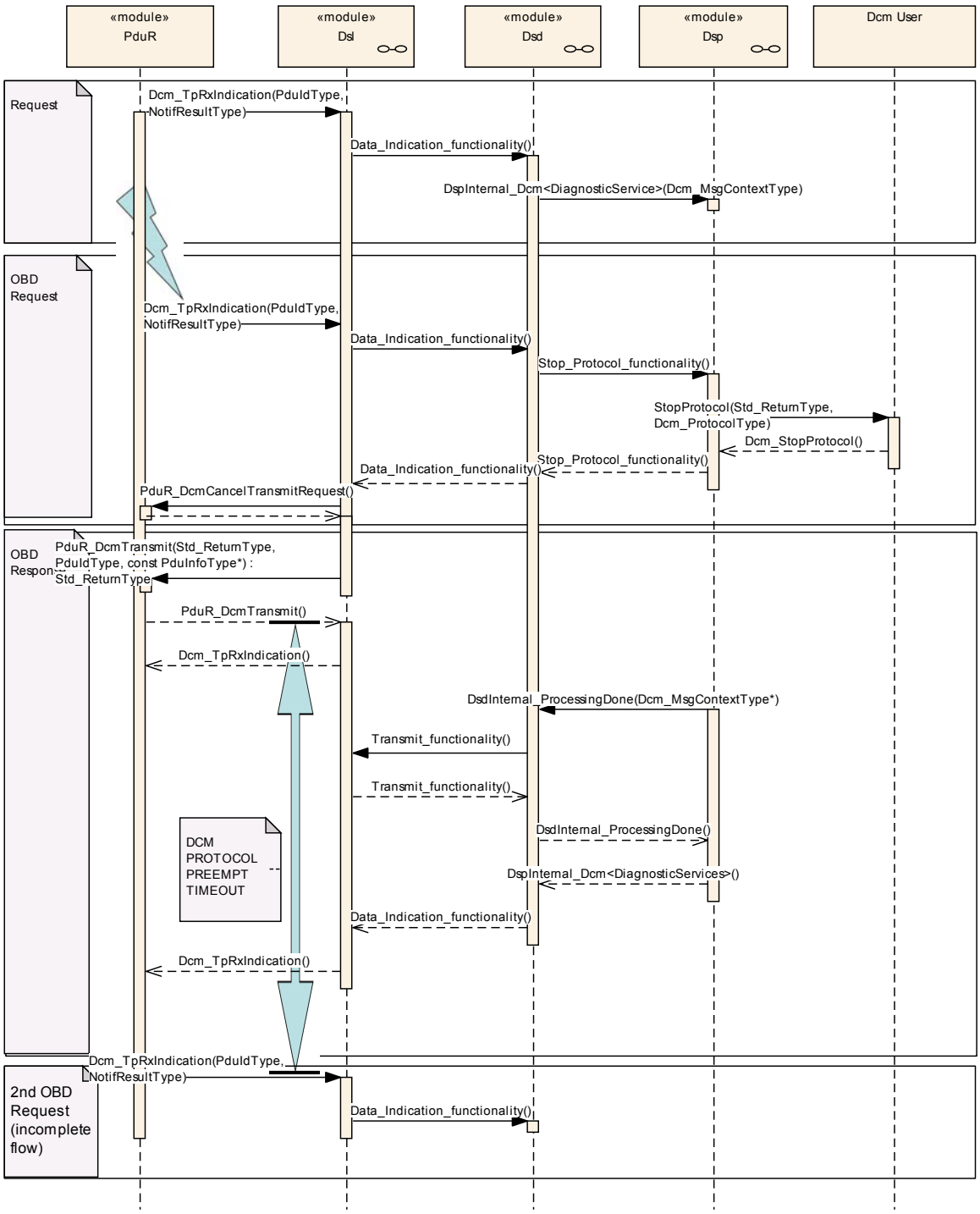
9.2.5 Process single event-triggered response of ResponseOnEvent



The DSP submodule requests transmission of ResponseOnEvent data, when ROE event occurs (`DslInternal_ResponseOnOneEvent()`).

Within this function the DSL submodule simulates a corresponding diagnostic request that has already been registered. This request will be processed by the DSP submodule.
The ReadData interfaces of the corresponding Datas from the DID is called to get the DID value.
The DCM module will not be able to process multiple event-triggered responses at one time.

9.2.6 Process concurrent requests

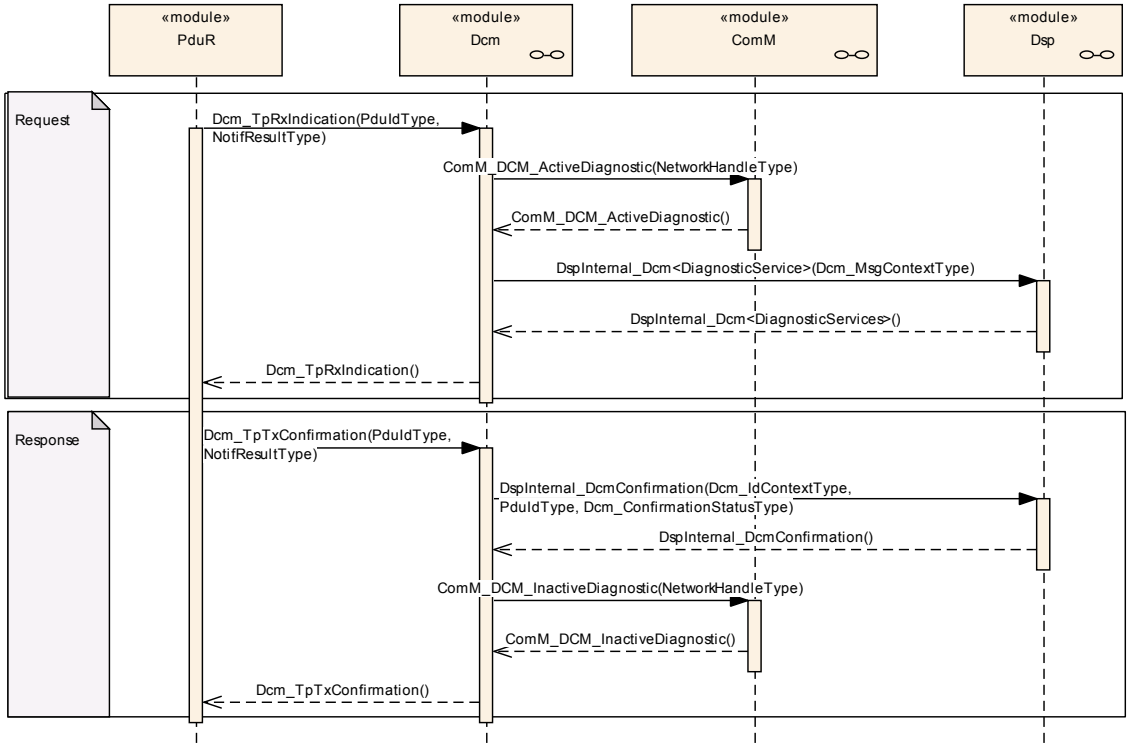


On reception of OBD request in parallel to processing of a normal diagnostic request (e.g enhanced diagnostic protocol, customer diagnostic protocol), running diagnostic request will be preempted. This is due to the configured higher priority of OBD protocol (see configuration parameter ***DcmDslProtocolPriority***).
The following is processed on reception of 1st OBD request:
- The Application is informed of the protocol stop (done with `Xxx_StopProtocol()`) and resets to a stable state (e.g. switch of digital Ios,..).

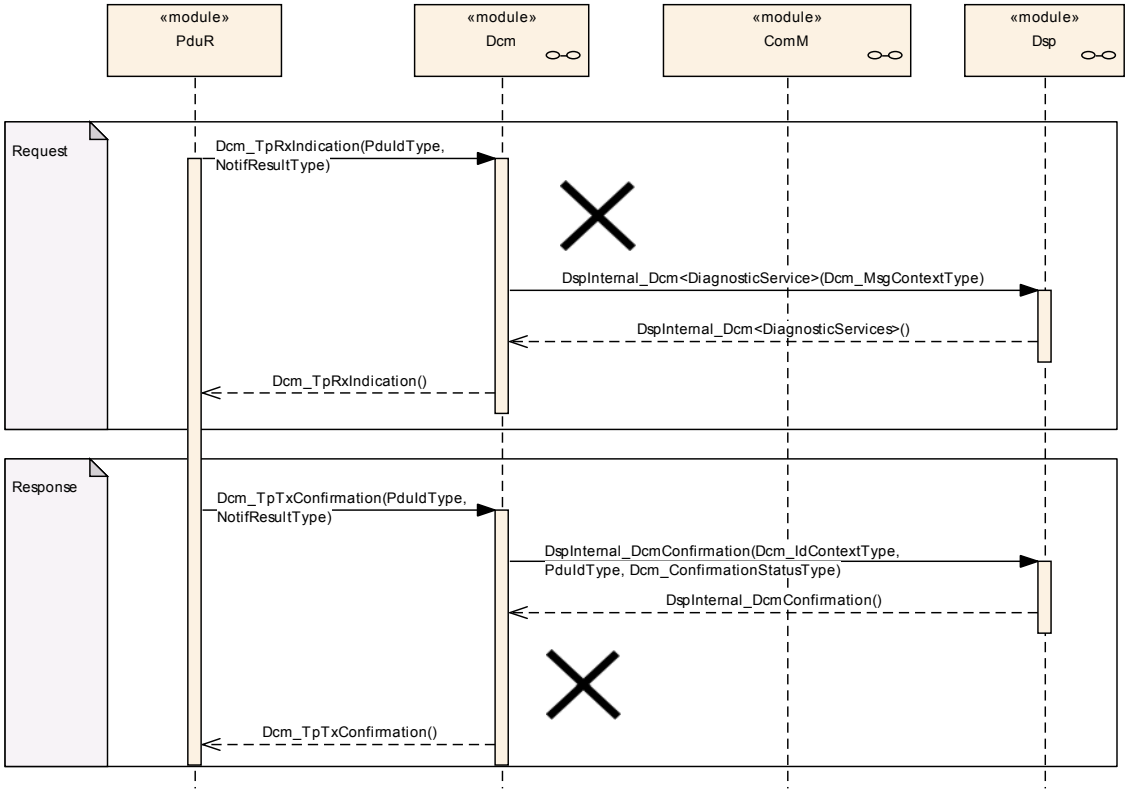
- Lower Layer is requested to cancel ongoing transmission on the same N-PDU (done with `PduR_DcmCancelTransmitRequest()`).
- If the DCM is not able to switch fast enough from non OBD to OBD protocol, the DSL submodule responses with a negative response “BusyRepeatRequest” (NRC 0x21) to OBD tester.
It 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

