Document Title	Specification of Diagnostic Communication Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	18

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.1

Document Change History

Date	Release	Changed by	Description
2017-12-08	4.3.1	AUTOSAR Release Management	• Cleanup SRS_Diagnostic requirement traceability • Fix Dcm/Dem interactions inconsistencies • Add constraints requirements for parameter configuration • minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	• Redesign interfaces between Dem and Dcm • Rework Security Access management • Add management for parallel support for OBD and UDS protocols • Clarify usage of Diagnosis scaling • minor corrections / clarifications / editorial changes; For details please refer to the BWCStatement

2015-07-31	4.2.2	AUTOSAR Release Management	• Specify the NRCs to be sent by the Dcm in case of Dem interfaces return negative values. • Clarify Routine operation prototypes • Debugging support marked as obsolete • Minor corrections / clarifications / editorial changes; For details please refer to the Change Documentation
2014-10-31	4.2.1	AUTOSAR Release Management	• Update to ISO 14229-1:2013 (Order of NRCs, SID 0x19 and 0x28 extended subfunctions, SID 0x38) • Specify security mechanisms (security Lock time, static seed). • Refine service ReadDataByPeriodicIdentifier (0x2A) and provide UUDT transfer. • Reorganize the configuration parameters for the routines.
2014-03-31	4.1.3	AUTOSAR Release Management	• Added functional description for DIDRange usage • Added support for bootloader interaction • Revised the header file structure • Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Created API tables for service interfaces • Provided synchronous and asynchronous APIs for DataServices callouts • Harmonization for the length parameter interpretation all over RDBI, WDBI and RC services to be in bytes • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Added Response on Event support • Rework configuration for S/R communication • Rework OBD Service $06 management

2011-12-22	4.0.3	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
2009-12-18	4.0.1	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
2010-02-02	3.1.4	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

2008-08-13	3.1.1	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
2007-12-21	3.0.1	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
2007-01-24	2.1.15	AUTOSAR Administration	• "Advice for users" revised • "Revision Information" added
	2.1.14	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
2006-11-28	2.1	AUTOSAR Administration	• Layout Adaptations
2006-05-16	2.0	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
2005-05-31	1.0	AUTOSAR Administration	• Initial release

Disclaimer

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

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

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

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

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 ISO 15031-5 [1] shall be processed prior to any enhanced diagnostic requests, the Dcm module provides a protocol switching mechanism based on protocol prioritization.
- UDS Service AccessTimingParameter (0x83) is not supported by the ISO standards in CAN and LIN. Also it is not planned to support this service with FlexRay. Therefore no support for this service is planned.
- Subfunction onComparisionOfValues of Service ResponseOnEvent is not supported in the current release.
- Subfunction onTimerInterrupt of Service ResponseOnEvent is not supported in the current release.
- UDS Service SecuredDataTransmission (0x84) is not supported in the current release.
- The Dcm SWS does not cover any SAE J1939 related diagnostic requirements.
- Due to DEM limitation, the diagnostic service $19 05 is limited to the OBD legislative freeze frame.
- Management of IOControl service without InputOutputControlParameter in request and response is not supported
- The length of controlState parameter in IOControl request and response has to be of same size (due to the one configuration parameter DcmDspDataByteSize)
- Same layout of a DID which is used in RDBI, WDBI or IOCBI services
- The user optional parameter DTCSettingControlOptionRecord in the ControlDTCSetting request is only supported if it corresponds to a groupOfDTC value. In other cases it has to be managed in a vendor specific implementation.
- Only the ControlDTCSetting sub-functions 0x01 and 0x02 are supported.
- The handling of infrastructure errors reported by the RTE during DCM/DEM <-> SW-C interactions is missing from the SWS and might have to be taken into account by implementers if they need it.
- The Dcm does not support DLT for ROE
- The ROE ServiceToRespondTo does not support PageBuffering
- ROE only supports sub-function listed in Table 2

- DID range feature cannot be applied for services DynamicallyDefineDataIdentifier, ReadDataByPeriodicIdentifier and InputOutputControlById
- AUTOSAR Dcm is not intended to be used in the bootloader
- PeriodicTransmission is not possible on FlexRay, as ISO 14229-4 demands header information (address information (source and target address) and FPL (Frame Payload length)). This information can't be filled with the specified concept of IF interface.
- The specification of the transformer for intra ecu communication between the Dcm module and the NvBlockSoftwareComponentType is not standardized in the current AUTOSAR release. For this scenario custom transformers implemented by a complex driver can be used. To elaborate on this the responsible stakeholder (usually the OEM) needs to specify the custom transformer from a behavioral point of view in a separate document (this might include definition of byte-ordering or alignment). If there is the necessity to define transformer specific attributes in the model this can be done using special data groups in UserDefinedTransformationDescription and UserDefinedTransformationISignalProps. For the configuration of this scenario, a DataPrototypeMapping shall exist for the affected SenderReceiverInterfaces of the Dcm module and the NvBlockSoftwareComponentType which refers to a DataTransformation in the role firstToSecondDataTransformation. This DataTransformation shall reference exactly one TransformationTechnology in the role transformerChain with the transformerClass attribute set to "serializer" and may compose a UserDefinedTransformationDescription in the role transformationDescription.
- In certain situations the Dcm module is capable to process diagnostic requests in parallel. This possibility is explicitly limited of OBD in parallel to UDS protocol processing. No other protocol combination can be processed in parallel. Particularly the use case of parallel processing of two or more UDS protocol requests or WWH-OBD and UDS protocols is not supported.

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.

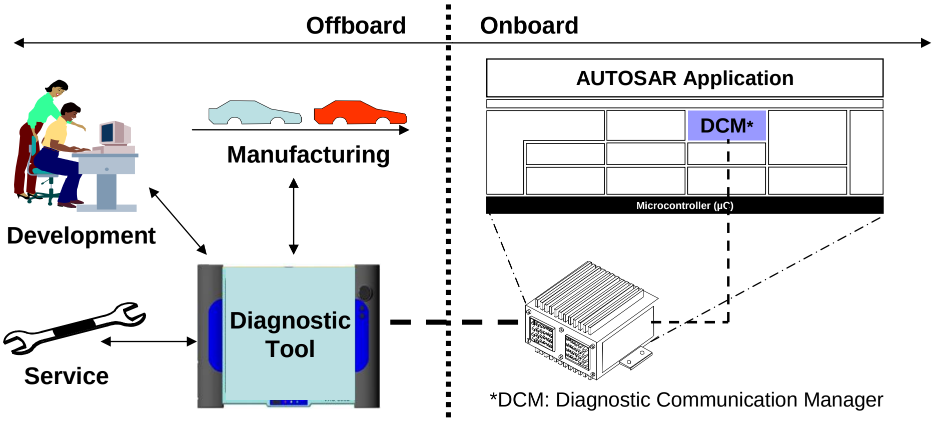


Figure 1.1: Overview of the communication between the external diagnostic tools and the onboard AUTOSAR Application

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 [2]				Legislated OBD - ISO15031-5 [1]
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 1.1: Diagnostic protocols and OSI-Layers

At OSI-level 7, the Dcm module provides an extensive set of ISO14229-1 [2] services. In addition, the Dcm module provides mechanisms to support the OBD services $01 - $0A defined in documents [3, SAE J1979] and [1, ISO 15031-5]. 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 [4]: Implementation of unified diagnostic services (UDS on CAN)
- ISO15765-4 [5]: 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).

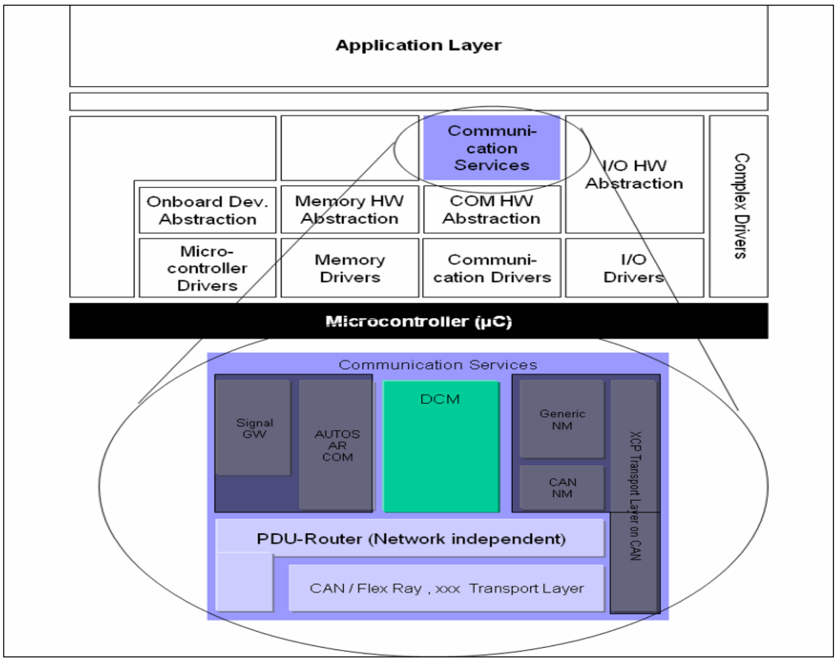


Figure 1.2: Position of the Dcm module in AUTOSAR Architecture

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

The glossary below includes acronyms and abbreviations relevant to the <MODULE_NAME> module that are not included in the [6, AUTOSAR glossary].

Abbreviation / Acronym:	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.

Abbreviation / Acronym:	Description:
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 • 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 [2].
OBD Service	This refers to an OBD Service as defined in ISO15031-5 [1].
AddressAndLengthFormat Identifier	Defines the number of bytes used for the memoryAddress and memorySize parameter in the request messages.
OBD Scan tool	See definition External Diagnostic Tool.

Terms:	Description:
API	Application Programming Interface
CAN	Controller Area Network
Dcm	Diagnostic Communication Manager
Dem	Diagnostic Event Manager
Det	Default Error Tracer
DID	Data Identifier

Terms:	Description:
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
RID	Routine 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
SPRMIB	suppressPosRspMsgIndicationBit

2.1 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 & related standards and norms

[1] Road vehicles – Communication between vehicle and external equipment for emission-related diagnostic – Part 5: Emission-related diagnostic services.
http://www.iso.org

[2] Unified diagnostic services (UDS) – Part 1: Specification and requirements (Release 2013-03)
http://www.iso.org

[3] SAE J1979

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

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

[6] Glossary
AUTOSAR_TR_Glossary

[7] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral

[8] ISO 17356-3: Road vehicles – Open interface for embedded automotive applications – Part 3: OSEK/VDX Operating System (OS)

[9] Specification of PDU Router
AUTOSAR_SWS_PDURouter

[10] Road vehicles – Diagnostics on Controller Area Networks (CAN) – Part2: Network layer services

[11] Specification of Diagnostic Event Manager
AUTOSAR_SWS_DiagnosticEventManager

[12] Road vehicles – Communication between vehicle and external equipment for emission-related diagnostic – Part 6: Diagnostic trouble code definitions
http://www.iso.org

[13] Specification of NVRAM Manager
AUTOSAR_SWS_NVRAMManager

[14] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture

[15] Specification of I/O Hardware Abstraction
AUTOSAR_SWS_IOHardwareAbstraction

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [7, SWS BSW General] , which is also valid for Diagnostic Communication Manager.

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

4 Constraints and assumptions

4.1 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 ISO 15031-5 [1] shall be processed prior to any enhanced diagnostic requests, the Dcm module provides a protocol switching mechanism based on protocol prioritization.
- UDS Service AccessTimingParameter (0x83) is not supported by the ISO standards in CAN and LIN. Also it is not planned to support this service with FlexRay. Therefore no support for this service is planned.
- Subfunction onComparisionOfValues of Service ResponseOnEvent is not supported in the current release.
- Subfunction onTimerInterrupt of Service ResponseOnEvent is not supported in the current release.
- UDS Service SecuredDataTransmission (0x84) is not supported in the current release.
- The Dcm SWS does not cover any SAE J1939 related diagnostic requirements.
- Due to DEM limitation, the diagnostic service $19 05 is limited to the OBD legislative freeze frame.
- Management of IOControl service without InputOutputControlParameter in request and response is not supported
- The length of controlState parameter in IOControl request and response has to be of same size (due to the one configuration parameter DcmDspDataByteSize)
- Same layout of a DID which is used in RDBI, WDBI or IOCBI services
- The user optional parameter DTCSettingControlOptionRecord in the ControlDTCSetting request is only supported if it corresponds to a groupOfDTC value. In other cases it has to be managed in a vendor specific implementation.
- Only the ControlDTCSetting sub-functions 0x01 and 0x02 are supported.
- The handling of infrastructure errors reported by the RTE during DCM/DEM <-> SW-C interactions is missing from the SWS and might have to be taken into account by implementers if they need it.
- The Dcm does not support DLT for ROE

- The ROE ServiceToRespondTo does not support PageBuffering
- ROE only supports sub-function listed in Table 2
- DID range feature cannot be applied for services DynamicallyDefineDataIdentifier, ReadDataByPeriodicIdentifier and InputOutputControlById
- AUTOSAR Dcm is not intended to be used in the bootloader
- PeriodicTransmission is not possible on FlexRay, as ISO 14229-4 demands header information (address information (source and target address) and FPL (Frame Payload length)). This information can't be filled with the specified concept of IF interface.
- The specification of the transformer for intra ecu communication between the Dcm module and the NvBlockSoftwareComponentType is not standardized in the current AUTOSAR release. For this scenario custom transformers implemented by a complex driver can be used. To elaborate on this the responsible stakeholder (usually the OEM) needs to specify the custom transformer from a behavioral point of view in a separate document (this might include definition of byte-ordering or alignment). If there is the necessity to define transformer specific attributes in the model this can be done using special data groups in UserDefinedTransformationDescription and UserDefinedTransformationISignalProps. For the configuration of this scenario, a DataPrototypeMapping shall exist for the affected SenderReceiverInterfaces of the Dcm module and the NvBlockSoftwareComponentType which refers to a DataTransformation in the role firstToSecondDataTransformation. This DataTransformation shall reference exactly one TransformationTechnology in the role transformerChain with the transformerClass attribute set to "serializer" and may compose a UserDefinedTransformationDescription in the role transformationDescription.
- In certain situations the Dcm module is capable to process diagnostic requests in parallel. This possibility is explicitly limited of OBD in parallel to UDS protocol processing. No other protocol combination can be processed in parallel. Particularly the use case of parallel processing of two or more UDS protocol requests or WWH-OBD and UDS protocols is not supported.

4.2 Applicability to car domains

The Dcm module can be used for all car domains.

4.3 Applicability to emission-related environments (OBD)

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

For WWH-OBD only the FunctionalGroupIdentifier 0x33 is currently supported.

5 Dependencies to other modules

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

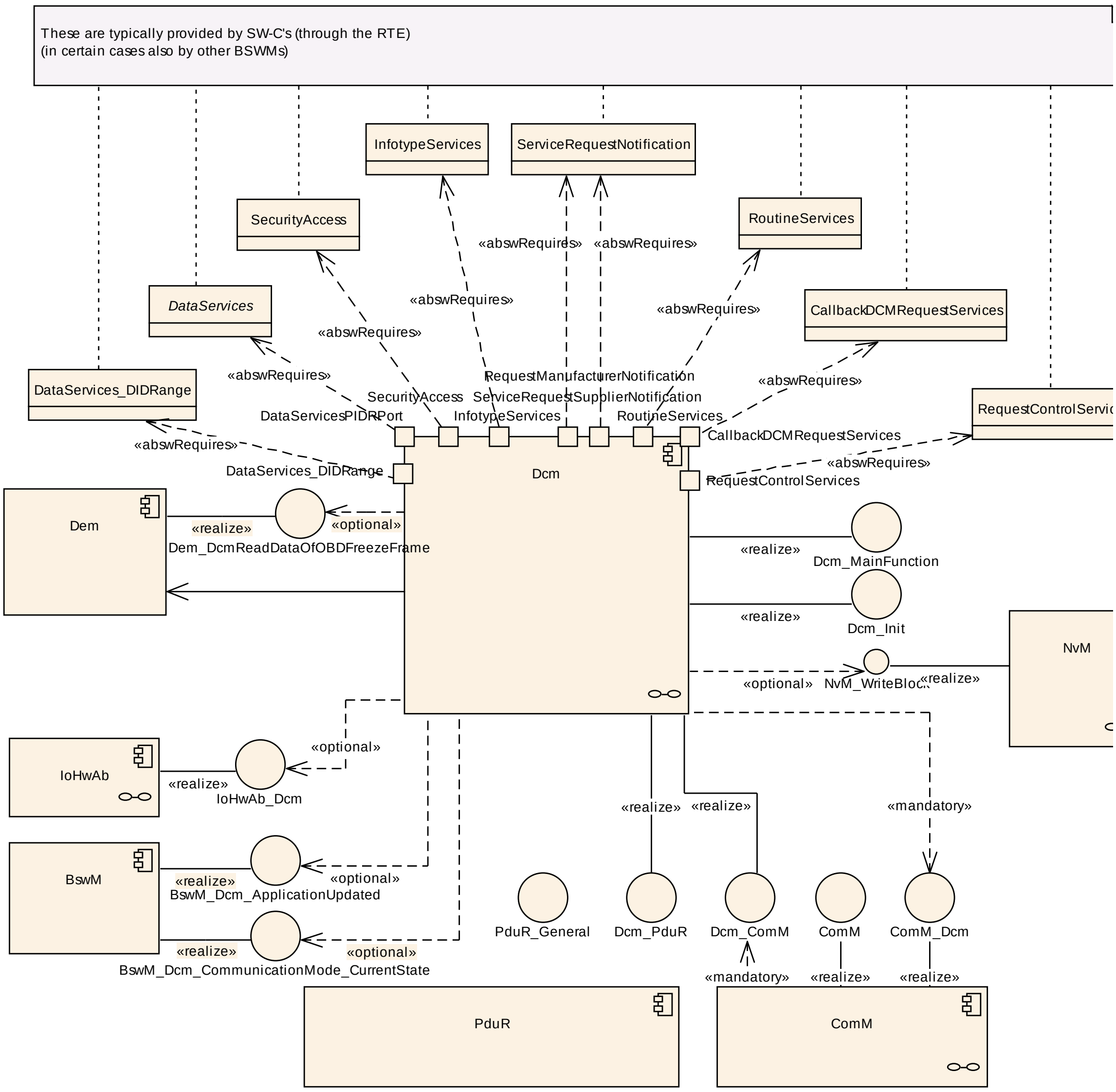


Figure 5.1: Interaction of the Dcm with other modules

- 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 [2], chapter 5 Application layer services).

- Communication Manager (ComM): The ComM module provides functions such that the Dcm module can indicate the states ”active” and ”inactive” for diagnostic communication. The Dcm module provides functionality to handle the communication requirements ”Full-/ Silent-/ No-Communication”. Additionally, the Dcm module provides the functionality to enable and disable Diagnostic Communication if requested by the ComM module.

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

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

5.1 File structure

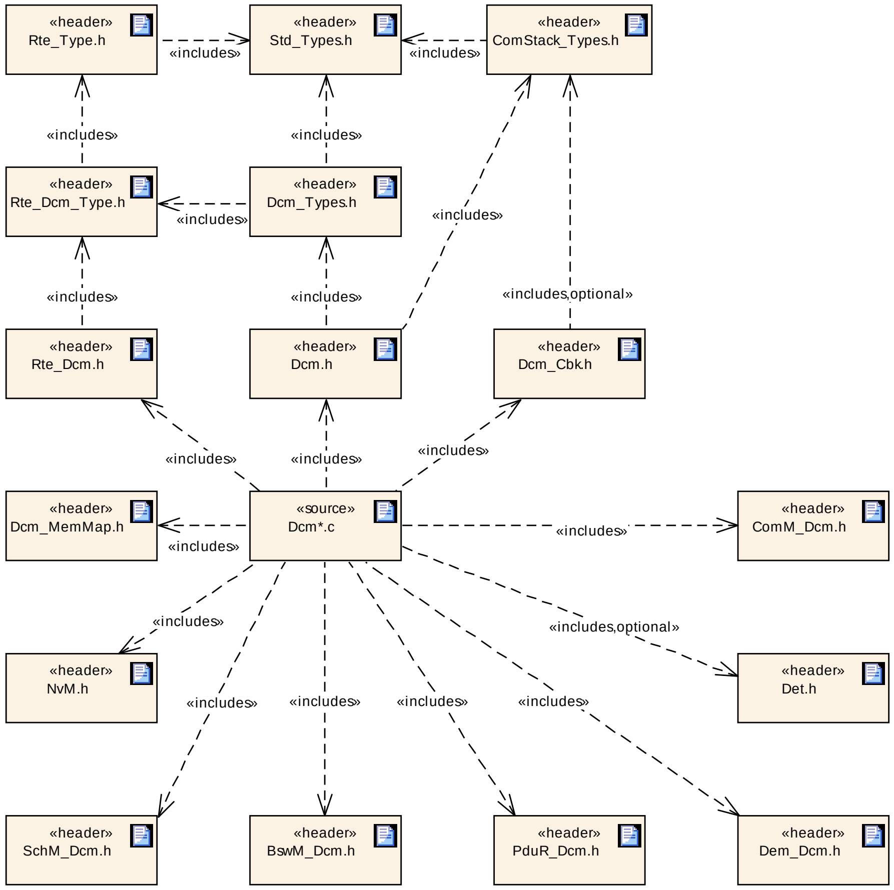


Figure 5.2: Dcm module file structure

[SWS_Dcm_00055] ⌈ The Dcm module shall use the header file structure shown in Figure 5.2. ⌋ *(SRS_BSW_00381, SRS_BSW_00412, SRS_BSW_00302)*

[SWS_Dcm_00683] ⌈ The file Dcm_Types.h shall provide all Dcm types definition. ⌋ *(SRS_BSW_00348)*

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

[SWS_Dcm_01065] ⌈ The file Dcm.h shall provide all type definitions and APIs used by other BSW modules for direct calls as described in chapter 8.3 Function definitions. ⌋ *()*

[SWS_Dcm_01066] ⌈ The file Dcm_Cbk.h shall provide all type definitions and APIs used by other BSW modules for direct calls as described in chapter 8.4 Callback Notifications. ⌋ *()*

6 Requirements Tracing

The following tables reference the requirements specified in <CITATIONS_OF_CONTRIBUTED_DOCUMENTS> and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00003]	All software modules shall provide version and identification information	[SWS_Dcm_00065]
[SRS_BSW_00005]	Modules of the μC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	[SWS_Dcm_NA_00999]
[SRS_BSW_00006]	The source code of software modules above the μC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	[SWS_Dcm_NA_00999]
[SRS_BSW_00007]	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	[SWS_Dcm_NA_00999]
[SRS_BSW_00009]	All Basic SW Modules shall be documented according to a common standard.	[SWS_Dcm_NA_00999]
[SRS_BSW_00010]	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	[SWS_Dcm_NA_00999]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_Dcm_00033] [SWS_Dcm_00034] [SWS_Dcm_00035] [SWS_Dcm_00036] [SWS_Dcm_00037]
[SRS_BSW_00158]	All modules of the AUTOSAR Basic Software shall strictly separate configuration from implementation	[SWS_Dcm_NA_00999]
[SRS_BSW_00159]	All modules of the AUTOSAR Basic Software shall support a tool based configuration	[SWS_Dcm_NA_00999]
[SRS_BSW_00160]	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	[SWS_Dcm_NA_00999]
[SRS_BSW_00161]	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	[SWS_Dcm_NA_00999]
[SRS_BSW_00162]	The AUTOSAR Basic Software shall provide a hardware abstraction layer	[SWS_Dcm_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00164]	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	[SWS_Dcm_NA_00999]
[SRS_BSW_00167]	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	[SWS_Dcm_NA_00999]
[SRS_BSW_00168]	SW components shall be tested by a function defined in a common API in the Basis-SW	[SWS_Dcm_NA_00999]
[SRS_BSW_00170]	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	[SWS_Dcm_NA_00999]
[SRS_BSW_00171]	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	[SWS_Dcm_NA_00999]
[SRS_BSW_00172]	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	[SWS_Dcm_NA_00999]
[SRS_BSW_00300]	All AUTOSAR Basic Software Modules shall be identified by an unambiguous name	[SWS_Dcm_NA_00999]
[SRS_BSW_00301]	All AUTOSAR Basic Software Modules shall only import the necessary information	[SWS_Dcm_NA_00999]
[SRS_BSW_00302]	All AUTOSAR Basic Software Modules shall only export information needed by other modules	[SWS_Dcm_00055]
[SRS_BSW_00304]	All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types	[SWS_Dcm_NA_00999]
[SRS_BSW_00305]	Data types naming convention	[SWS_Dcm_NA_00999]
[SRS_BSW_00306]	AUTOSAR Basic Software Modules shall be compiler and platform independent	[SWS_Dcm_NA_00999]
[SRS_BSW_00307]	Global variables naming convention	[SWS_Dcm_NA_00999]
[SRS_BSW_00308]	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	[SWS_Dcm_NA_00999]
[SRS_BSW_00309]	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	[SWS_Dcm_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00310]	API naming convention	[SWS_Dcm_NA_00999]
[SRS_BSW_00312]	Shared code shall be reentrant	[SWS_Dcm_NA_00999]
[SRS_BSW_00314]	All internal driver modules shall separate the interrupt frame definition from the service routine	[SWS_Dcm_NA_00999]
[SRS_BSW_00318]	Each AUTOSAR Basic Software Module file shall provide version numbers in the header file	[SWS_Dcm_NA_00999]
[SRS_BSW_00321]	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	[SWS_Dcm_NA_00999]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_Dcm_NA_00999]
[SRS_BSW_00325]	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	[SWS_Dcm_NA_00999]
[SRS_BSW_00327]	Error values naming convention	[SWS_Dcm_NA_00999]
[SRS_BSW_00328]	All AUTOSAR Basic Software Modules shall avoid the duplication of code	[SWS_Dcm_NA_00999]
[SRS_BSW_00330]	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	[SWS_Dcm_NA_00999]
[SRS_BSW_00331]	All Basic Software Modules shall strictly separate error and status information	[SWS_Dcm_NA_00999]
[SRS_BSW_00333]	For each callback function it shall be specified if it is called from interrupt context or not	[SWS_Dcm_NA_00999]
[SRS_BSW_00334]	All Basic Software Modules shall provide an XML file that contains the meta data	[SWS_Dcm_NA_00999]
[SRS_BSW_00335]	Status values naming convention	[SWS_Dcm_NA_00999]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_Dcm_NA_00999]
[SRS_BSW_00337]	Classification of development errors	[SWS_Dcm_00040]
[SRS_BSW_00339]	Reporting of production relevant error status	[SWS_Dcm_NA_00999]
[SRS_BSW_00341]	Module documentation shall contains all needed informations	[SWS_Dcm_NA_00999]
[SRS_BSW_00342]	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	[SWS_Dcm_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00343]	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	[SWS_Dcm_NA_00999]
[SRS_BSW_00344]	BSW Modules shall support link-time configuration	[SWS_Dcm_NA_00999]
[SRS_BSW_00345]	BSW Modules shall support pre-compile configuration	[SWS_Dcm_NA_00999]
[SRS_BSW_00346]	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	[SWS_Dcm_NA_00999]
[SRS_BSW_00347]	A Naming seperation of different instances of BSW drivers shall be in place	[SWS_Dcm_NA_00999]
[SRS_BSW_00348]	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	[SWS_Dcm_00683]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/disabling of detection and reporting of development errors.	[SWS_Dcm_NA_00999]
[SRS_BSW_00351]	Encapsulation of compiler specific methods to map objects	[SWS_Dcm_NA_00999]
[SRS_BSW_00353]	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	[SWS_Dcm_NA_00999]
[SRS_BSW_00357]	For success/failure of an API call a standard return type shall be defined	[SWS_Dcm_NA_00999]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_Dcm_NA_00999]
[SRS_BSW_00359]	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	[SWS_Dcm_NA_00999]
[SRS_BSW_00360]	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	[SWS_Dcm_NA_00999]
[SRS_BSW_00361]	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header	[SWS_Dcm_NA_00999]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_Dcm_00044]