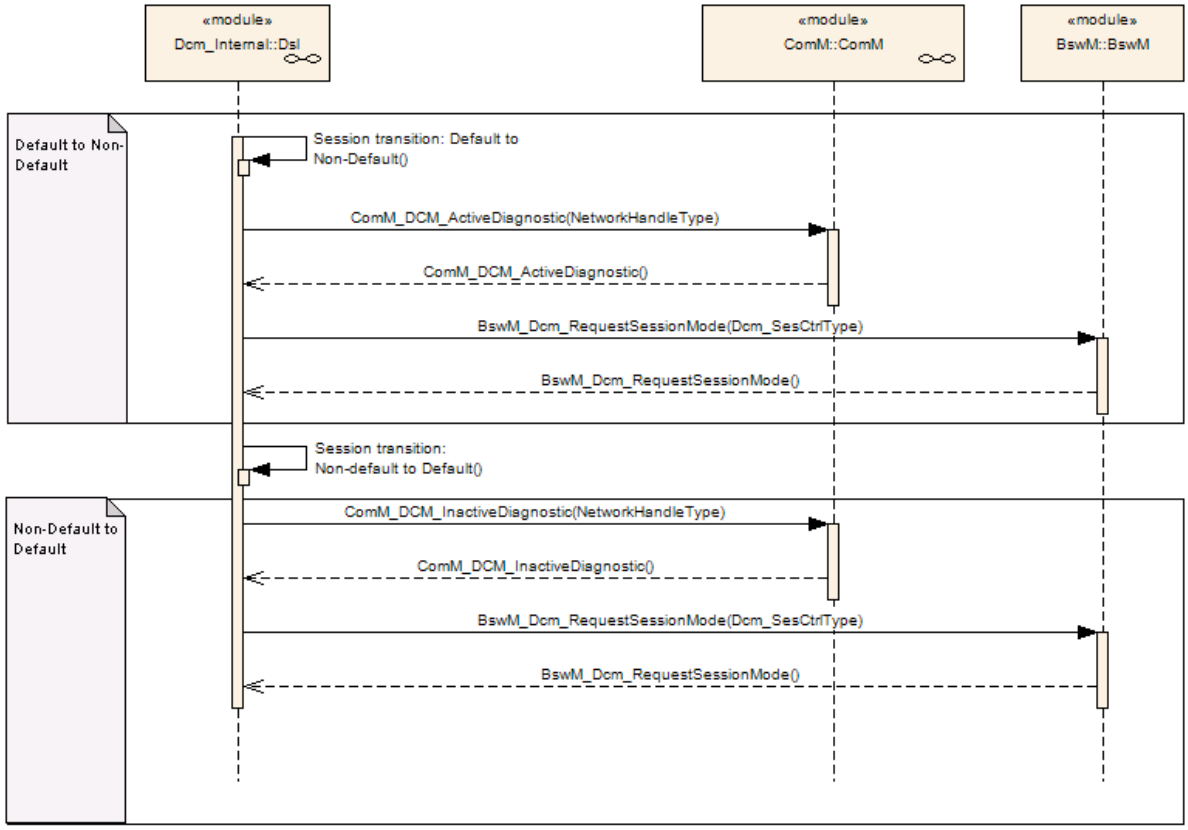
9.2.7.1 Handling in Default Session



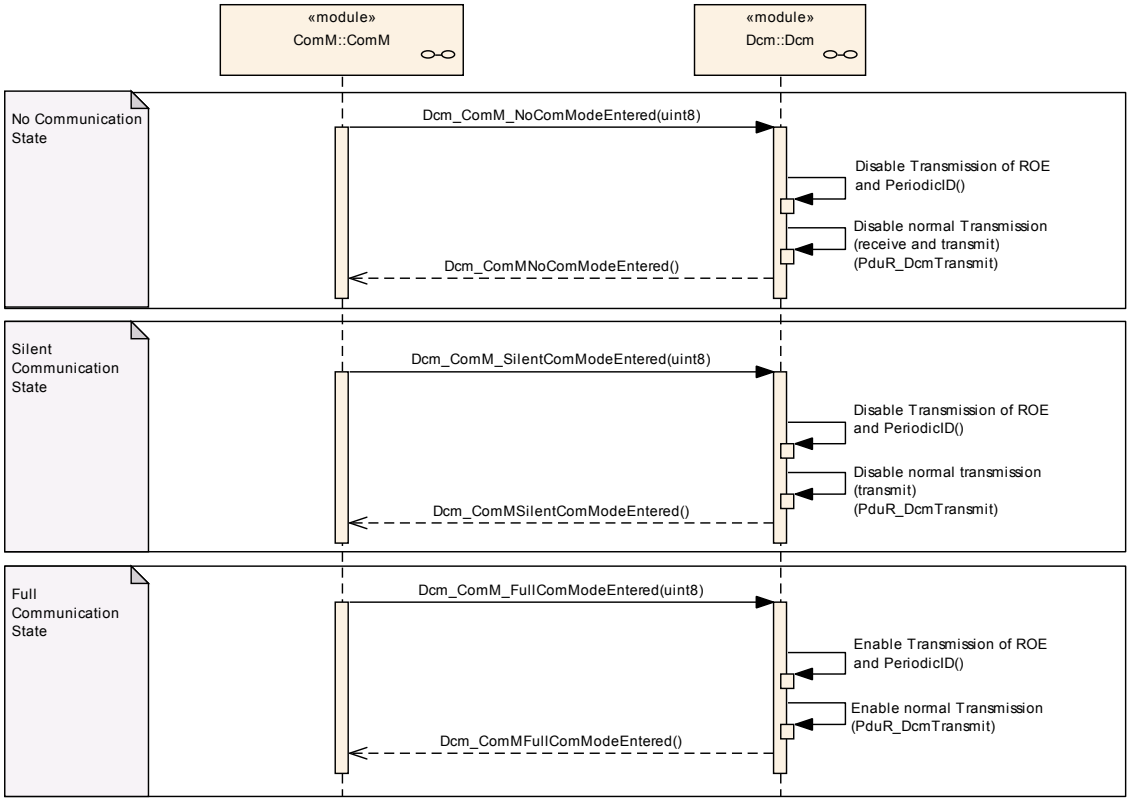
9.2.7.2 Handling in Non-Default Session



9.2.7.3 Session transitions

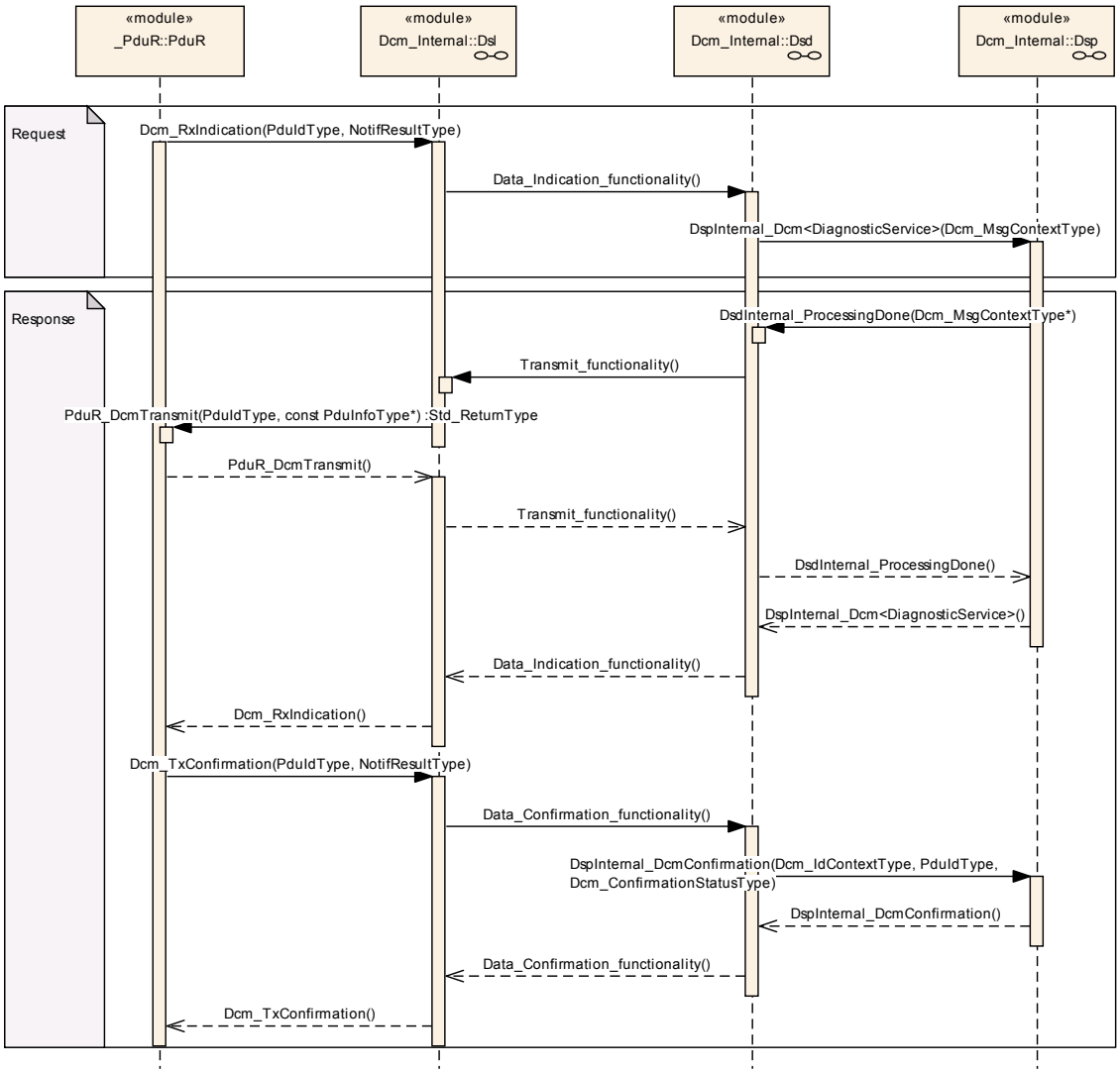


9.2.7.4 Communication States

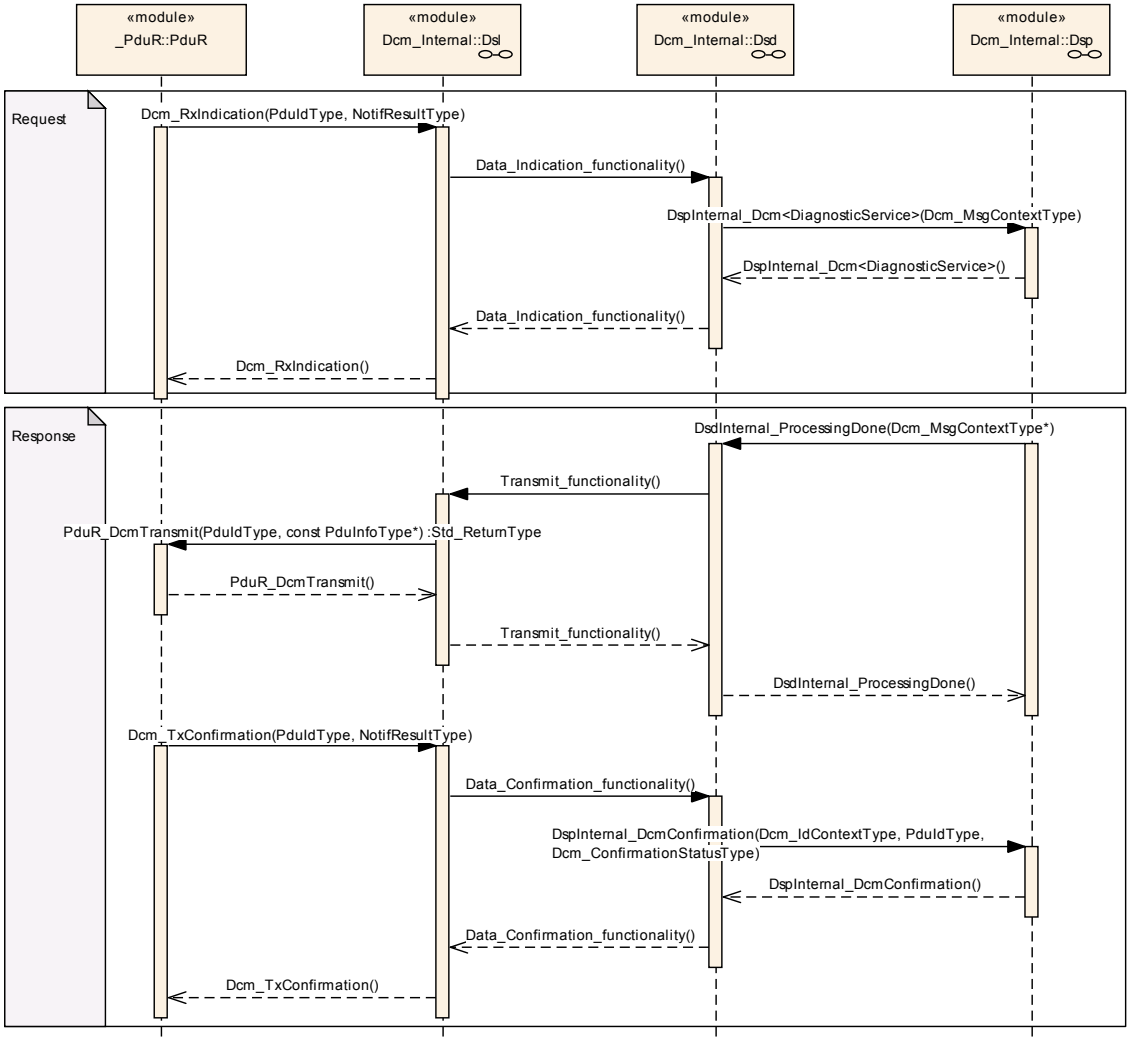


9.3 DSD (Diagnostic Service Dispatcher)

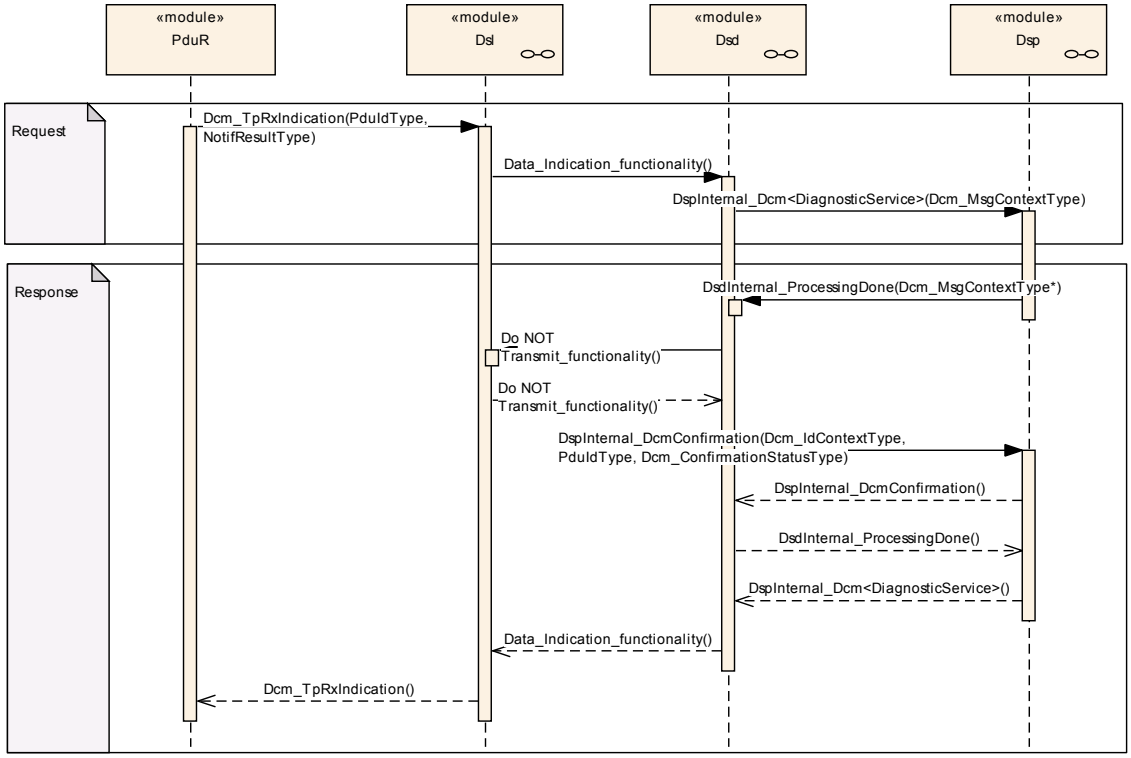
Receive a request message and transmit a positive response message – synchronous transmission



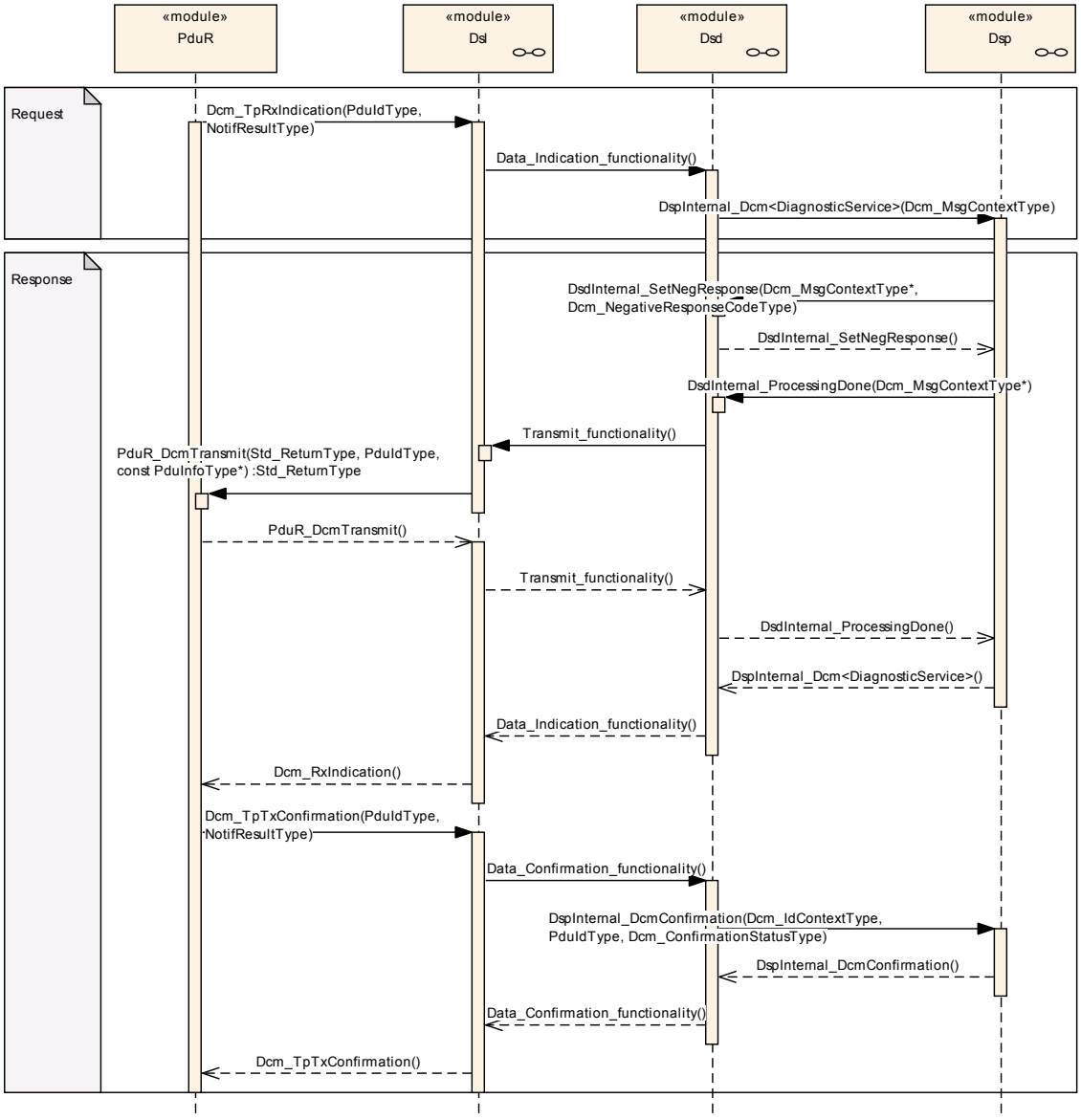
Receive a request message and transmit a positive response message – asynchronous transmission