Requirement	Description	Satisfied by
[SRS_BSW_00371]	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	[SWS_Dcm_NA_00999]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[SWS_Dcm_00053]
[SRS_BSW_00374]	All Basic Software Modules shall provide a readable module vendor identification	[SWS_Dcm_NA_00999]
[SRS_BSW_00375]	Basic Software Modules shall report wake-up reasons	[SWS_Dcm_NA_00999]
[SRS_BSW_00377]	A Basic Software Module can return a module specific types	[SWS_Dcm_NA_00999]
[SRS_BSW_00378]	AUTOSAR shall provide a boolean type	[SWS_Dcm_NA_00999]
[SRS_BSW_00379]	All software modules shall provide a module identifier in the header file and in the module XML description file.	[SWS_Dcm_NA_00999]
[SRS_BSW_00380]	Configuration parameters being stored in memory shall be placed into separate c-files	[SWS_Dcm_NA_00999]
[SRS_BSW_00381]	The pre-compile time parameters shall be placed into a separate configuration header file	[SWS_Dcm_00055]
[SRS_BSW_00383]	The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description	[SWS_Dcm_NA_00999]
[SRS_BSW_00384]	The Basic Software Module specifications shall specify at least in the description which other modules they require	[SWS_Dcm_NA_00999]
[SRS_BSW_00385]	List possible error notifications	[SWS_Dcm_NA_00999]
[SRS_BSW_00386]	The BSW shall specify the configuration for detecting an error	[SWS_Dcm_NA_00999]
[SRS_BSW_00388]	Containers shall be used to group configuration parameters that are defined for the same object	[SWS_Dcm_NA_00999]
[SRS_BSW_00389]	Containers shall have names	[SWS_Dcm_NA_00999]
[SRS_BSW_00390]	Parameter content shall be unique within the module	[SWS_Dcm_NA_00999]
[SRS_BSW_00392]	Parameters shall have a type	[SWS_Dcm_NA_00999]
[SRS_BSW_00393]	Parameters shall have a range	[SWS_Dcm_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00394]	The Basic Software Module specifications shall specify the scope of the configuration parameters	[SWS_Dcm_NA_00999]
[SRS_BSW_00395]	The Basic Software Module specifications shall list all configuration parameter dependencies	[SWS_Dcm_NA_00999]
[SRS_BSW_00396]	The Basic Software Module specifications shall specify the supported configuration classes for changing values and multiplicities for each parameter/container	[SWS_Dcm_NA_00999]
[SRS_BSW_00397]	The configuration parameters in pre-compile time are fixed before compilation starts	[SWS_Dcm_NA_00999]
[SRS_BSW_00398]	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	[SWS_Dcm_NA_00999]
[SRS_BSW_00399]	Parameter-sets shall be located in a separate segment and shall be loaded after the code	[SWS_Dcm_NA_00999]
[SRS_BSW_00400]	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	[SWS_Dcm_NA_00999]
[SRS_BSW_00401]	Documentation of multiple instances of configuration parameters shall be available	[SWS_Dcm_NA_00999]
[SRS_BSW_00402]	Each module shall provide version information	[SWS_Dcm_NA_00999]
[SRS_BSW_00403]	The Basic Software Module specifications shall specify for each parameter/container whether it supports different values or multiplicity in different configuration sets	[SWS_Dcm_NA_00999]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_Dcm_NA_00999]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[SWS_Dcm_NA_00999]
[SRS_BSW_00406]	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	[SWS_Dcm_NA_00999]
[SRS_BSW_00407]	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	[SWS_Dcm_00065]

Requirement	Description	Satisfied by
[SRS_BSW_00408]	All AUTOSAR Basic Software Modules configuration parameters shall be named according to a specific naming rule	[SWS_Dcm_NA_00999]
[SRS_BSW_00409]	All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from Dem configuration	[SWS_Dcm_NA_00999]
[SRS_BSW_00410]	Compiler switches shall have defined values	[SWS_Dcm_NA_00999]
[SRS_BSW_00411]	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	[SWS_Dcm_NA_00999]
[SRS_BSW_00412]	References to c-configuration parameters shall be placed into a separate h-file	[SWS_Dcm_00055]
[SRS_BSW_00413]	An index-based accessing of the instances of BSW modules shall be done	[SWS_Dcm_NA_00999]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_Dcm_NA_00999]
[SRS_BSW_00415]	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	[SWS_Dcm_NA_00999]
[SRS_BSW_00416]	The sequence of modules to be initialized shall be configurable	[SWS_Dcm_NA_00999]
[SRS_BSW_00417]	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	[SWS_Dcm_NA_00999]
[SRS_BSW_00419]	If a pre-compile time configuration parameter is implemented as "const" it should be placed into a separate c-file	[SWS_Dcm_NA_00999]
[SRS_BSW_00422]	Pre-de-bouncing of error status information is done within the DEM	[SWS_Dcm_NA_00999]
[SRS_BSW_00423]	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	[SWS_Dcm_NA_00999]
[SRS_BSW_00424]	BSW module main processing functions shall not be allowed to enter a wait state	[SWS_Dcm_00053]
[SRS_BSW_00425]	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	[SWS_Dcm_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00426]	BSW Modules shall ensure data consistency of data which is shared between BSW modules	[SWS_Dcm_NA_00999]
[SRS_BSW_00427]	ISR functions shall be defined and documented in the BSW module description template	[SWS_Dcm_NA_00999]
[SRS_BSW_00428]	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	[SWS_Dcm_NA_00999]
[SRS_BSW_00429]	Access to OS is restricted	[SWS_Dcm_NA_00999]
[SRS_BSW_00432]	Modules should have separate main processing functions for read/receive and write/transmit data path	[SWS_Dcm_NA_00999]
[SRS_BSW_00433]	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	[SWS_Dcm_NA_00999]
[SRS_BSW_00437]	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	[SWS_Dcm_NA_00999]
[SRS_BSW_00438]	Configuration data shall be defined in a structure	[SWS_Dcm_00037] [SWS_Dcm_00037]
[SRS_BSW_00439]	Enable BSW modules to handle interrupts	[SWS_Dcm_NA_00999]
[SRS_BSW_00440]	The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via Rte_Call API	[SWS_Dcm_NA_00999]
[SRS_BSW_00441]	Naming convention for type, macro and function	[SWS_Dcm_NA_00999]
[SRS_BSW_00447]	Standardizing Include file structure of BSW Modules Implementing Autosar Service	[SWS_Dcm_NA_00999]
[SRS_BSW_00448]	Module SWS shall not contain requirements from Other Modules	[SWS_Dcm_NA_00999]
[SRS_BSW_00449]	BSW Service APIs used by Autosar Application Software shall return a Std_ReturnType	[SWS_Dcm_NA_00999]
[SRS_BSW_00450]	A Main function of a un-initialized module shall return immediately	[SWS_Dcm_NA_00999]
[SRS_BSW_00451]	Hardware registers shall be protected if concurrent access to these registers occur	[SWS_Dcm_NA_00999]
[SRS_BSW_00452]	Classification of runtime errors	[SWS_Dcm_01416] [SWS_Dcm_NA_00999]
[SRS_BSW_00453]	BSW Modules shall be harmonized	[SWS_Dcm_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00454]	An alternative interface without a parameter of category DATA_REFERENCE shall be available.	[SWS_Dcm_NA_00999]
[SRS_BSW_00456]	- A Header file shall be defined in order to harmonize BSW Modules	[SWS_Dcm_NA_00999]
[SRS_BSW_00457]	- Callback functions of Application software components shall be invoked by the Basis SW	[SWS_Dcm_NA_00999]
[SRS_BSW_00458]	Classification of production errors	[SWS_Dcm_NA_00999]
[SRS_BSW_00459]	It shall be possible to concurrently execute a service offered by a BSW module in different partitions	[SWS_Dcm_NA_00999]
[SRS_BSW_00460]	Reentrancy Levels	[SWS_Dcm_NA_00999]
[SRS_BSW_00461]	Modules called by generic modules shall satisfy all interfaces requested by the generic module	[SWS_Dcm_NA_00999]
[SRS_BSW_00462]	All Standardized Autosar Interfaces shall have unique requirement Id / number	[SWS_Dcm_NA_00999]
[SRS_BSW_00463]	Naming convention of callout prototypes	[SWS_Dcm_NA_00999]
[SRS_BSW_00464]	File names shall be considered case sensitive regardless of the filesystem in which they are used	[SWS_Dcm_NA_00999]
[SRS_BSW_00465]	It shall not be allowed to name any two files so that they only differ by the cases of their letters	[SWS_Dcm_NA_00999]
[SRS_BSW_00466]	Classification of extended production errors	[SWS_Dcm_NA_00999]
[SRS_BSW_00467]	The init / deinit services shall only be called by BswM or EcuM	[SWS_Dcm_NA_00999]
[SRS_BSW_00469]	Fault detection and healing of production errors and extended production errors	[SWS_Dcm_NA_00999]
[SRS_BSW_00470]	Execution frequency of production error detection	[SWS_Dcm_NA_00999]
[SRS_BSW_00471]	Do not cause dead-locks on detection of production errors - the ability to heal from previously detected production errors	[SWS_Dcm_NA_00999]
[SRS_BSW_00472]	Avoid detection of two production errors with the same root cause.	[SWS_Dcm_NA_00999]
[SRS_BSW_00473]	Classification of transient faults	[SWS_Dcm_NA_00999]
[SRS_BSW_00477]	The functional interfaces of AUTOSAR BSW modules shall be specified in C90	[SWS_Dcm_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00478]	Timing limits of main functions	[SWS_Dcm_NA_00999]
[SRS_BSW_00479]	Interfaces for handling request from external devices	[SWS_Dcm_NA_00999]
[SRS_BSW_00480]	NullPointer Errors shall follow a naming rule	[SWS_Dcm_NA_00999]
[SRS_BSW_00481]	Invalid configuration set selection errors shall follow a naming rule	[SWS_Dcm_NA_00999]
[SRS_BSW_00482]	Get Version Informationfunction shall follow a naming rule	[SWS_Dcm_00065]
[SRS_Diag_04002]	The Diagnostic event (fault) management shall be established as Basic SW Module	[SWS_Dcm_NA_00999]
[SRS_Diag_04003]	Network independent design	[SWS_Dcm_00030]
[SRS_Diag_04005]	Manage Security Access level handling	[SWS_Dcm_00020] [SWS_Dcm_00033] [SWS_Dcm_00252] [SWS_Dcm_00338]
[SRS_Diag_04006]	Manage session handling	[SWS_Dcm_00022] [SWS_Dcm_00250] [SWS_Dcm_00339] [SWS_Dcm_01373] [SWS_Dcm_01374] [SWS_Dcm_01375] [SWS_Dcm_01376] [SWS_Dcm_01377] [SWS_Dcm_01378]
[SRS_Diag_04007]	Provide a diagnostic service handling for the applications involved in diagnostic functionality	[SWS_Dcm_NA_00999]
[SRS_Diag_04011]	Provide diagnostic state information to applications	[SWS_Dcm_00338] [SWS_Dcm_00339] [SWS_Dcm_00340] [SWS_Dcm_01321] [SWS_Dcm_01322]
[SRS_Diag_04015]	Timing handling according to ISO15765-3	[SWS_Dcm_00027] [SWS_Dcm_00030] [SWS_Dcm_00143] [SWS_Dcm_00144] [SWS_Dcm_00311]
[SRS_Diag_04016]	Support ”Busy handling” by sending a negative response 0x78	[SWS_Dcm_00024]
[SRS_Diag_04019]	Confirm transmitting if complete to continue processing	[SWS_Dcm_NA_00999]
[SRS_Diag_04020]	Suppress responses to diagnostic tool requests	[SWS_Dcm_00001] [SWS_Dcm_00200]
[SRS_Diag_04021]	Handling of different diagnostic sessions in parallel	[SWS_Dcm_00015]
[SRS_Diag_04024]	Access and handle specific data elements and data element groups if requested by an external scan tool	[SWS_Dcm_NA_00999]
[SRS_Diag_04031]	Notify the Function Inhibition Manager (FIM) upon changes of the event status in order to process them according to the SW components dependencies	[SWS_Dcm_NA_00999]
[SRS_Diag_04032]	Different diagnostic addresses shall be supported by multiple (physical) channels	[SWS_Dcm_NA_00999]

Requirement	Description	Satisfied by
[SRS_Diag_04033]	Support the upload/download services for reading/writing data in an ECU in an extended and manufacturer specific diagnostic session	[SWS_Dcm_00496] [SWS_Dcm_00499] [SWS_Dcm_00502] [SWS_Dcm_00503] [SWS_Dcm_00504] [SWS_Dcm_00505] [SWS_Dcm_01395] [SWS_Dcm_01396] [SWS_Dcm_01417] [SWS_Dcm_01418] [SWS_Dcm_01419] [SWS_Dcm_01420] [SWS_Dcm_01421] [SWS_Dcm_01422]
[SRS_Diag_04057]	Classification of events for series production, OBD and expert usage	[SWS_Dcm_NA_00999]
[SRS_Diag_04058]	Ability to access different event memories	[SWS_Dcm_00004] [SWS_Dcm_00005] [SWS_Dcm_00077] [SWS_Dcm_00279] [SWS_Dcm_00293] [SWS_Dcm_00295] [SWS_Dcm_00378] [SWS_Dcm_00383] [SWS_Dcm_00384] [SWS_Dcm_00388] [SWS_Dcm_00389] [SWS_Dcm_00393] [SWS_Dcm_00465] [SWS_Dcm_01147] [SWS_Dcm_01263]
[SRS_Diag_04059]	Configuration of timing parameters	[SWS_Dcm_NA_00999]
[SRS_Diag_04063]	Process a dedicated event identifier for each monitoring path to support an autonomous handling of different events/faults	[SWS_Dcm_NA_00999]
[SRS_Diag_04064]	Buffers of scalable sizes for the storage of the events, status information and environmental data	[SWS_Dcm_NA_00999]
[SRS_Diag_04067]	Provide the diagnostic status information according to ISO 14229-1	[SWS_Dcm_00293] [SWS_Dcm_00378]
[SRS_Diag_04068]	Event specific debounce algorithms	[SWS_Dcm_NA_00999]
[SRS_Diag_04071]	Process events according to their defined importance like priority and/or severity	[SWS_Dcm_NA_00999]
[SRS_Diag_04077]	Uses standard mechanisms provided by persistency modules	[SWS_Dcm_NA_00999]
[SRS_Diag_04085]	The Default Error Tracer shall provide an interface to receive error reports	[SWS_Dcm_NA_00999]
[SRS_Diag_04086]	Report errors shall contain a dedicated set of information	[SWS_Dcm_NA_00999]
[SRS_Diag_04087]	The Default Error Tracer shall provide a development error report reception service	[SWS_Dcm_NA_00999]
[SRS_Diag_04089]	The DET module shall support fan-out of received error reports	[SWS_Dcm_NA_00999]
[SRS_Diag_04090]	A configurable list of error report receivers shall be provided	[SWS_Dcm_NA_00999]
[SRS_Diag_04091]	Notification about valid freeze frame data to applications	[SWS_Dcm_NA_00999]
[SRS_Diag_04093]	Memory overflow indication	[SWS_Dcm_NA_00999]

Requirement	Description	Satisfied by
[SRS_Diag_04097]	Decentralized and modular diagnostic configuration in applications	[SWS_Dcm_NA_00999]
[SRS_Diag_04098]	Interact with standard bootloader	[SWS_Dcm_00532] [SWS_Dcm_00535] [SWS_Dcm_00536] [SWS_Dcm_00592] [SWS_Dcm_00654] [SWS_Dcm_00767] [SWS_Dcm_01163] [SWS_Dcm_01177] [SWS_Dcm_01423] [SWS_Dcm_01424] [SWS_Dcm_01425] [SWS_Dcm_CONSTR_6080]
[SRS_Diag_04100]	Interface for logging and tracing	[SWS_Dcm_NA_00999]
[SRS_Diag_04101]	The DET module shall forward its trace events to the DLT	[SWS_Dcm_NA_00999]
[SRS_Diag_04105]	Event memory management	[SWS_Dcm_NA_00999]
[SRS_Diag_04107]	Provide defensive behavior	[SWS_Dcm_NA_00999]
[SRS_Diag_04109]	Provide an interface to retrieve the number of event memory entries	[SWS_Dcm_NA_00999]
[SRS_Diag_04110]	SAE J1939 lamp status	[SWS_Dcm_NA_00999]
[SRS_Diag_04111]	SAE J1939 Expanded-Freeze Frame	[SWS_Dcm_NA_00999]
[SRS_Diag_04112]	The DEM module shall support DTCs according to SAE J1939	[SWS_Dcm_NA_00999]
[SRS_Diag_04113]	Support a set of SAE J1939 DM-messages	[SWS_Dcm_NA_00999]
[SRS_Diag_04115]	The optional parameter DTCSettingControlOption Record as part of UDS service ControlDTCSetting shall be limited to GroupOfDTC	[SWS_Dcm_00406] [SWS_Dcm_01063] [SWS_Dcm_NA_00999]
[SRS_Diag_04117]	Configurable behavior for DTC deletion	[SWS_Dcm_NA_00999]
[SRS_Diag_04118]	Optionally support event displacement	[SWS_Dcm_NA_00999]
[SRS_Diag_04119]	Handle the execution of diagnostic services according to the assigned diagnostic session	[SWS_Dcm_NA_00999]
[SRS_Diag_04120]	Support a predefined Address AndLengthFormatIdentifier	[SWS_Dcm_NA_00999]
[SRS_Diag_04121]	Provide the handling of service DynamicallyDefineDataIdentifier according to ISO 14229-1	[SWS_Dcm_NA_00999]
[SRS_Diag_04123]	Harmonized Driving//WarmUp cycles	[SWS_Dcm_NA_00999]
[SRS_Diag_04124]	Store the current debounce counter value non-volatile to over a power-down cycle	[SWS_Dcm_NA_00999]
[SRS_Diag_04125]	Event debounce counter shall be configurable	[SWS_Dcm_NA_00999]
[SRS_Diag_04126]	Configurable suppression of events	[SWS_Dcm_NA_00999]

Requirement	Description	Satisfied by
[SRS_Diag_04127]	Configurable record numbers and trigger options for DTCSnapshotRecords and DTCExtendedDataRecords	[SWS_Dcm_NA_00999]
[SRS_Diag_04129]	Provide OBD-specific configuration capabilities	[SWS_Dcm_NA_00999]
[SRS_Diag_04131]	Consistent event management mechanisms	[SWS_Dcm_NA_00999]
[SRS_Diag_04133]	Aging for event memory entries	[SWS_Dcm_NA_00999]
[SRS_Diag_04135]	Support UDS service $38 (RequestFileTransfer)	[SWS_Dcm_NA_00999]
[SRS_Diag_04136]	Configurable ”confirmed” threshold	[SWS_Dcm_NA_00999]
[SRS_Diag_04137]	Definition of replacement failure	[SWS_Dcm_NA_00999]
[SRS_Diag_04139]	Support subfunction 0x42 of UDS service 0x19	[SWS_Dcm_NA_00999]
[SRS_Diag_04140]	Aging for UDS status bits ”confirmedDTC” and ”testFailed SinceLastClear”	[SWS_Dcm_NA_00999]
[SRS_Diag_04143]	The Default Error Tracer shall provide an interface to receive runtime error reports	[SWS_Dcm_NA_00999]
[SRS_Diag_04144]	The Default Error Tracer shall provide an interface to receive transient fault reports	[SWS_Dcm_NA_00999]
[SRS_Diag_04145]	The Default Error Tracer shall forward received runtime error reports to configured integrator code	[SWS_Dcm_NA_00999]
[SRS_Diag_04146]	The Default Error Tracer shall forward received transient fault reports to configured integrator code	[SWS_Dcm_NA_00999]
[SRS_Diag_04147]	Communication with the transport layers to receive and send diagnostic data	[SWS_Dcm_00642] [SWS_Dcm_01186]
[SRS_Diag_04148]	Provide capabilities to inform applications about diagnostic data changes	[SWS_Dcm_NA_00999]
[SRS_Diag_04150]	Support the primary fault memory defined by ISO 14229-1	[SWS_Dcm_NA_00999]
[SRS_Diag_04151]	Event status handling	[SWS_Dcm_NA_00999]
[SRS_Diag_04153]	Support generic connections	[SWS_Dcm_00849] [SWS_Dcm_01347] [SWS_Dcm_01348]
[SRS_Diag_04159]	Control of DTC storage	[SWS_Dcm_00249] [SWS_Dcm_01399]
[SRS_Diag_04162]	Parallel fault memory access	[SWS_Dcm_01369] [SWS_Dcm_01370] [SWS_Dcm_01371] [SWS_Dcm_01372]
[SRS_Diag_04163]	Parallel OBD and UDS processing	[SWS_Dcm_01365] [SWS_Dcm_01366] [SWS_Dcm_01367]

Requirement	Description	Satisfied by
[SRS_Diag_04215]	Support of UDS service Read DataByPeriodicIdentifier (0x2A)	[SWS_Dcm_00254] [SWS_Dcm_00716] [SWS_Dcm_00721] [SWS_Dcm_00722] [SWS_Dcm_00820] [SWS_Dcm_00843] [SWS_Dcm_00851] [SWS_Dcm_01093] [SWS_Dcm_01094] [SWS_Dcm_01095] [SWS_Dcm_01096] [SWS_Dcm_01097] [SWS_Dcm_01098] [SWS_Dcm_01099] [SWS_Dcm_01100] [SWS_Dcm_01101] [SWS_Dcm_01102] [SWS_Dcm_01103] [SWS_Dcm_01104] [SWS_Dcm_01105] [SWS_Dcm_01106] [SWS_Dcm_01107] [SWS_Dcm_01108] [SWS_Dcm_01109] [SWS_Dcm_01110] [SWS_Dcm_01111] [SWS_Dcm_01112] [SWS_Dcm_01113] [SWS_Dcm_01114] [SWS_Dcm_01115] [SWS_Dcm_01116] [SWS_Dcm_01117] [SWS_Dcm_01118] [SWS_Dcm_01426] [SWS_Dcm_01427] [SWS_Dcm_01428]
[SRS_Diag_04218]	Support of UDS service 0x2F InputOutputControlByIDentifier.	[SWS_Dcm_00680] [SWS_Dcm_01280] [SWS_Dcm_01337]

7 Functional specification

7.1 Error Classification

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

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

[SWS_Dcm_00044] ⌈ The error values shall be the unique for all error types. The Dcm shall use only the values given in this chapter. ⌋*(SRS_BSW_00369)*

7.1.1 Development Errors

[SWS_Dcm_00040] Development Error Types ⌈ The errors and exceptions described in Table 7.1 shall be detectable by the Dcm module depending on its build version (development/production mode) ⌋*(SRS_BSW_00337)*

Type of error	Related error code	Value [hex]
Interface: Timeout occurred during interaction with another module (e.g. maximum number of response pending is reached, refer to [SWS_Dcm_00120])	DCM_E_INTERFACE_TIMEOUT	0x01
Interface return-value is out of range	DCM_E_INTERFACE_RETURN_VALUE	0x02

Type of error	Related error code	Value [hex]
Interface: Boundary check of buffers provided by the Dcm failed during interaction with another module (application, Dem, PduR, etc.)	DCM_E_INTERFACE_BUFFER_OVERFLOW	0x03
Internal:Dcm not initialized	DCM_E_UNINIT	0x05
Dcm API function with invalid input parameter	DCM_E_PARAM	0x06
Dcm API service invoked with NULL POINTER as parameter	DCM_E_PARAM_POINTER	0x07
Dcm initialisation failed	DCM_E_INIT_FAILED	0x08
Storing the ProgConditions failed	DCM_E_SET_PROG_CONDITIONS_FAIL	0x09

Table 7.1: Dcm development errors

7.1.2 Runtime Errors

[SWS_Dcm_01416] Runtime Error Types ⌈ The errors and exceptions described in Table 7.2 shall be detectable by the Dcm module depending on its build version (development/production mode). ⌋*(SRS_BSW_00452)*

Type of error	Related error code	Value [hex]
The Dcm is getting called with an invalid input parameter value or the Dcm has called an function and this function returns an invalid out parameter or return value.	DCM_E_INVALID_VALUE	0x01

Table 7.2: Dcm Runtime errors

7.1.3 Transient Faults

There are no transient faults.

7.1.4 Production Errors

There are no production errors.

7.1.5 Extended Production Errors

There are no extended production errors.

7.2 General design elements

7.2.1 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.

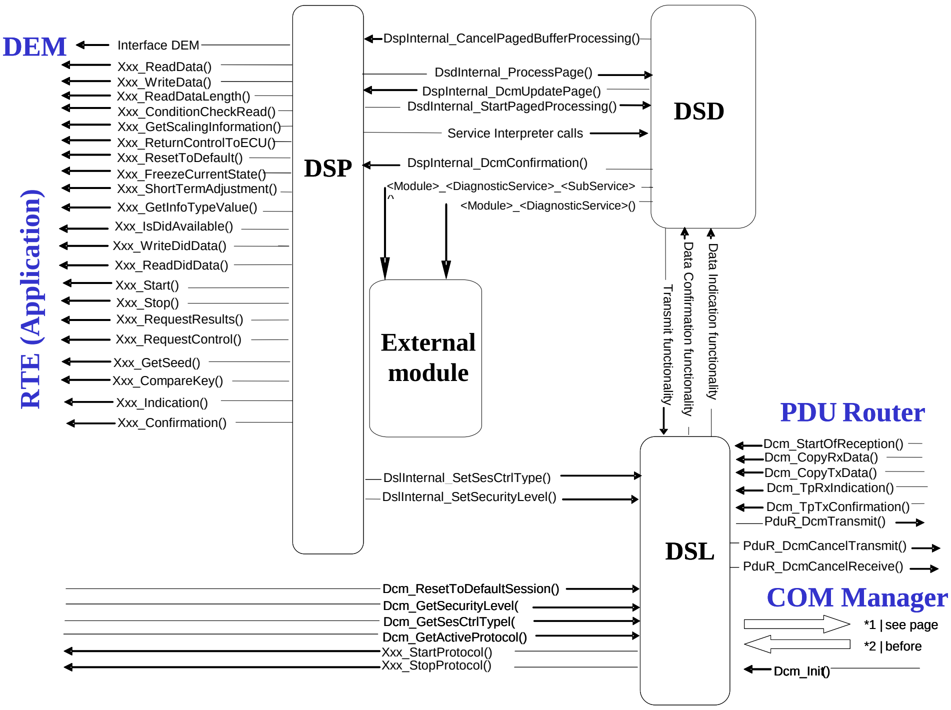


Figure 7.1: Possible interaction between the submodules in the DCM

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.

7.2.2 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.

[SWS_Dcm_01075] ⌈ The order of the transmitted NRC shall be compliant with the one described in ISO14229-1 [2]. ⌋*()*

7.2.3 Non-volatile information

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

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

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

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

7.2.4 Types

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

- boolean
- uint8
- uint16
- uint32
- sint8
- sint16
- sint32

- uint8[n]

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

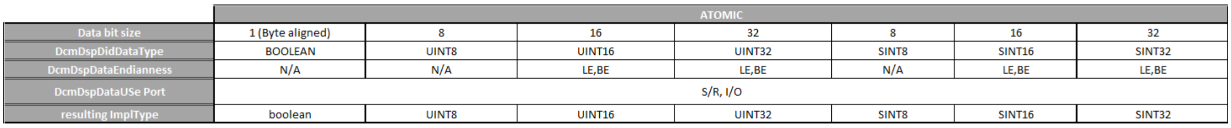
[SWS_Dcm_00969] ⌈ The Dcm shall treat non-integer data types (e.g. uint8[n]) either like integer data types of the matching size or leave their contents uninterpreted in case DcmDspDataEndianness is configured to OPAQUE. ⌋()

[SWS_Dcm_00970] ⌈ The Dcm module shall interpret opaque data as uint8[n] and shall always map it to an n-bytes sized signal. For opaque data endianness, DcmDspDataEndianness has to be configured to OPAQUE. ⌋()

[SWS_Dcm_00971] ⌈ The Dcm shall extend the endianness conversion defined in [8], to signed data types. ⌋()

In [8] (Chapter 2.4) the endianness conversion is defined for unsigned data types. The associated configurations can be found in the configuration 10.3.5.5.1 DcmDspData.