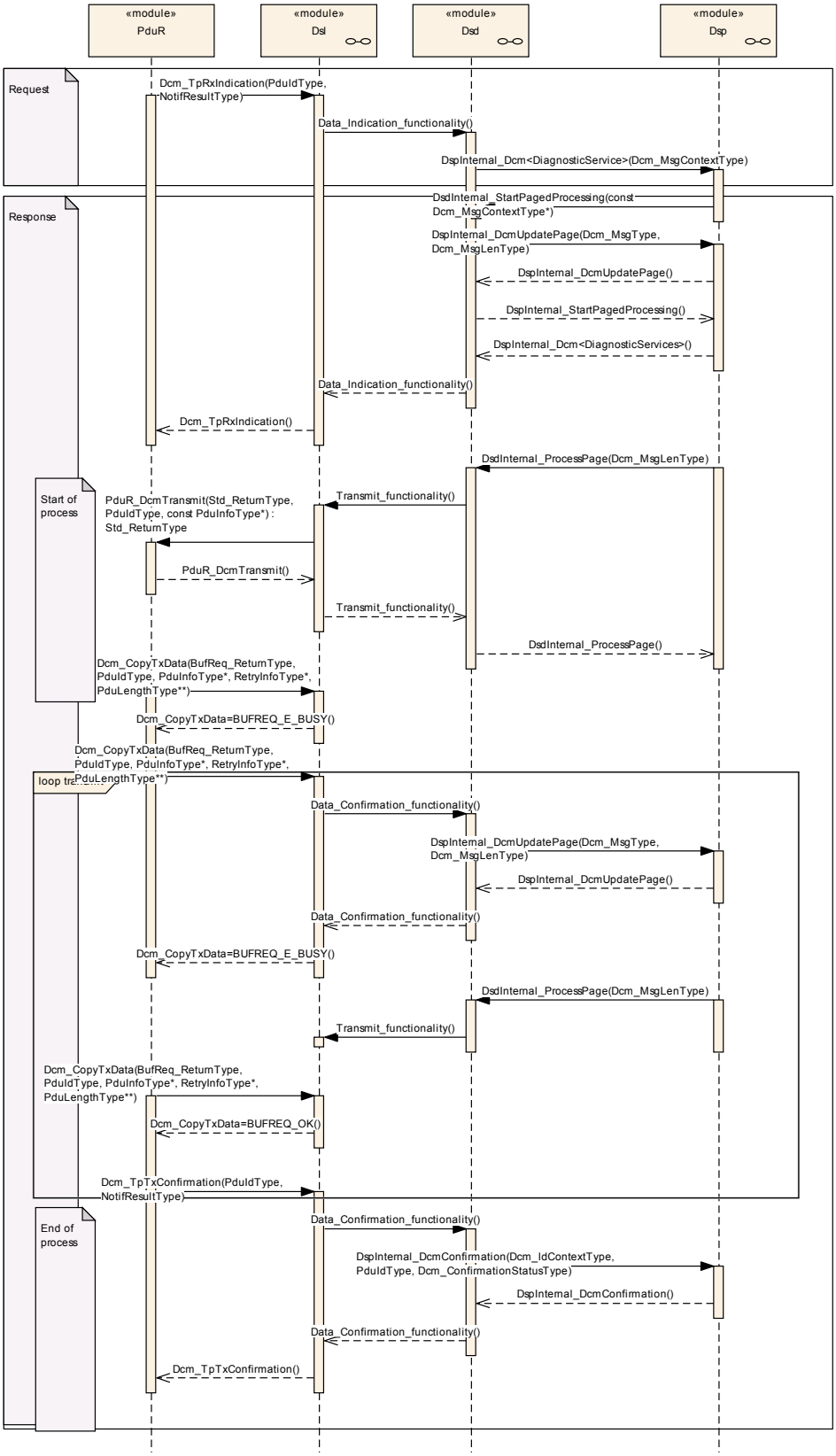
Receive a request message and suppress a positive response


«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType, NotifResultType)
Data_Indication_functionality()
DspInternal_Dcm<DiagnosticService>(Dcm_MsgContextType)
Response
DsdInternal_ProcessingDone(Dcm_MsgContextType*)
Do NOT Transmit_functionality()
Do NOT Transmit_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
DsdInternal_ProcessingDone()
DspInternal_Dcm<DiagnosticService>()
Data_Indication_functionality()
Dcm_TpRxIndication()

9.3.1 Receive request message and transmit negative response message



9.3.2 Process Service Request with paged-buffer



The following flow is processed in case no error occurs on the Application side:

Start of process:

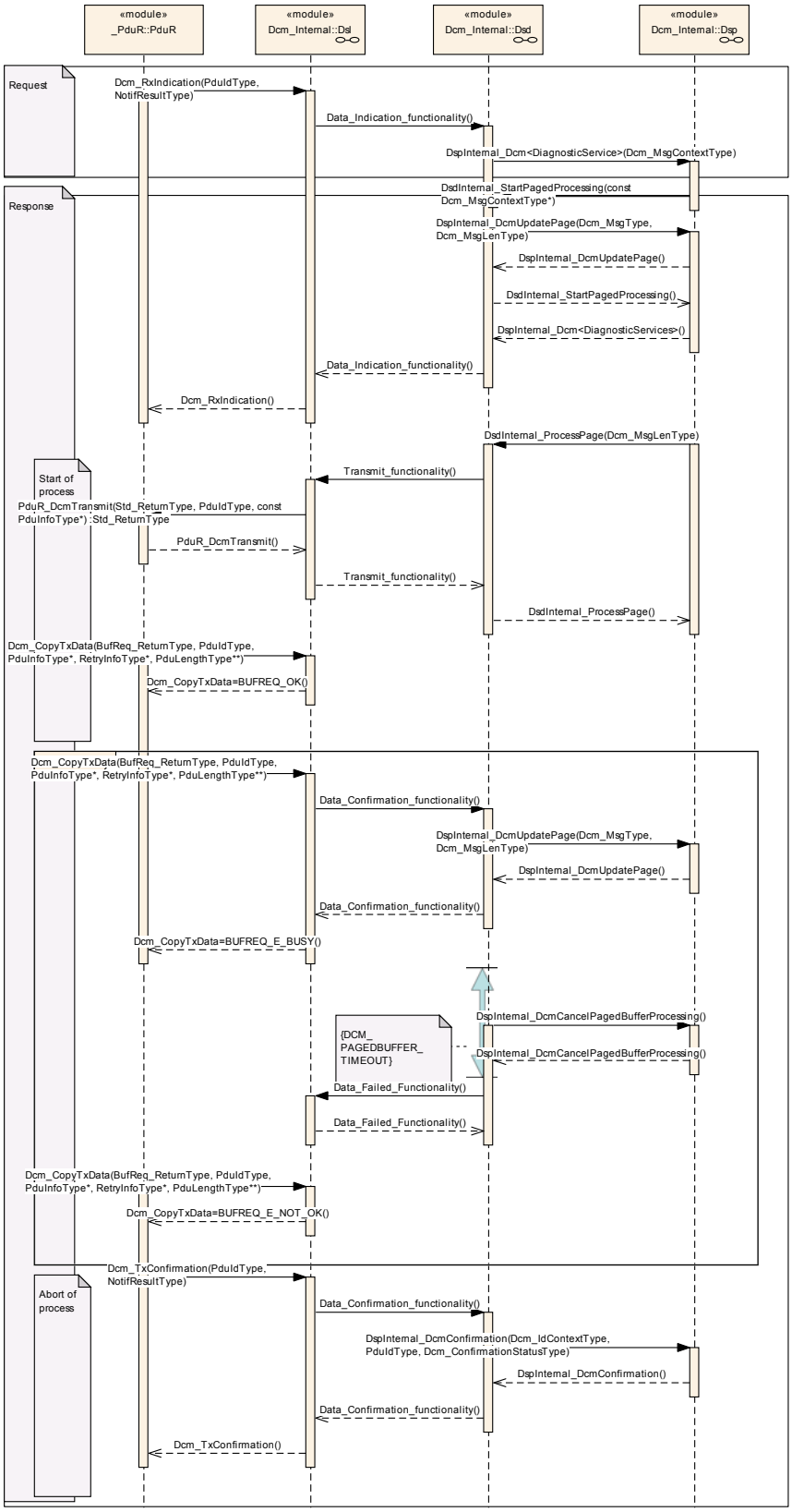
4) `DsdInternal_StartPagedProcessing()`: With this API, the DSP submodule gives the complete response length to the DCM module and starts paged-buffer handling. This API starts no transmission!
5) UpdatePage(): The DCM module requests data to be transmitted.
6) `DsdInternal_ProcessPage()`: With this API, the DSP submodule requests transmission of the current page.
8) `PduR_DcmTransmit()`: The DCM module requests transmission to the lower layers.
9) `Dcm_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::PduR
«module»
Dcm_Internal::Dsl
«module»
Dcm_Internal::Dsd
«module»
Dcm_Internal::Dsp
Request
Dcm_RxIndication(PduIdType, NotifResultType)
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_RxIndication()
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, PduInfoType*, RetryInfoType*, PduLengthType**)
Dcm_CopyTxData=BUFREQ_OK()
Dcm_CopyTxData(BufReq_ReturnType, PduIdType, 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, PduInfoType*, RetryInfoType*, PduLengthType**)
Dcm_CopyTxData=BUFREQ_E_NOT_OK()
Dcm_TxConfirmation(PduIdType, NotifResultType)
Abort of process
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TxConfirmation()

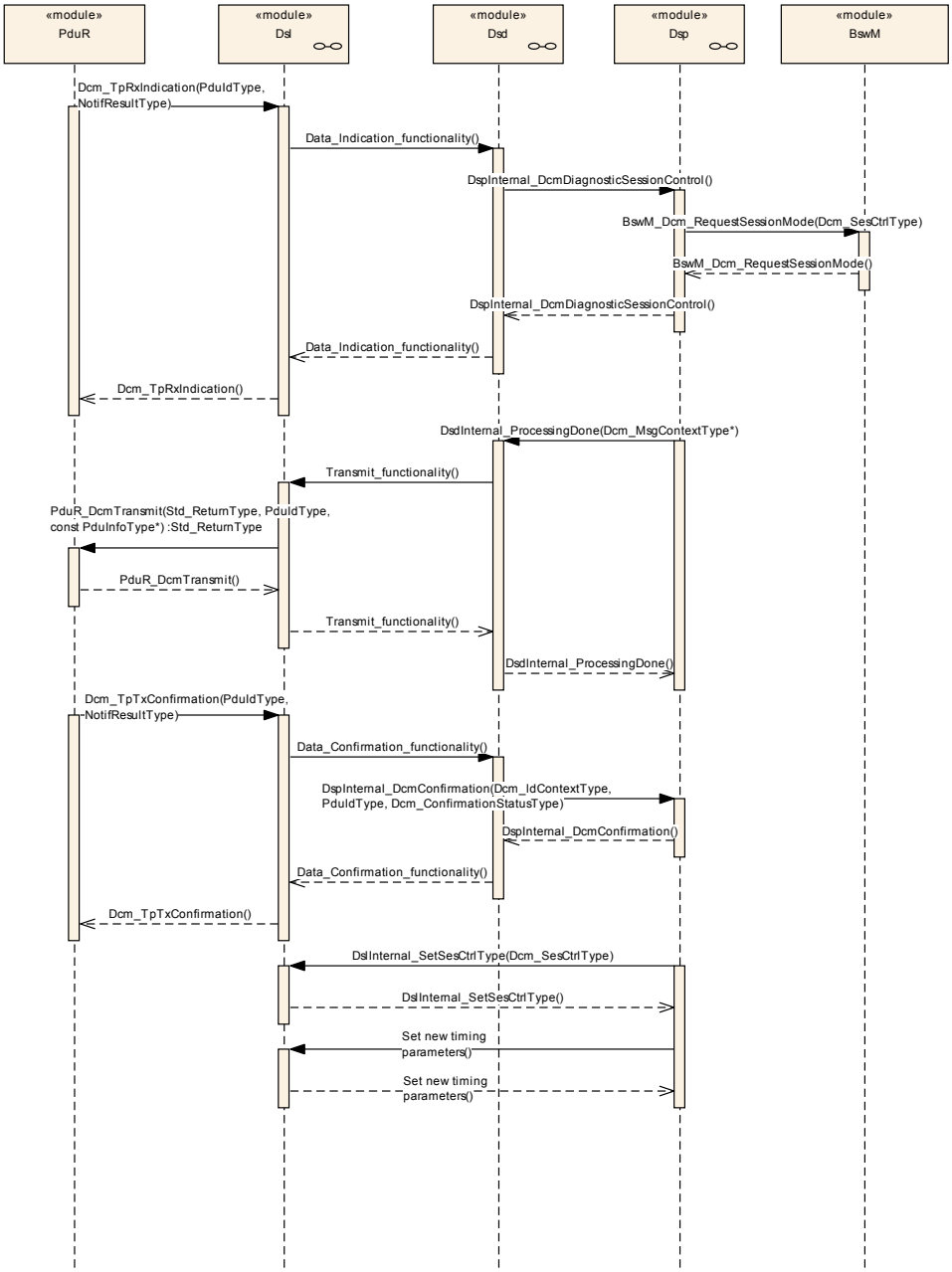
9.4 DSP (Diagnostic Service Processing)

9.4.1 Interface DSP – DEM (service 0x19, 0x14, 0x85)

Please refer to Section 9. In [7].

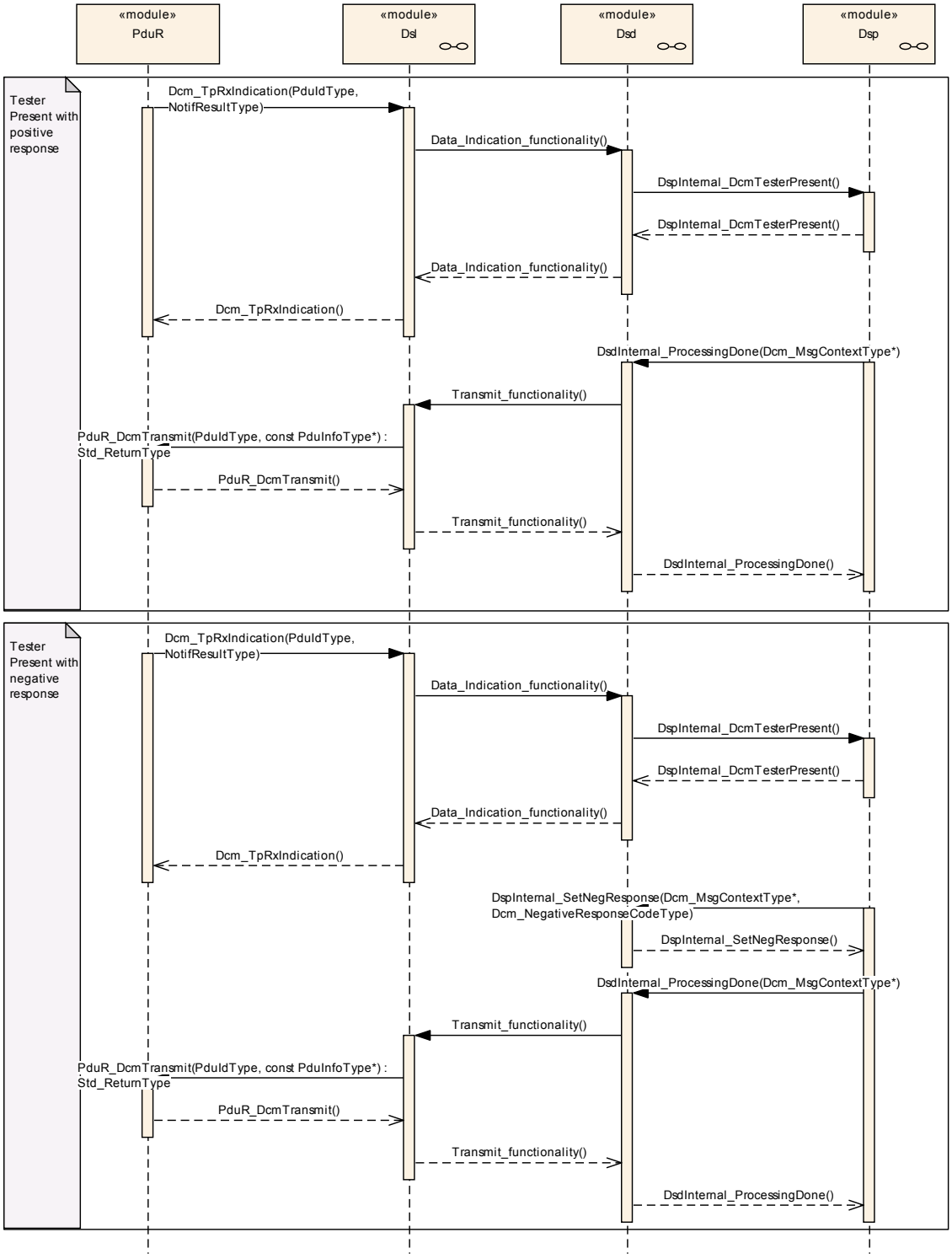
9.4.2 Interface special services

9.4.2.1 Process Diagnostic Session Control



On a Diagnostic Session Control request from a tester, the DSP submodule requests the Application to get the permission to change the session.
With the permission from the Application, a positive response is given to the PduR module. In the data confirmation function the new session type and the new timing values are set.
The DSL submodule indicates the session change to the Application (`Xxx_ChangeIndication()`).