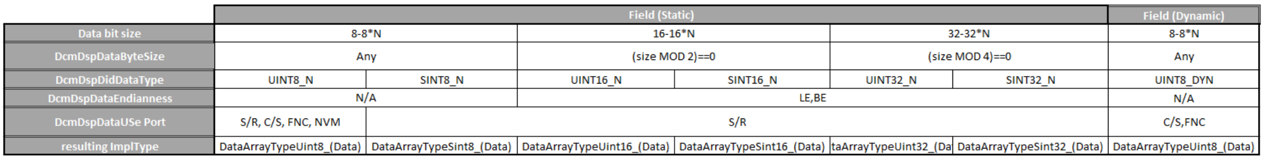
7.2.4.1 Atomic types overview



	ATOMIC						
Data bit size	1 (Byte aligned)	8	16	32	8	16	32
DcmDspDidDataType	BOOLEAN	UINT8	UINT16	UINT32	SINT8	SINT16	SINT32
DcmDspDataEndianness	N/A	N/A	LE,BE	LE,BE	N/A	LE,BE	LE,BE
DcmDspDataUSe Port	S/R, I/O						
resulting ImplType	boolean	UINT8	UINT16	UINT32	SINT8	SINT16	SINT32

Figure 7.2: Atomic types overview

7.2.4.2 Data array types overview



	Field (Static)						Field (Dynamic)
Data bit size	8-8*N		16-16*N		32-32*N		8-8*N
DcmDspDataByteSize	Any		(size MOD 2)==0		(size MOD 4)==0		Any
DcmDspDidDataType	UINT8_N	SINT8_N	UINT16_N	SINT16_N	UINT32_N	SINT32_N	UINT8_DYN
DcmDspDataEndianness	N/A		LE,BE				N/A
DcmDspDataUSe Port	S/R, C/S, FNC, NVM	S/R					C/S,FNC
resulting ImplType	DataArrayTypeUint8_(Data)	DataArrayTypeSint8_(Data)	DataArrayTypeUint16_(Data)	DataArrayTypeSint16_(Data)	taArrayTypeUint32_(Da	DataArrayTypeSint32_(Data)	DataArrayTypeUint8_(Data)

Figure 7.3: Data array types overview

7.2.4.3 Data types constraints

[SWS_Dcm_CONSTR_6002] Existence of size parameter ⌈ DcmDspDataByteSize shall be present if DcmDspDataType is set to: UINT8_N, SINT8_N, UINT16_N, SINT16_N, UINT32_N, SINT32_N or UINT8_DYN. ⌋()

Note: DcmDspDataByteSize is not required for primitive datatypes

[SWS_Dcm_CONSTR_6035] Restrictions on size parameter for 16 Bit arrays ⌈ DcmDspDataByteSize shall be a multiple of 2 if the value is greater than 2 and DcmDspDataType is UINT16_N or SINT16_N. ⌋*()*

[SWS_Dcm_CONSTR_6036] Restrictions on size parameter for 32 Bit arrays ⌈ DcmDspDataByteSize shall be a multiple of 4 if the value is greater than 4 and DcmDspDataType is UINT32_N or SINT32_N. ⌋*()*

[SWS_Dcm_CONSTR_6008] Define the usage of DcmDspRoutineParameterSize parameter ⌈ DcmDspRoutineParameterSize is only required if DcmDspRoutineSignalType is set to SINT8_N, SINT16_N, SINT32_N, UINT8_N, UINT16_N, UINT32_N or VARIABLE_LENGTH. ⌋*()*

[SWS_Dcm_CONSTR_6011] Only last parameters in RID may have a variable length ⌈ DcmDspRoutineSignalType with VARIABLE_LENGTH is only valid for the last signal. ⌋*()*

[SWS_Dcm_CONSTR_6012] Existence of size parameter ⌈ DcmDspPidDataByteSize shall be present if DcmDspPidDataType is set to: UINT8_N, SINT8_N, UINT16_N, SINT16_N, UINT32_N or SINT32_N. ⌋*()*

Note: DcmDspPidDataByteSize is not required for primitive datatypes

[SWS_Dcm_CONSTR_6040] Restrictions on size parameter for 16 Bit arrays ⌈ DcmDspPidDataByteSize shall be a multiple of 2 if the value is greater than 2 and DcmDspPIDDataType is UINT16_N or SINT16_N. ⌋*()*

[SWS_Dcm_CONSTR_6041] Restrictions on size parameter for 32 Bit arrays ⌈ DcmDspPidDataByteSize shall be a multiple of 4 if the value is greater than 4 and DcmDspPIDDataType is UINT32_N or SINT32_N. ⌋*()*

UINT8 shall be used as (implementation) data type for bit lengths between 1 and 8

[SWS_Dcm_CONSTR_6042] UINT8 shall be used as (implementation) data type for Client-Server interface ⌈ In case DcmDspPidDataUsePort parameter is set to USE_DATA_SYNCH_CLIENT_SERVER, DcmDspPIDDataType shall use UINT8_N. ⌋*()*

[SWS_Dcm_CONSTR_6043] Restrictions on datatype usage ⌈ DcmDspPIDDataType shall be UINT8_N in case DcmDspPidDataUsePort is equal to USE_DATA_SYNCH_FNC. ⌋*()*

[SWS_Dcm_CONSTR_6024] UINT8 shall be used as (implementation) data type for Client-Server interface ⌈ In case DcmDspDataUsePort parameter is set to USE_DATA_SYNCH_CLIENT_SERVER, USE_DATA_ASYNCH_CLIENT_SERVER, USE_DATA_ASYNCH_CLIENT_SERVER_ERROR, DcmDspDataType shall use UINT8_N or UINT8_DYN. ⌋*()*

[SWS_Dcm_CONSTR_6037] Restrictions on datatype usage ⌈ DcmDspDataType shall be UINT8_N or UINT8_DYN, in case DcmDspDataUsePort is equal to USE_DATA_ASYNCH_FNC_ERROR ||USE_DATA_SYNCH_FNC ||USE_DATA_ASYNCH_FNC. ⌋*()*

[SWS_Dcm_CONSTR_6038] Restrictions on datatype usage ⌈ DcmDspDataType shall be UINT8_N, in case DcmDspDataUsePort is equal to USE_BLOCK_ID. ⌋*()*

[SWS_Dcm_CONSTR_6026] Usage of variable data length in case of S/R communication, NvRam access or ECU signal access ⌈ In case DcmDspDataUsePort is set to {USE_DATA_SENDER_RECEIVER, USE_DATA_SENDER_RECEIVER_AS_SERVICE, USE_BLOCK_ID, USE_ECU_SIGNAL}, the usage of variable data length shall be not allowed. ⌋*()*

[SWS_Dcm_CONSTR_6031] ⌈ The DcmDspData.SHORT-NAME and DcmDspPidData.SHORT-NAME shall be distinct. ⌋*()*

Note: Variable data length is only possible with UINT8 arrays with DcmDspDataType set to UINT8_DYN.

7.2.4.4 Dcm_OpStatusType

For the operation using the Dcm_OpStatusType, the Dcm shall work as follow : **[SWS_Dcm_00527]** ⌈ At first call of an operation using the Dcm_OpStatusType, the Dcm call the operation with OpStatus = DCM_INITIAL ⌋*()*

[SWS_Dcm_00528] ⌈ If the value DCM_E_FORCE_RCRRP is returned from an operation using Dcm_OpStatusType, the Dcm shall invoke the transmit request for RCR-RP (NRC 0x78 transmission) and the Dcm shall not realize further invocation of the operation till RCR-RP is transmitted. ⌋*()*

[SWS_Dcm_00529] ⌈ After transmit confirmation of a RCR-RP transmitted on the context of [SWS_Dcm_00528], the Dcm calls, from Dcm_MainFunction (due to call context), the operation again with OpStatus = DCM_FORCE_RCRRP_OK. ⌋*()*

[SWS_Dcm_00530] ⌈ If a DCM_E_PENDING value is returned from an operation using the Dcm_OpStatusType, the Dcm call the operation on each Dcm_MainFunction call with OpStatus = DCM_PENDING as long as DCM_E_PENDING is returned. ⌋*()*

7.2.4.5 Dcm_SesCtrlType

7.3 Diagnostic Session Layer (DSL)

7.3.1 Introduction

[SWS_Dcm_00030] ⌈ All functional areas of the DSL submodule shall be in conformance with the specifications ISO14229-1 [2] and the network-independent part of ISO15765-3 [4]. ⌋*(SRS_Diag_04003, SRS_Diag_04015)*

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

7.3.2 Use cases

The DSL submodule provides the following functionalities:

- Session handling (as required by ISO14229-1 [2] and ISO 15765-3 [4])
- Application layer timing handling (as required by ISO14229-1 [2] and ISO 15765-3 [4])
- Specific response behavior (as required by ISO14229-1 [2] and ISO 15765-3 [4])

7.3.3 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

7.3.4 Functional description

7.3.4.1 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.

7.3.4.2 Forward requests from the PduR module to the DSD submodule

The PduR module indicates the Dcm module whenever a reception of new diagnostic request content is started on a DcmRxPduId, which is assigned to the Dcm module. This is done by calling Dcm_StartOfReception, which inform the Dcm module of the data size to be received and provides the data of the first frame or single frame, and allows the Dcm to reject the reception if the data size overflows its buffer size, or if the requested service is not available. The further call to Dcm_CopyRxData request the Dcm module to copy the data from the provided buffer to the Dcm buffer. If the reception of a diagnostic request is finished (when Dcm_StartOfReception succeeded) the PduR module will call Dcm_TpRxIndication to give a receive indication to the Dcm module. The Dcm shall be able to use generic connections, where the addressing information is provided to Dcm by Dcm_StartOfReception via the MetaData of the DcmRxPdu. This addressing information must be stored and used for the response and for detection of requests from the same tester. see section 7.3.4.5 Generic Connection Handling for further details.

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

[SWS_Dcm_00241] ⌈ As soon as a request message is received (after a call of Dcm_TpRxIndication with parameter Result = E_OK (see [SWS_Dcm_00093]) and until a call to Dcm_TpTxConfirmation (see [SWS_Dcm_00351]) for the associated Tx-DcmPduId), the DSL submodule shall block the corresponding DcmPduId. During the processing of this request, no other request of the same DcmDslConnection (e.g. an enhanced session can be ended by a OBD session) can be received, until the corresponding response message is sent and the DcmPduId is released again (except for concurrent TesterPresent requests). ⌋*()*

More descriptions of the APIs (prototype, input/output parameter) can be found in the interface description of PduR module [9].

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).

7.3.4.2.1 Dcm_StartOfReception

[SWS_Dcm_00444] ⌈ If the requested size is large than the buffer available in the DCM, the function Dcm_StartOfReception shall return BUFREQ_E_OVFL (see [SWS_Dcm_00094]). ⌋*()*

[SWS_Dcm_00788] ⌈ When processing a diagnostic request and in case DcmDslDiagRespOnSecondDeclinedRequest is set to TRUE, the Dcm module shall return BUFREQ_OK on Dcm_StartOfReception received on new request using a different DcmDslConnection. ⌋*()*

[SWS_Dcm_00789] ⌈ In case [SWS_Dcm_00788], the Dcm respond with a NRC 0x21 ⌋*()*

[SWS_Dcm_00790] ⌈ When processing a diagnostic request, the Dcm module shall reject (Dcm_StartOfReception shall return BUFREQ_E_NOT_OK) any new request using a different DcmDslConnection in case DcmDslDiagRespOnSecondDeclinedRequest is set to FALSE until the current diagnostic request processing is over. ⌋*()*

[SWS_Dcm_00557] ⌈ When processing a diagnostic request, the Dcm module shall reject (Dcm_StartOfReception shall return BUFREQ_E_NOT_OK) any new diagnostic request with the same DcmDslConnection until the current diagnostic request processing is over. Concurrent TesterPresent requests will be accepted with a BUFREQ_OK, but not further processed, as the running diagnostic request already resets the session timeout timer (S3Server). ⌋*()*

[SWS_Dcm_01145] ⌈ If the current session is a non-default session and a concurrent TesterPresent received on a different DcmDslConnection, this request will be accepted with a BUFREQ_OK, but not further processed. E.g. it is not resetting the session timeout timer (S3Server) ⌋*()*

[SWS_Dcm_01146] ⌈ In case of [SWS_Dcm_01145] with reception on a higher priority protocol, this will not lead to protocol preemption. ⌋*()*

[SWS_Dcm_00642] ⌈ When the API Dcm_StartOfReception is invoked with TpSduLength equal to 0, the value BUFREQ_E_NOT_OK shall be returned and no further action shall be taken. ⌋*(SRS_Diag_04147)*

[SWS_Dcm_00655] ⌈ If the current session is a non-default session and a new diagnostic request with same or lower priority protocol than active one is detected, the Dcm shall act according [SWS_Dcm_00788], [SWS_Dcm_00789] and [SWS_Dcm_00790]. ⌋*()*

[SWS_Dcm_00656] ⌈ If the current session is the default session and a diagnostic request is in execution, for any new diagnostic request with same or lower priority protocol than active one, the Dcm shall act according [SWS_Dcm_00788], [SWS_Dcm_00789] and [SWS_Dcm_00790]. ⌋*()*

[SWS_Dcm_00833] ⌈ Dcm_StartOfReception () shall be callable in interrupt context. ⌋*()*

7.3.4.2.2 Dcm_CopyRxData

[SWS_Dcm_00443] ⌈ If Dcm_StartOfReception returns BUFREQ_OK, the further call to Dcm_CopyRxData shall copy the data from the buffer provided in info parameter) to the Dcm buffer and update the bufferSizePtr parameter with remaining free place in Dcm receive buffer after completion of this call. ⌋*()*

[SWS_Dcm_00996] ⌈ When the APIDcm_CopyRxData is invoked with SduLength from info equal to 0, the value BUFREQ_OK shall be returned and bufferSizePtr shall be filled with the remaining size of the Rx buffer. ⌋*()*

Note: The size of the Rx buffer is based on the buffer length, which is returned in the parameter RxBufferSizePtr of API Dcm_StartOfReception. **[SWS_Dcm_00342]** ⌈ After starting to copy the received data (see [SWS_Dcm_00443]), the Dcm module shall not access the receive buffer until it is notified by the service Dcm_TpRxIndication about the successful completion or unsuccessful termination of the reception. ⌋*()*

Note: Dcm_TpRxIndication is only expected when Dcm_StartOfReception succeeded

[SWS_Dcm_00831] ⌈ Dcm_CopyRxData shall be callable in interrupt context. ⌋*()*

7.3.4.2.3 Dcm_TpRxIndication

[SWS_Dcm_00344] ⌈ If Dcm_TpRxIndication is called with parameter Result different from E_OK, then the Dcm module shall not evaluate the buffer assigned to the I-PDU, which is referenced in parameter DcmRxPduId. ⌋*()*

Rationale for [SWS_Dcm_00344]: It is undefined which part of the buffer contains valid data in this case

[SWS_Dcm_00345] ⌈ Dcm_TpRxIndication shall be callable in interrupt context. ⌋*()*

7.3.4.3 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 [2]. This functional ”TesterPresent” will be received on a separate DcmRxPduId (UDS func DcmRxPduId)), which is belonging to the same DcmDslConnection as the physical request. A Dcm-internal receive buffer which is not configured explicitly, is used in this case. Due to that reason, the functional TesterPresent (and only functional TesterPresent without response) is handled in the following way:

[SWS_Dcm_00112] ⌈ When the PduR module calls Dcm_TpRxIndication with parameter Result=E_OK (see [SWS_Dcm_00093]) and if the request is a ”TesterPresent” command with ”suppressPosRspMsgIndicationBit” set to TRUE (SID equal to 0x3E, subfunction equal to 0x80), the DSL submodule shall reset the session timeout timer (S3Server). ⌋*()*

[SWS_Dcm_00113] ⌈ When the PduR module calls Dcm_TpRxIndication with parameter Result = E_OK (see [SWS_Dcm_00093]) and if the request is a ”TesterPresent” command with ”suppressPosRspMsgIndicationBit” set to TRUE (SID equal to 0x3E, subfunction equal to 0x80), the DSL submodule shall not forward this request to the DSD submodule for further interpretation. ⌋*()*

Rationale for [SWS_Dcm_00113]: Because of bypassing the functional ”TesterPresent” in the DSL submodule, the Dcm module is able to receive and process next physical requests without any delay.

[SWS_Dcm_01168] ⌈ The Dcm shall handle a tester present request as concurrent request only if it was received on a functional address with ”suppressPosRspMsgIndicationBit” set to TRUE. ⌋*()*

7.3.4.3.1 Dcm_CopyTxData

If the copied data is smaller than the length requested to transmit within the service PduR_DcmTransmit() the Dcm module will be requested by the service Dcm_CopyTxData to provide another data when the current copied data have been transmitted.

[SWS_Dcm_00346] ⌈ If the function Dcm_CopyTxData is called and the Dcm module successfully copied the data in the buffer provided in info parameter, then the function shall return BUFREQ_OK. ⌋*()*

[SWS_Dcm_00350] ⌈ Caveats of Dcm_CopyTxData:

- The value of parameter availableDataPtr 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.

⌋*()*

[SWS_Dcm_00832] ⌈ Dcm_CopyTxData shall be callable in interrupt context. ⌋*()*

7.3.4.3.2 Dcm_TpTxConfirmation

[SWS_Dcm_00352] ⌈ If the function Dcm_TpTxConfirmation is called, then the Dcm module shall unlock the transmit buffer. ⌋*()*

[SWS_Dcm_00353] ⌈ If the function Dcm_TpTxConfirmation is called, then the Dcm module shall stop error handling (Page buffer timeout, P2ServerMax/P2*ServerMax timeout). ⌋*()*

[SWS_Dcm_00354] ⌈ Dcm_TpTxConfirmation shall be callable in interrupt context (e.g. from a transmit interrupt) ⌋*()*

For transmission via FlexRay the following restriction has to be considered: Since the FlexRay Specification does not mandate the existence of a transmit interrupt, the exact meaning of this confirmation (i.e. "transfer into the FlexRay controller's send buffer" OR "transmission onto the FlexRay network") depends on the capabilities of the FlexRay communication controller and the configuration of the FlexRay Interface.

7.3.4.4 Forward responses from the DSD submodule to the PduR module

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

[SWS_Dcm_00115] ⌈ When the diagnostic response of a DcmDslMainConnection is ready, the DSL submodule shall trigger the transmission of the diagnostic response to the PduR module by calling PduR_DcmTransmit() using the corresponding DcmDslProtocolTxPduRef parameter as PduId. ⌋*()*

[SWS_Dcm_01072] ⌈ In case of PeriodicTransmission, the Dcm shall provide in the call to PduR_DcmTransmit() the full payload data and expect no call to Dcm_CopyTxData. ⌋*()*

[SWS_Dcm_01073] ⌈ In case of PeriodicTransmission, the Dcm will be called for periodic transmission with Dcm_TxConfirmation to indicate the transmission result. ⌋*()*

Responses are sent with the DcmTxPduId, which is linked in the Dcm module configuration to the DcmRxPduId, i.e. the ID the request was received with (see configuration parameter DcmDslProtocolTx) Within PduR_DcmTransmit() only the length information and, for generic connections, the addressing information, is given to the PduR module. After the Dcm module has called successfully PduR_DcmTransmit(), the PduR module will call Dcm_CopyTxData to request the Dcm module to provide the data to be transmitted and will call Dcm_TpTxConfirmation after the complete PDU has successfully been transmitted or an error occurred. see section 7.3.4.5 "Generic Connection Handling for further details on address information handling within generic connections".

[SWS_Dcm_00117] ⌈ If the DSL submodule receives a confirmation after the complete Dcm PDU has successfully been transmitted or an error occurred by a call of Dcm_TpTxConfirmation, then the DSL submodule shall forward this confirmation to the DSD submodule. ⌋*()*

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

Note: Dcm_TpTxConfirmation is only expected when PduR_DcmTransmit succeeded.

[SWS_Dcm_01166] ⌈ If the Multiplicity of DcmDslProtocolTx is set to "0" the Dcm shall process the received diagnostic request without sending a response. ⌋*()*

More descriptions of the APIs (prototype, input/output parameter) can be found in the interface description of the PduR module [9].

7.3.4.5 Generic Connection Handling

The Dcm shall be able to handle generic connections, identified by DcmPdus with MetaDataItems of type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16. These connections carry the actual tester address at run time. Please note that this address is not provided to the application. If the application needs to discern different testers, separate connections have to be created. Generic connections are supported for diagnostics over IP and FlexRay diagnostics, and CAN diagnostics using normal fixed or mixed 29 bit addressing formats according to ISO15765-2 [10]. Depending on the actual layout of the CAN IDs, generic connections could also be used for extended or normal and mixed 11 bit addressing formats. The Dcm is not aware of the actual addressing format used by CanTp. Several connections may reference the same DcmPdus.

[SWS_Dcm_CONSTR_6044] ⌈ Generic connections shall be consistent. This means that the MetaDataItems and the PduLength of all referenced PDUs of a DcmDslConnection (DcmDslProtocolRxPduRef, DcmDslProtocolTxPduRef, DcmDslPeriodicTxPduRef, DcmDslRoeTxPduRef)are identical. ⌋*()*

[SWS_Dcm_00848] ⌈ The source address of diagnostic requests received via a generic connection must be stored. It is provided in the MetaDataItem SOURCE_ADDRESS_16 provided via Dcm_StartOfReception. ⌋*()*

[SWS_Dcm_00849] Target address for generic connection transmission ⌈ If the Dcm is about to send a response, response on event, or periodic message for a generic connection request, the Dcm shall set TARGET_ADDRESS_16 to the value of the stored source address in the MetaDataPtr in the PduR_DcmTransmit(). ⌋*(SRS_Diag_04153)*

[SWS_Dcm_01429] ⌈ The source address of diagnostic requests received via a generic connection shall be provided in the parameter TesterSourceAddress to the application [SWS_Dcm_01339], [SWS_Dcm_01340], [SWS_Dcm_01341], [SWS_Dcm_01342], [SWS_Dcm_00692], [SWS_Dcm_00694], [SWS_Dcm_00340], [SWS_Dcm_00698]). ⌋*()*

[SWS_Dcm_01347] ⌈ The target address of diagnostic requests received via a generic connection can be provided in the MetaDataItem TARGET_ADDRESS_16 received via Dcm_StartOfReception(). In this case, the Dcm shall ignore physical requests where the target address is not equal to the configured ECU address DcmDspProtocolEcuAddr. ⌋*(SRS_Diag_04153)*

[SWS_Dcm_01348] ⌈ The source address of the response transmitted via generic connections can be read from the configuration parameter DcmDspProtocolEcuAddr. It shall be provided to PduR_DcmTransmit() in the MetaDataItem SOURCE_ADDRESS_16, if that is configured for the transmit PDU. ⌋*(SRS_Diag_04153)*

Note: If different source addresses are required for certain transmitted diagnostic messages of the same DcmDslProtocolRow, the MetaDataItem SOURCE_ADDRESS_16 can be omitted from the PDUs, and the address can then be configured in the lower layers. The same is possible for physical requests, where the TARGET_ADDRESS_16 can be omitted from the PDUs.

7.3.4.6 Guarantee timing to tester by sending busy responses

[SWS_Dcm_00024] ⌈ If the Application (or the DSP submodule) is able to perform a requested diagnostic task, but needs additional time to finish the task and prepare the response, then the DSL submodule shall send a negative response with NRC 0x78 (Response pending) when reaching the response time (DcmDspSessionP2ServerMax - DcmTimStrP2ServerAdjust respectively DcmDspSessionP2StarServerMax - DcmTimStrP2StarServerAdjust). ⌋*(SRS_Diag_04016)*

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

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

Rationale for [SWS_Dcm_00119]: This is needed in order to avoid overwriting the ongoing processing of requests, e.g. the application already prepared response contents in the diagnostic buffer. The number of negative responses with NRC 0x78 (Response pending) for one diagnostic request is limited by the configuration parameter DcmDslDiagRespMaxNumRespPend. This avoids deadlocks in the Application.

7.3.4.7 Support of periodic transmission

The UDS service ReadDataByPeriodicIdentifier (0x2A) allows the tester to request the periodic transmission of data record values from the ECU identified by one or more periodicDataIdentifiers.

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

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

[SWS_Dcm_00123] ⌈ Periodic transmission communication shall only take place in Full Communication Mode. ⌋*()*

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

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

[SWS_Dcm_00126] ⌈ Periodic transmission events shall not activate the Full Communication Mode. ⌋*()*

7.3.4.8 Support of ROE transmission

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

[SWS_Dcm_00595] ⌈ The ROE functionality is enabled only if the container DcmDslResponseOnEvent exists. ⌋*()*

7.3.4.8.1 ResponseOnEvent StateChar

[SWS_Dcm_00871] ⌈ The Dcm shall support several RoeEvents. Each RoeEvent can have the states "ROE cleared", "ROE stopped" and "ROE started". The transitions from state to state are described in the following section. The Labels in Figure 7.4 represents the numbers of the sections. ⌋*()*

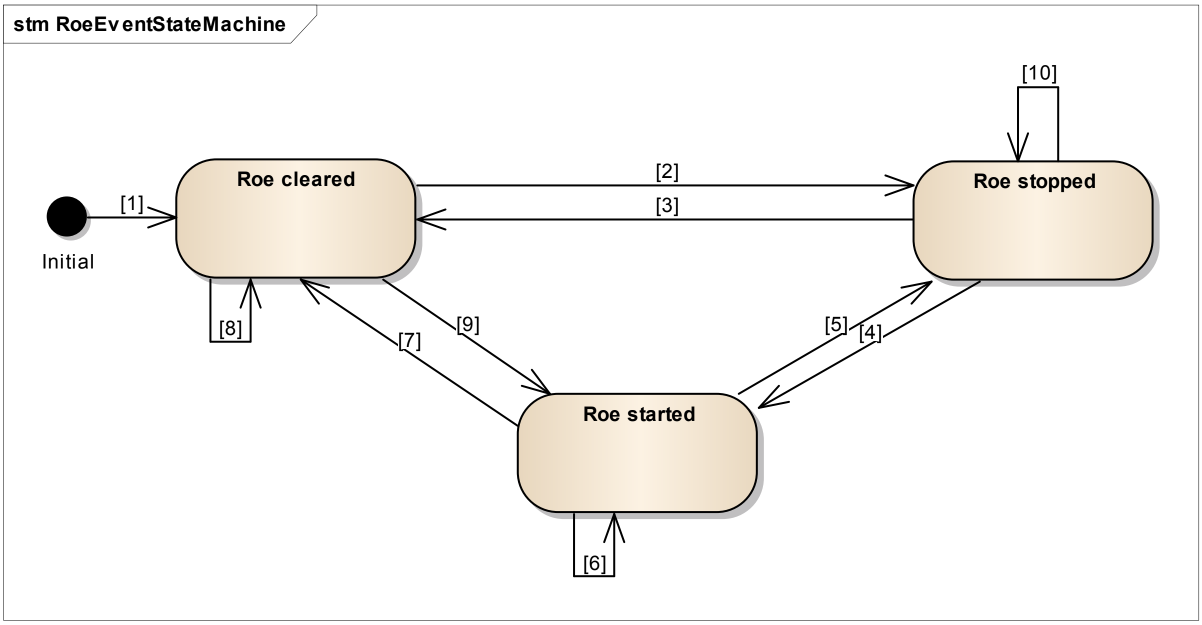


Figure 7.4: RoeEvent State Chart

7.3.4.8.1.1 Initializing Dcm (1)

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

7.3.4.8.1.2 Transition from 'ROE cleared' to 'ROE stopped' (2)

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

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

Note: If an Event is initialized once with StorageState set to 'StoreEvent', it will stay initialized until it is cleared by a ClearResponseOnEvent request (see also [SWS_Dcm_00897]).