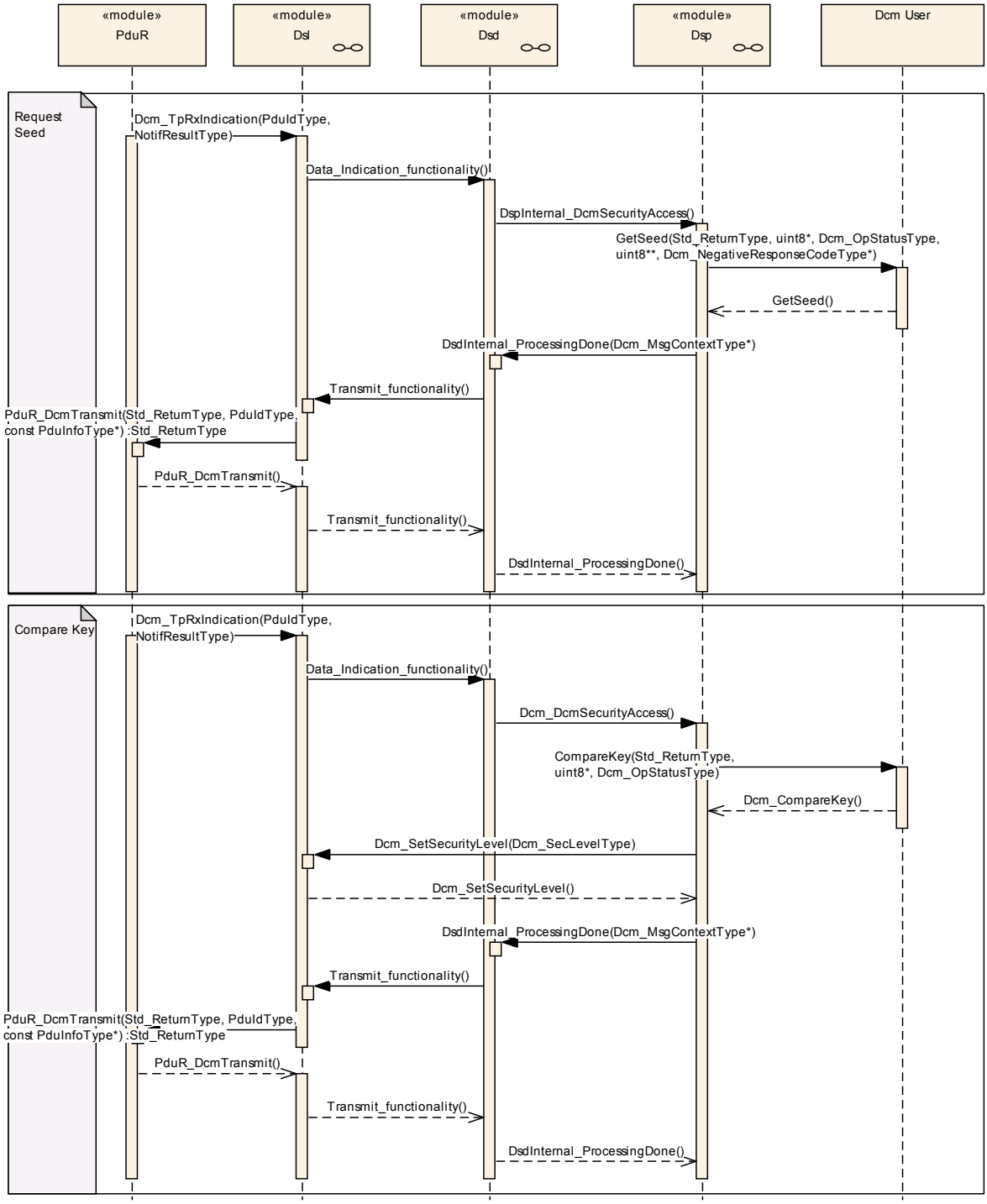
9.4.2.2 Process Tester Present



Above sequence diagram shows processing of TesterPresent commands, which are not of type functional addressed with subfunction 0x80. These TesterPresent commands are interpreted in the DSL submodule (more details can be found in Section 7.2.4.3 Concurrent “TesterPresent” (“keep alive logic”))

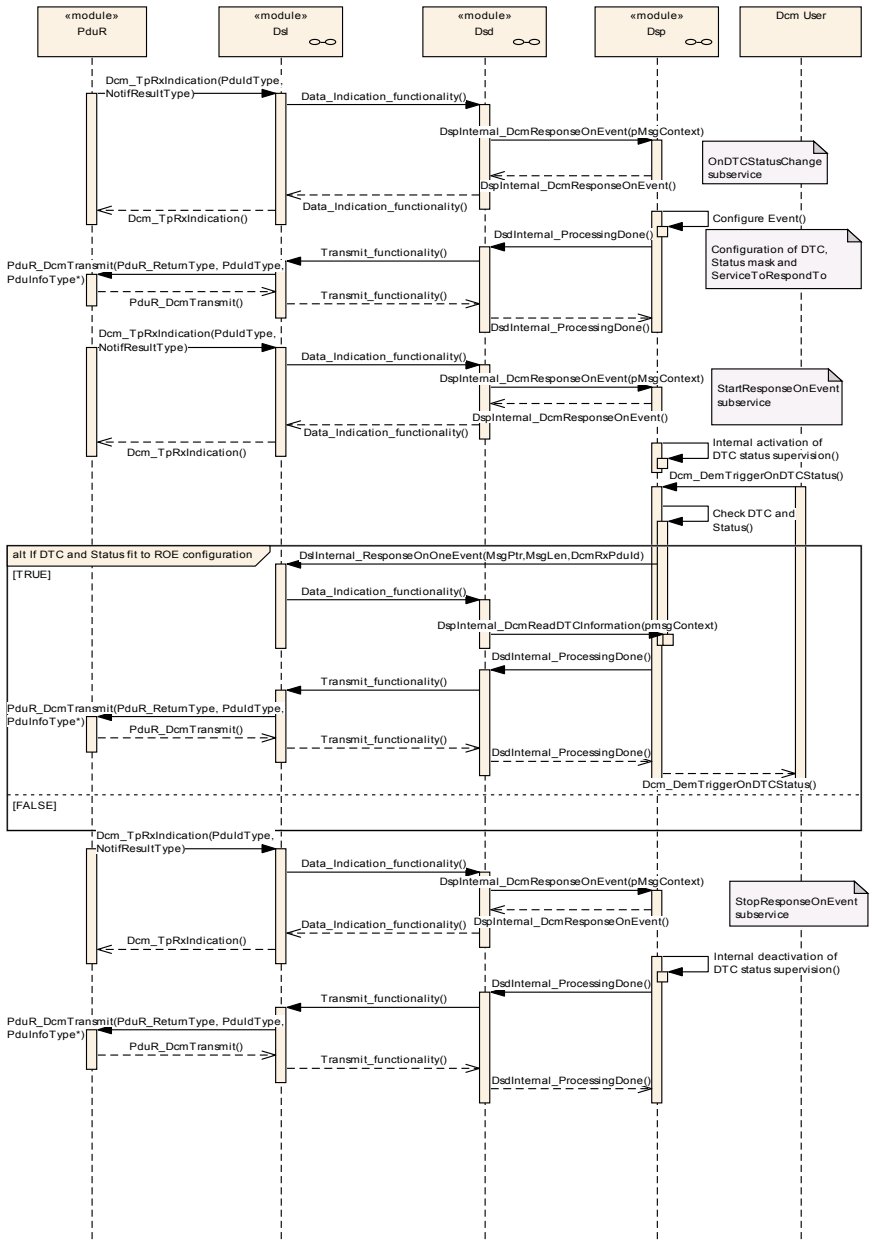
All the other TesterPresent commands are processed in the following way:
On a command TesterPresent the DSD submodule calls the DSP submodule with the function TesterPresent(). The sequence chart also shows the case when an error occurs and a negative response is sent.

9.4.2.3 Process Security Access



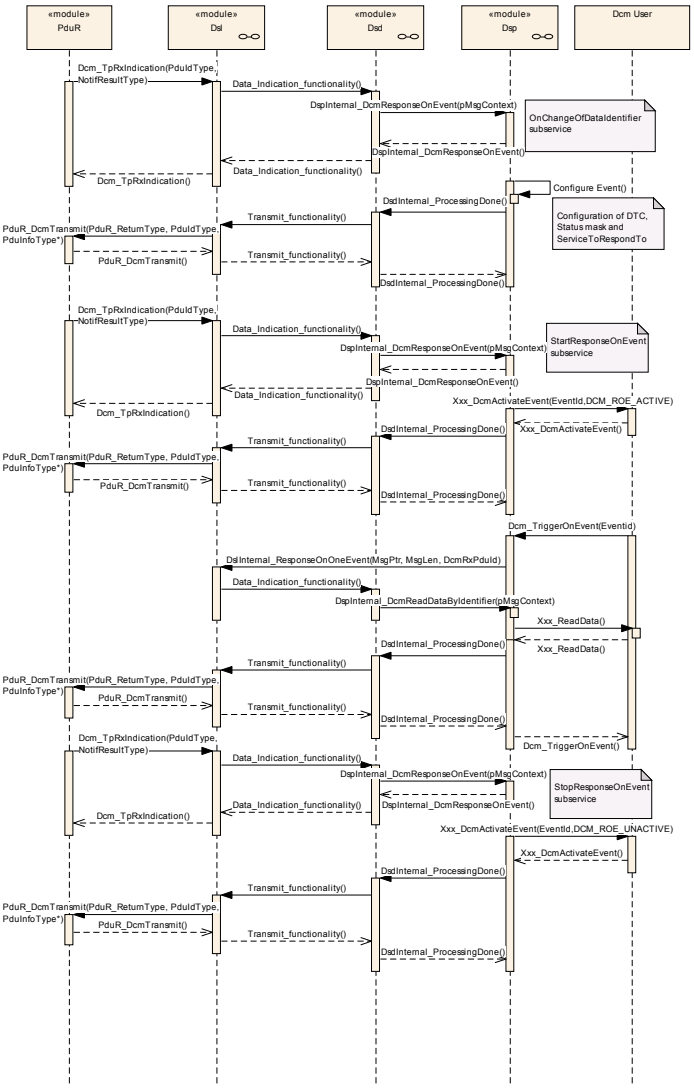
To get the security access, the DSD submodule has to call the DSP submodule to get the seed value from the application. If no error is detected, the seed value is sent in the positive response.
In a second step, the DSP submodule gets the key calculated by the tester and requests the application to compare this key with the internal calculated key. If no error occurs, the new access type is set in the DSL submodule and a positive response is sent.

9.4.2.4 Process ResponseOnEvent OnDtcChange



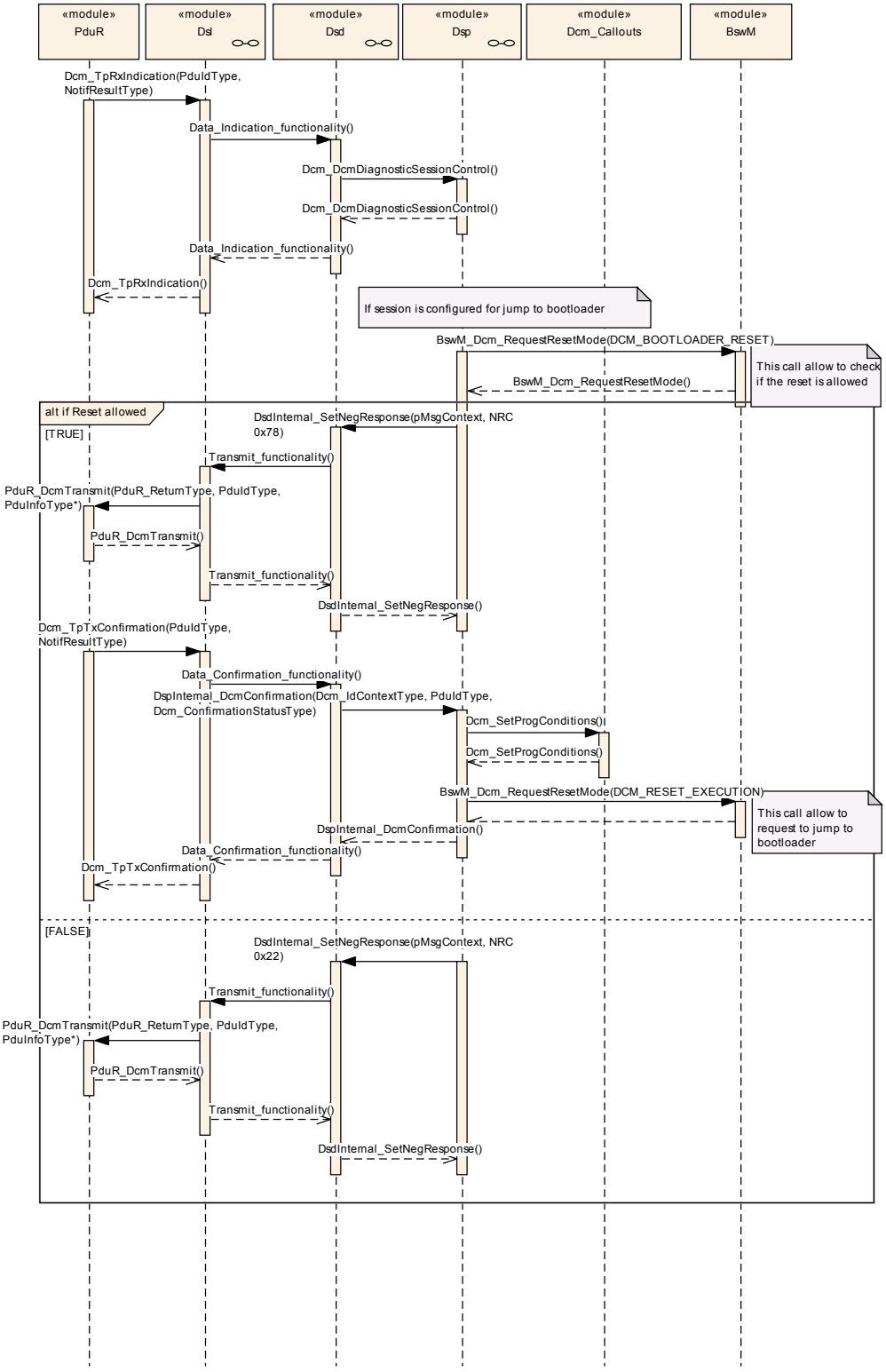
Above sequence diagram shows processing of ResponseOnEvent service for sub-service OnDtcChange.
After configuration and activation of the event by the service ResponseOnEvent, the DCM checks the status of the configured DTC on every call to interface Dcm_DemTriggerOnDTCStatus() in order to identify if the event shall be trigger. This interface is called by DEM for any DTC status change and independing of the activation/unactivation of ResponseOnEvent.

9.4.2.5 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.

9.4.2.6 Process Jump to Bootloader



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
- DCM request the jump to bootloader to BswM module

10 Configuration specification

10.1 How to read this section

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [5]
- AUTOSAR ECU Configuration Specification [6]. This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.
The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.
Pre-compile time specifies whether the configuration parameter shall be of configuration class Pre-compile time or not.

Label	Description
x	The configuration parameter shall be of configuration class Pre-compile time.
--	The configuration parameter shall never be of configuration class Pre-compile time.

Link time specifies whether the configuration parameter shall be of configuration class Link time or not.

Label	Description
x	The configuration parameter shall be of configuration class Link time.
--	The configuration parameter shall never be of configuration class Link time.

Post Build specifies whether the configuration parameter shall be of configuration class Post Build or not.

Label	Description
x	The configuration parameter shall be of configuration class Post Build and no specific implementation is required.
L	Loadable – the configuration parameter shall be of configuration class Post Build and only one configuration parameter set resides in the ECU.
M	Multiple – the configuration parameter shall be of configuration class Post Build and is selected out of a set of multiple parameters by passing a

Label	Description
	dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class Post Build.

10.1.2 Variants

The DCM module has the following variants:

[Dcm171] ⌈ VARIANT-PRE-COMPILE: Only parameters with “Pre-compile time” configuration are allowed in this variant.⌋(BSW00396)

[Dcm172] ⌈ VARIANT-LINK-TIME: Only parameters with “Pre-compile time” and “Link time” are allowed in this variant.⌋(BSW00396)

[Dcm173] ⌈ VARIANT-POST-BUILD: Parameters with “Pre-compile time”, “Link time” and “Post-build time” are allowed in this variant.⌋(BSW00396)

In variant POST-BUILD the pre-compile parameter shall be used to enable or disable the functionality (e.g. ROE transmission DCM_ROE_ENABLED). With the post build parameter the functionality shall be configured (e.g. DCM_ROE_TRANS_TYPE).
Please note: This pre-compile configuration parameters are mandatory for all variants. The pre-compile parameter shall be used to enable or disable DCM functionality (e.g. ROE transmission DCM_ROE_ENABLED) or are DCM global configuration data that shall be configured at pre complile time (e.g. memory influence).

10.1.3 Containers

The Containers structure the set of configuration parameters. This means:

- all configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

Containers have names indicating what kinds of parameters are handled inside.

10.2 DCM configurations

10.2.1 Dcm

Module Name	**Dcm**
Module Description	Configuration of the Dcm (Diagnostic Communications Manager) module.

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmConfigSet	1	This container contains the configuration parameters and sub containers of the DCM module supporting multiple configuration sets. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.

10.2.2 DcmConfigSet

SWS Item	**Dcm819_Conf :**
Container Name	DcmConfigSet [Multi Config Container]
Description	This container contains the configuration parameters and sub containers of the DCM module supporting multiple configuration sets. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
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	**Dcm688_Conf :**
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	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDsdServiceTable	1..256	This container contains the configuration (DSD parameters) for Service Identifier Table.

10.2.4 DcmDsdServiceTable

SWS Item	**Dcm732_Conf :**
Container Name	DcmDsdServiceTable
Description	This container contains the configuration (DSD parameters) for Service Identifier Table.
Configuration Parameters	

SWS Item	**Dcm736_Conf :**		
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	**Dcm689_Conf :**
Container Name	DcmDsdService
Description	This container contains the configuration (DSD parameters) for Service.
Configuration Parameters	

SWS Item	**Dcm777_Conf :**
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	**Dcm735_Conf :**		
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	**Dcm737_Conf :**		
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	**Dcm918_Conf :**		
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	**Dcm733_Conf :**		
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	**Dcm734_Conf :**		
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	**Dcm802_Conf :**
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	**Dcm942_Conf :**		
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: Module		

SWS Item	**Dcm803_Conf :**		
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	**Dcm924_Conf :**		
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	**Dcm812_Conf :**
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	**Dcm804_Conf :**		
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	**Dcm690_Conf :**
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.
DcmDslServiceRequestManufacturerNotification	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 DcmDslServiceRequestManufacturerNotification. The lowerMultiplicity is 0: If DcmRequestManufacturerNotificationEnabled = false the Indication API is not available.
DcmDslServiceRequestSupplierNotification	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 DcmDslServiceRequestSupplierNotification. The lowerMultiplicity is 0: If DcmRequestSupplierNotification = false the Indication API is not available.

10.2.8 DcmDslBuffer

SWS Item	**Dcm739_Conf :**
Container Name	DcmDslBuffer
Description	This container contains the configuration (parameters) for the diagnostic buffer.
Configuration Parameters	

SWS Item	**Dcm738_Conf :**		
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	**Dcm679_Conf :**

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	**Dcm691_Conf :**
Container Name	DcmDslDiagResp
Description	This container contains the configuration (parameters) for the ResponsePending handling
Configuration Parameters	

SWS Item	**Dcm693_Conf :**		
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	**Dcm914_Conf :**		
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	**Dcm694_Conf :**
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	**Dcm695_Conf :**
Container Name	DcmDslProtocolRow
Description	Definition of a single Row of configuration for the protocol configuration (for each protocol)
Configuration Parameters	

SWS Item	**Dcm886_Conf :**		
Name	DcmDslProtocolEndiannessConvEnabled		
Description	Enables /disables the endianness conversion, per diagnostic protocol.		
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: Module		

SWS Item	**Dcm696_Conf :**	
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_IP	DCM_PERIODICTRANS_ON_IP
	DCM_PERIODIC_ON_CAN	DCM_PERIODIC_ON_CAN
	DCM_PERIODIC_ON_FLEXRAY	DCM_PERIODIC_ON_FLEXRAY
	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	**Dcm697_Conf :**		
Name	DcmDslProtocolIsParallelExecutab		
Description	Enables the parallel processing of ROE or Periodic Transmission protocol. Only these both protocols are allowed to run in parallel to normal protocol (UDS, OBD).		
Multiplicity	1		
Type	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: Module		

SWS Item	**Dcm698_Conf :**		
Name	DcmDslProtocolPreemptTimeout		
Description	This is the value for the timeout (in milliseconds) of preempting protocol until protocol needs to be started. This is defined in the AUTOSAR SWS for DCM as a uint16. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM. min: A negative value is not allowed. upperMultiplicity: Exactly one DcmDslProtocolPreemptTimeout value per DSL protocol timing structure. lowerMultiplicity: Exactly one DcmDslProtocolPreemptTimeout value per DSL timing structure. origin: Standard AUTOSAR configuration parameter.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 1000		
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	**Dcm699_Conf :**		
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	**Dcm700_Conf :**		
Name	DcmDslProtocolTransType		
Description	Selects the transmission type for protocol.		
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	**Dcm910_Conf :**		
Name	DcmSendRespPendOnTransToBoot		
Description	Parameter specifying if the ECU should send a NRC 0x78 (response pending) before transitioning to the bootloader (parameter set to TRUE) or if the transition shall be initiated without sending NRC 0x78 (parameter set to FALSE).		
Multiplicity	1		
Type	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: module		

SWS Item	**Dcm729_Conf :**
Name	DcmTimStrP2ServerAdjust

Description	This parameter is used to gurantee that the DCM response is available on the bus before reaching P2.This parameter value in milliseconds 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 .. 1000		
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	**Dcm728_Conf :**		
Name	DcmTimStrP2StarServerAdjust		
Description	This parameter is used to gurantee that the DCM response is available on the bus before reaching P2*.This parameter value in milliseconds 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 .. 5000		
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	**Dcm701_Conf :**		
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	**Dcm702_Conf :**		
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	**Dcm851_Conf :**		
Name	DcmDslProtocolSessionRef		
Description	Reference to the DcmDspSession table used for this protocol ConfigurationClass.		
Multiplicity	0..1		
Type	Reference to [DcmDspSession]		
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: DcmDslProtocolRxPduRef, DcmDslBuffer		

SWS Item	**Dcm704_Conf :**		
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	**Dcm705_Conf :**
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 (DcmDslPeriodicTranmissionConRef) 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	**Dcm706_Conf :**
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 (DcmDslPeriodicTranmissionConRef) relevant for protocols with DcmDslProtocolTransType = TYPE2 or TYPE1.
Configuration Parameters	

SWS Item	**Dcm826_Conf :**		
Name	DcmDslProtocolRxTesterSourceAddr		
Description	Tester source address uniquely describes a client and will be used e.g. within the jump to Bootloader interfaces.		
Multiplicity	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	**Dcm707_Conf :**		
Name	DcmDslPeriodicTranmissionConRef		
Description	Configures the link to the connection of PeriodicTransmission protocol for the processing of the PeriodicTransmission events.		
Multiplicity	0..1		
Type	Reference to [DcmDslPeriodicTransmission]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dcm708_Conf :**
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	**Dcm709_Conf :**
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	**Dcm710_Conf :**		
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	**Dcm778_Conf :**		
Name	DcmDslProtocolRxChannelId		
Description	Channel Identifier associated to the received Pdu.		
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	**Dcm687_Conf :**		
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	**Dcm906_Conf :**		
Name	DcmDslProtocolRxComMChannelRef		
Description	Reference to the ComMChannel on which the DcmDslProtocolRxPdu is received.		
Multiplicity	1		
Type	Reference to [ComMChannel]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**Dcm770_Conf :**		
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	**Dcm711_Conf :**
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	**Dcm864_Conf :**		
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	**Dcm772_Conf :**		
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	**Dcm741_Conf :**
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	**Dcm897_Conf :**
Container Name	DcmDslPeriodicConnection
Description	Holding the TxPduId configuration for PeriodicTransmission.
Configuration Parameters	

SWS Item	**Dcm862_Conf :**		
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	**Dcm742_Conf :**		
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	**Dcm744_Conf :**
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	**Dcm863_Conf :**
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	**Dcm743_Conf :**		
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 DcmDslServiceRequestManufacturerNotification

SWS Item	**Dcm681_Conf :**
Container Name	DcmDslServiceRequestManufacturerNotification
Description	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 DcmDslServiceRequestManufacturerNotification. The lowerMultiplicity is 0: If DcmRequestManufacturerNotificationEnabled = false the Indication API is not available.
Configuration Parameters	

No Included Containers

10.2.21 DcmDslServiceRequestSupplierNotification

SWS Item	**Dcm816_Conf :**
Container Name	DcmDslServiceRequestSupplierNotification
Description	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 DcmDslServiceRequestSupplierNotification. The lowerMultiplicity is 0: If DcmRequestSupplierNotification = false the Indication API is not available.

Configuration Parameters

No Included Containers

10.2.22 DcmDsp

SWS Item	**Dcm712_Conf :**
Container Name	DcmDsp
Description	These parameters apply to Diagnostic Service Processing. There will always be one set of these parameters per DCM.
Configuration Parameters	

SWS Item	**Dcm638_Conf :**		
Name	DcmDspMaxDidToRead		
Description	Indicates the maximum allowed DIDs in a single "ReadDataByIdentifier" request. If set to 0, then no limitation is applied.		
Multiplicity	0..1		
Type	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	**Dcm818_Conf :**		
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.		
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.
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
DcmDspTestResultByObdmid	0..1	This container contains the configuration (parameters) of the "Request on-board monitoring test results" service (Service $06).
DcmDspVehInfo	0..*	This container contains the configuration (parameters) of the "Request vehicle information service" (service $09).

10.2.23 DcmDspComControl

SWS Item	**Dcm900_Conf :**
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.

10.2.24 DcmDspComControlAllChannel

SWS Item	**Dcm901_Conf :**
Container Name	DcmDspComControlAllChannel
Description	Collection of ComM channels which shall be controlled if all networks are addressed.
Configuration Parameters	

SWS Item	**Dcm902_Conf :**		
Name	DcmDspAllComMChannelRef		
Description	Reference to ComM channel.		
Multiplicity	1		
Type	Reference to [ComMChannel]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.25 DcmDspComControlSetting

SWS Item	**Dcm943_Conf :**
Container Name	DcmDspComControlSetting
Description	Provide the configuration of the Communication control.
Configuration Parameters	

SWS Item	**Dcm944_Conf :**		
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

10.2.26 DcmDspComControlSpecificChannel

SWS Item	**Dcm903_Conf :**
Container Name	DcmDspComControlSpecificChannel
Description	Assigns subnet number to ComM channel which will be controlled.
Configuration Parameters	

SWS Item	**Dcm905_Conf :**	
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	**Dcm904_Conf :**		
Name	DcmDspSpecificComMChannelRef		
Description	Reference to ComM channel.		
Multiplicity	1		
Type	Reference to [ComMChannel]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.27 DcmDspDid

SWS Item	**Dcm601_Conf :**
Container Name	DcmDspDid
Description	This container contains the configuration (parameters) of the DID.
Configuration Parameters	

SWS Item	**Dcm602_Conf :**		
Name	DcmDspDidIdentifier		
Description	2 byte Identifier of the DID		
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	**Dcm859_Conf :**		
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 dependency: DcmDspRoeQueueEnabled		

SWS Item	**Dcm805_Conf :**		
Name	DcmDspDidUsed		
Description	Allow to activate or deactivate the usage of a DID, for multi purpose ECUs true = DID available false = DID not available		
Multiplicity	1		
Type	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	**Dcm604_Conf :**		
Name	DcmDspDidInfoRef		
Description	Reference to DcmDspDidInfo containing information on this DID.		
Multiplicity	1		
Type	Reference to [DcmDspDidInfo]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Dcm606_Conf :**		
Name	DcmDspDidRef		
Description	Reference to DcmDspDid in case this DID refer to one or serveral other DID's		
Multiplicity	0..*		
Type	Reference to [DcmDspDid]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDspDidExtRoe	0..1	Provide information to manage a responseOnEvent request on this DID externally. If this container doesn't exist, an ROE event on this DID will be managed internally.
DcmDspDidSignal	0..*	This container defines the reference to 1 DcmDspData container and position relevant for this DID.