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

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

7.3.4.8.1.3 Transition from 'ROE stopped' to 'ROE cleared' (3)

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

7.3.4.8.1.4 Transition from 'ROE stopped' to 'ROE started' (4)

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

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

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

7.3.4.8.1.5 Transition from 'ROE started' to 'ROE stopped' (5)

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

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

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

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

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

7.3.4.8.1.6 Transition from 'ROE started' to 'ROE started' (6)

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

7.3.4.8.1.7 Transition from 'ROE started' to 'ROE cleared' (7)

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

7.3.4.8.1.8 Transition from 'ROE cleared' to 'ROE cleared' (8)

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

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

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

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

7.3.4.8.1.9 Transition from 'ROE cleared' to 'ROE started' (9)

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

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

7.3.4.8.1.10 Transition from 'ROE stopped' to 'ROE stopped' (10)

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

[SWS_Dcm_00953] ⌈ If a ROE request is received with the sub-function OnDTCStatusChange and the RoeEvent is already 'ROE stopped' the RoeEvent for OnDTCStatusChange shall stay in 'ROE stopped' state but the event logic shall be updated with the newly received DTCStatusMask. ⌋*()*

7.3.4.8.2 ROE sub-functions

[SWS_Dcm_00892] ⌈ The Dcm shall support all ROE sub-functions marked as supported in Table 7.3. ⌋*()*

Sub function ID	Sub-function name	Kind of sub-function	ServiceTo RespondTo	Support status
0x00/0x40	stopResponseOnEvent	Control		Supported
0x01/0x41	onDTCStatusChange	Setup	0x19, 0x0E	Supported
0x02/0x42	onTimerInterrupt	Setup		Not supported
0x03/0x43	onChangeOfDataIdentifier	Setup	0x22	Supported
0x04	reportActivatedEvents	Control		Supported

Sub function ID	Sub-function name	Kind of sub-function	ServiceTo RespondTo	Support status
0x05/0x45	StartResponseOnEvent	Control		Supported
0x06/0x46	clearResponseOnEvent	Control		Supported
0x07/0x47	onComparisonOfValues	Setup		Not supported
Other	OEM Specific	Setup		Not supported

Table 7.3: Supported sub function of Response on Event (0x86)

Note: If a user wants to support a sub-function with StorageState bit set, then it has to be explicitly configured in the DSD. The Dcm will not mask the StorageState bit internally.

[SWS_Dcm_00893] ⌈ For each setup sub function the Dcm shall only support the one fixed ServiceToRespondTo. The supported ServiceToRespondTo is listed in Table 7.3. ⌋*()*

7.3.4.8.3 EventWindowTime and StorageState

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

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

[SWS_Dcm_00894] ⌈ he Dcm shall support in general the EventWindowTimes defined in Table 7.4. ⌋*()*

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

Table 7.4: Supported ROE EventWindowTime

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

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

[SWS_Dcm_01076] ⌈ If the Roe request has a storageState equal to storeEvent and contains an EventWindowTime that is not infinite, the Dcm shall reject the request with a negative response with the NRC 0x31 (RequestOutOfRange). ⌋*()*

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

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

- The RoeEvent was started in default session
- The StartResponseOnEventRequest has a storageState set to 'StoreEvent'
- The setup request has the EventWindowTime inifinity and the storageState was set to 'StoreEvent'.

⌋ *()*

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

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

⌋ *()*

[SWS_Dcm_00900] ⌈ If a RoeEvent set up with the EventWindowTime set to CurrentAndFollowingCycle is started in default session, the EventWindowTime shall end at the end of the next power cycle or with a clearResponseOnEvent/stopResponseOnEvent request. ⌋ *()*

[SWS_Dcm_00901] ⌈ If a RoeEvent set up with the EventWindowTime set to CurrentCycle is started in default session, the EventWindowTime shall end at the end of the current power cycle or with a clearResponseOnEvent/stopResponseOnEvent. ⌋ *()*

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

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

⌋ *()*

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

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

7.3.4.8.4 Pre-configuration of ResponseOnEvent

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

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

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

[SWS_Dcm_00954] Pre-configuraton of ROE events ⌈ If DcmDspRoeInitialEventStatus is set to DCM_ROE_STOPPED, the Dcm shall behave according RoeEvent set-up:

- StorageState set to "StoreEvent"
- EventWindowTime set to "infinity"
- DTCStatusMask set to value configured in DcmDspRoeDTCStatusMask in case of onDTCStatusChange and
- ARTechTermRefDID set to the value given with DcmDspRoeDidRef in case of onChangeOfDataIdentifier

⌋*()*

[SWS_Dcm_01323] ⌈ Likewise, when responding to the reportActivatedEvents (0x04) subfunction of the ResponseOnEvent (0x86) service, preconfigured events shall have the storageState bit set within the corresponding eventTypeOfActiveEvent byte. ⌋*()*

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

7.3.4.8.5 Handling of event-trigger

7.3.4.8.5.1 ROE event-trigger onDTCStatusChange (0x01)

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

[SWS_Dcm_00912] ⌈ If the state of one RoeEvent that is configured for onDTCStatusChange changes to 'ROE started' the Dcm shall evaluate the callback Dcm_DemTriggerOnDTCStatus. ⌋*()*

[SWS_Dcm_00913] ⌈ If the state of the RoeEvent, configured to OnDTCStatusChange, leaves 'ROE started' the Dcm shall ignore the callback Dcm_DemTriggerOnDTCStatus. ⌋*()*

[SWS_Dcm_01410] ⌈ In case a request to clear the EventMemory is processed, the Dcm shall ignore the callback Dcm_DemTriggerOnDTCStatus. ⌋*()*

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

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

[SWS_Dcm_CONSTR_6054] Existence of DTCStatusMask ⌈ DcmDspRoeDTCStatusMask shall be present if DcmDspRoeInitialEventStatus is set to DCM_ROE_STOPPED. ⌋*()*

7.3.4.8.5.2 ROE event-trigger onChangeOfDataIdentifier (0x03)

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

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

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

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

7.3.4.8.6 Trigger a ServiceToRespondTo

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

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

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

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

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

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

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

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

7.3.4.8.7 Send a ServiceToRespondTo

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

[SWS_Dcm_00131] ⌈ The configured protocol buffer shall be used for transmission of the ROE messages (as the reception shall use a separate protocol, a separate buffer needs to be used for reception). ⌋*()*

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

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

[SWS_Dcm_00928] ⌈ If the referred Protocol for ResponseOnEvent (DcmDslROEConnectionRef) is configured for TYPE2 the Dcm shall send the ServiceToRespondTo to the configured TxPduID (see configuration parameter DcmDslRoeTxPduRef). ⌋*()*

[SWS_Dcm_00132] ⌈ The content of the pMsgContext pointer (ROE message) shall be copied into the buffer. ⌋*()*

[SWS_Dcm_00133] ⌈ ROE communication shall only be performed in Full Communication Mode. The Dcm shall check the communication mode of the DcmDslProtocolComMChannelRef in the DcmDslMainConnection. ⌋*()*

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

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

[SWS_Dcm_00136] ⌈ ROE events requested by the Application shall not activate the Full Communication Mode. ⌋*()*

[SWS_Dcm_CONSTR_6025] Reference to DcmDslResponseOnEvent connection ⌈ Only one DcmDslROEConnectionRef shall reference DcmDslResponseOnEvent connection. ⌋*()*

[SWS_Dcm_CONSTR_6056] Dependency for DcmDslProtocolTransType ⌈ DcmDslProtocolTransType shall be only present if the Dcm_ProtocolType is configured to DCM_ROE_ON_CAN or DCM_ROE_ON_FLEXRAY or DCM_ROE_ON_IP. ⌋*()*

7.3.4.8.7.1 Roe transmission cycle

[SWS_Dcm_00601] ⌈ The Dcm module shall respect a minimum time between two (2) consecutive Roe transmissions (see configuration parameter DcmDspRoeInterMessageTime) ⌋*()*

7.3.4.8.8 ResponseOnEvent in multiple client environments

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

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

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

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

7.3.4.9 Support of segmented response (paged-buffer)

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

[SWS_Dcm_CONSTR_6055] Dependency for DcmDslProtocolMaximumResponseSize ⌈ DcmDslProtocolMaximumResponseSize shall be only present if DcmPagedBufferEnabled is set to TRUE. ⌋*()*

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

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

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

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

[SWS_Dcm_01186] ⌈ The Dcm shall provide the correct amount of Data requested by the TP or return BUFREQ_E_BUSY in case the requested amount of data is not available. ⌋*(SRS_Diag_04147)*

Note: In case the requested amount of data is not available, the Dcm should fill up the paged buffer immediately.

7.3.4.10 Support of ResponsePending response triggered by the Application

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

When the Dcm module calls an operation and gets an error status DCM_E_FORCE_RCRRP, the DSL submodule will trigger the transmission of a negative response with NRC 0x78 (Response pending). This response needs to be sent from a separate buffer, in order to avoid overwriting the ongoing processing of the request.

7.3.4.11 Manage security level

[SWS_Dcm_00020] ⌈ The DSL submodule shall save the level of the current active security level. ⌋*(SRS_Diag_04005)*

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

- get the current active security level: Dcm_GetSecurityLevel
- set a new security level: DslInternal_SetSecurityLevel()

[SWS_Dcm_00033] ⌈ During Dcm initialization the security level is set to the value 0x00 (DCM_SEC_LEV_LOCKED). ⌋*(SRS_BSW_00101, SRS_Diag_04005)*

[SWS_Dcm_00139] ⌈ The DSL shall reset the security level to the value 0x00 (i.e. the security is enabled) under one of the following conditions: - if a transition from any diagnostic session other than the defaultSession to another session other than the defaultSession (including the currently active diagnostic session) is performed or - if a transition from any diagnostic session other than the defaultSession to the defaultSession (DslInternal_SetSecurityLevel()) (initiated by UDS Service DiagnosticSessionControl (0x10) or S3Server timeout) is performed. ⌋*()*

Only one security level can be active at a time.

[SWS_Dcm_01329] ⌈ On every security level change the Dcm shall update the ModeDeclarationGroup DcmSecurityAccess with the new security level. ⌋*()*

7.3.4.11.1 Initialization sequence

[SWS_Dcm_01154] ⌈ At initialization, for each DcmDspSecurityRow entry for which the DcmDspSecurityAttemptCounterEnabled configuration parameter is set to TRUE, the corresponding Xxx_GetSecurityAttemptCounter shall be called in order to get the value of the AttemptCounter for each of these DcmDspSecurityRow entries. ⌋*()*

[SWS_Dcm_01156] ⌈ If Xxx_GetSecurityAttemptCounter has returned E_NOT_OK the attempt counter shall be set to the value configured in DcmDspSecurityNumAttDelay of the according SecurityLevel. ⌋*()*

[SWS_Dcm_01351] ⌈ If any Xxx_GetSecurityAttemptCounter operation returns a DCM_E_PENDING value, the Dcm shall interrupt calling the

Xxx_GetSecurityAttemptCounter() in order to resume this chain of calls within the next Dcm_MainFunction() cycle. ⌋*()*

Note: this may be the case when these values are stored within some specific non-volatile memory.

[SWS_Dcm_CONSTR_6076] Dependency for DcmDspSecurityGetAttemptCounterFnc ⌈ DcmDspSecurityGetAttemptCounterFnc shall be present only if DcmDspSecurityUsePort is set to USE_ASYNCH_FNC and DcmDspSecurityAttemptCounterEnabled is set to TRUE. ⌋*()*

[SWS_Dcm_01352] ⌈ If the delay after the first call of the Dcm_MainFunction() which is configured in DcmDspSecurityMaxAttemptCounterReadoutTime has been reached and all the Xxx_GetSecurityAttemptCounter have not been called yet (i.e. one operation has returned a DCM_E_PENDING status in the previous Dcm_MainFunction() cycle), the pending operation shall be cancelled by a call with the OpStatus set to DCM_CANCEL. ⌋*()*

[SWS_Dcm_01353] ⌈ In the conditions of [SWS_Dcm_01352], the AttemptCounters of remaining security levels (which have not been obtained via the calls to their Xxx_GetSecurityAttemptCounter) shall be initialized with the value configured in DcmDspSecurityNumAttDelay of the according SecurityLevel. ⌋*()*

[SWS_Dcm_01354] ⌈ While not all Xxx_GetSecurityAttemptCounter operations have returned a final status and the operation chain has not been cancelled, the conditionsNotCorrect (0x22) NRC shall be returned to any SecurityAccess (0x27) requestSeed subfunction request. ⌋*()*

[SWS_Dcm_01355] ⌈ Once all the AttemptCounter values have been successfully or unsuccessfully retrieved (all the Xxx_GetSecurityAttemptCounter() operations have been executed and have returned a final, non-PENDING error value or the operation chain has been cancelled), if at least one of the restored AttemptCounter values is greater than or equal to the DcmDspSecurityNumAttDelay configured for its corresponding DcmDspSecurityRow, the Dcm shall start the SecurityDelayTimer with the higher value of DcmDspSecurityDelayTimeOnBoot / DcmDspSecurityDelayTime of the according DcmDspSecurityRow. ⌋*()*

[SWS_Dcm_01356] ⌈ A timer (DcmDspSecurityDelayTime, DcmDspSecurityMaxAttemptCounterReadoutTime) which is configured with 0 shall be considered to have timed out instantaneously when it is started, i.e. shall have no delay effect. ⌋*()*

[SWS_Dcm_CONSTR_6074] Dependency for DcmDspSecurityMaxAttemptCounterReadoutTime ⌈ DcmDspSecurityMaxAttemptCounterReadoutTime shall be a multiple and at minimum equal to DcmTaskTime. ⌋*()*

7.3.4.11.2 AttemptCounter update

[SWS_Dcm_01357] ⌈ A successful sendKey subfunction request or an expired SecurityDelayTimer shall reset that security level's specific AttemptCounter. ⌋*()*

[SWS_Dcm_01155] ⌈ The Dcm shall call Xxx_SetSecurityAttemptCounter() (in case the configuration parameter DcmDspSecurityAttemptCounterEnabled for the according DcmDspSecurityRow is set to TRUE) when the Dcm has changed the attempt counter to inform the application about the counter change. ⌋*()*

[SWS_Dcm_CONSTR_6078] Dependency for DcmDspSecuritySetAttemptCounterFnc ⌈ DcmDspSecuritySetAttemptCounterFnc shall be present only if DcmDspSecurityUsePort is set to USE_ASYNCH_FNC and the DcmDspSecurityAttemptCounterEnabled set to TRUE. ⌋*()*

7.3.4.12 Manage session state

[SWS_Dcm_00022] ⌈ The DSL submodule shall save the state of the current active session. ⌋*(SRS_Diag_04006)*

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

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

[SWS_Dcm_00034] ⌈ During Dcm initialization, the session state is set to the value 0x01 ("DefaultSession"). ⌋*(SRS_BSW_00101)*

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

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

7.3.4.13 Keep track of active non-default sessions

[SWS_Dcm_00140] ⌈ Whenever a non-default session is active and when the session timeout (S3Server) is reached without receiving any diagnostic request, the DSL submodule shall reset to the default session state ("DefaultSession", 0x01) and invoke the the mode switch of the ModeDeclarationGroupPrototype DcmDiagnosticSessionControl by calling SchM_Switch_<bsnp>_DcmDiagnosticSessionControl(RTE_MODE_DcmDiagnosticSessionControl_DEFAULT_SESSION) . ⌋*()*

Note: <bsnp> is the BSW Scheduler Name Prefix

The start / stop of S3Server timeout timer is processed as follows:

[SWS_Dcm_00141] ⌈

Subsequent 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)

Subsequent 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. ⌋*()*

7.3.4.14 Allow to modify timings

[SWS_Dcm_00027] ⌈ The Dcm module shall handle the following protocol timing parameters in compliance with [4]: P2ServerMin, P2ServerMax, P2*ServerMin, P2*ServerMax, S3Server ⌋*(SRS_Diag_04015)*

[SWS_Dcm_00143] ⌈ P2min / P2*min and S3Server shall be set to defined values: P2min = 0ms, P2*min = 0ms, S3Server = 5s. ⌋*(SRS_Diag_04015)*

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.

7.3.4.15 Handling of different diagnostic protocols

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

[SWS_Dcm_01365] ⌈ If the API Dcm_GetActiveProtocol is called, the Dcm shall return the active UDS protocol as Dcm_ProtocolType in the ActiveProtocol parameter. ⌋*(SRS_Diag_04163)*

[SWS_Dcm_01366] ⌈ If only an OBD protocol or no protocol is started and the API Dcm_GetActiveProtocol is called, the Dcm shall return with ActiveProtocol parameter set to DCM_NO_ACTIVE_PROTOCOL. ⌋*(SRS_Diag_04163)*

Note: The Dcm_GetActiveProtocol API doesn't supply information about running OBD protocols.

7.3.4.15.1 Different service tables

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

[SWS_Dcm_00035] ⌈ With every protocol initialization, the DSL submodule sets a link to the corresponding service table (see configuration parameter DcmDslProtocolSIDTable). ⌋*(SRS_BSW_00101)*

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

7.3.4.15.2 Prioritization of protocol

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

[SWS_Dcm_00015] ⌈ A protocol with higher priority is allowed to preempt the already running protocol. ⌋*(SRS_Diag_04021)*

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

7.3.4.15.3 Preemption of protocol

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

[SWS_Dcm_01144] ⌈ Protocol preemption can't be activated with a concurrent Tester-Present of a higher priority protocol (see also [SWS_Dcm_01146]). ⌋*()*

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

[SWS_Dcm_00460] ⌈ When PduR_DcmCancelTransmit() returns E_NOT_OK, the Dcm module shall assume that the ongoing transmission cannot be cancelled and shall not retry to cancel the transmit request. The current protocol shall be stopped and the new one started. ⌋*()*

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

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

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

[SWS_Dcm_00576] ⌈ When PduR_DcmCancelReceive () returns E_NOT_OK, the Dcm module shall assume that the ongoing reception cannot be cancelled and shall not retry to cancel the receiverequest. The current protocol shall be stopped and the new one started. ⌋*()*

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

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

[SWS_Dcm_00727] ⌈ If a diagnostic request is already running and a second request (ClientB) can not be processed (e.g.due to priority assessment), the response behaviour depends on the configuration option parameter DcmDslDiagRespOnSecondDeclinedRequest (see [SWS_Dcm_00914]_Conf). If this configuration parameter is TRUE, a negative response with NRC 0x21 (BusyRepeatRequest) shall be issued for the second request (see [SWS_Dcm_00788] and [SWS_Dcm_00789]). If the configuration parameter is FALSE, no response shall be issued (see [SWS_Dcm_00790]). ⌋*()*

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

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

7.3.4.15.4 Parallel OBD and UDS protocol processing

[SWS_Dcm_01367] ⌈ The Dcm shall process incoming OBD requests in parallel to a running UDS request. In this case the protocol priority check according to [SWS_Dcm_00015] is skipped and no protocol pre-emption is done. ⌋*(SRS_Diag_04163)*

With the container DcmDslProtocolRow, the Dcm configuration supports multiple protocols. Each protocol has a configured DcmDemClientRef defining the Dem client interacting with the Dem. This client Id allows the Dem to distinguish between concurrent calls of the Dcm of the same function or set of functions to process a certain request.

[SWS_Dcm_01369] ⌈ While processing a diagnostic request received from a given protocol, the Dcm shall determine the DcmDemClientRef of the DcmDslProtocolRow of the processed protocol. The Dcm shall use this value in all Dem API calls that have a ClientId as parameter. ⌋*(SRS_Diag_04162)*

[SWS_Dcm_01370] ⌈ The Dcm shall internally serialise all asynchronous C/S interface calls by the same port interface between the OBD and UDS protocol processors and return a pending to the re-entrant caller. ⌋*(SRS_Diag_04162)*

[SWS_Dcm_01371] ⌈ If the Dcm received an OBD request and the Dcm is processing a diagnostic service in a non-default session, the Dcm shall cancel the running UDS request, make a transition into default session and process the OBD request. ⌋*(SRS_Diag_04162)*

[SWS_Dcm_01372] ⌈ If the Dcm processes an OBD request and the Dcm is receiving an UDS diagnostic request to change in a non-default session, the Dcm shall delay the UDS request until the OBD service is finished according to [SWS_Dcm_01371]. After the OBD service is finished, the Dcm shall make a transition into the requested non-default session. ⌋*(SRS_Diag_04162)*

7.3.4.15.5 Detection of protocol start

[SWS_Dcm_00036] ⌈ With first request of a diagnostic protocol, the DSL submodule shall call all configured Xxx_StartProtocol() functions (see configuration parameter DcmDslCallbackDCMRequestService). ⌋*(SRS_BSW_00101)*

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

[SWS_Dcm_00144] ⌈ After all Xxx_StartProtocol() functions have returned E_OK (meaning all components have allowed the start of the protocol), the default timing parameters are loaded from the default session configuration (see configuration parameter DcmDspSessionRow). ⌋*(SRS_Diag_04015)*

[SWS_Dcm_00145] ⌈ After all Xxx_StartProtocol() functions have returned E_OK (meaning all components have allowed the start of the protocol), the service table is set (see configuration parameter DcmDslProtocolSIDTable). ⌋*()*

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

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

Note: <bsnp> is the BSW Scheduler Name Prefix

[SWS_Dcm_00674] ⌈ If Xxx_StartProtocol() doesn't return E_OK, the Dcm shall return NRC 0x22. ⌋*()*

7.3.4.15.6 Protocol stop

A protocol stop can appear only in case of protocol preemption (see chapter 7.3.4.15.3 Preemption of protocol).

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

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

[SWS_Dcm_01190] ⌈ If Xxx_StopProtocol() doesn't return E_OK, the Dcm shall return NRC 0x22. ⌋*()*

7.3.4.16 Manage resources

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

- It is allowed to use and allocate only one diagnostic buffer in the Dcm module. This buffer is then used for processing the diagnostic requests and responses.

- Output of NRC 0x78 (Response pending) responses is done with a separate buffer.
- paged-buffer handling (see [SWS_Dcm_00028]).

7.3.4.17 Communication Mode Handling

Communication Mode Handling is an interface between Dcm and ComM. The ComM informs the Dcm about the current communication state of a channel. The Dcm is calling the ComM about active Diagnostic which shall prevent an Ecu shutdown/sleep.

The status ActiveDiagnostic shows if diagnostic requests shall keep the ECU awake (ActiveDiagnostic ==’DCM_COMM_ACTIVE’) or if diagnostic requests shall not prevent an Ecu shutdown/sleep (ActiveDiagnostic ==’DCM_COMM_NOT_ACTIVE’). Application can change the status ActiveDiagnostic regarding to system conditions.

[SWS_Dcm_CONSTR_6027] ⌈ The application will inform the Dcm by calling Xxx_SetActiveDiagnostic() about the ActiveDiagnostic status. ⌋*()*

[SWS_Dcm_01069] ⌈ After Dcm_Init, the Dcm shall set ActiveDiagnostic to ’DCM_COMM_ACTIVE’. ⌋*()*

[SWS_Dcm_01070] ⌈ If Xxx_SetActiveDiagnostic() is called with ’false’ the Dcm set ActiveDiagnostic to ’DCM_COMM_NOT_ACTIVE’. ⌋*()*

[SWS_Dcm_01071] ⌈ If Xxx_SetActiveDiagnostic() is called with ’true’ the Dcm set ActiveDiagnostic to ’DCM_COMM_ACTIVE’. ⌋*()*

[SWS_Dcm_01142] ⌈ The Dcm shall wait the Full Communication mode indication from the ComM (call to Dcm_ComM_FullComModeEntered) before initiating the transmission of the diagnostic answer. The time to wait should be no longer than the P2ServerMax calculated from the moment the request was received. ⌋*()*

[SWS_Dcm_01143] ⌈ If the Dcm fails to confirm a response pending transmission (DCM_E_FORCE_RCRRP) due to [SWS_Dcm_01142], the Dcm shall trigger the Det error DCM_E_FORCE_RCRRP_IN_SILENT_COMM. ⌋*()*

Note : On the reception side a silent communication mode can lead to the lost of the request in case of segmented transmission.

7.3.4.17.1 No Communication

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

[SWS_Dcm_00148] ⌈ Dcm_ComM_NoComModeEntered shall disable all kinds of transmissions (receive and transmit) of communication. This means that the message reception and also the message transmission shall be off. ⌋*()*

[SWS_Dcm_00149] ⌈ Dcm_ComM_NoComModeEntered shall disable the ResponseOnEvent transmissions. ⌋*()*

[SWS_Dcm_00150] ⌈ Dcm_ComM_NoComModeEntered shall disable the periodicId transmissions (ReadDataByPeriodicIdentifier). ⌋*()*

[SWS_Dcm_00151] ⌈ Dcm_ComM_NoComModeEntered shall disable normal transmissions. ⌋*()*

[SWS_Dcm_00152] ⌈ After Dcm_ComM_NoComModeEntered has been called, the Dcm module shall not call the function PduR_DcmTransmit(). ⌋*()*

[SWS_Dcm_01324] ⌈ In case Dcm_ComM_NoComModeEntered is called with a NetworkId for a ComM channel not referenced within the Dcm (see configuration parameter DcmDslProtocolComMChannelRef), the Dcm shall return without performing any further action. ⌋*()*

7.3.4.17.2 Silent Communication

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

[SWS_Dcm_00153] ⌈ Dcm_ComM_SilentComModeEntered shall disable all transmission. This means that the message transmission shall be off. ⌋*()*

[SWS_Dcm_00154] ⌈ Dcm_ComM_SilentComModeEntered shall disable the ResponseOnEvent transmissions. ⌋*()*

[SWS_Dcm_00155] ⌈ Dcm_ComM_SilentComModeEntered shall disable the periodicId transmissions (ReadDataByPeriodicIdentifier) shall be disabled. ⌋*()*

[SWS_Dcm_00156] ⌈ Dcm_ComM_SilentComModeEntered shall disable the normal transmissions. ⌋*()*

[SWS_Dcm_01325] ⌈ In case Dcm_ComM_SilentComModeEntered is called with a NetworkId for a ComM channel not referenced within the Dcm (see configuration parameter DcmDslProtocolComMChannelRef), the Dcm shall return without performing any further action. ⌋*()*

7.3.4.17.3 Full Communication

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

[SWS_Dcm_00157] ⌈ Dcm_ComM_FullComModeEntered shall enable all kind of communication. This means that the message reception and also the message transmission shall be on. ⌋*()*

[SWS_Dcm_00159] ⌈ Dcm_ComM_FullComModeEntered shall enable the ResponseOnEvent transmissions. ⌋*()*

[SWS_Dcm_00160] ⌈ Dcm_ComM_FullComModeEntered shall enable the periodicId transmissions (ReadDataByPeriodicIdentifier). ⌋*()*

[SWS_Dcm_00161] ⌈ Dcm_ComM_FullComModeEntered shall enable the normal transmissions. ⌋*()*

[SWS_Dcm_00162] ⌈ After Dcm_ComM_FullComModeEntered has been called, the Dcm shall handle the functions DslInternal_ResponseOnOneDataByPeriodicId() or DslInternal_ResponseOnOneEvent() without restrictions. ⌋*()*

[SWS_Dcm_01326] ⌈ In case Dcm_ComM_FullComModeEntered is called with a NetworkId for a ComM channel not referenced within the Dcm (see configuration parameter DcmDslProtocolComMChannelRef), the Dcm shall return without performing any further action. ⌋*()*

7.3.4.17.4 Diagnostic Activation State

The Dcm notifies the ComM module about the internal diagnostic state for all networks. There are two options for the diagnostic state on a network. In 'active' diagnostic state, the Dcm is processing one or more diagnostic requests from this network or the Dcm is in a non-default session. In 'inactive' diagnostic state, the Dcm is in default session and is not processing a diagnostic request on that network.

When a network has no communication in progress, the Dcm will set the diagnostic activation state to 'inactive'. When there is a diagnostic communication on a network the Dcm sets the diagnostic state to 'active'. In any non-default session, the diagnostic state remains in state 'active'. The communication state can also be controlled by the API Xxx_SetActiveDiagnostic according to [SWS_Dcm_01070] and [SWS_Dcm_01071].