10.2.28 DcmDspDidExtRoe

SWS Item	**Dcm780_Conf :**
Container Name	DcmDspDidExtRoe
Description	Provide information to manage a responseOnEvent request on this DID externally. If this container doesn't exist, an ROE event on this DID will be managed internally.
Configuration Parameters	

SWS Item	**Dcm782_Conf :**
Name	DcmDspDidRoeActivateFnc

Description	Function name to activate/deactivate an ROE event managed externally. Only relevant if DcmDspDataUsePort== USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC. This parameter is related to the interface xxx_ActivateEvent.		
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 dependency: DcmDspDataUsePort		

SWS Item	**Dcm781_Conf :**		
Name	DcmDspDidRoeEventId		
Description	EventId to be used within Apis Dcm_TriggerOnEvent() and xxx_ActivateEvent()		
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

10.2.29 DcmDspDidSignal

SWS Item	**Dcm813_Conf :**
Container Name	DcmDspDidSignal
Description	This container defines the reference to 1 DcmDspData container and position relevant for this DID.
Configuration Parameters	

SWS Item	**Dcm814_Conf :**		
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	**Dcm808_Conf :**		
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

10.2.30 DcmDspDidRange

SWS Item	**Dcm937_Conf :**
Container Name	DcmDspDidRange
Description	This container defines the DID Range
Configuration Parameters	

SWS Item	**Dcm941_Conf :**		
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 dependency: DcmDspRoeQueueEnabled		

SWS Item	**Dcm938_Conf :**		
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	**Dcm939_Conf :**
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	**Dcm946_Conf :**		
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	**Dcm940_Conf :**		
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	**Dcm947_Conf :**		
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	**Dcm945_Conf :**
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	**Dcm948_Conf :**		
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	**Dcm950_Conf :**		
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

10.2.31 DcmDspControlDTCSetting

SWS Item	**Dcm935_Conf :**
Container Name	DcmDspControlDTCSetting
Description	Provide the configuration of the ControlDTCSetting mechanism.
Configuration Parameters	

SWS Item	**Dcm936_Conf :**

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_EnableDTCSetting()) 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

10.2.32 DcmDspData

SWS Item	**Dcm869_Conf :**
Container Name	DcmDspData
Description	This container contains the configuration (parameters) of a Data belonging to a DID
Configuration Parameters	

SWS Item	**Dcm677_Conf :**		
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 DcmDspDataUsePort=="USE_DATA_SYNCH_FNC or 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		

SWS Item	**Dcm825_Conf :**
Name	DcmDspDataEcuSignal
Description	Function name to control the access to a certain ECU Signal by the DCM. (IoHwAb_Dcm_<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	**Dcm674_Conf :**		
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	**Dcm676_Conf :**		
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	**Dcm671_Conf :**		
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	**Dcm824_Conf :**		
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	**Dcm669_Conf :**		
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 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		

SWS Item	**Dcm673_Conf :**		
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	**Dcm672_Conf :**		
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	**Dcm675_Conf :**		
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	**Dcm605_Conf :**		
Name	DcmDspDataSize		
Description	Length of data in bits associated to the Data. If Data has variable datalength, that corresponds to the maximum datalength.		
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	**Dcm879_Conf :**		
Name	DcmDspDataType		
Description	Provide the data type of Data belonging to a DID.		
Multiplicity	0..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.	
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	**Dcm713_Conf :**	
Name	DcmDspDataUsePort	
Description	Define wich 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. 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 named DataServices_<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. 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	**Dcm670_Conf :**		
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	**Dcm809_Conf :**		
Name	DcmDspDataBlockIdRef		
Description	NRAM blockId to access the data. Only relevant if DcmDspDataUsePort==USE_BLOCK_ID.		
Multiplicity	0..1		
Type	Reference to [NvMBlockDescriptor]		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	dependency: DcmDspDataUsePort		

SWS Item	**Dcm811_Conf :**		
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		

No Included Containers

10.2.33 DcmDspDataInfo

SWS Item	**Dcm810_Conf :**
Container Name	DcmDspDataInfo
Description	This container contains the configuration (parameters) of a Data
Configuration Parameters	

SWS Item	**Dcm608_Conf :**		
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	**Dcm611_Conf :**		
Name	DcmDspDataScalingInfoSize		
Description	If Scaling information service is available for this Data, it provides the size 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

10.2.34 DcmDspDidInfo

SWS Item	**Dcm607_Conf :**
Container Name	DcmDspDidInfo
Description	This container contains the configuration (parameters) of the DID's Info
Configuration Parameters	

SWS Item	**Dcm612_Conf :**		
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

10.2.35 DcmDspDidAccess

SWS Item	**Dcm609_Conf :**
Container Name	DcmDspDidAccess
Description	This container contains the configuration (parameters) of the DID access
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDspDidControl	0..1	This container contains the configuration (parameters) of the DID control.
DcmDspDidRead	0..1	This container contains the configuration (parameters) of the DID read.
DcmDspDidWrite	0..1	This container contains the configuration (parameters) of the DID write.

10.2.36 DcmDspDidControl

SWS Item	**Dcm619_Conf :**
Container Name	DcmDspDidControl
Description	This container contains the configuration (parameters) of the DID control.
Configuration Parameters	

SWS Item	**Dcm624_Conf :**		
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	**Dcm623_Conf :**		
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	**Dcm622_Conf :**		
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	**Dcm625_Conf :**		
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	**Dcm923_Conf :**		
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	**Dcm620_Conf :**		
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	**Dcm621_Conf :**		
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

10.2.37 DcmDspDidRead

SWS Item	**Dcm613_Conf :**
Container Name	DcmDspDidRead
Description	This container contains the configuration (parameters) of the DID read.
Configuration Parameters	

SWS Item	**Dcm917_Conf :**		
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	**Dcm614_Conf :**
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	**Dcm615_Conf :**		
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

10.2.38 DcmDspDidWrite

SWS Item	**Dcm616_Conf :**
Container Name	DcmDspDidWrite
Description	This container contains the configuration (parameters) of the DID write.
Configuration Parameters	

SWS Item	**Dcm922_Conf :**		
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	**Dcm617_Conf :**		
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	**Dcm618_Conf :**

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

10.2.39 DcmDspMemory

SWS Item	**Dcm784_Conf :**
Container Name	DcmDspMemory
Description	This container contains the configuration of the memory access.
Configuration Parameters	

SWS Item	**Dcm912_Conf :**		
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.		
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: module		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
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

10.2.40 DcmDspMemoryIdInfo

SWS Item	**Dcm911_Conf :**
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	**Dcm913_Conf :**		
Name	DcmDspMemoryIdValue		
Description	Value of the memory device identifer used. This parameter is only relevant if the memory Id parameter use is enabled. (DcmDspUseMemoryId = TRUE)		
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 dependency: DcmDspUseMemoryId		

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.

10.2.41 DcmDspReadMemoryRangeInfo

SWS Item	**Dcm785_Conf :**
Container Name	DcmDspReadMemoryRangeInfo
Description	Provides the range of memory address allowed for reading
Configuration Parameters	

SWS Item	**Dcm787_Conf :**		
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	**Dcm786_Conf :**
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	**Dcm920_Conf :**			
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	**Dcm788_Conf :**			
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

10.2.42 DcmDspWriteMemoryRangeInfo

SWS Item	**Dcm789_Conf :**
Container Name	DcmDspWriteMemoryRangeInfo
Description	Provides the range of memory address allowed for writting.
Configuration Parameters	

SWS Item	**Dcm791_Conf :**	
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	**Dcm790_Conf :**			
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	**Dcm916_Conf :**			
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	**Dcm793_Conf :**			
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

10.2.43 DcmDspPid

SWS Item	**Dcm626_Conf :**		
Container Name	DcmDspPid		
Description	This container defines the availability of a PID to the DCM.		
Configuration Parameters			

SWS Item	**Dcm627_Conf :**		
Name	DcmDspPidIdentifier		
Description	1 byte Identifier of the PID		
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	**Dcm893_Conf :**		
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	**Dcm870_Conf :**		
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	**Dcm806_Conf :**
Name	DcmDspPidUsed

Description	Allow to activate or deactivate the usage of a PID, for multi purpose ECUs true = PID available false = PID not available		
Multiplicity	1		
Type	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).

10.2.44 DcmDspPidSupportInfo

SWS Item	**Dcm871_Conf :**
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	**Dcm873_Conf :**		
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	**Dcm872_Conf :**		
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

10.2.45 DcmDspPidData

SWS Item	**Dcm865_Conf :**
Container Name	DcmDspPidData
Description	This container defines the parameter for a Signal in the PID.
Configuration Parameters	

SWS Item	**Dcm866_Conf :**		
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	**Dcm628_Conf :**		
Name	DcmDspPidDataSize		
Description	Length of data associated to the PID in bit.		
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		

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.

10.2.46 DcmDspPidService01

SWS Item	**Dcm894_Conf :**
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	**Dcm629_Conf :**		
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	**Dcm720_Conf :**		
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

10.2.47 DcmDspPidService02

SWS Item	**Dcm895_Conf :**
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	**Dcm887_Conf :**
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			

No Included Containers

10.2.48 DcmDspPidDataSupportInfo

SWS Item	**Dcm874_Conf :**
Container Name	DcmDspPidDataSupportInfo
Description	This container defines the supported information.
Configuration Parameters	

SWS Item	**Dcm876_Conf :**		
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	**Dcm875_Conf :**		
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

10.2.49 DcmDspRequestControl

SWS Item	**Dcm637_Conf :**
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	**Dcm722_Conf :**		
Name	DcmDspRequestControlInBufferSize		
Description	--		
Multiplicity	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		

SWS Item	**Dcm723_Conf :**		
Name	DcmDspRequestControlOutBufferSize		
Description	--		
Multiplicity	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		

SWS Item	**Dcm656_Conf :**		
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

10.2.50 DcmDspRoe

SWS Item	**Dcm858_Conf :**
Container Name	DcmDspRoe
Description	Provide the configuration of the ResponseOnEvent mechanism.
Configuration Parameters	

SWS Item	**Dcm857_Conf :**

Name	DcmDspRoeBufSize		
Description	Size of the buffer used to managed internally the subservice OnChangeOfDataIdentifier of ResponseOnEvent Service.		
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	**Dcm933_Conf :**		
Name	DcmDspRoeInitFnc		
Description	Function name to initialize an ROE event managed externally.		
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	**Dcm860_Conf :**		
Name	DcmDspRoeInitOnDSC		
Description	If set to TRUE the ROE functionality is re-initialized on reception of service DiagnosticSessionControl.		
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	**Dcm856_Conf :**		
Name	DcmDspRoeInterMessageTime		
Description	Provide the minimum time in ms between two transmissions of ROE event. It is used for the delay between two different consecutive Roe transmissions and between two transmissions of the retry mechanism. 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.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 5000		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-

			POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**Dcm853_Conf :**		
Name	DcmDspRoeMaxEventLength		
Description	Provide the max size in byte of an ROE message that will be stored in the ROE transmission queue.		
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 dependency: DcmDspRoeQueueEnabled		

SWS Item	**Dcm855_Conf :**		
Name	DcmDspRoeMaxNumberOfRetry		
Description	Provide the max number of re-transmission of an ROE event in case of transmission failure. If set to 0, no retry mechanism will be managed		
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	**Dcm854_Conf :**		
Name	DcmDspRoeMaxQueueLength		
Description	Provide the max number of element in the ROE transmission queue.		
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 dependency: DcmDspRoeQueueEnabled		

SWS Item	**Dcm852_Conf :**		
Name	DcmDspRoeQueueEnabled		
Description	If set to TRUE, the ROE queue mechanism is enabled on DCM. If set to FALSE, the ROE event will be sent without queuing.		
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	**Dcm934_Conf :**		
Name	DcmDspRoeStopFnc		
Description	Function name to stop an ROE event managed externally.		
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		

No Included Containers

10.2.51 DcmDspRoutine

SWS Item	**Dcm640_Conf :**
Container Name	DcmDspRoutine
Description	This container contains the configuration (parameters) for Routines
Configuration Parameters	

SWS Item	**Dcm753_Conf :**		
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	**Dcm899_Conf :**
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	**Dcm817_Conf :**		
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	**Dcm641_Conf :**		
Name	DcmDspRoutineIdentifier		
Description	2 bytes Identifier of the RID		
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	**Dcm724_Conf :**		
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	**Dcm807_Conf :**		
Name	DcmDspRoutineUsed		
Description	Allow to activate or deactivate the usage of a Routine, for multi purpose ECUs true = Routine available false = Routine not available		
Multiplicity	1		
Type	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	**Dcm664_Conf :**		
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	**Dcm752_Conf :**		
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	**Dcm898_Conf :**		
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	**Dcm642_Conf :**		
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

10.2.52 DcmDspRoutineInfo

SWS Item	**Dcm643_Conf :**
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.

10.2.53 DcmDspRoutineAuthorization

SWS Item	**Dcm644_Conf :**
Container Name	DcmDspRoutineAuthorization
Description	This container contains the configuration (parameters) for the Routine Authorization.
Configuration Parameters	

SWS Item	**Dcm915_Conf :**		
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	**Dcm648_Conf :**		
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	**Dcm649_Conf :**		
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

10.2.54 DcmDspRoutineRequestResOut

SWS Item	**Dcm831_Conf :**
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.

10.2.55 DcmDspRoutineRequestResOutSignal

SWS Item	**Dcm836_Conf :**
Container Name	DcmDspRoutineRequestResOutSignal
Description	Provide description of a routine signal used in RoutineControl service.
Configuration Parameters	

SWS Item	**Dcm838_Conf :**

Name	DcmDspRoutineSignalLength		
Description	Provide the length in bits of the signal in the RoutineControl request/response		
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	**Dcm837_Conf :**		
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	**Dcm881_Conf :**		
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

10.2.56 DcmDspRoutineStopIn

SWS Item	**Dcm832_Conf :**
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.

10.2.57 DcmDspRoutineStopInSignal

SWS Item	**Dcm839_Conf :**
Container Name	DcmDspRoutineStopInSignal
Description	Provide description of a routine signal used in RoutineControl service.
Configuration Parameters	

SWS Item	**Dcm841_Conf :**		
Name	DcmDspRoutineSignalLength		
Description	Provide the length in bits of the signal in the RoutineControl request/response		
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	**Dcm840_Conf :**		
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	**Dcm882_Conf :**
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

10.2.58 DcmDspRoutineStopOut

SWS Item	**Dcm833_Conf :**
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.

10.2.59 DcmDspRoutineStopOutSignal

SWS Item	**Dcm842_Conf :**
Container Name	DcmDspRoutineStopOutSignal
Description	Provide description of a routine signal used in RoutineControl service.
Configuration Parameters	

SWS Item	**Dcm844_Conf :**		
Name	DcmDspRoutineSignalLength		
Description	Provide the length in bits of the signal in the RoutineControl request/response		
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	**Dcm843_Conf :**		
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	**Dcm883_Conf :**		
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

10.2.60 DcmDspStartRoutineIn

SWS Item	**Dcm834_Conf :**
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.

10.2.61 DcmDspStartRoutineInSignal

SWS Item	**Dcm845_Conf :**
Container Name	DcmDspStartRoutineInSignal
Description	Provide description of a routine signal used in RoutineControl service.
Configuration Parameters	

SWS Item	**Dcm847_Conf :**		
Name	DcmDspRoutineSignalLength		
Description	Provide the length in bits of the signal in the RoutineControl request/response		
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	**Dcm846_Conf :**		
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	**Dcm884_Conf :**		
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

10.2.62 DcmDspStartRoutineOut

SWS Item	**Dcm835_Conf :**
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.

10.2.63 DcmDspStartRoutineOutSignal

SWS Item	**Dcm848_Conf :**
Container Name	DcmDspStartRoutineOutSignal
Description	Provide description of a routine signal used in RoutineControl service.
Configuration Parameters	

SWS Item	**Dcm850_Conf :**		
Name	DcmDspRoutineSignalLength		
Description	Provide the length in bits of the signal in the RoutineControl request/response.		
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	**Dcm867_Conf :**		
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	**Dcm885_Conf :**		
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

10.2.64 DcmDspSecurity

SWS Item	**Dcm764_Conf :**
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_<LEVEL>. The R-Port is named SecurityAccess_<LEVEL> where _<LEVEL> is the name of the container DcmDspSecurityRow. If there is no reference, no check of security level shall be done.

10.2.65 DcmDspSecurityRow

SWS Item	**Dcm759_Conf :**

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_<LEVEL>. The R-Port is named SecurityAccess_<LEVEL> where _<LEVEL> is the name of the container DcmDspSecurityRow. If there is no reference, no check of security level shall be done.
Configuration Parameters	

SWS Item	**Dcm725_Conf :**		
Name	DcmDspSecurityADRSize		
Description	Size 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: Module		

SWS Item	**Dcm757_Conf :**		
Name	DcmDspSecurityDelayTime		
Description	Delay time after failed security access (in ms). This is started after DcmDspSecurityNumAttDelay number of failed security accesses. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM. min: A negative value is not allowed. upperMultiplicity: Exactly one delay time must be specified per security levbel configuration. lowerMultiplicity: Exactly one delay time must be specified per security levbel configuration. origin: Standard AUTOSAR configuration parameter.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 20000		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**Dcm726_Conf :**
Name	DcmDspSecurityDelayTimeOnBoot
Description	Start delay timer on power on (in ms) The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM. min: A negative value is not allowed. upperMultiplicity: Exactly one delay time must be specified per security level configuration. lowerMultiplicity: Exactly one delay time must be specified per security level configuration.
Multiplicity	1

Type	EcucFloatParamDef		
Range	0 .. 20000		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**Dcm760_Conf :**		
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: Module		

SWS Item	**Dcm754_Conf :**		
Name	DcmDspSecurityLevel		
Description	Value of Security level. Level 1 is the highest 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: Module		

SWS Item	**Dcm762_Conf :**		
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: Module dependency: DcmDspSecurityDelayTime		

SWS Item	**Dcm755_Conf :**
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: Module		

No Included Containers

10.2.66 DcmDspSession

SWS Item	**Dcm769_Conf :**
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)

10.2.67 DcmDspSessionRow

SWS Item	**Dcm767_Conf :**
Container Name	DcmDspSessionRow
Description	Definition of a single Row of session control configuration (per session control)
Configuration Parameters	

SWS Item	**Dcm815_Conf :**		
Name	DcmDspSessionForBoot		
Description	This parameter defines whether this diagnostic session allows to jump to Bootloader (OEM Bootloader or System Supplier Bootloader). If this diagnostic session doesn't allow to jump to Bootloader the value DCM_NO_BOOT shall be chosen.		
Multiplicity	1		
Type	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: Module		

SWS Item	**Dcm765_Conf :**		
Name	DcmDspSessionLevel		
Description	Value of the Session control. The symbolicName represents the Name of the Session SWS defines the Dcm_SesCtrlType: 1 DEFAULT_SESSION 2 PROGRAMMING_SESSION 3 EXTENDED_DIAGNOSTIC_SESSION 4 SAFETY_SYSTEM_DIAGNOSTIC_SESSION 5...63 Reserved by Document 64...126 <configuration dependent (according "diagnosticSessionType" parameter DiagnosticSessionControl request> 127...254 Reserved by Document 255 ALL_SESSION_LEVEL		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**Dcm766_Conf :**		
Name	DcmDspSessionP2ServerMax		
Description	This is the session value for P2ServerMax in milliseconds (per Session Control). This is defined in the AUTOSAR SWS for DCM as a uint16. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 1000		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**Dcm768_Conf :**		
Name	DcmDspSessionP2StarServerMax		
Description	This is the session value for P2*ServerMax in milliseconds (per Session Control). This is defined in the AUTOSAR SWS for DCM as a uint16. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 100000		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME

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

No Included Containers

10.2.68 DcmDspTestResultByObdmid

SWS Item	**Dcm771_Conf :**
Container Name	DcmDspTestResultByObdmid
Description	This container contains the configuration (parameters) of the "Request on-board monitoring test results" service (Service $06).
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDspTestResultObdmidTid	1..*	This container contains the configuration (parameters) of a OBDMID and the assigned TIDs for the "Request on-board monitoring test results" service (Service $06).
DcmDspTestResultTid	1..*	This container contains the configuration (parameters) of a single TID for the "Request on-board monitoring test results" service (Service $06). The DCM will access the Data using an R-Port requiring a PortInterface DTRServices. The R-Port is named DtrServices_<OBDMID>_<TID> where <TID> is the name of the container DcmDspTestResultTid and <OBDMID> is the name of the container DcmDspTestResultObdmidTid.

10.2.69 DcmDspTestResultObdmidTid

SWS Item	**Dcm773_Conf :**
Container Name	DcmDspTestResultObdmidTid
Description	This container contains the configuration (parameters) of a OBDMID and the assigned TIDs for the "Request on-board monitoring test results" service (Service $06).
Configuration Parameters	

SWS Item	**Dcm684_Conf :**		
Name	DcmDspTestResultObdmid		
Description	OBDMID for Service $06		
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
DcmDspTestResultObdmidTids	1..*	This container contains the configuration (parameters) of the TIDs belonging to the OBDMID for the "Request on-board

		monitoring test results" service (Service $06).

10.2.70 DcmDspTestResultObdmidTids

SWS Item	**Dcm892_Conf :**
Container Name	DcmDspTestResultObdmidTids
Description	This container contains the configuration (parameters) of the TIDs belonging to the OBDMID for the "Request on-board monitoring test results" service (Service $06).
Configuration Parameters	

SWS Item	**Dcm686_Conf :**		
Name	DcmDspTestResultObdmidTidUaSid		
Description	Unit And Scaling ID for Service $06		
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	**Dcm685_Conf :**		
Name	DcmDspTestResultObdmidTidRef		
Description	Link to reported TIDs of this OBDMID upperMultiplicity / lowerMultiplicity:: At least one TID must be reported for an OBDMID		
Multiplicity	1..*		
Type	Reference to [DcmDspTestResultTid]		
ConfigurationClass	Pre-compile 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		

No Included Containers

10.2.71 DcmDspTestResultTid

SWS Item	**Dcm634_Conf :**
Container Name	DcmDspTestResultTid
Description	This container contains the configuration (parameters) of a single TID for the "Request on-board monitoring test results" service (Service $06). The DCM will access the Data using an R-Port requiring a PortInterface DTRServices. The R-Port is named DtrServices_<OBDMID>_<TID> where <TID> is the name of the container DcmDspTestResultTid and <OBDMID> is the name of the container DcmDspTestResultObdmidTid.
Configuration Parameters	

SWS Item	**Dcm635_Conf :**

Name	DcmDspTestResultTestId		
Description	Test Id for Service $06		
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

10.2.72 DcmDspVehInfo

SWS Item	**Dcm630_Conf :**
Container Name	DcmDspVehInfo
Description	This container contains the configuration (parameters) of the "Request vehicle information service" (service $09).
Configuration Parameters	

SWS Item	**Dcm631_Conf :**		
Name	DcmDspVehInfoInfoType		
Description	value of InfoType.		
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.

10.2.73 DcmDspVehInfoData

SWS Item	**Dcm888_Conf :**
Container Name	DcmDspVehInfoData
Description	Data Item of an InfoType; post-fix of the port interface name.
Configuration Parameters	

SWS Item	**Dcm891_Conf :**
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	**Dcm889_Conf :**		
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	**Dcm890_Conf :**		
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	**Dcm727_Conf :**		
Name	DcmDspVehInfoDataUsePort		
Description	When the parameter DcmDspVehInfoUsePort is set to true the DCM will access the Data using an R-Port requiring a PortInteface InfotypeServices_<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 the parameter DcmDspVehInfoDataUsePort is 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

10.2.74 DcmGeneral

SWS Item	**Dcm822_Conf :**		
Container Name	DcmGeneral		
Description	This container contains the configuration (parameters) for Component wide parameters		
Configuration Parameters			

SWS Item	**Dcm823_Conf :**		
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: module		

SWS Item	**Dcm783_Conf :**		
Name	DcmRequestManufacturerNotificationEnabled		
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	**Dcm868_Conf :**		
Name	DcmRequestSupplierNotificationEnabled		
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		

SWS Item	**Dcm600_Conf :**		
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	**Dcm820_Conf :**		
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 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	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: module		

SWS Item	**DCM821_Conf :**		
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: module		

No Included Containers

10.2.75 DcmPageBufferCfg

SWS Item	**Dcm775_Conf :**
Container Name	DcmPageBufferCfg
Description	This container contains the configuration (parameters) for Page Buffer handling
Configuration Parameters	

SWS Item	**Dcm776_Conf :**		
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	**Dcm774_Conf :**		
Name	DcmPagedBufferTimeout		
Description	Allow to configure the Timeout (in ms) towards the application for filling the next page. This parameter is only relevant if the Page Buffer handling is enabled. (DcmPagedBufferEnabled = TRUE) Defined in the DCM SWS as uint16, this parameter contains a value in milliseconds. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. DCM configuration tools must convert this float value to the appropriate value format for the use in the software implementation of DCM. min: A negative value is not allowed. upperMultiplicity: Exactly one Timeout must be specified per configuration. lowerMultiplicity: Exactly one Timeout must be specified per configuration.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	0 .. 1000		
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

10.2.76 DcmProcessingConditions

SWS Item	**Dcm932_Conf :**
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

10.2.77 DcmModeCondition

SWS Item	**Dcm928_Conf :**
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	**Dcm929_Conf :**		
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: Instance		

SWS Item	**Dcm931_Conf :**		
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: Instance		

SWS Item	**Dcm930_Conf :**		
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: Instance		

No Included Containers

10.2.78 DcmModeRule

SWS Item	**Dcm925_Conf :**
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	**Dcm926_Conf :**		
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: Instance		

SWS Item	**Dcm949_Conf :**		
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	0 .. 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	**Dcm927_Conf :**		
Name	DcmArgumentRef		
Description	This is a choice reference either to a mode condition or a an other mode rule serving as sub-expression.		
Multiplicity	1..*		
Type	Choice reference to [DcmModeCondition , DcmModeRule]		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.3 Protocol Configuration Example

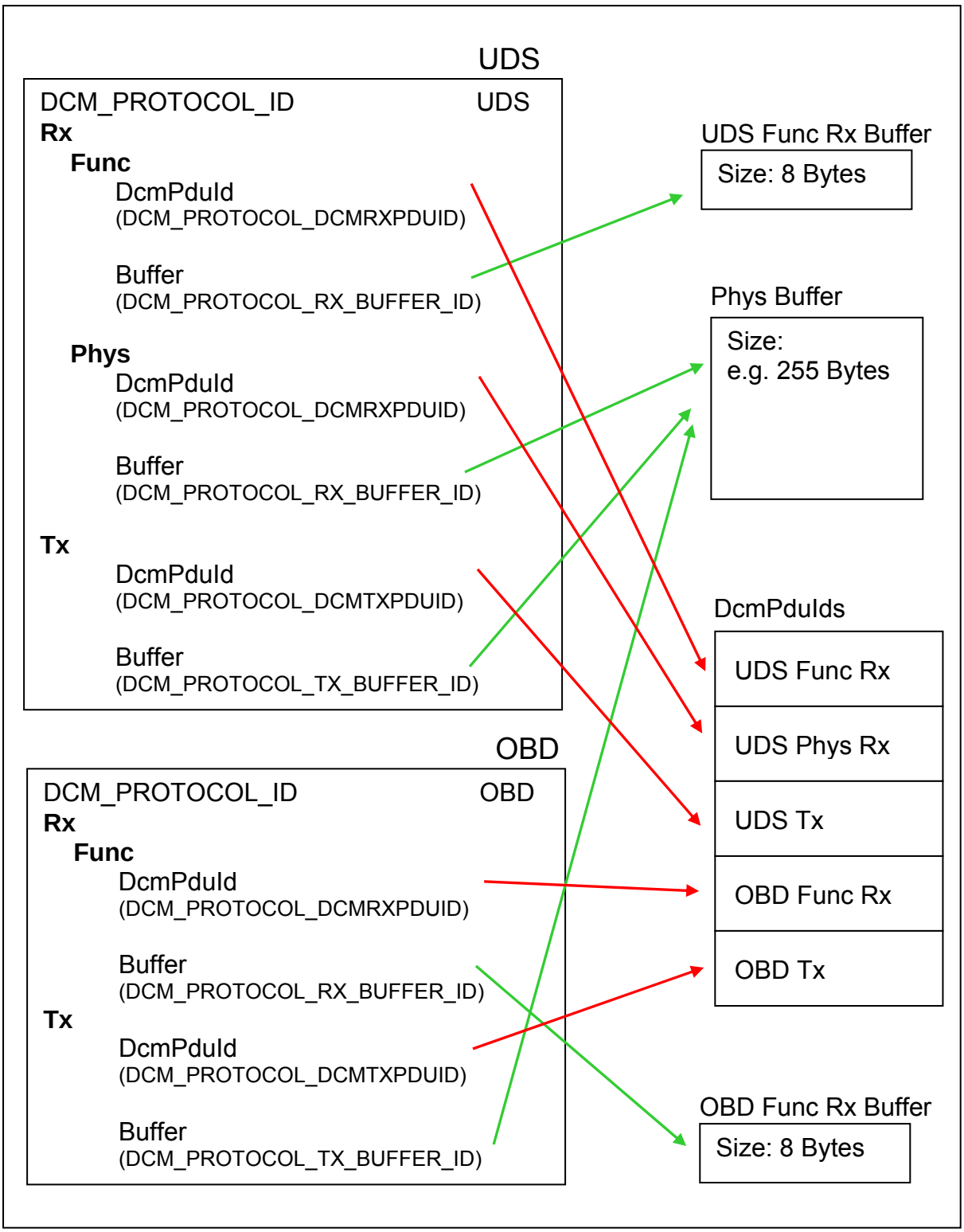


Figure 10 Examples of protocol configuration with focus on buffer / DcmPduId settings

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

[Dcm914] 「 The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [1] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [9].」()

Additional module-specific published parameters are listed below if applicable.

11 Changes to Release 3.1

11.1 Deleted SWS Items

SWS Item	Rationale
Dcm427	DEM Api change
Dcm432	DEM Api change
Dcm428	DEM Api change
Dcm138	Inconsistency with Api definition
Dcm471	Not needed Api
Dcm374	Rework of Reset interfaces
Dcm375	Rework of Reset interfaces
Dcm477	Rework of Reset interfaces
Dcm142	Session change indication is now done by BswM module
Dcm472	Data has fixed length for writting
Dcm274	Session check is now done in DSD
Dcm391	Limitation of service 0x19 with subfunction 0x05 to OBD freeze frame
Dcm429	Limitation of service 0x19 with subfunction 0x05 to OBD freeze frame
Dcm430	Limitation of service 0x19 with subfunction 0x05 to OBD freeze frame
Dcm431	Limitation of service 0x19 with subfunction 0x05 to OBD freeze frame
Dcm479	Redundancy with Dcm478
Dcm480	Redundancy with Dcm478
Dcm347	changes in buffer management between PduR and DCM
Dcm348	changes in buffer management between PduR and DCM
Dcm278	Availability PID for service $02 is now managed DCM internally
Dcm497	Delete container DcmDspDownloadMemoryRangeInfo
Dcm498	Delete container DcmDspUploadMemoryRangeInfo
Dcm661	Requirement replaced by API table
Dcm662	Requirement replaced by API table
Dcm663	Requirement replaced by API table
Dcm664	Requirement replaced by API table
Dcm665	Requirement replaced by API table
Dcm666	Requirement replaced by API table
Dcm667	Requirement replaced by API table
Dcm546	The callout Dcm_<DiagnosticService> no longer exists

11.2 Changed SWS Items

SWS Item	Rationale
Dcm384	DEM Api change
Dcm385	DEM Api change
Dcm439	Inconsistency
Dcm438	Inconsistency with ISO 14229-1
Dcm137	Inconsistency with ISO 15765-3
Dcm004	Wrong DTC number
Dcm024	New configuration parameter
Dcm473	Data has fixed length for writting