[SWS_Dcm_01373] ⌈ The Dcm shall go into 'active' diagnostic state on a network, if a diagnostic request is received on a network or the diagnostic session is changed to any non-default session. ⌋*(SRS_Diag_04006)*

[SWS_Dcm_01374] ⌈ The Dcm shall go into 'inactive' diagnostic state on a network when the current diagnostic request processing is finished and the Dcm is not processing a diagnostic request of another protocol on this network and if the Dcm is in default session. ⌋*(SRS_Diag_04006)*

[SWS_Dcm_01375] ⌈ The Dcm shall go into 'inactive' diagnostic state on all networks if a S3Server timeout occurs and the Dcm makes a transition into default session. ⌋*(SRS_Diag_04006)*

[SWS_Dcm_01376] ⌈ If ActiveDiagnostic is 'DCM_COMM_ACTIVE' and the Dcm is doing a transition into 'active' diagnostic state of a diagnostic protocol, the Dcm shall call ComM_DCM_ActiveDiagnostic(NetworkId), with the networkId associated to the received Pdu (see DcmDslProtocolComMChannelRef), with every request, to inform the ComM module about the need to stay in Full Communication Mode. ⌋*(SRS_Diag_04006)*

[SWS_Dcm_01377] ⌈ Upon a diagnostic state transition into 'inactive', the Dcm shall notify the ComM module about an inactive diagnostic state on a network by calling ComM_DCM_InactiveDiagnostic(NetworkId), with the networkId associated to the received Pdu (see DcmDslProtocolComMChannelRef). ⌋*(SRS_Diag_04006)*

[SWS_Dcm_01378] ⌈ The definition of a finished diagnostic request according to [SWS_Dcm_01374], shall be as follows:

- the Dcm has sent a positive or negative response unequal to NRC 0x78 by receiving the Dcm_TpTxConfirmation connected to the response given by the DSL submodule
- the Dcm has processed the service with SPRMIB=true and the positive response was suppressed
- in case of functional addressing, the Dcm has processed the service and the negative response was suppressed.

⌋*(SRS_Diag_04006)*

7.4 Diagnostic Service Dispatcher (DSD)

7.4.1 Introduction

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

[SWS_Dcm_00178] ⌈ The DSD submodule shall only process valid requests and shall reject invalid ones. ⌋*()*

7.4.2 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

7.4.2.1 Receive a request message and transmit a positive response message

This is the standard use case of normal communication ("ping-pong"). The server receives a diagnostic request message. The DSD submodule ensures the validity of the request message. In this use case, the request is valid and the response will be positive. The request will be forwarded to the appropriate data processor in the DSP submodule. When the data processor has finished all actions of data processing, it triggers the transmission of the response message by the DSD submodule.

If the data processor processes the service immediately as part of the request indication function, the data processor can trigger the transmission inside this indication function ("synchronous"). If the processing takes a longer time (e. g. waiting on EEPROM driver), the data processor defers some processing ("asynchronous"). The response pending mechanism is covered by the DSL submodule. The data processor triggers the transmission explicitly, but from within the data processor's context.

As soon as a request message is received, the corresponding DcmPduId is blocked by the DSL submodule (see [SWS_Dcm_00241]). During the processing of this request, no other request of the same protocol type (e.g. an enhanced session can be ended by a OBD session) can be received, until the corresponding response message is sent and the DcmPduId is released again.

7.4.2.2 Receive a request message and suppress a positive response

This is a sub-use-case of the previous one. Within the UDS protocol it is possible to suppress the positive response by setting a special bit in the request message (see [SWS_Dcm_00200]). This special suppression handling is completely performed within the DSD submodule.

7.4.2.3 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, 0x31, 0x7E and 0x7F (see [SWS_Dcm_00001]).

7.4.2.4 Receive a request message and transmit a negative response message

There are a many different reasons why a request message is rejected and a negative response is to be sent. If a diagnostic request is not valid or if a request may not be executed in the current session, the DSD submodule will reject the processing and a negative response will be returned.

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

In case of a request composed of several parameters (e.g. a UDS Service ReadDataByIdentifier (0x22) request with more than one identifier to read), each parameter is treated separately. And each of these parameters can return an error. This kind of request returns a positive response if at least one of the parameters was processed successfully.

[SWS_Dcm_00827] ⌈ The DSD sub-module shall check the received diagnostic request in the order given by ISO14229-1 [2]. If one of the computations failed the Dcm shall stop the execution of the NRC check sequence then stop or do not start the execution of the received diagnostic request and finally transmit the NRC for which the computation failed. ⌋*()*

7.4.2.5 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 [2]). 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 2 = UUDT message on a separate DcmTxPduId.

- ResponseOnEvent (0x86). This service requests a server to start or stop transmission of responses on a specified event.

  Type 1 = USDT messages on the DcmTxPduId already used for normal diagnostic responses,

  Type 2 = USDT messages on separate DcmTxPduId.

  For Type 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.

7.4.2.6 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.

7.4.3 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 or external modules outside the Dcm
- keeps track of request processing (Return the status on <Module>_<DiagnosticService>() and <Module>_<DiagnosticService>_<SubService>() APIs call or "Service Interpreter calls")
- transmits the response of the Application to the DSL submodule (Transmit functionality)

7.4.3.1 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 7.5: Interaction between the DSD submodule and the DSL submodule

7.4.3.2 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 7.6: Interaction of the DSD with the DSP

7.4.4 Functional Description of the DSD

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

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

[SWS_Dcm_00084] ⌈ If configured (configuration parameter DcmRespondAllRequest=FALSE), if the Dcm module receives a diagnostic request that contains a service ID that is in the range from 0x40 to 0x7F or in the range from 0xC0 to 0xFF, the Dcm shall not respond to such a request. ⌋*()*

This range corresponds to the diagnostic response identifier.

[SWS_Dcm_00192] ⌈ The DSD submodule shall analyze the (incoming) diagnostic message for the diagnostic service identifier (based on first byte of the diagnostic message) and shall check the supported services with the newly received diagnostic service identifier. ⌋*()*

[SWS_Dcm_00193] ⌈ During this check, the DSD submodule shall search the newly received diagnostic service identifier in the "Service Identifier Table". ⌋*()*

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

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

Rationale for [SWS_Dcm_00195]: The "Service Identifier Table" and the information about the supported services will be generated out of the configuration. More than one Service Identifier Table can be configured for selection. At one time only one Service Identifier Table can be active.

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

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

[SWS_Dcm_00197] ⌈ If the newly received diagnostic service identifier is not supported, the DSD submodule shall transmit a negative response with NRC 0x11 (Service not supported) to the DSL submodule. ⌋()

[SWS_Dcm_00198] ⌈ The DSD submodule shall store the newly received diagnostic service identifier for later use. ⌋()

For example: WriteDataByIdentifier (for writing VIN number):

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

[SWS_Dcm_CONSTR_6047] ⌈ Id of the Service identifier configured in DcmDsdSidTabServiceId shall be unique within one DcmDsdServiceTable. ⌋()

[SWS_Dcm_00732] ⌈ For the first call of <Module>_<DiagnosticService> the opStatus shall be set to DCM_INITIAL ⌋()

[SWS_Dcm_00733] ⌈ The Dcm shall not accept further requests (on same or lower priority) while <Module>_<DiagnosticService>() returns DCM_E_PENDING. Dcm-internal timeout handling (based on RCR-RP limitation) may lead to a cancellation of the external diagnostic service processing. ⌋()

[SWS_Dcm_00735] ⌈ In case of cancellation the API <Module>_<DiagnosticService> is called again with the parameter opStatus set to DCM_CANCEL ⌋()

7.4.4.2 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 [2] Section 6.5: Server response implementation rules).

[SWS_Dcm_00200] ⌈ If the "suppressPosRspMsgIndicationBit" is TRUE, the DSD submodule shall NOT send a positive response message. ⌋*(SRS_Diag_04020)*

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

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

[SWS_Dcm_00203] ⌈ In case of responsePending the Dcm module shall clear the "suppressPosRspMsgIndicationBit." ⌋*()*

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

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

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

Rationale for [SWS_Dcm_00204]: The "suppressPosRspMsgIndicationBit" is only available if a service has a subfunction.

7.4.4.3 Verification functionality

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

1. Verification of Manufacturer permission (Call of the manufacturer interface indication operation)
2. Verification of the SID
3. Verification of the Diagnostic Session
4. Verification of the Service Security Access levels
5. Verification of the Supplier permission (Call of the Supplier interface indication operation)
6. Verification of the Mode rules for service IDs

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

7.4.4.3.1 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 subservice is allowed in the current diagnostic session or not.

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

[SWS_Dcm_00211] ⌈ If the newly received diagnostic service is not allowed in the current Diagnostic Session (according to the configuration parameter DcmDsdSidTabSessionLevelRef), the DSD submodule shall transmit a negative response with NRC 0x7F (serviceNotSupportedInActiveSession) to the DSL submodule. ⌋*()*

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

7.4.4.3.2 Verification of the Service Security Access levels

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

The management of the security level is not part of the DSD submodule.

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

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

[SWS_Dcm_00617] ⌈ If the newly received diagnostic service is allowed in the current Security level (see [SWS_Dcm_00217]), but the requested subservice is not allowed in the current Security level (according to the configuration parameter DcmDsdSubServiceSecurityLevelRef), the DSD submodule shall transmit a negative response with NRC 0x33 (Security access denied) to the DSL submodule. ⌋*()*

7.4.4.3.3 Verification of the Service mode dependencies

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

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

7.4.4.4 Check format and subfunction support

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

[SWS_Dcm_00273] General sub-function supported NRC check ⌈ The DSD shall send the negative response NRC 0x12 (sub-functionNotSupported), if for the processed service no configured DcmDsdSubService exists with the DcmDsdSubServiceId of the processed service. This NRC check shall not be done for UDS Service 0x31 (RoutineControl). ⌋*()*

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

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

[SWS_Dcm_01411] ⌈ If DcmDsdSubService is configured for a DcmDsdService, the Dcm shall support the sub-function configured in DcmDsdSubServiceId with SPRMIB set to 0 or 1. ⌋*()*

7.4.4.4.1 Verification of the Manufacturer Application environment/permission

The purpose of this functionality is that, just after receiving the diagnostic request, the Manufacturer Application is requested to check permission/environment.

E.g. in after-run ECU state, it might be not allowed to process OBD requests.

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

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

[SWS_Dcm_01172] ⌈ In case of [SWS_Dcm_00462], the DSD shall only call Xxx_Confirmation but not DspInternal_DcmConfirmation. ⌋*()*

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

[SWS_Dcm_01321] ⌈ If more than one Xxx_Indication function called, according to [SWS_Dcm_00218], has returned E_NOT_OK and no function has returned E_REQUEST_NOT_ACCEPTED, the DSD submodule shall trigger a negative response using the ErrorCode parameter from the first Xxx_Indication returning E_NOT_OK (according to the sorting criteria of TPS_ECUC_06067). ⌋*(SRS_Diag_04011)*

7.4.4.4.2 Verification of the Supplier Application environment/permission

The purpose of this functionality is that, right before processing the diagnostic message, the Supplier Application is requested to check permission/environment.

E.g. in after-run ECU state, it might be not allowed to process OBD requests.

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

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

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

[SWS_Dcm_01322] ⌈ If more than one Xxx_Indication function called, according to [SWS_Dcm_00516], has returned E_NOT_OK and no function has returned E_REQUEST_NOT_ACCEPTED, the DSD submodule shall trigger a negative response using the ErrorCode parameter from the first Xxx_Indication returning E_NOT_OK (according to the sorting criteria of TPS_ECUC_06067). ⌋ *(SRS_Diag_04011)*

7.4.4.5 Distribution of diagnostic message to DSP submodule

[SWS_Dcm_00221] ⌈ The DSD submodule shall search for the executable functionality of the DSP submodule for newly received diagnostic service identifier and shall call the corresponding DSP service interpreter. ⌋ *()*

7.4.4.6 Assemble positive or negative response

[SWS_Dcm_00222] ⌈ When the DSP submodule has finished the execution of the requested Diagnostic Service the DSD submodule shall assemble the response. ⌋ *()*

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

- positive Result or
- negative Result.

Following possible Responses can be assembled:

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

7.4.4.6.1 Positive Response

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

[SWS_Dcm_00224] ⌈ The DSD submodule shall therefore transfer the Dcm_MsgContextType into a (response) buffer and shall add the service identifier at the first byte of the buffer. ⌋ *()*

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

7.4.4.6.2 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 [2] (see Section 4.2.4 Response code parameter definition Table 12) and ISO15031-5 [1]. The DSP and the Application have to take care of the correct use of NRC of the executed Service ID.

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

7.4.4.6.3 Suppression of response

[SWS_Dcm_00231] ⌈ In the case that the "suppressPosRspMsgIndicationBit" is indicated in the functionality "Handling of suppressPosRspMsgIndicationBit" (stored in the Variable Dcm_MsgContextType (Element: Dcm_MsgAddInfo)), the DSD submodule shall activate the suppression of Positive Responses. ⌋*()*

[SWS_Dcm_00001] ⌈ In the case of a Negative Result of the execution and active Functional Addressing the DSD submodule shall activate the suppression of the following Negative Responses:

- NRC 0x11 (Service not supported),
- NRC 0x12 (SubFunction not supported),
- NRC 0x31 (Request out of range),
- NRC 0x7E (Subfunction not supported in active session),
- NRC 0x7F (Service not supported in active session)

⌋*(SRS_Diag_04020)*

7.4.4.7 Initiate transmission

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

[SWS_Dcm_00237] ⌈ The DSL submodule shall forward the diagnostic (response) message (positive or negative response) further to the PduR module by executing a DSL transmit functionality. ⌋*()*

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

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

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

[SWS_Dcm_00238] ⌈ In the case that no diagnostic (response) message shall be sent (Suppression of Responses) the DSL submodule shall not transmit any response. ⌋*()*

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

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

Rationale for [SWS_Dcm_00240]: The DSP submodule is waiting for the execution of the DspInternal_DcmConfirmation() functionality. So it has to be sent, also when no Data Confirmation is provided. Altogether this means that in any of the following cases:

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

The DSD submodule will finish by calling DspInternal_DcmConfirmation() (refer to 8.10.3 DspInternal_DcmConfirmation).

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

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

[SWS_Dcm_00677] ⌈ If the operation Indication() returns value E_REQUEST_NOT_ACCEPTED, the Dcm module shall not send any diagnostic response and shall end the current diagnostic request management. ⌋*()*

[SWS_Dcm_00678] ⌈ If the operation Indication() returns value E_NOT_OK, the Dcm module shall send a negative response with NRC value equal to ErrorCode parameter value. ⌋*()*

7.5 Diagnostic Service Processing (DSP)

7.5.1 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.

7.5.1.1 Check format and subfunction support

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

[SWS_Dcm_00272] ⌈ The DSP submodule shall trigger a negative response with NRC 0x13 (Incorrect message length or invalid format), when the analysis of the request message results in formatting or length failure. ⌋*()*

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

7.5.1.2 Assemble response

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

[SWS_Dcm_00038] ⌈ If the paged-buffer mechanism is used, the DSP submodule shall determine the overall response length before any data is passed to the DSD submodule or the DSL submodule respectively. ⌋*()*

Requirement [SWS_Dcm_00038] is needed because of segmented diagnostic data transmission on CAN using ISO15765-2 [10], 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 7.3.4.9 for details about the paged-buffer mechanism).

7.5.1.3 Negative Response Codes handling

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

[SWS_Dcm_01414] Accepted range of Dcm_NegativeResponseCodeType for negative responses ⌈ If the Dcm calls an external application by any of the APIs having the out parameter Dcm_NegativeResponseCodeType ErrorCode, the Dcm shall accept only values in the range 0x01-0xFF in case the return value is E_NOT_OK. ⌋*()*

[SWS_Dcm_01415] Behavior on application returning unexpected return code ⌈ If the Dcm calls an API with the out parameter Dcm_NegativeResponseCodeType ErrorCode and the application sets this parameter to DCM_POS_RESP and E_NOT_OK is returned, the Dcm shall report the runtime error DCM_E_INVALID_VALUE. ⌋*()*

[SWS_Dcm_00275] ⌈ The DSP submodule shall trigger a negative response with NRC 0x31 (Request out of range), when the analysis of the request message results in other unsupported message parameters. ⌋*()*

7.5.1.4 Diagnostic mode declaration groups

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

1. DcmDiagnosticSessionControl (service 0x10)
2. DcmEcuReset (partly service 0x11)
3. DcmSecurityAccess (service 0x27)
4. DcmModeRapidPowerShutDown (partly service 0x11)
5. DcmCommunicationControl_<symbolic name of ComMChannelId>. (service 0x28)
6. DcmControlDTCSetting (service 0x85)
7. DcmResponseOnEvent_<RoeEventID> (service 0x86)

⌋*()*

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

[SWS_Dcm_01327] ⌈ The Dcm shall define the ModeDeclarationGroupPrototype DcmSecurityAccess as provided-ModeGroup based on the following ModeDeclarationGroup:

```
ModeDeclarationGroup DcmSecurityAccess {
    {
    DCM_SEC_LEV_LOCKED
    DCM_SEC_LEV_1
    ...
    DCM_SEC_LEV_63
    }
initialMode = DCM_SEC_LEV_LOCKED
};
```

⌋*()*

[SWS_Dcm_01328] ⌈

```
ModeSwitchInterface SchM_Switch_<bsnp>_DcmSecurityAccess {
  isService = true;
  SecLevel currentMode;
};
```

⌋*()*

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

[SWS_Dcm_00777] ⌈ The `Dcm` shall define the ModeDeclarationGroupPrototype DcmEcuReset as provided-ModeGroup in its Basic Software Module instance based on the ModeDeclarationGroup DcmEcuReset. ⌋*()*

[SWS_Dcm_00807] ⌈ The `Dcm` shall define the ModeDeclarationGroupPrototype DcmModeRapidPowerShutDown as provided-ModeGroup in its Basic Software Module instance based on the ModeDeclarationGroup DcmModeRapidPowerShutDown. ⌋*()*

[SWS_Dcm_00780] ⌈ The `Dcm` shall define for each network which is considered in the CommunicationControl service a separate ModeDeclarationGroupPrototype DcmCommunicationControl_<symbolic name of ComMChannelId> as provided-ModeGroup in its Basic Software Module instance based on the ModeDeclarationGroup DcmCommunicationControl. ⌋*()*

[SWS_Dcm_00781] ⌈ The `Dcm` shall define the ModeDeclarationGroupPrototype DcmControlDTCSetting as provided-ModeGroup in its Basic Software Module instance based on the ModeDeclarationGroup DcmControlDTCSetting. ⌋*()*

[SWS_Dcm_00933] ⌈ The `Dcm` shall define for each RoeEvent a separate ModeDeclarationGroupPrototype DcmResponseOnEvent_<Symbolic name of RoeEventId> as provided-ModeGroup in its Basic Software Module instance based on the ModeDeclarationGroup DcmResponseOnEvent. ⌋*()*

The `Dcm` provides a state machine for each RoeEvent (see Figure 7.4). The state for a RoeEvent is needed by SWC to activate event reporting or report the Roe status to a Did. Therefore the `Dcm` provides for each state of each RoeEvent a ModeDeclarationGroupPrototype which reports the current state of the state machine as mode.

[SWS_Dcm_00934] ⌈ The ModeDeclarationGroupPrototype shall represent the current state of the ROE state machine for this RoeEvent. ⌋*()*

7.5.1.5 Mode dependent request execution

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

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

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

[SWS_Dcm_CONSTR_6028] ⌈ DcmModeCondition shall either have a DcmBswModeRef or a DcmSwcModeRef or a DcmSwcSRDataElementRef as external reference. ⌋ *()*

[SWS_Dcm_00810] ⌈ The DcmSwcModeRef and DcmBswModeRef of 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. ⌋ *()*

[SWS_Dcm_01119] ⌈ The DcmSwcSRDataElementRef of DcmModeCondition shall be evaluated if the referenced data element (by DcmDspExternalSRDataElementClass):

- is equal to the value represented by the reference DcmSwcSRDataElementValueRef in case of DcmConditionType is set to DCM_EQUALS
- is unequal to the value represented by the reference DcmSwcSRDataElementValueRef in case of DcmConditionType is set to DCM_EQUALS_NOT
- is greater than the value represented by the reference DcmSwcSRDataElementValueRef in case of DcmConditionType is set to DCM_GREATER_THAN
- is greater or equal than the value represented by the reference DcmSwcSRDataElementValueRef in case of DcmConditionType is set to DCM_GREATER_OR_EQUAL
- is less than the value represented by the reference DcmSwcSRDataElementValueRef in case of DcmConditionType is set to DCM_LESS_THAN
- is less or equal than the value represented by the reference DcmSwcSRDataElementValueRef in case of DcmConditionType is set to DCM_LESS_OR_EQUAL

⌋ *()*

[SWS_Dcm_CONSTR_6029] ⌈ The values DCM_GREATER_THAN, DCM_GREATER_OR_EQUAL, DCM_LESS_OR_EQUAL and DCM_LESS_THAN shall not used with a Mode reference (DcmBswModeRef or DcmSwcModeRef) . ⌋*()*

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

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

Note: This implies the priority of NRCs

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

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

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

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

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

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

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

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

Examples on using mode dependent request execution:

General assumptions:

1. DcmModeRule1 consists of DcmModeCondition1, DcmModeRule2 and DcmModeRule3
2. DcmModeRule1 defines NRC 0x22

3. DcmModeRule2 and DcmModeRule3 do not have any sub-rules
4. DcmModeRule2 defines NRC 0x72
5. DcmModeRule3 does not define a NRC value

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

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

7.5.1.6 Sender/Receiver Communication

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

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

[SWS_Dcm_01336] ⌈ The Dcm shall create for each configured element DcmDspPidService01 having a sender/receiver interface (if parameter DcmDspPidDataUsePort is set to USE_DATA_SENDER_RECEIVER or USE_DATA_SENDER_RECEIVER_AS_SERVICE) a corresponding R-Port DataInterface with one data element having an ImplementationDataType of type DcmDspPidDataType. ⌋*()*

[SWS_Dcm_00964] ⌈ The created ports of [SWS_Dcm_00962], [SWS_Dcm_00963] and [SWS_Dcm_01336] shall derive the CompuMethod from the DcmDspDiagnosisScaling container (if present) and add it to the DataType in their respective port interface. ⌋*()*

7.5.1.7 Passing SwDataDefProps properties from DEXT file to the Dcm Service SW-C

UseCase: Pass the SwDataDefProps details like CompuMethod, DataContraints and Units to the Dcm Service SW-C and make them there available per DID DataElement / per RoutineControl signal. Two alternative work flows are available.

[SWS_Dcm_CONSTR_6079] Dependency for DcmSwcSRDataElementValueRef
⌈ DcmSwcSRDataElementValueRef shall be present only if DcmSwcSRDataElementRef is configured. ⌋*()*

7.5.1.7.1 DcmDspDiagnosticDataElementRef workflow

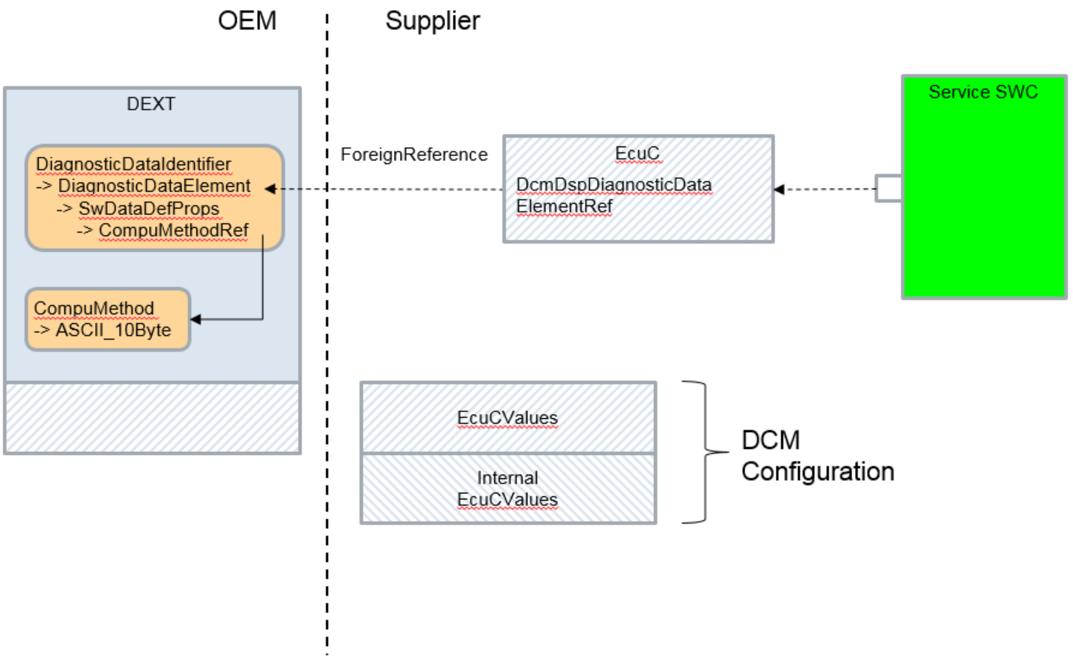


Figure 7.5: DcmDspDiagnosticDataElementRef Workflow

The feature of the DcmDspDiagnosticDataElementRef workflow is the use of a EcucForeignReference inside the generated EcuC values. While importing the DEXT information, a dedicated EcuC parameter is generated, which holds a EcucForeignReference named DcmDspDiagnosticDataElementRef to a DiagnosticDataElement in the DEXT file. This EcucForeignReference enables the access to all SwDataDefProps (BaseType, CompuMethod, DataConstr, etc.) of the corresponding DiagnosticDataElement. The container DcmDspAlternativeDiagnosticDataElement aggregates this EcucForeignReference. In the process step of generating the corresponding Service SWC all needed content will be copied directly based on the EcucForeignReference from DEXT to the Service SW-C. In this work flow the existence of the DEXT file while the generation of the Service SW-C is required.

7.5.1.7.2 DcmDspAlternativeDataType.DcmApplicationDataType workflow

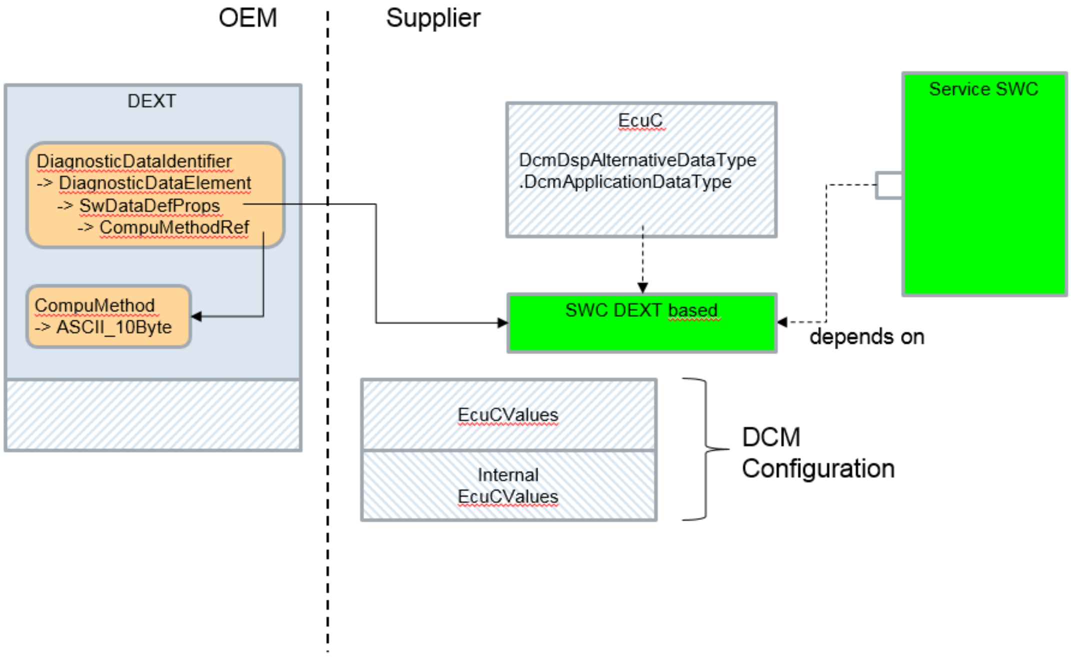


Figure 7.6: DcmDspAlternativeDataType.DcmApplicationDataType Workflow

The feature of the DcmDspAlternativeDataType.DcmApplicationDataType workflow is that while importing the DEXT information beside the EcuC values also a SW-C fragment is generated. In this SW-C fragment all needed SwDataDefProps are directly copied from the DEXT file. Inside the generated EcuC values the EcuC parameter DcmDspAlternativeDataType.DcmApplicationDataType refers to the SWC fragment and enables the access to all SwDataDefProps (BaseType, CompuMethod, DataConstr, etc.). In the process step of generating the corresponding Service SW-C, all needed content will be included based on the reference from DcmDspAlternativeDataType.DcmApplicationDataType to the SW-C fragment. In this work flow the existence of the DEXT file while the generation of the Service SW-C is not required.