Dcm388	Limitation of service 0x19 with subfunction 0x05 to OBD freeze frame
Dcm389	Limitation of service 0x19 with subfunction 0x05 to OBD freeze frame
Dcm390	Limitation of service 0x19 with subfunction 0x05 to OBD freeze frame
Dcm092	Update of PduR interface
Dcm556	Update of PduR interface
Dcm557	Clarification
Dcm445	Clarification
Dcm284	Availability PID for service $02 is now managed DCM internally
Dcm054	File name reference corrected
Dcm403	Configuration parameter correction
Dcm405	Configuration parameter correction
Dcm055	Correction of header file names
Dcm040	Update description of errors
Dcm530	Change condition on Dcm__MainFunction call
Dcm642	“BUFREQ_E_OK” has been replaced by “BUFREQ_OK”
Dcm094	“BUFREQ_E_OK” has been replaced by “BUFREQ_OK”
Dcm092	Delete TP_NOENTRY reference from RetryInfoPtr defnition
Dcm687	Correct SenderReceiver interface DataServices_<Data> definition
Dcm054	Correction of header file names
Dcm055	Correction of header file names
Dcm461	Add clarification on transmit cancellation behavior
Dcm577	Add clarification on receive cancellation behavior
Dcm371	Update for reading extended data
Dcm372	Update for reading freeze frame data
Dcm135	Modify this requirement according to the storageState parameter
Dcm638	Update this requirement to clarify endianess conversion management
Dcm639	Update this requirement to clarify endianess conversion management
Dcm640	Update this requirement to clarify endianess conversion management
Dcm641	Update this requirement to clarify endianess conversion management
Dcm654	Update this requirement with DcmSendRespPendOnTransToBoot parameter management
Dcm535	Update this requirement with DcmSendRespPendOnTransToBoot parameter management
Dcm690	Update the interfaces
Dcm400	Specifiy that currentDataLength is provided in bits
Dcm401	Specifiy that currentDataLength is provided in bits
Dcm092	Clarification added to the description of Dcm_CopyTxData
Dcm686	Update the interfaces
Dcm685	Update description of API GetSeed and signature definition according to DcmDspSecurityADRSize parameter value
Dcm333	Add the type TPParameterType
Dcm269	Replace DsdInternal_ProcessingDone by Dcm_ExternalProcessingDone

11.3 Added SWS Items

SWS Item	Rationale
Dcm481	New requirement
Dcm482	New requirement
Dcm483	New requirement
Dcm484	Debugging concept
Dcm485	Debugging concept

Dcm486	Debugging concept
Dcm487	Debugging concept
Dcm488	Add support of service WriteMemoryByAdress
Dcm489	Add support of service WriteMemoryByAdress
Dcm490	Add support of service WriteMemoryByAdress
Dcm491	Add support of service WriteMemoryByAdress
Dcm492	Add support of service ReadMemoryByAdress
Dcm493	Add support of service ReadMemoryByAdress
Dcm494	Add support of service ReadMemoryByAdress
Dcm495	Add support of service ReadMemoryByAdress
Dcm496	Add support of service RequestDownload
Dcm497	Add support of service RequestDownload
Dcm498	Add support of service RequestDownload
Dcm499	Add support of service RequestUpload
Dcm500	Add support of service RequestUpload
Dcm501	Add support of service RequestUpload
Dcm502	Add support of service TransferData
Dcm503	Add support of service TransferData
Dcm504	Add support of service TransferData
Dcm505	Add support of service RequestTransferExit
Dcm506	Debugging concept
Dcm507	Debugging concept
Dcm508	Debugging concept
Dcm509	Debugging concept
Dcm511	Add support of service CommunicationControl
Dcm512	Add support of service CommunicationControl
Dcm513	Add support of service CommunicationControl
Dcm514	Add support of service CommunicationControl
Dcm516	Split application check in Manufacturer application check and Supplier application check
Dcm517	Split application check in Manufacturer application check and Supplier application check
Dcm518	Split application check in Manufacturer application check and Supplier application check
Dcm519	
Dcm520	New interface
Dcm521	New interface
Dcm522	Roe requirement
Dcm523	Roe requirement
Dcm524	Roe requirement
Dcm525	Roe requirement
Dcm526	Roe requirement
Dcm527	Dcm_OpStatusType requirement
Dcm528	Dcm_OpStatusType requirement
Dcm529	Dcm_OpStatusType requirement
Dcm530	Dcm_OpStatusType requirement
Dcm531	Bootloader Interaction
Dcm532	Bootloader Interaction
Dcm533	Bootloader Interaction
Dcm535	Bootloader Interaction
Dcm536	Bootloader Interaction
Dcm537	Bootloader Interaction
Dcm538	Bootloader Interaction

Dcm539	Dcm_ReadMemory callout
Dcm540	Dcm_WriteMemory callout
Dcm548	SRS General requirement
Dcm549	SRS General requirement
Dcm550	SRS General requirement
Dcm551	SRS General requirement
Dcm552	SRS General requirement
Dcm553	SRS General requirement
Dcm554	SRS General requirement
Dcm555	SRS General requirement
Dcm556	Dcm_CopyRxData Api
Dcm557	Management of BUFREQ_E_NOT_OK status
Dcm558	Roe requirement
Dcm559	
Dcm560	Usage of NvM_ReadBlock
Dcm561	Usage of DcmDspDidUsed parameter
Dcm562	Usage of DcmDspDidUsed parameter
Dcm563	IOCBID requirement
Dcm564	Usage of DcmDspDidUsed parameter
Dcm565	IOCBID requirement
Dcm566	IOCBID requirement
Dcm567	IOCBID requirement
Dcm568	RoutineControl requirement
Dcm569	RoutineControl requirement
Dcm570	RoutineControl requirement
Dcm571	RoutineControl requirement
Dcm574	Duplicate requirement Id
Dcm575	Reception cancellation requirement
Dcm576	Reception cancellation requirement
Dcm577	Reception cancellation requirement
Dcm578	IoHwAb access requirements
Dcm579	IoHwAb access requirements
Dcm580	IoHwAb access requirements
Dcm581	IoHwAb access requirements
Dcm582	ROE requirement
Dcm583	Reset Interface
Dcm584	Reset Interface
Dcm585	Session change Interface
Dcm586	Communication control requirement
Dcm587	ISO 15031-6 evolution
Dcm588	ISO 15031-6 evolution
Dcm589	EcuReset requirement
Dcm590	Signal based concept for RoutineControl
Dcm591	Communication control requirement
Dcm592	System supplier bootloader
Dcm593	Main Function Processing for Un-Initialized Module
Dcm594	EcuReset interface with BswM
Dcm595	ROE support
Dcm597	ROE support
Dcm598	ROE support
Dcm599	ROE support
Dcm600	ROE support
Dcm601	ROE support

Dcm602	ROE support
Dcm603	ROE support
Dcm604	ROE support
Dcm605	ROE support
Dcm606	ROE support
Dcm607	ROE support
Dcm608	ROE support
Dcm609	ROE support
Dcm610	ROE support
Dcm611	ROE support
Dcm612	ROE support
Dcm613	ROE support
Dcm614	New DEM Interface for ROE
Dcm615	New Interface for BswM
Dcm616	Check of subservice allowance according to diagnostic session
Dcm617	Security level check for subservices
Dcm618	Initialization of ROE event on S3Server timeout
Dcm619	Initialization of ROE event on S3Server timeout
Dcm620	Limitation for dynamic datallength of a DID
Dcm621	PID Support information
Dcm622	PID Support information
Dcm623	PID Support information
Dcm624	Clarification of protocol stop
Dcm625	Clarification of protocol stop
Dcm626	Stop of IOControl
Dcm627	Stop of IOControl
Dcm628	Stop of IOControl
Dcm629	Naming of Dcm_<ModeName>ModeEntry
Dcm630	Naming of Dcm_<ModeName>ModeEntry
Dcm631	Naming of Dcm_<ModeName>ModeEntry
Dcm632	Limitation of service 0x19 with subfunction 0x05 to OBD freeze frame
Dcm001_PI	Rework of Published Information
Dcm643	Negative response for Dcm_WriteMemory
Dcm644	Negative response for Dcm_ReadMemory
Dcm645	Negative response for block sequence error
Dcm646	DynamicallyDefineDataIdentifier requirement
Dcm647	DynamicallyDefineDataIdentifier requirement
Dcm648	DynamicallyDefineDataIdentifier requirement
Dcm649	DynamicallyDefineDataIdentifier requirement
Dcm650	DynamicallyDefineDataIdentifier requirement
Dcm651	DynamicallyDefineDataIdentifier requirement
Dcm652	DynamicallyDefineDataIdentifier requirement
Dcm653	DynamicallyDefineDataIdentifier requirement
Dcm654	Sending of NRC 0x78, before switching to bootloader
Dcm655	Clarification on return value of Dcm_StartOfReception
Dcm656	Clarification on return value of Dcm_StartOfReception
Dcm657	Add condition check by BswM before jumping to bootloader
Dcm658	Add condition check by BswM before jumping to bootloader
Dcm659	Add DCM behaviour in case of E_NOT_OK return value
Dcm660	Add DCM behaviour in case of E_NOT_OK return value
Dcm661	Add DCM behaviour in case of E_NOT_OK return value
Dcm662	Add DCM behaviour in case of E_NOT_OK return value

Dcm663	Add DCM behaviour in case of E_NOT_OK return value
Dcm664	Add DCM behaviour in case of E_NOT_OK return value
Dcm665	Add DCM behaviour in case of E_NOT_OK return value
Dcm666	Add DCM behaviour in case of E_NOT_OK return value
Dcm667	Add DCM behaviour in case of E_NOT_OK return value
Dcm668	Add DCM behaviour in case of E_NOT_OK return value
Dcm669	Add DCM behaviour in case of E_NOT_OK return value
Dcm670	Add DCM behaviour in case of E_NOT_OK return value
Dcm671	Add DCM behaviour in case of E_NOT_OK return value
Dcm672	Add DCM behaviour in case of E_NOT_OK return value
Dcm673	Add DCM behaviour in case of E_NOT_OK return value
Dcm674	Add DCM behaviour in case of E_NOT_OK return value
Dcm675	Add DCM behaviour in case of E_NOT_OK return value
Dcm676	Add DCM behaviour in case of E_NOT_OK return value
Dcm677	Add DCM behaviour in case of E_NOT_OK return value
Dcm678	Add DCM behaviour in case of E_NOT_OK return value
Dcm679	Add DCM behaviour in case of E_NOT_OK return value
Dcm680	IoControl management
Dcm682	IoControl management
Dcm683	Add of Dcm_Types.h file definition
Dcm684	Computation of NOfDataItems for Mode $9
Dcm685	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm686	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm687	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm688	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm689	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm690	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm691	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm692	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm693	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm694	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm695	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm696	Add DSD request length check
Dcm697	Add requirement for negative response suppression
Dcm698	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm699	Add requirement number for Standardized AUTOSAR Interface definitions
Dcm700	Add generic requirement to avoid misuse of Dem_SetDTCFilter() special DTCStatusMask value 0x00.
Dcm701	Add this requirement to define the expected range for a TID in a Routine

	control request with the OBD Service 0x08
Dcm702	Add this requirement to manage Dem_DisableDTCRecordUpdate return value
Dcm703	Clarify diagnostic answer according to Dem_ClearDTC() return
Dcm704	Clarify diagnostic answer according to Dem_ClearDTC() return
Dcm705	Clarify diagnostic answer according to Dem_ClearDTC() return
Dcm706	Clarify diagnostic answer according to Dem_ClearDTC() return
Dcm707	Clarify diagnostic answer according to Dem_ClearDTC() return
Dcm708	Clarify diagnostic answer according to Dem_ClearDTC() return
Dcm709	Specify the storage state parameter for Response on Event
Dcm710	Add this requirement to specify FIFO queue management for ROE events
Dcm711	Add this requirement to specify the *EventWindowTime* parameter management
Dcm712	Add this requirement to specify the *EventWindowTime* parameter management
Dcm713	Add this requirement to specify the *EventWindowTime* parameter management
Dcm714	Add this requirement to specify the *EventWindowTime* parameter management
Dcm715	Add this requirement to specify the behaviour in case of Dcm_SetProgConditions() returns E_NOT_OK
Dcm716	Add this requirement for endianess conversion management
Dcm717	Add this requirement for endianess conversion management
Dcm718	Add this requirement for endianess conversion management
Dcm719	Add this requirement for DcmSendRespPendOnTransToBoot parameter management
Dcm720	Add this requirement for DcmSendRespPendOnTransToBoot parameter management
Dcm721	Add session and security check for service $2A
Dcm722	Add session and security check for service $2A
Dcm723	Add session and security check for service $2C
Dcm724	Add session and security check for service $2C
Dcm725	Add session and security check for service $2C
Dcm726	Add session and security check for service $2C
Dcm727	Add for handling of parallel requests with the same prioritization
Dcm728	Add for handling of parallel requests with the same prioritization
Dcm729	Add for handling of parallel requests with the same prioritization
Dcm730	Add Dcm_StopROE description
Dcm731	Add Dcm_RestartROE description
Dcm732	Add during rework of chapter 8.9
Dcm733	Add during rework of chapter 8.9
Dcm734	Add during rework of chapter 8.9
Dcm735	Add during rework of chapter 8.9
Dcm738	Add during rework of chapter 8.9
Dcm739	Add this requirement to clarification the expected behavior if Dem_GetStatusOfDTC returns DEM_STATUS_PENDING
Dcm740	Add this requirement to clarification the expected behavior if Dem_GetNextFilteredRecord returns DEM_FILTERED_PENDING
Dcm741	Add functional description of confirmation to SW-C
Dcm742	Add functional description of confirmation to SW-C
Dcm743	Add requirement on LinkControl service
Dcm744	Add requirement on LinkControl service

Dcm747	Add this requirement to specify the *EventWindowTime* parameter management
Dcm748	Add this requirement to specify the *EventWindowTime* parameter management
Dcm750	Add this requirement for header file inclusion
Dcm760	Add this requirement for API tables in chapter 8.9
Dcm761	Add this requirement for API tables in chapter 8.9
Dcm762	Add this requirement for API tables in chapter 8.9
Dcm763	Add this requirement for API tables in chapter 8.9
Dcm764	Add this requirement for API tables in chapter 8.9
Dcm765	Create the new port interface "GeneralROE"
Dcm766	Add requirement related to Dem_GetDTCByOccurrenceTime call
Dcm767	Added for bootloader interaction
Dcm768	Added for bootloader interaction
Dcm769	Add ClientServerInterface DataServices_DIDRange
Dcm770	Add description to chapter 7.3.4.7 Initiate transmission
Dcm772	Create the new interface GeneralRequiredROEServices
Dcm772	Create the new interface GeneralRequiredROEServices
Dcm773	Added for mode management
Dcm774	Added for mode management
Dcm775	Added for mode management
Dcm776	Added for mode management
Dcm777	Added for mode management
Dcm778	Added for mode management
Dcm779	Added for mode management
Dcm780	Added for mode management
Dcm781	Added for mode management
Dcm782	Added for mode management
Dcm783	Added for mode management
Dcm784	Added for mode management
Dcm785	Added for mode management
Dcm786	Added for mode management
Dcm788	Added for parallel request management
Dcm789	Added for parallel request management
Dcm790	Added for parallel request management
Dcm793	Add requirement ID for API tables
Dcm794	Add requirement ID for API tables
Dcm795	Add requirement ID for API tables
Dcm796	Add requirement ID for API tables
Dcm797	Add requirement ID for API tables
Dcm798	Add requirement ID for API tables
Dcm799	Add requirement ID for API tables
Dcm800	Add requirement ID for API tables
Dcm801	Add requirement ID for API tables
Dcm802	Add requirement ID for API tables
Dcm803	Added for mode management
Dcm804	Added for mode management
Dcm805	Added for mode management
Dcm806	Added for mode management
Dcm807	Added for mode management
Dcm808	Added for mode management
Dcm809	Added for mode management

Dcm810	Added for mode management
Dcm811	Added for mode management
Dcm812	Added for mode management
Dcm813	Added for mode management
Dcm814	Added for mode management
Dcm815	Added for mode management
Dcm818	Added for mode management
Dcm819	Added for mode management
Dcm820	Added for mode management
Dcm821	Added for mode management
Dcm822	Added for mode management
Dcm823	Added for mode management
Dcm824	Added for mode management
Dcm825	Added for mode management
Dcm826	Added for mode management

12 Not applicable requirements

[Dcm999] ⌈ 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)