7.5.1.8 Asynchronous call behavior

[SWS_Dcm_01412] ⌈ If a Dem function returns DEM_PENDING, the Dcm shall call this function again at a later point in time as long as DEM_PENDING is returned. ⌋*()*

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

[SWS_Dcm_01184] ⌈ The Dcm_SetProgConditions API shall be called again in the next Dcm main function cycle if previous return status was E_PENDING. ⌋*()*

[SWS_Dcm_00760] ⌈ The return of DCM_E_PENDING shall do a re-triggering (e.g. in the next MainFunction cycle). ⌋*()*

[SWS_Dcm_01413] ⌈ The return values of interfaces called with an OpStatus equal to DCM_CANCEL shall be ignored ⌋*()*

7.5.2 UDS Services

[SWS_Dcm_00442] ⌈ The Dcm module shall implement the services of UDS according to Table 7.7.⌋*()*

SID	Service	Subfunction	Supported
0x10	DiagnosticSessionControl		Supported
0x11	ECUReset		Supported
0x14	ClearDiagnosticInformation		Supported
0x19	ReadDTCInformation		Supported
0x22	ReadDataByIdentifier		Supported
0x23	ReadMemoryByAddress		Supported (callout)
0x24	ReadScalingDataByIdentifier		Supported
0x27	SecurityAccess		Supported
0x28	CommunicationControl		Supported
0x2A	ReadDataByPeriodicIdentifier		Supported
0x2C	DynamicallyDefineDataIdentifie		Supported
0x2E	WriteDataByIdentifier		Supported
0x2F	InputOutputControlByIdentifier		Supported
0x31	RoutineControl		Supported
0x34	RequestDownload		Supported (callout)
0x35	RequestUpload		Supported (callout)
0x36	TransferData		Supported
0x37	RequestTransferExit		Supported
0x38	RequestFileTransfer		Supported (callout)
0x3D	WriteMemoryByAddress		Supported (callout)
0x3E	TesterPresent		Supported
0x83	AccessTimingParameter		NRC "ServiceNotSupported"
0x84	SecuredDataTransmission		NRC "ServiceNotSupported"
0x85	ControlDTCSetting	On, off	Supported
0x86	ResponseOnEvent	All excepted onComparisionOfValues and OnTimerInterrupt	Supported
0x87	LinkControl		User optional

Table 7.7: Support of UDS Services

7.5.2.1 General behavior using DEM interfaces

[SWS_Dcm_00007] ⌈ The Dcm module shall retrieve the DTCStatusAvailabilityMask by using the function Dem_GetDTCStatusAvailabilityMask(). ⌋*()*

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

Note : Masking is performed in the module Dem and does not need to be done on Dcm side (see SWS_Dem_00657 in [11]).

[SWS_Dcm_00371] ⌈ To ensure consistent event related data during the reading sequence, the Dcm module shall lock the update of event related data before reading freeze frames or extended data records. The Dcm shall lock the update using the Dem API Dem_DisableDTCRecordUpdate(). After the locking the Dcm shall read the event related data by calls to:

- Dem_SelectExtendedDataRecord()
- Dem_GetSizeOfExtendedDataRecordSelection()
- Dem_GetNextExtendedDataRecord()
- Dem_SelectFreezeFrameData()
- Dem_GetSizeOfFreezeFrameSelection() and
- Dem_GetNextFreezeFrameData() After the event related data is read, the Dcm shall re-enable updates by calling Dem_EnableDTCRecordUpdate().

⌋*()*

[SWS_Dcm_00702] ⌈ If function Dem_DisableDTCRecordUpdate() returns DEM_PENDING, the Dcm shall retry to get the lock in the next Dcm_MainFunction. ⌋*()*

[SWS_Dcm_00700] ⌈ When the Dcm module receives a request with the DTCStatusMask set to 0x00, it shall send positive response 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 [12] DTC format, the DTCFormat parameter shall be equal to DEM_DTC_FORMAT_OBD. For the 2-byte ISO14229-1 DTC format, the DTCFormat parameter shall be equal to DEM_DTC_FORMAT_UDS.

[SWS_Dcm_01160] ⌈ When the Dcm module receives a request with the DTCSeverityMask set to 0x00, it shall send a positive response as specified in ISO14229-1 [2] and shall not use the Dem interface Dem_SetDTCFilter() ⌋*()*

[SWS_Dcm_00835] ⌈ The Dcm shall call Dem_SetDTCFilter prior to Dem_GetNumberOfFilteredDTC, any sequence of Dem_GetNextFilteredDTC, any sequence of Dem_GetNextFilteredDTCAndFDC, as well as any sequence of Dem_GetNextFilteredDTCAndSeverity. ⌋*()*

[SWS_Dcm_00836] ⌈ The Dcm shall call Dem_SetFreezeFrameRecordFilter prior to any sequence of Dem_GetNextFilteredRecord. ⌋*()*

[SWS_Dcm_01127] ⌈ The Dcm module shall retrieve the DTCSeverityAvailabilityMask by using the function Dem_GetDTCSeverityAvailabilityMask() ⌋*()*

Note: The mask DTCSeverityAvailabilityMask reflects the severity bits supported by the ECU.

[SWS_Dcm_01212] ⌈ If Dem_DisableDTCRecordUpdate() returns DEM_WRONG_DTC, the Dcm shall send a NRC 0x31 (RequestOutOfRange). ⌋*()*

[SWS_Dcm_01213] ⌈ If Dem_DisableDTCRecordUpdate() returns DEM_WRONG_DTCORIGIN, the Dcm shall send a NRC 0x31 (RequestOutOfRange). ⌋*()*

[SWS_Dcm_01234] ⌈ If Dem_GetNextFilteredDTCAndSeverity() returns DEM_NO_SUCH_ELEMENT and at least one matching element could be retrieved before, the Dcm shall send a positive response including these data elements. ⌋*()*

[SWS_Dcm_01235] ⌈ If Dem_GetNextFilteredDTCAndSeverity() returns DEM_NO_SUCH_ELEMENT and no matching element could be retrieved before, the Dcm shall send a positive response only for service, subservice and mandatory data specified in ISO 14229-1 [2]. ⌋*()*

[SWS_Dcm_01242] ⌈ If Dem_GetSizeOfExtendedDataRecordSelection() returns DEM_WRONG_DTC, DEM_WRONG_DTCORIGIN or DEM_NO_SUCH_ELEMENT, the Dcm shall send a NRC 0x31 (RequestOutOfRange) ⌋*()*

[SWS_Dcm_01250] ⌈ If Dem_GetStatusOfDTC() returns DEM_WRONG_DTC or DEM_WRONG_DTCORIGIN, the Dcm shall send a NRC 0x31 (RequestOutOfRange). ⌋*()*

[SWS_Dcm_01409] ⌈ If Dem_GetStatusOfDTC() returns DEM_NO_SUCH_ELEMENT, the Dcm shall send a positive response only for service and subservice. ⌋*()*

[SWS_Dcm_01255] ⌈ If Dem_SetDTCFilter() returns E_NOT_OK, the Dcm shall send a NRC 0x31 (RequestOutOfRange). ⌋*()*

7.5.2.2 Service 0x10 - Diagnostic Session Control

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

- diagnosticSessionType

[SWS_Dcm_00250] ⌈ The Dcm module shall implement the UDS Service 0x10. ⌋ *(SRS_Diag_04006)*

[SWS_Dcm_00307] ⌈ When responding to UDS Service 0x10, if the requested subfunction value is not configured in the ECU (configuration parameter DcmDspSessionLevel), the DSP submodule shall trigger a negative response with NRC 0x12 (SubFunction not supported). ⌋*()*

If the requested subfunction value is configured, the following steps are processed even if the requested session type is equal to the already running session type (see ISO14229-1 [2] Section 9.2).

[SWS_Dcm_00311] ⌈ The send confirmation function shall set the new diagnostic session type with DslInternal_SetSesCtrlType() and shall set the new timing parameters (P2ServerMax, P2ServerMax*) (see configuration parameters DcmDspSessionP2ServerMax and DcmDspSessionP2StarServerMax) and do the mode switch of the ModeDeclarationGroupPrototype DcmDiagnosticSessionControl by calling SchM_Switch_<bsnp>_DcmDiagnosticSessionControl() with the new diagnostic session type (see [SWS_Dcm_91019]). ⌋*(SRS_Diag_04015)*

[SWS_Dcm_00085] ⌈ The DSP submodule shall manage internally a read access for the dataIdentifier 0xF186 (ActiveDiagnosticSessionDataIdentifier) defined in ISO14229-1 [2]. ⌋*()*

7.5.2.3 Service 0x11 - ECUReset

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

- resetType

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

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

Note: By this mode switch the Dcm informs the BswM to carry out necessary actions for the handling of this individual reset type. These actions can be configured within the BswM action list corresponding to the requested reset type. Here the integrator can also define if an ECU reset will finally be performed or not.

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

Note: By this mode switch the Dcm requests the BswM to perform the final processing on the reset type according to the configured action list.

[SWS_Dcm_00818] ⌈ On reception of a request for UDS Service 0x11 with the sub functions enableRapidPowerShutdown (0x04) or disableRapidPowerShutdown (0x05), the Dcm module shall trigger the mode switch of ModeDeclarationGroupPrototype DcmRapidPowerShutDown: Sub function enableRapidPowerShutDown (0x04) to ENABLE_RAPIDPOWERSHUTDOWN, Sub function disableRapidPowerShutDown (0x05) to DISABLE_RAPIDPOWERSHUTDOWN ⌋*()*

Note: If EnableRapidPowerShutdown is enabled, the ECU should shorten its power-down time.

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

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

[SWS_Dcm_CONSTR_6080] DcmDspEcuResetRow container configuration ⌈ One container DcmDspEcuResetRow shall be configured for each DcmDsdSubService configured for the UDS service ECUReset (0x11). ⌋*(SRS_Diag_04098)*

7.5.2.4 Service 0x14 - Clear Diagnostic Information

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

- groupOfDTC.

[SWS_Dcm_00247] ⌈ The Dcm module shall implement UDS Service 0x14. ⌋*()*

[SWS_Dcm_01263] ⌈ Upon reception of a UDS Service ClearDiagnosticInformation (0x14) request with parameter groupOfDTC, the Dcm module shall call the API Dem_SelectDTC() with the following parameter values:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef)
- DTC: groupOfDTC from the service request
- DTCFormat: DEM_DTC_FORMAT_UDS
- DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY

⌋*(SRS_Diag_04058)*

[SWS_Dcm_01400] ⌈ After call of Dem_SelectDTC() the Dcm shall call Dem_GetDTCSelectionResult() with the following parameter value:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef.

⌋*()*

[SWS_Dcm_01265] ⌈ In case Dem_GetDTCSelectionResult() returns DEM_WRONG_DTC, the Dcm shall send a NRC 0x31 (RequestOutOfRange). ⌋*()*

[SWS_Dcm_01268] ⌈ In case Dem_GetDTCSelectionResult() returns E_OK, the Dcm module shall check if application allows to clear the DTC (according to the configuration parameter DcmDspClearDTCCheckFnc). If not, the Dcm module shall send a negative response with NRC set to value from the parameter "ErrorCode". ⌋*()*

[SWS_Dcm_01269] ⌈ In case application allows to clear the DTC, the Dcm module shall check if the DTC can be cleared in the current mode condition (according to the configuration parameter DcmDspClearDTCModeRuleRef). If not, the Dcm module shall send the calculated negative response code of the referenced DcmModeRule. ⌋*()*

[SWS_Dcm_00005] ⌈ If the condition checks are successfully done, the Dcm module shall call Dem_ClearDTC with the following parameter values:

- ClientId = Client Id for this Dcm instance (see DcmDemClientRef)

⌋*(SRS_Diag_04058)*

[SWS_Dcm_00705] ⌈ In case Dem_ClearDTC() returns E_OK, the Dcm module shall send a positive response. ⌋*()*

[SWS_Dcm_00707] ⌈ In case Dem_ClearDTC() returns DEM_CLEAR_FAILED, the Dcm shall send a negative response 0x22 (conditionsNotCorrect). ⌋*()*

[SWS_Dcm_00708] ⌈ In case Dem_ClearDTC() returns DEM_WRONG_DTC, the Dcm shall send a negative response 0x31 (requestOutOfRange). ⌋*()*

[SWS_Dcm_00966] ⌈ In case Dem_ClearDTC() returns DEM_CLEAR_BUSY, the Dcm shall send a negative response 0x22 (conditionsNotCorrect). ⌋*()*

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

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

[SWS_Dcm_01408] ⌈ In case Dem_ClearDTC() returns DEM_WRONG_DTCORIGIN, the Dcm shall trigger a negative response 0x31 (requestOutOfRange). ⌋*()*

7.5.2.5 Service 0x19 - Read DTC Information

Service 0x19 allows a client to read the status of server resident Diagnostic Trouble Code (DTC) information.

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

To setup the retrieval of specific data from the Dem module, the Dcm will call different filter APIs (Dem_SetDTCFilter(), Dem_SetFreezeFrameRecordFilter(), Dem_SelectFreezeFrameData() and Dem_SelectExtendedDataRecord()).

[SWS_Dcm_01043] ⌈ In case E_NOT_OK is returned by Dem_SetDTCFilter(), the Dcm module shall send a negative response with NRC 0x31 (requestOutOfRange). ⌋*()*

[SWS_Dcm_01334] ⌈ For all sub-functions addressing user defined fault memory, before calling the appropriate Dem API, the Dcm shall add the value 0x0100 to the received selection request parameter MemorySelection in order to match the Dem_DTCOriginType. ⌋*()*

7.5.2.5.1 Subfunctions 0x01, 0x07, 0x11 and 0x12

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

- DTCStatusMask

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

- DTCSeverityMask
- DTCStatusMask

[SWS_Dcm_00376] ⌈ When sending a positive response to UDS Service 0x19 with subfunction 0x01, 0x07, 0x11 or 0x12, the Dcm module shall use the data in the response message according to Table 7.8⌋*()*

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see [SWS_Dcm_00007]).
DTCFormatIdentifier	Value returned by Dem_GetTranslationType()
DTCCount	Value calculated according to [SWS_Dcm_00293]

Table 7.8: Subfaunction 0x01, 0x07, 0x11 and 0x12 response values

[SWS_Dcm_00293] ⌈ When responding to UDS Service 0x19 with subfunction 0x01, 0x07, 0x11 or 0x12, the Dcm module shall calculate the number of DTCs using Dem_GetNumberOfFilteredDTC() after having set the DEM-filter with Dem_SetDTCFilter() using the parameter values according to Table 7.9.⌋ *(SRS_Diag_04058, SRS_Diag_04067)*

	reportNumberOfDTCByStatusMask	**reportNumberOfDTCBySeverityMaskRecord**	**reportNumberOfMirrorMemoryDTCByStatusMask**	**reportNumberOfEmissionsRelatedOBDDTCByStatusMask**
	0x01	0x07	0x11	0x12
ClientId	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)
DTCStatusMask	DTCStatusMask from request (see [SWS_Dcm_00700])	DTCStatusMask from request (see [SWS_Dcm_00700])	DTCStatusMask from request (see [SWS_Dcm_00700])	DTCStatusMask from request (see [SWS_Dcm_00700])
DTCFormat	DEM_DTC_FORMAT_UDS	DEM_DTC_FORMAT_UDS	DEM_DTC_FORMAT_UDS	DEM_DTC_FORMAT_UDS
DTCOrigin	PRIMARY_MEMORY	PRIMARY_MEMORY	MIRROR_MEMORY	DEM_DTC_ORIGIN_OBD_RELEVANT_MEMORY
FilterWithSeverity	NO	YES	NO	NO
DTCSeverityMask	Not relevant	DTCSeverityMask from request	Not relevant	Not relevant
FilterForFaultDetectionCounter	NO	NO	NO	NO

Table 7.9: Dem_SetDTCFilter() parameters values for subfunctions 0x01, 0x07, 0x11 and 0x12

7.5.2.5.2 Subfunctions 0x02, 0x0A, 0x0F, 0x13, 0x15 and 0x17

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

- DTCStatusMask

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

[SWS_Dcm_00377] ⌈ When sending a positive response to UDS Service 0x19 with subfunction 0x02, 0x0A, 0x0F, 0x13, 0x15 or 0x17, the Dcm module shall use the data in the response message according to Table 7.10.⌋ *()*

Parameter name	Value

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see [SWS_Dcm_00007])
DTCAndStatusRecord	As defined in [SWS_Dcm_00008] and [SWS_Dcm_00378]
MemorySelection (subservice 0x17 only)	From request

Table 7.10: Subfunction 0x02, 0x0A, 0x0F, 0x13, 0x15 and 0x17 eesponse values

[SWS_Dcm_00008] ⌈ On reception of a UDS Service 0x19 request with subfunction 0x02, 0x0F and 0x13 and if the result of the bitwise AND operation between the DTCStatusMask received within the request message and the DTCStatusAvailabilityMask reported by the DEM is equal to 0, the Dcm module shall answer positively with 0 DTC. ⌋*()*

[SWS_Dcm_00378] ⌈ When responding to UDS Service 0x19 with subfunctions 0x02, 0x0A, 0x0F, 0x13, 0x15 or 0x17, 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 according to Table 7.11.⌋*(SRS_Diag_04058, SRS_Diag_04067)*

	reportDTC ByStatus Mask	report Supported DTCs	report Mirror Memory DTCBy StatusMask	report Emissions Related OBDDTC ByStatus Mask	report DTCWith Permanent Status	report UserDef Memory DTCBy StatusMask
	0x02	0x0A	0x0F	0x13	0x15	0x17
ClientId	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)
DTCStatus Mask	DTCStatus Mask from request (see SWS_Dcm_00700)	0x00	DTCStatus Mask from request (see SWS_Dcm_00700)	DTCStatus Mask from request (see SWS_Dcm_00700)	0x00	DTCStatus Mask from request (see SWS_Dcm_00700)
DTCFormat	DEM_DTC_FORMAT_UDS	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	DEM_DTC_ORIGIN_OBD_RELEVANT_MEMORY	PERMANENT_MEMORY	Memory Selection from request + 0x0100
FilterWith Severity	NO	NO	NO	NO	NO	NO

	reportDTC ByStatus Mask	report Supported DTCs	report Mirror Memory DTCBy StatusMask	report Emissions Related OBDDTC ByStatus Mask	report DTCWith Permanent Status	report UserDef Memory DTCBy StatusMask
DTCSeverity Mask	Not relevant	Not relevant	Not relevant	Not relevant	Not relevant	Not relevant
FilterFor FaultDetectionCounter	NO	NO	NO	NO	NO	NO

Table 7.11: Dem_SetDTCFilter() parameters values for subfunctions 0x02, 0x0A, 0x0F, 0x13,0x15 and0x17

Note:

- The Dcm module can get an indication of 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().

[SWS_Dcm_00828] ⌈ In case of paged buffer support is disabled, the Dcm module shall not insert zero-padded DTCs to the response of UDS Service 0x19 with subfunctions 0x02, 0x0A, 0x0F, 0x13, 0x15 or 0x17. ⌋*()*

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

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

[SWS_Dcm_00588] ⌈ In case of paged buffer support is enabled,The Dcm shall pad the response with the size calculated when sending the first page. If less DTC match the filter after this sending, the missing DTCs shall be padded with 0 value as defined in 15031-6 [12]. ⌋*()*

[SWS_Dcm_01229] ⌈ If Dem_GetNextFilteredDTC() returns DEM_NO_SUCH_ELEMENT and at least one matching element could be retrieved before, the Dcm shall send a positive response including these data elements. ⌋*()*

[SWS_Dcm_01230] ⌈ If Dem_GetNextFilteredDTC() returns DEM_NO_SUCH_ELEMENT and at no matching element could be retrieved before, the Dcm shall send a positive response only for service and subservice and additional parameters required within a positive response. ⌋*()*

7.5.2.5.3 Subfunction 0x08

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

- DTCSeverityMask
- DTCStatusMask

[SWS_Dcm_00379] ⌈ When sending a positive response to UDS Service 0x19 with subfunction 0x08, the Dcm module shall use the data in the response message according to Table 7.12.⌋ *()*

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see [SWS_Dcm_00007])
DTCAndSeverityRecord	As defined in [SWS_Dcm_00380]

Table 7.12: Subfunction 0x08 response values

[SWS_Dcm_00380] ⌈ 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 according to Table 7.13.⌋ *()*

	reportDTCBySeverityMaskRecord
ClientId	Client Id for this Dcm instance (see DcmDemClientRef)
DTCStatusMask	DTCStatusMask from request (see [SWS_Dcm_00700]
DTCFormat	DEM_DTC_FORMAT_UDS
DTCOrigin	PRIMARY_MEMORY
FilterWithSeverity	YES
DTCSeverityMask	DTCSeverityMask from request
FilterForFaultDetectionCounter	NO

Table 7.13: Dem_SetDTCFilter() parameters values for Subfunction 0x08

Note: The Dcm module can get an indication of the number of records that will be found using Dem_GetNextFilteredDTCAndSeverity() by using Dem_GetNumberOfFilteredDTC().

7.5.2.5.4 Subfunction 0x09

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

- DTCMaskRecord

[SWS_Dcm_00381] ⌈ When sending a positive response to UDS Service 0x19 with subfunction 0x09, the Dcm module shall use the data in the response message according to Table 7.14.⌋ *()*

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see [SWS_Dcm_00007])
DTCAndSeverityRecord	DTCSeverityMask: see [SWS_Dcm_01402] DTCFunctionalUnit: see [SWS_Dcm_01403] DTC: the given DTC of the request statusOfDTC : see [SWS_Dcm_01404]

Table 7.14: Subfunction 0x09 response values

[SWS_Dcm_01402] ⌈ To select the DTC, the Dcm module shall call the API Dem_SelectDTC() with the following parameter values:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef)
- DTC: DTC from the service request
- DTCFormat: DEM_DTC_FORMAT_UDS
- DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY

⌋ *()*

[SWS_Dcm_01403] ⌈ To retrieve the DTCSeverityMask of the selected DTC, the Dcm shall call Dem_GetSeverityOfDTC() with the following parameter value:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef)

⌋ *()*

[SWS_Dcm_01404] ⌈ To retrieve the DTCFunctionalUnit of the selected DTC, the Dcm shall call Dem_GetFunctionalUnitOfDTC() with the following parameter value:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef)

⌋ *()*

[SWS_Dcm_01405] ⌈ To retrieve the statusOfDTC of the selected DTC, the Dcm shall call Dem_GetStatusOfDTC() with the following parameter value:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef)

⌋ *()*

[SWS_Dcm_01226] ⌈ If Dem_GetFunctionalUnitOfDTC() returns DEM_WRONG_DTC or DEM_WRONG_DTCORIGIN, the Dcm shall send a NRC 0x31 (requestOutOfRange). ⌋ *()*

[SWS_Dcm_01240] ⌈ If Dem_GetSeverityOfDTC() returns DEM_WRONG_DTC, the Dcm shall send a NRC 0x31 (requestOutOfRange) ⌋ *()*

[SWS_Dcm_01406] ⌈ If Dem_GetStatusOfDTC() returns DEM_WRONG_DTC or DEM_WRONG_DTCORIGIN, the Dcm shall send a NRC 0x31 (requestOutOfRange). ⌋*()*

7.5.2.5.5 Subfunctions 0x06/0x10/0x19

The UDS Service 0x19 with subfunction 0x06, 0x10 or 0x19 requests a specific Extended Data Records for a specific DTC. The service request contains the parameters:

- DTCMaskRecord
- DTCExtendedDataRecordNumber

[SWS_Dcm_00386] ⌈ Upon reception of UDS Service 0x019 with subfunction 0x06 or 0x10 or 0x19, the Dcm shall retrieve from the Dem the stored extended data records for the requested DTC and origin. ⌋*()*

[SWS_Dcm_00295] ⌈ When responding to UDS Service 0x19 with subfunction 0x06, 0x10 or 0x19, the Dcm module shall calculate the statusOfDTC by first calling Dem_SelectDTC() with the parameters values set according to Table 7.15 and then Dem_GetStatusOfDTC() with ClientId = Client Id for this Dcm instance (see DcmDemClientRef).⌋*(SRS_Diag_04058)*

	reportDTC ExtendedData Record ByDTCNumber	**report MirrorMemoryDTC ExtendedData Record ByDTCNumber**	**reportUserDef MemoryDTC ExtDataRecord ByDTCNumber**
	0x06	0x10	0x19
ClientId	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)
DTC	DTCMaskRecord from request	DTCMaskRecord from request	DTCMaskRecord from request
DTCOrigin	PRIMARY_MEMORY	MIRROR_MEMORY	MemorySelection from request + 0x0100

Table 7.15: Dem_SelectDTC() parameters values for subfunctions 0x06, 0x10 and 0x19

[SWS_Dcm_00841] ⌈ If Dem_GetNextExtendedDataRecord() returns E_OK and BufSize 0 (empty buffer), the Dcm module shall omit the DTCExtendedDataRecordNumber for the related record in the response of service 0x19 0x06/0x10/0x19. ⌋*()*

[SWS_Dcm_00382] ⌈ When responding to UDS Service 0x19 with subfunction 0x06, 0x10 or 0x19, the Dcm module shall calculate the DTCExtendedDataRecord by first calling Dem_SelectExtendedDataRecord() with the parameter values set according to Table 7.16 and then call Dem_GetNextExtendedDataRecord() repeatedly until DEM_NO_SUCH_ELEMENT is returned.⌋*()*

	reportDTCExtended-DataRecord ByDTCNumber	reportMirrorMemoryD Extended-DataRecord ByDTCNumber	reportUserDefMemory DTCExtDataRecord ByDTCNumber
	0x06	0x10	0x19
ClientId	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)
DTC	DTCMaskRecord from request	DTCMaskRecord from request	DTCMaskRecord from request
DTCOrigin	PRIMARY_MEMORY	MIRROR_MEMORY	Memory Selection from request + 0x0100
ExtendedDataNumber	DTCExtendedData RecordNumber from request	DTCExtendedData RecordNumber from request	DTCExtendedData RecordNumber from request

Table 7.16: Dem_SelectExtendedDataRecord() parameters values for subfunctions 0x06, 0x10 and 0x19

As required in [SWS_Dcm_00371], the Dcm module shall obtain the size of the extended data record by using Dem_GetSizeOfExtendedDataRecordSelection().

7.5.2.5.6 Subfunction 0x03

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

[SWS_Dcm_00300] ⌈ When sending a positive response to UDS Service 0x19 with subfunction 0x03, the Dcm module shall use the data in the response message according to Table 7.17. ⌋*()*

Parameter name	Value
DTCRecord / DTCSnapshotRecord-Number	As defined in [SWS_Dcm_00299]

Table 7.17: Subfunction 0x03 response values

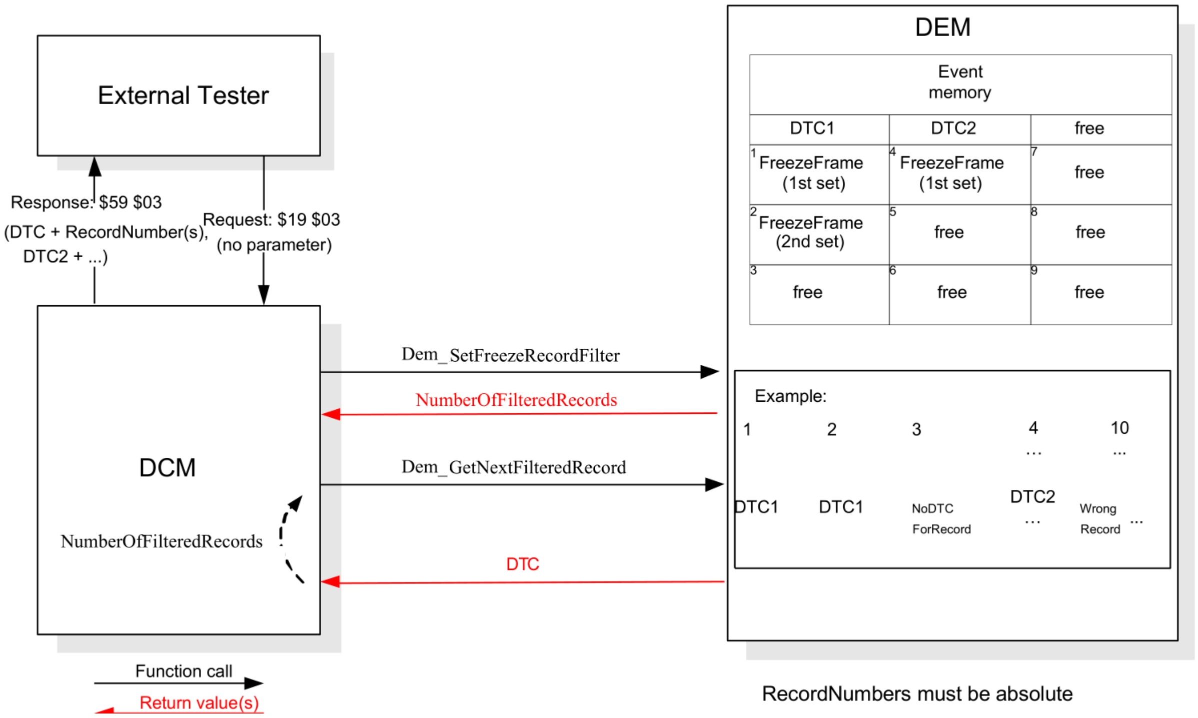


Figure 7.7: Request DTC Snapshot Record Identification

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

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

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

[SWS_Dcm_01237] ⌈ If Dem_GetNextFilteredRecord() returns DEM_NO_SUCH_ELEMENT and at least one matching element could be retrieved before, the Dcm shall send a positive response including these data elements. ⌋*()*

[SWS_Dcm_01238] ⌈ If Dem_GetNextFilteredRecord() returns DEM_NO_SUCH_ELEMENT and no matching element could be retrieved before, the Dcm shall send a positive response only for service and subservice. ⌋ *()*

[SWS_Dcm_01256] ⌈ If Dem_SetFreezeFrameRecordFilter() returns E_NOT_OK, the Dcm shall send a NRC 0x31 (RequestOutOfRange). ⌋*()*

7.5.2.5.7 Subfunctions 0x04 and 0x18

Using UDS Service 0x19 with subfunction 0x04 or 0x18, an external tester can request FreezeFrame information for one or all FreezeFrames of a specific DTC. The service request contains parameters:

- DTCMaskRecord
- DTCSnapshotRecordNumber

The subfunction 0x18 has an additional MemorySelection.

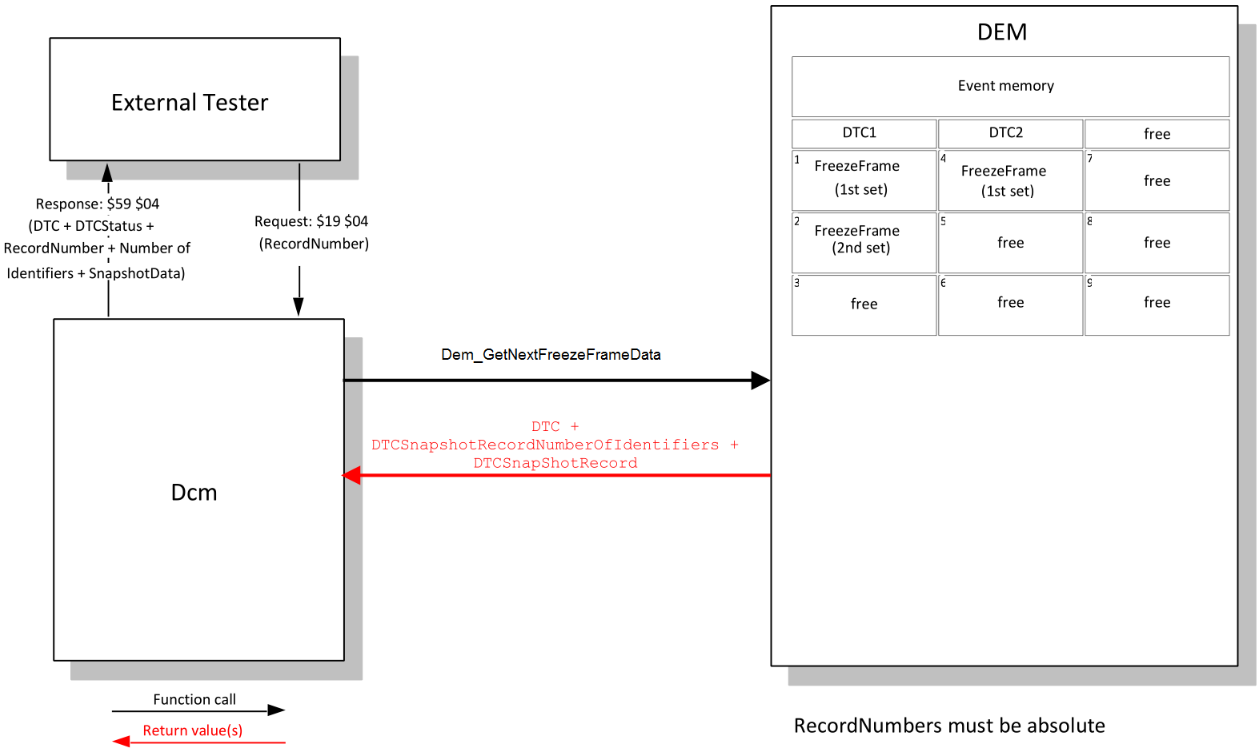


Figure 7.8: Request DTC Snapshot Record by Snapshot Record Number

[SWS_Dcm_00302] ⌈ When sending a positive response to UDS Service 0x19 with subfunction 0x04 or 0x18, the Dcm module shall use the data in the response message according to Table 7.18.⌋*()*

Parameter name	Value in Subservice 0x04	Value in Subservice 0x18
DTCAndStatusRecord	DTC from the request, statusOfDTC according to [SWS_Dcm_00383]	DTC from the request, statusOfDTC according to [SWS_Dcm_01147]
DTCSnapshotRecordNumber	The DTCSnapshotRecord Number is contained in the output buffer from the Dem_GetNextFreezeFrame() call. see [SWS_Dcm_00384]	The DTCSnapshotRecord Number is contained in the output buffer from the Dem_GetNextFreezeFrame() call. see [SWS_Dcm_00384].
DTCSnapshotRecordNumberOfIdentifiers / DTCSnapshotRecord	As defined in [SWS_Dcm_00384]	As defined in [SWS_Dcm_00384]

MemorySelection	n/a	From request

Table 7.18: Subfunction 0x04 and 0x18 response values

[SWS_Dcm_00383] ⌈ When responding to UDS Service 0x19 with subfunction 0x04, the Dcm module shall obtain the status of the DTC by first calling Dem_SelectDTC() with the following parameters:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef)
- DTC: DTC from the request
- DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY

and then Dem_GetStatusOfDTC() with ClientId = Client Id for this Dcm instance (see DcmDemClientRef) ⌋*(SRS_Diag_04058)*

[SWS_Dcm_01147] ⌈ When responding to UDS Service 0x19 with subfunction 0x18, the Dcm module shall obtain the status of the DTC by first calling Dem_GetStatusOfDTC() with the following parameters:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef
- DTC: DTC from the request
- DTCOrigin: Memory Selection from request + 0x0100

and then Dem_GetStatusOfDTC() with ClientId = DcmDemClientRef ⌋*(SRS_Diag_04058)*

[SWS_Dcm_00384] ⌈ Upon reception of UDS Service 0x019 with subfunction 0x04 or 0x18, the Dcm shall retrieve from the Dem the stored snapshot records for the requested DTC and origin. ⌋*(SRS_Diag_04058)*

[SWS_Dcm_00441] ⌈ The Dcm module shall obtain the size of the data returned by Dem in Dem_GetNextFreezeFrameData() call by using Dem_GetSizeOfFreezeFrameSelection(). ⌋*()*

To get the size of all FreezeFrame data, the Dcm module calls Dem_SelectFreezeFrameData() with RecordNumber set to 0xFF.

[SWS_Dcm_01220] ⌈ If Dem_GetNextFreezeFrameData() returns DEM_WRONG_DTC or DEM_WRONG_DTCORIGIN the Dcm shall send a NRC 0x31 (RequestOutOfRange) ⌋*()*

[SWS_Dcm_01430] ⌈ When responding to UDS Service 0x19 with subfunction 0x04, or 0x18, the Dcm shall collect the freeze frame data by first calling Dem_SelectFreezeFrameData() and then call Dem_GetNextFreezeFrameData() repeatedly until DEM_NO_SUCH_ELEMENT is returned. ⌋*()*

[SWS_Dcm_01224] ⌈ If at least one of the requested freeze frame data is supported, the Dcm shall send a positive response. Otherwise the Dcm shall send a NRC 0x31 (RequestOutOfRange). ⌋*()*

[SWS_Dcm_01246] ⌈ If Dem_GetSizeOfFreezeFrameSelection() returns DEM_WRONG_DTC, DEM_WRONG_DTCORIGIN or DEM_NO_SUCH_ELEMENT the Dcm shall send a NRC 0x31 (RequestOutOfRange). ⌋*()*

7.5.2.5.8 Subfunction 0x05

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:

- DTCStoredDataRecordNumber

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

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

[SWS_Dcm_00574] ⌈ When sending a positive response to UDS Service 0x19 with subfunction 0x05 and DTCStoredDataRecordNumber is 0x00, the Dcm module shall use the data in the response message according to Table 7.19. ⌋*()*

Parameter name	Value
DTCStoredDataRecordNumber	DTCStoredDataRecordNumber from request (0x00)
DTCAndStatusRecord	DTC according to [SWS_Dcm_01193], statusOfDTC according to [SWS_Dcm_00389]
DTCStoredDataRecordNumberOfIdentifiers / DTCStoredDataRecord	As defined in [SWS_Dcm_00388]

Table 7.19: Subfunction 0x05 response values

[SWS_Dcm_00388] ⌈ When responding to UDS Service 0x19 with subfunction 0x05 and DTCStoredDataRecordNumber is 0x00, the Dcm shall compose the OBD Freezeframe by looping all DcmDspPid and collecting all DcmDspPidData which are configured for service 0x02 by calling Dem_DcmReadDataOfOBDFreezeFrame() for the Data Element. The Dcm shall compose the DidId by adding 0xF400 to the Pid, and calculate padding and supported informations. ⌋*(SRS_Diag_04058)*

[SWS_Dcm_01193] ⌈ When responding to UDS Service 0x19 with subfunction 0x05 and DTCStoredDataRecordNumber is 0x00, the Dcm shall call Dem_DcmGetDTCOfOBDFreezeFrame() with FrameNumber 0x00 and DTCFormat DEM_DTC_FORMAT_UDS to retrieve the DTC of the provided FreezeFrame. ⌋*()*

[SWS_Dcm_00389] ⌈ When responding to UDS Service 0x19 with subfunction 0x05 and DTCStoredDataRecordNumber is 0x00, the Dcm module shall obtain the status of the DTC by first calling Dem_SelectDTC() with the following parameters:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef)

- DTC: DTC as defined in [SWS_Dcm_00388]
- DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY

and then Dem_GetStatusOfDTC() with the following parameter:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef)

⌋*(SRS_Diag_04058)*

7.5.2.5.9 Subfunctions 0x0B, 0x0C, 0x0D and 0x0E

An external test tool can request the first occurred or most recent failed or confirmed DTC and associated status, by sending the UDS Service request 0x19 including one of the following sub-functions 0x0B, 0x0C, 0x0D, 0x0E

[SWS_Dcm_00392] ⌈ When sending a positive response to UDS Service 0x19 with subfunction 0x0B, 0x0C, 0x0D or 0x0E, the Dcm module shall use the data in the response message according to Table 7.20.⌋*()*

Parameter name	Value
DTCStatusAvailabilityMask	DTCStatusAvailabilityMask (see [SWS_Dcm_00007])
DTCAndStatusRecord	The DTC is obtained according to [SWS_Dcm_00466], the StatusOfDtc is obtained according to [SWS_Dcm_00393]

Table 7.20: Subfunctions 0x0B, 0x0C, 0x0D and 0x0E response values

[SWS_Dcm_00393] ⌈ For the purpose of responding to UDS Service 0x19 with subfunctions 0x0B, 0x0C, 0x0D or 0x0E, the Dcm module shall obtain the StatusOfDtc by calling Dem_GetStatusOfDTC() with the following parameter values:

- ClientId :Client Id for this Dcm instance (see DcmDemClientRef
- DTC: the DTC value as defined in [SWS_Dcm_00466]
- DTCOrigin: DEM_DTC_ORIGIN_PRIMARY_MEMORY

⌋*(SRS_Diag_04058)*

[SWS_Dcm_00466] ⌈ For the purpose of responding to UDS Service 0x19 with subfunctions 0x0B, 0x0C, 0x0D or 0x0E, the Dcm shall obtain the DTC with Dem_GetDTCByOccurrenceTime() using the parameter values according to Table 7.21.⌋*()*

	reportFirstTest FailedDTC	reportFirst ConfirmedDTC	reportMost RecentTest FailedDTC	reportMost Recent ConfirmedDTC
	0x0B	0x0C	0x0D	0x0E
ClientId	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)

	reportFirstTest FailedDTC	reportFirst ConfirmedDTC	reportMost RecentTest FailedDTC	reportMost Recent ConfirmedDTC
DTCRequest	DEM_FIRST_ FAILED_DTC	DEM_FIRST_ DET_ CONFIRMED_ DTC	DEM_MOST_ RECENT_ FAILED_DTC	DEM_MOST_ REC_DET_ CONFIRMED_ DTC

Table 7.21: Dem_GetDTCByOccurrenceTime() parameters values for subfunctions 0x0B, 0x0C, 0x0D and 0x0E

[SWS_Dcm_00766] ⌈ If the Dcm received DEM_NO_SUCH_ELEMENT by calling Dem_GetDTCByOccurrenceTime it shall reply with a positive response and empty DTCAndStatusRecord. ⌋*()*

7.5.2.5.10 Subfunction 0x14

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

[SWS_Dcm_00464] ⌈ When sending a positive response to UDS Service 0x19 with subfunction 0x14, the Dcm module shall use the data in the response message according to Table 7.22.⌋*()*

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()

Table 7.22: Subfunction 0x14 response values

[SWS_Dcm_00465] ⌈ When responding to UDS Service 0x19 with subfunctions 0x14, the Dcm module shall obtain the DTCFaultCounter of every DTCs with status "prefailed" by repeatedly calling Dem_GetNextFilteredDTCAndFDC() after having configured the filter with Dem_SetDTCFilter() using the parameter values according to Table 7.23.⌋ *(SRS_Diag_04058)*

Parameter name	Value
ClientId	Client Id for this Dcm instance (see DcmDemClientRef)
DTCStatusMask	0x00
DTCFormat	DEM_DTC_FORMAT_UDS
DTCOrigin	PRIMARY_MEMORY
FilterWithSeverity	NO
DTCSeverityMask	Not relevant
FilterForFaultDetectionCounter	YES

Table 7.23: Dem_GetNextFilteredDTCAndFDC() parameters values for subfunctions 0x14

[SWS_Dcm_00681] ⌈ The Dcm module shall obtain the number of records that will be found using Dem_GetNextFilteredDTCAndFDC() by using Dem_GetNumberOfFilteredDTC(). ⌋*()*

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

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

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

[SWS_Dcm_01232] ⌈ If Dem_GetNextFilteredDTCAndFDC() returns DEM_NO_SUCH_ELEMENT and at least one matching element could be retrieved before, the Dcm shall send a positive response including these data elements. ⌋*()*

[SWS_Dcm_01233] ⌈ If Dem_GetNextFilteredDTCAndFDC() returns DEM_NO_SUCH_ELEMENT and no matching element could be retrieved before, the Dcm shall send a positive response only for service and subservice. ⌋*()*

7.5.2.5.11 Subfunction 0x42

UDS Service 0x19 with subfunction 0x42 requests WWH OBD DTCs matching a DTC status mask a severity mask record. The service request contains the following parameters:

- FunctionalGroupIdentifier
- DTCSeverityMask
- DTCStatusMask

[SWS_Dcm_01128] ⌈ The Dcm shall reject request messages for subFunction 0x42 with FunctionalGroupIdentifier unequal to 0x33 by returning NRC 0x31 (requestOutOfRange) ⌋*()*

[SWS_Dcm_01129] ⌈ When sending a positive response to UDS Service 0x19 with subfunction 0x42, the Dcm module shall use the data in the response message according to Table 7.24.⌋*()*

Parameter name	Value
FunctionalGroupIdentifier	0x33

Parameter name	Value
DTCStatusAvailabilityMask	Dem_GetDTCStatusAvailabilityMask (see [SWS_Dcm_00007])
DTCSeverityAvailabilityMask	Dem_GetDTCSeverityAvailabilityMask (see [SWS_Dcm_01127])
DTCFormatIdentifier	Dem_GetTranslationType (limited to values 0x04 and 0x02)
DTCAndSeverityRecord	As defined in [SWS_Dcm_01130]

Table 7.24: Subfunction 0x42 response values

[SWS_Dcm_01130] ⌈ When responding to UDS Service 0x19 with subfunction 0x42, the Dcm module shall obtain the DTCAndSeverityRecords by repeatedly calling Dem_GetNextFilteredDTCAndSeverity() after having configured the filter with Dem_SetDTCFilter() using the parameter values according to Table 7.25.⌋*()*

Parameter name	Value
ClientId	Client Id for this Dcm instance (see DcmDemClientRef)
DTCStatusMask	DTCStatusMask from request (see [SWS_Dcm_00700])
DTCFormat	DEM_DTC_FORMAT_UDS
DTCOrigin	DEM_DTC_ORIGIN_OBD_RELEVANT_MEMORY
FilterWithSeverity	YES
DTCSeverityMask	DTCSeverityMask from request
FilterForFaultDetectionCounter	NO

Table 7.25: Dem_GetNextFilteredDTCAndSeverity() parameters values for subfunctions 0x42

[SWS_Dcm_01131] ⌈ The return values of Dem_GetNextFilteredDTCAndSeverity shall be filled according to Table 7.26. ⌋*()*

Parameter name	Value
DTCSeverity	DTCSeverity
DTCHighByte (MSB)	DTC (high byte)
DTCMiddleByte	DTC (middle byte)
DTCLowByte	DTC (low byte)
statusOfDTC	DTCStatus

Table 7.26: Dem_GetNextFilteredDTCAndSeverity return values

Note: The Dcm module can get an indication of the number of records that will be found using Dem_GetNextFilteredDTCAndSeverity() by using Dem_GetNumberOfFilteredDTC().

7.5.2.5.12 Subfunction 0x55

With UDS Service 0x19 with sub-function 0x55 a client can retrieve a list of WWH-OBD DTCs with the "permanent DTC" status. The service request contains the following parameter:

- FunctionalGroupIdentifier

[SWS_Dcm_01343] ⌈ The Dcm shall only process request messages for sub-function 0x55 with FunctionalGroupIdentifier equal to 0x33. ⌋*()*

[SWS_Dcm_01344] ⌈ The Dcm shall reject request messages for sub-function 0x55 with FunctionalGroupIdentifier unequal to 0x33 by returning NRC 0x31 (RequestOutOfRange). ⌋*()*

[SWS_Dcm_01345] ⌈ When sending a positive response to UDS Service 0x19 with sub-function 0x55, the Dcm module shall use the following data in the response message according to Table 7.27. ⌋*()*

Parameter name	Value
FunctionalGroupIdentifier	0x33
DTCStatusAvailabilityMask	Dem_GetDTCStatusAvailabilityMask (see [SWS_Dcm_00007])
DTCFormatIdentifier	Dem_GetTranslationType (limited to values 0x04 and 0x02)
DTCAndStatusRecord	As returned by Dem_GetNextFilteredDTC()

Table 7.27: Subfunction 0x55 response values

Note : When responding to UDS Service 0x19 with sub-function 0x55, the Dcm module could obtain the DTCAndStatusRecords by repeatedly calling Dem_GetNextFilteredDTC() after having configured the filter with Dem_SetDTCFilter() using the parameter values according to Table 7.28.

Parameter name	Value
ClientId	See DcmDemClientRef
DTCStatusMask	0x00
DTCFormat	DEM_DTC_FORMAT_UDS
DTCOrigin	DEM_DTC_ORIGIN_PERMANENT_MEMORY
FilterWithSeverity	NO
DTCSeverityMask	Not relevant
FilterForFaultDetectionCounter	NO

Table 7.28: Dem_GetNextFilteredDTCAndSeverity() parameters values for subfunctions 0x42

The Dcm module can get an indication of the number of records that will be found using Dem_GetNextFilteredDTC() by using Dem_GetNumberOfFilteredDTC().

[SWS_Dcm_01346] ⌈ When responding to UDS Service 0x19 with sub-function 0x55 and Dem_GetTranslationType returns a Dem_DTCTranslationFormatType different to 0x02 (DEM_DTC_TRANSLATION_SAEJ1939_73) or 0x04 (DEM_DTC_TRANSLATION_J2012DA_FORMAT_04), the Dcm module shall return NRC 0x10 (generalReject). ⌋*()*

7.5.2.6 Service 0x22 - ReadDataByIdentifier

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

[SWS_Dcm_01335] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), if the number of requested DID exceeds the configured maximum number of data identifiers (refer to configuration parameter DcmDspMaxDidToRead), the Dcm module shall send NRC 0x13 (Incorrect message length or invalid format) ⌋ *()*

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

[SWS_Dcm_00438] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22) , for every requested DID the Dcm module shall check if the DID is supported (see configuration parameter DcmDspDid and DcmDspDidRange) If none of the requested DIDs is supported, the Dcm module shall send NRC 0x31 (Request out of range). ⌋ *()*

[SWS_Dcm_00651] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22) with DID in the range 0xF200 to 0xF3FF, the Dcm module shall check if the DID can be dynamically defined (the DcmDspDidInfo it references has the DcmDspDidDynamicallyDefined set to true). If yes, if this DID has not been dynamically defined yet by calls to the DynamicallyDefineDataIdentifier (0x2C) service, i.e. it has no data sources defined, the Dcm module shall send NRC 0x31 (Requestout of range) ⌋ *()*

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

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

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

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

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

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

[SWS_Dcm_00433] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID the Dcm module shall check if the DID has a Read access configured (see configuration parameter DcmDspDidRead in DcmDspDidInfo). If none of the DID has a Read access, the Dcm module shall send NRC 0x31 (Request out of range). ⌋ *()*

[SWS_Dcm_00434] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID the Dcm module shall check if the DID can be read in the current session (see configuration parameter DcmDspDidReadSessionRef). If none of the DID can be renden in the current session, the Dcm module shall send a NRC 0x31 (RequestOutOfRange). ⌋ *()*

[SWS_Dcm_00435] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID the Dcm module shall check if the DID can be read in the current security level (see configuration parameter DcmDspDidReadSecurityLevelRef). If not, the Dcm module shall send NRC 0x33 (Security access denied). ⌋ *()*

[SWS_Dcm_00819] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID the Dcm module shall check if the DID can be read in the current mode condition (according to the configuration parameter DcmDspDidReadModeRuleRef). If not, the Dcm module shall send the calculated negative response of the referenced DcmModeRule. ⌋ *()*

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

[SWS_Dcm_CONSTR_6023] DcmDspDidRef shall not reference the same DID reference twice ⌈ DcmDspDid container shall not include the same DcmDspDidRef parameters more than once. ⌋ *()*

[SWS_Dcm_CONSTR_6057] Dependency for DcmDspDataEcuSignal ⌈ DcmDspDataEcuSignal shall be only present if DcmDspDataUsePort is set to USE_ECU_SIGNAL. ⌋ *()*

[SWS_Dcm_CONSTR_6058] Dependency for DcmDspDataEndianness ⌈ In case DcmDspDataEndianness is not configured, the DcmDspDataDefaultEndianness shall be used instead. ⌋ *()*

[SWS_Dcm_CONSTR_6061] Dependency for DcmDspDataReadDataLengthFnc ⌈ DcmDspDataReadDataLengthFnc shall be only present if:

- DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC_ERROR

⌋()

[SWS_Dcm_CONSTR_6062] Dependency for DcmDspDataReadFnc ⌈ DcmDspDataReadFnc shall be only present if:

- DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC_ERROR

⌋()

7.5.2.6.1 Non-OBD DID

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

[SWS_Dcm_00439] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID outside the OBD range (F400-F8FF), the Dcm module shall request the application if the DID can be read by calling the configured function (if parameter DcmDspDataUsePort set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC or USE_DATA_ASYNCH_FNC_ERROR; 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 or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR). If not (one function returns E_NOT_OK) , the Dcm module shall send a negative response with NRC set to value from the parameter "ErrorCode" of DcmDspDataConditionCheckReadFnc function or ConditionCheckRead operation. ⌋()

[SWS_Dcm_00436] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID outside the OBD range (F400-F8FF), the Dcm module shall for each signal (DcmDspDidSignal) with a dynamic data length (DcmDspDataType is set to UINT8_DYN): call either the configured function DcmDspDataReadDataLengthFnc (if parameter DcmDspDataUsePort set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC or USE_DATA_ASYNCH_FNC_ERROR) or the associated ReadDataLength operation (if parameter DcmDspDataUsePort set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR) to get the data length in byte. ⌋()

[SWS_Dcm_00437] ⌈ After all verification (see [SWS_Dcm_00433], [SWS_Dcm_00434], [SWS_Dcm_00435] and [SWS_Dcm_00436]) the

Dcm module shall get for every requested DID outside the OBD range (F400-F8FF), all the data values by calling all the configured function (if parameter DcmDspDataUsePort set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC or USE_DATA_ASYNCH_FNC_ERROR; 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 USE_DATA_ASYNCH_CLIENT_SERVER_ERROR) or read all the associated SenderReceiver interfaces (if parameter DcmDspDataUsePort set to USE_DATA_SENDER_RECEIVER or to USE_DATA_SENDER_RECEIVER_AS_SERVICE). ⌋*()*

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

Note : For more information, refer to [13, SWS-NVRAMManager].

[SWS_Dcm_00638] ⌈ To serialize the required AUTOSAR data types (signed- and unsigned integer) into the response message of ReadDataByIdentifier responses, the target endianness configured in DcmDspDataEndianness shall be considered for DcmDspData elements having DcmDspDataUsePort set to USE_DATA_SENDER_RECEIVER, USE_DATA_SENDER_RECEIVER_AS_SERVICE, USE_ECU_SIGNAL. ⌋*()*

[SWS_Dcm_CONSTR_6070] Dependency for DcmDspDataEndianness ⌈ In case DcmDspDataEndianness is not present, the DcmDspDataDefaultEndianness shall be used instead. ⌋*()*

7.5.2.6.2 OBD DID

[SWS_Dcm_00481] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID inside the OBD range (F400-F4FF), the Dcm module shall get the DID value as defined for OBD Service $01 (see [SWS_Dcm_00407], [SWS_Dcm_00408], [SWS_Dcm_00943], [SWS_Dcm_00621], [SWS_Dcm_00622], [SWS_Dcm_00623], [SWS_Dcm_00944] and [SWS_Dcm_00718]), if DcmDspEnableObdMirror is set to true. ⌋*()*

[SWS_Dcm_00482] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID inside the OBD Monitor range (F600-F6FF), the Dcm module shall get the DID value as defined for OBD Service $06 (see [SWS_Dcm_00957], [SWS_Dcm_00958], [SWS_Dcm_00945] and [SWS_Dcm_00956]) ⌋*()*

[SWS_Dcm_00483] ⌈ On reception of the UDS Service ReadDataByIdentifier (0x22), for every requested DID inside the OBD InfoType range (F800-F8FF), the Dcm module shall get the DID value as defined for OBD Service $09 (see [SWS_Dcm_00422], [SWS_Dcm_00423] and [SWS_Dcm_00949] without including the number of data items within the response, if DcmDspEnableObdMirror is set to true. ⌋*()*

[SWS_Dcm_01195] ⌈ If DcmDspEnableObdMirror is set to true, an explicitly configured DID inside the OBD range (F400-F4FF) and the OBD InfoType range (F800-F8FF) shall use the UDS interface. ⌋*()*

[SWS_Dcm_01197] ⌈ If DcmDspEnableObdMirror is set to FALSE, all requests within the OBD DID range shall use the UDS interface. ⌋*()*

If DcmDspEnableObdMirror is set to FALSE ([SWS_Dcm_01197]) or the DID is explicitly configured inside the OBD PID range (F400-F4FF) ([SWS_Dcm_01195]), the access to the OBD data shall be given in the following way:

[SWS_Dcm_01379] ⌈ On reception of an UDS Service ReadDataByIdentifier (0x22) request with only ”availability OBDDataIdentifier” as parameter, the Dcm shall respond with the corresponding supported (=configured) DIDs in the OBD range (F400-F4FF). ⌋*()*

[SWS_Dcm_01380] ⌈ On reception of an UDS Service ReadDataByIdentifier (0x22) request with only OBDDataIdentifier that are not ”availability OBDDataIdentifier”, the Dcm shall obtain the current value of these OBDDataIdentifier by invoking the configured Xxx_ReadData() functions for every data of the OBDDataIdentifier and shall return these values as response to Service 0x22. ⌋*()*

[SWS_Dcm_01381] ⌈ On reception of an UDS Service ReadDataByIdentifier (0x22) request with a mixture of ”availability OBDDataIdentifier” and not ”availability OBDDataIdentifier”, this request shall be ignored by the Dcm. ⌋*()*

[SWS_Dcm_01382] ⌈ If an OBDDataIdentifier contains support information (presence of DcmDspDidDataSupportInfo container), the Dcm shall add the support information in the diagnostic response. ⌋*()*

[SWS_Dcm_01383] ⌈ If an OBDDataIdentifier contains support information (presence of DcmDspDidDataSupportInfo container), the Dcm shall calculate the support information value according to the available data for this DID: for every DcmDspData container existing for this DID, the associated support information bits, referenced in DcmDspDidDataSupportInfo, shall be set to one. ⌋*()*

[SWS_Dcm_01384] ⌈ When responding to UDS Service ReadDataByIdentifier (0x22) with OBDDataIdentifier, the Dcm shall put fill-bytes between DcmDspData in the OBDDataIdentifier whenever content bytes are missing, in order to fit to the DID size (see configuration parameter DcmDspDidSize). ⌋*()*

[SWS_Dcm_01385] ⌈ The Dcm shall set the fill bytes to 0x00. ⌋*()*

[SWS_Dcm_01386] ⌈ To serialize the required AUTOSAR data types (signed and unsigned integer) into the response message of ReadDataByIdentifier (0x22) OBDDataIdentifier responses the target endianness configured in DcmDspDataEndianness shall be considered for DcmDspData elements having DcmDspDataUsePort set to {USE_DATA_SENDER_RECEIVER, USE_DATA_SENDER_RECEIVER_AS_SERVICE, USE_ECU_SIGNAL}. In case DcmDspDataEndianness is not present, the DcmDspDataDefaultEndianness shall be used instead. ⌋*()*

If DcmDspEnableObdMirror is set to FALSE or the DID is explicitly configured inside the OBD InfoType range (F800-F8FF), the access to the OBD data shall be given in the following way:

[SWS_Dcm_01387] ⌈ On reception of an UDS Service ReadDataByIdentifier (0x22) request with one or more "availability OBDInfoTypeDataIdentifier" as parameter, the Dcm module shall respond with the corresponding supported (=configured) DIDs in the OBD range (F800-F8FF). ⌋*()*

[SWS_Dcm_01388] ⌈ On reception of an UDS Service ReadDataByIdentifier (0x22) request with "availability OBDInfoTypeDataIdentifier" together with other OBDInfoTypeDataIdentifier as parameter, the Dcm module shall ignore the request. ⌋*()*

[SWS_Dcm_01389] ⌈ On reception of an UDS Service ReadDataByIdentifier (0x22) request with an OBDInfoTypeDataIdentifier that is not an "availability OBDInfoTypeDataIdentifier", the Dcm module shall obtain the value of this OBDInfoTypeDataIdentifier by invoking all the configured Xxx_ReadData() function for every data of this OBDInfoTypeDataIdentifier and shall return the value as response to Service 0x22. ⌋*()*

7.5.2.7 Service 0x24 - ReadScalingDataByIdentifier

[SWS_Dcm_00258] ⌈ The Dcm module shall implement the UDS Service ReadScalingDataByIdentifier (0x24) ⌋*()*

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

- The 2-byte DID (see configuration parameter DcmDspDidIdentifier)
- For every data of the DID :
- The function GetScalingInformation (see configuration parameters DcmDspDataGetScalingInfoFnc and DcmDspDataUsePort)
- The length of the ScalingInfo returned by the GetScalingInformation function (see configuration parameter DcmDspDataScalingInfoSize)

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

[SWS_Dcm_CONSTR_6060] Dependency for DcmDspDataGetScalingInfoFnc
⌈ DcmDspDataGetScalingInfoFnc shall be only present if:

- DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC_ERROR

⌋()

7.5.2.8 Service 0x27 - SecurityAccess

[SWS_Dcm_00252] ⌈ The Dcm module shall implement the UDS Service SecurityAccess (0x27) ⌋*(SRS_Diag_04005)*

The purpose of this service is to provide a means to access data and/or diagnostic services, which have restricted access for security, emissions, or safety reasons.

[SWS_Dcm_00321] ⌈ If the request length is correct, the DSP submodule shall check if the requested subfunction value (access type) is configured in the ECU (see configuration parameter DcmDspSecurityLevel). If the requested subfunction value is not configured, the DSP submodule shall trigger a negative response with NRC 0x12 (SubFunction not supported). ⌋*()*

[SWS_Dcm_00323] ⌈ If the requested subfunction value is configured and a service with subfunction type "requestSeed "(= odd value) has been received and if the requested access type is already active (see Dcm_GetSecurityLevel), the DSP submodule shall set the seed content to 0x00. ⌋*()*

[SWS_Dcm_00324] ⌈ In the other case than the one described in [SWS_Dcm_00323] (access type is not active or "send key" request), if DcmDspSecurityUsePort is set to USE_ASYNCH_CLIENT_SERVER, the DSP submodule shall call the configured operation Xxx_GetSeed() (in case "request seed" is received) or Xxx_CompareKey() (in case "send key" is received). ⌋*()*

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

Note : If the static seed mechanism is used, the processing needs to be done by the application implementing the Xxx_GetSeed() and Xxx_CompareKey() functions.

[SWS_Dcm_CONSTR_6077] Dependency for DcmDspSecurityGetSeedFnc ⌈ DcmDspSecurityGetSeedFnc shall be present only if DcmDspSecurityUsePort is set to USE_ASYNCH_FNC. ⌋*()*

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

[SWS_Dcm_CONSTR_6075] Dependency for DcmDspSecurityCompareKeyFnc ⌈ DcmDspSecurityCompareKeyFnc shall be configured only if DcmDspSecurityUsePort is set to USE_ASYNCH_FNC. ⌋*()*

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),
- ExceededNumberOfAttempts (NRC 0x36), when number of attempts to get security access reaches or exceeds the configured limit (see configuration parameter DcmDspSecurityNumAttDelay), and
- InvalidKey (NRC 0x35), when invalid key is send at "send key".

[SWS_Dcm_00325] ⌈ If the operation CompareKey() returns E_OK, the DSP submodule shall set the new access type with DslInternal_SetSecurityLevel()(see the conversion formula given in ECUC_Dcm_00754 DcmDspSecurityLevel). ⌋*()*

[SWS_Dcm_01397] ⌈ If Xxx_CompareKey() returns value DCM_E_COMPARE_KEY_FAILED, the Dcm shall increment the attempt counter of the security level for which the sendKey subfunction request failed. ⌋*()*

[SWS_Dcm_00660] ⌈ If Xxx_CompareKey() returns value DCM_E_COMPARE_KEY_FAILED and if the number of failed attempts to enter the requested security level (AttemptCounter) is less than the value configured for the DcmDspSecurityNumAttDelay parameter of the requested security level, the Dcm module shall send a negative response with NRC 0x35 (InvalidKey) and shall not change the Dcm internal security level. ⌋*()*

[SWS_Dcm_01349] ⌈ If Xxx_CompareKey() returns value DCM_E_COMPARE_KEY_FAILED and if the number of failed attempts to enter the requested security level (AttemptCounter) is greater or equal than the value configured for the DcmDspSecurityNumAttDelay parameter of the requested security level, the Dcm module shall start the SecurityDelayTimer with the value configured in DcmDspSecurityDelayTime for the SecurityLevel which was requested in the failed request, send a negative response with NRC 0x36 (exceededNumberOfAttempts) and shall not change the Dcm internal security level. ⌋*()*

[SWS_Dcm_01150] ⌈ If Xxx_CompareKey() returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the ErrorCode parameter value and shall not increment the attempt counter or change the Dcm internal security level. ⌋*()*

[SWS_Dcm_01350] ⌈ While the SecurityDelayTimer of SecurityLevel is not yet elapsed, the Dcm module shall send a negative response with NRC 0x37 (required-

TimeDelayNotExpired) on a SecurityAccess (0x27) requestSeed subfunction request for that Security Level. ⌋*()*

[SWS_Dcm_00659] ⌈ If Xxx_GetSeed() returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the ErrorCode parameter value. ⌋*()*

7.5.2.9 Service 0x28 - CommunicationControl

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

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

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

2. call the Api BswM_Dcm_CommunicationMode_CurrentState with the parameters NetworkHandleType and Dcm_CommunicationModeType corresponding to the communicationType and controlType parameter from the CommunicationControl request (see Dcm_CommunicationModeType definition).

⌋*()*

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

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

2. call the Api BswM_Dcm_CommunicationMode_CurrentState with the parameters NetworkHandleType and Dcm_CommunicationModeType corresponding to the communicationType and controlType parameter from the CommunicationControl request (see Dcm_CommunicationModeType definition)

⌋*()*

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

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

⌋*()*

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

[SWS_Dcm_00753] ⌈ In case that the referenced ModeRule (see ECUC_Dcm_00943) is not fulfilled anymore for a NetworkHandle which is currently in a state other than DCM_ENABLE_RX_TX_NORM_NM, the Dcm shall:

1. switch the mode group Dcm_CommunicationControl_<Network> ModeDeclarationGroupPrototype toDCM_ENABLE_RX_TX_NORM_NM
2. call BswM_Dcm_CommunicationMode_CurrentState with the parameters NetworkHandleType set to the corresponding NetworkHandle of the network and RequestedCommunicationMode set to DCM_ENABLE_RX_TX_NORM_NM

⌋*()*

[SWS_Dcm_00860] ⌈ For a NetworkHandle which is currently in a state other than DCM_ENABLE_RX_TX_NORM_NM if the Dcm is transitioning to default session or upon any diagnostic session change where the new session does not support UDS Service CommunicationControl anymore, the Dcm shall:

1. switch the mode group Dcm_CommunicationControl_<Network> ModeDeclarationGroupPrototype to DCM_ENABLE_RX_TX_NORM_NM
2. call BswM_Dcm_CommunicationMode_CurrentState with the parameters NetworkHandleType set to the corresponding NetworkHandle of the network and RequestedCommunicationMode set to DCM_ENABLE_RX_TX_NORM_NM

⌋*()*

Note: the NetworkHandles to be considered are all ComM channels which are referenced from either DcmDspSpecificComMChannelRef, DcmDspAllComMChannelRef or DcmDspComControlSubNodeComMChannelRef.

[SWS_Dcm_01077] ⌈ If a CommunicationControl Request with the sub-function "enableRxAndDisableTxWithEnhancedAddressInformation" is received, the Dcm shall check the "nodeIdentification-Number" listed as DcmDspComControlSubNodeId and for the referenced network (see DcmDspComControlSubNodeComMChannelRef), it shall do the followings:

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).

The analogue controlType enableRxAndDisableTx shall be used with the the following existing Dcm_CommunicationModeType values:

- DCM_ENABLE_RX_DISABLE_TX_NORM
- DCM_ENABLE_RX_DISABLE_TX_NM
- DCM_ENABLE_RX_DISABLE_TX_NORM_NM.

⌋*()*

[SWS_Dcm_01078] ⌈ The Dcm shall trigger a negative response with NRC 0x31 (RequestOutOfRange), if a CommunicationControl Request with the sub-function "enableRxAndDisableTxWithEnhancedAddressInformation" and a "nodeIdentification-Number" which is not listed as DcmDspComControlSubNodeId is received. ⌋*()*

[SWS_Dcm_01079] ⌈ If a CommunicationControl Request with the sub-function "enableRxAndTxWithEnhancedAddressInformation" is received, the Dcm shall check the "nodeIdentification-Number" listed as DcmDspComControlSubNodeId and for the referenced network (see DcmDspComControlSubNodeComMChannelRef) it shall do the followings:

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).

The analogue controlType enableRxAndTx shall be used with this the following existing Dcm_CommunicationType values :

- DCM_ENABLE_RX_TX_NORM
- DCM_ENABLE_RX_TX_NM
- DCM_ENABLE_RX_TX_NORM_NM.

⌋*()*

[SWS_Dcm_01080] ⌈ The Dcm shall trigger a negative response with NRC 0x31 (RequestOutOfRange), if a CommunicationControl Request with the sub-function "enableRxAndTxWithEnhancedAddressInformation" and a "nodeIdentification-Number" which is not listed as DcmDspComControlSubNodeId is received. ⌋*()*

[SWS_Dcm_01081] ⌈ If DcmDspComControlSubNodeUsed is set to FALSE the subsystem (DcmDspComControlSubNode) is not available in this configuration. ⌋*()*

[SWS_Dcm_01082] ⌈ If DcmDspComControlSubNodeUsed is set to TRUE the subsystem (DcmDspComControlSubNode) is available in this configuration. ⌋*()*

Note : Condition checks (i.e. NRC 22 checks) on CommunicationType and NetworkType as well as check of CommunicationType support (i.e. NRC 0x31 check for CommunicationType) are not directly supported by the Dcm. Supplier/manufacturer notifications can be used.

7.5.2.10 Service 0x2A - ReadDataByPeriodicIdentifier

[SWS_Dcm_00254] ⌈ The DSP submodule shall implement the UDS Service ReadDataByPeriodicIdentifier (0x2A) ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01093] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), the Dcm module shall check the request minimum length. If length of the request is wrong, the Dcm module shall send a NRC 0x13 (Incorrect message length or invalid format). ⌋*(SRS_Diag_04215)*

[SWS_Dcm_00721] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), for every requested periodicDIDs, 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 (RequestOutOfRange). ⌋*(SRS_Diag_04215)*

[SWS_Dcm_00722] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), for every requested periodicDIDs, 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). ⌋*(SRS_Diag_04215)*

[SWS_Dcm_00820] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), for every requested periodicDIDs, 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 ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01097] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), if verification has been successfully done ([SWS_Dcm_00721], [SWS_Dcm_00722] and [SWS_Dcm_00820]), and if the request contains one or more dynamically defined DID(s), the Dcm module shall do the session, security and mode dependencies checks for all source data in case the configuration parameter DcmDspDDDIDcheckPerSourceDID is set to TRUE. ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01098] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), for every requested periodicDIDs, the Dcm module shall invoke the ConditionCheckRead operation (or the respective C-Function) if configured. In case of a negative result, the returned ErrorCode shall be used as final negative response code. ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01099] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), for every requested periodicDIDs, with a configured dynamic length the Dcm module shall invoke the ReadDataLength operation (or the respective C-Function) to retrieve the length of the periodicDID. This length is valid for each ReadData operation till the periodicDID is removed from the scheduler or updated via a new request. This length shall further be used to check against the UUDT size. ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01100] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmissionMode different than stopSending, the Dcm shall do the verification for session, security and mode rule. ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01426] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmissionMode = stopSending, the Dcm shall skip the verification for security and mode rule. ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01427] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmissionMode = stopSending and no periodicDataIdentifier in the request, the Dcm shall stop all scheduled periodicDataIdentifier transmissions. ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01428] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmissionMode = stopSending and at least one periodicDataIdentifier is in the request, the Dcm shall stop the scheduled periodic data transmissions for all requested and in the current session supported periodicDataIdentifiers. ⌋*(SRS_Diag_04215)*

[SWS_Dcm_00716] ⌈ To serialize the required AUTOSAR data types (signed- and unsigned integer) into the response message of ReadDataByPeriodicIdentifier responses the target endianness configured in DcmDspDataEndianness shall be considered for DcmDspData elements having DcmDspDataUsePort set to USE_DATA_SENDER_RECEIVER, USE_DATA_SENDER_RECEIVER_AS_SERVICE, USE_ECU_SIGNAL. In case DcmDspDataEndianness is not present, the DcmDspDataDefaultEndianness shall be used instead. ⌋*(SRS_Diag_04215)*

[SWS_Dcm_00843] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A),the Dcm module shall check if the periodicDataIdentifiers requested in a single request do not exceed the configured DcmDspMaxPeriodicDidToRead (maximum length check). Otherwise (in case the number of elements is exceeded) the Dcm module shall send a NRC 0x13 (Incorrect message length or invalid format). ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01094] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier(0x2A), the Dcm module shall check if the transmissionMode is supported, otherwise the Dcm module shall send a NRC 0x31(Request out of range). ⌋*(SRS_Diag_04215)*

Note: With UDS Service ReadDataByPeriodicIdentifier (0x2A), the tester can start one or more periodicDIDs.

Note: A request of UDS Service ReadDataByPeriodicIdentifier will contain DIDs represented by only one byte, the low byte of the configured DID. In all cases the complete DID will be constructed by adding 0xF2 as high byte.

[SWS_Dcm_01095] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), for every requested periodicDIDs, the Dcm module shall check if the periodicDID is supported (see configuration parameter DcmDspDid). If none of the periodicDIDs are supported, the Dcm module shall send NRC 0x31 (Request out of range). ⌋ *(SRS_Diag_04215)*

[SWS_Dcm_01096] ⌈ If a DID is set as unused (DcmDspDidUsed set to FALSE), the Dcm shall consider the DID as not supported. ⌋*(SRS_Diag_04215)*

[SWS_Dcm_00851] ⌈ On reception of the UDS Service ReadDataByPeriodicIdentifier(0x2A) with transmissionMode different than 0x04 "stopSending", the Dcm module shall check if all requested periodicDataIdentifiers not currently in the periodic scheduler can be added to the scheduler considering the free space of the scheduler (maximum size is defined by configuration parameter DcmDspMaxPeriodicDidScheduler). Otherwise (in case the requested periodicDataIdentifiers can not be added to the scheduler) the Dcm module shall send a NRC 0x31 (RequestOutOfRange). ⌋ *(SRS_Diag_04215)*

Note : To optimize the resource consumption AUTOSAR has chosen a simplified approach to validate the request message. ISO recommends to check the size only for currently supported dataIdentifers.

7.5.2.10.1 Scheduler PeriodicTransmission

Note: The periodic responses will only contain the DID and its data, and no Service ID (no A_PCI byte).

[SWS_Dcm_01101] ⌈ All periodic responses (scheduled responses, not the initial response) will use dedicated IF-PDU's and transmission will be done through PduR. Each time PduR_DcmTransmit is called the data pointer shall be valid. ⌋*(SRS_Diag_04215)*

Note : Only UUDT messages (IF-PDUs) are supported

[SWS_Dcm_01102] ⌈ After triggering the transmission request to the PduR the corresponding periodicDID counter shall be reloaded. ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01103] ⌈ The Dcm shall not trigger a transmission request to the PduR unless the transmit confirmation for the previously transmitted periodic response is received. ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01104] ⌈ In case of multiple configured UUDT messages, the Dcm shall use always the same order of periodicDIDs per client. Transmission errors shall not influence this order, the Dcm shall continue to retry the transmission. The Dcm shall consider the priority inversion of message transmission as well. ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01105] ⌈ After the periodicDIDs are started, initial request was responded positively, no negative response will be sent for those periodicDID's (when periodically triggered). ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01106] ⌈ Each time the counter of a periodicDataIdentifiers elapses, the Dcm shall retrieve the data via the ReadData operation (or respective C-Function) without validating the other conditions (i.e. session, security, mode dependencies, ConditionCheckRead and ReadDataLength). ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01107] ⌈ When the diagnostic session changes to DefaultSession, any scheduled periodic DID shall be stopped (see [SWS_Dcm_01113], [SWS_Dcm_01114], [SWS_Dcm_01115], [SWS_Dcm_01116], [SWS_Dcm_01117] and [SWS_Dcm_01118]). ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01108] ⌈ When the diagnostic session changes to a non-defaultSession, any scheduled periodic DID that was restricted by security access shall be stopped (see [SWS_Dcm_01113], [SWS_Dcm_01114], [SWS_Dcm_01115], [SWS_Dcm_01116], [SWS_Dcm_01117] and [SWS_Dcm_01118]). ⌋*(SRS_Diag_04215)*

[SWS_Dcm_01109] ⌈ When the diagnostic session changes to a non-defaultSession, any scheduled periodic DID that is not supported in the new session shall be stopped (see [SWS_Dcm_01113], [SWS_Dcm_01114], [SWS_Dcm_01115], [SWS_Dcm_01116], [SWS_Dcm_01117] and [SWS_Dcm_01118]). ⌋*(SRS_Diag_04215)*

Note: The rate for a specific transmissionMode (DcmDspPeriodicTransmissionSlowRate, DcmDspPeriodicTransmissionMediumRate, DcmDspPeriodicTransmissionFastRate) is defined as the time between any two consecutive response messages with the same periodicDataIdentifier, when only a single periodicDID is scheduled. If multiple periodicDIDs are scheduled concurrently, the effective period between the same periodicDataIdentifier will vary based upon the following design parameters:

- The main function recurrence (see configuration parameter DcmTaskTime)
- The number of available periodic connections
- The number of periodicDIDs that can be scheduled concurrently (see configuration parameter DcmDspMaxPeriodicDidScheduler).

[SWS_Dcm_01110] ⌈ On any security level change, the Dcm shall stop any scheduled periodic DID (see [SWS_Dcm_01113], [SWS_Dcm_01114], [SWS_Dcm_01115], [SWS_Dcm_01116], [SWS_Dcm_01117] and [SWS_Dcm_01118]), that was restricted by security access, but not supported by the new security level anymore. ⌋ *(SRS_Diag_04215)*

[SWS_Dcm_01111] ⌈ On any Session change, the Dcm shall stop any scheduled periodic DDDID (see [SWS_Dcm_01114], [SWS_Dcm_01116], [SWS_Dcm_01117] and [SWS_Dcm_01118]), that contains source data, not supported in the current session or requires security access, in case the configuration parameter DcmDspDDDIDcheckPerSourceDID is set to TRUE ⌋ *(SRS_Diag_04215)*

[SWS_Dcm_01112] ⌈ On any security level change, the Dcm shall stop any scheduled periodic DDDID (see [SWS_Dcm_01114], [SWS_Dcm_01116], [SWS_Dcm_01117] and [SWS_Dcm_01118]), that contains source data, not supported in the current security level, in case the configuration parameter DcmDspDDDIDcheckPerSourceDID is set to TRUE. ⌋ *(SRS_Diag_04215)*

[SWS_Dcm_01113] ⌈ On a static periodic DID stop event, the Dcm shall no longer call the "ReadData" function of this DID's data (i.e. periodic DID is removed from scheduler). ⌋ *(SRS_Diag_04215)*

[SWS_Dcm_01114] ⌈ On a dynamically defined periodic DID stop event, the Dcm shall no longer call any source data "ReadMemory" or "ReadData" function of the periodic DDDID (i.e. periodic DDDID is removed from scheduler). ⌋ *(SRS_Diag_04215)*

[SWS_Dcm_01115] ⌈ On a static periodic DID stop event, after the asynchronous call of its data service port has already been initiated (i.e. its "ReadData" port operation already returned E_PENDING), the corresponding service port shall be immediately aborted by signaling OpStatus=DCM_CANCEL. ⌋ *(SRS_Diag_04215)*

[SWS_Dcm_01116] ⌈ On a dynamically defined periodic DID stop event, after the asynchronous call of its source data service port/callout has already been initiated (e.g. a "ReadMemory" callout already returned DCM_READ_PENDING), the corresponding service port/callout shall be immediately aborted by signaling OpStatus=DCM_CANCEL. ⌋ *(SRS_Diag_04215)*

[SWS_Dcm_01117] ⌈ On a periodic DID stop event, all its data in a Dcm queue (waiting to be transmitted) is cleared. ⌋ *(SRS_Diag_04215)*

[SWS_Dcm_01118] ⌈ On a periodic DID stop event, Dcm will NOT try to cancel any data transmission already initiated by the call of PduR_DcmTransmit. ⌋ *(SRS_Diag_04215)*

7.5.2.11 Service 0x2C - DynamicallyDefineDataIdentifier

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

The DynamicallyDefineDataIdentifier service is implemented internally in Dcm module.
[SWS_Dcm_00866] ⌈ If DcmDDDIDStorage configuration parameter is set to FALSE, the Dcm shall initialize all DDDIDs as not present at power-up Dcm_Init). ⌋*()*

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

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

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

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

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

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

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

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

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

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

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

In case of memory address(es), on reception of ReadDataByIdentifier or ReadDataByPeriodicIdentifier request for a dynamically defined DID, the Dcm will use the callout Dcm_ReadMemory for all contained memory addresses to access the data.

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

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

7.5.2.12 Service 0x2E - WriteDataByIdentifier

[SWS_Dcm_00255] ⌈ The Dcm module shall implement the UDS Service WriteDataByIdentifier (0x2E) of the Unified Diagnostic Services. ⌋*()*

When using Service 0x2E, the request of the tester contains a 2-byte DID and a dataRecord with the data to be written. The configuration of the Dcm contains a list of supported DIDs and defines for each configured DID:

- The 2-byte DID (see configuration parameter DcmDspDidIdentifier)
- For every data of the DID:
  - The function WriteData to be used for this data (see configuration parameters DcmDspDataWriteFnc and DcmDspDataUsePort)

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

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

[SWS_Dcm_00468] ⌈ On reception of the UDS Service WriteDataByIdentifier (0x2E), the Dcm module shall check if the DID has a Write access configured (see configuration parameter DcmDspDidWrite in DcmDspDidInfo). If not, the Dcm module shall send NRC 0x31 (Request out of range). ⌋*()*

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

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

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

[SWS_Dcm_00473] ⌈ On reception of the UDS Service WriteDataByIdentifier (0x2E), if all signals (DcmDspDidSignal) of the DID have fixed length (DcmDspDataType is different than UINT8_DYN), the Dcm module shall check if the received data length corresponds to the DID data length (addition of all DcmDspDataByteSize). ⌋*()*

[SWS_Dcm_00395] ⌈ After all verifications (see [SWS_Dcm_00467], [SWS_Dcm_00468], [SWS_Dcm_00469], [SWS_Dcm_00470], [SWS_Dcm_00473]) the Dcm module shall write all the signals (DcmDspDidSignal) of the DID by either calling the configured function DcmDspDataWriteFnc (if parameter DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC or USE_DATA_ASYNCH_FNC or USE_DATA_ASYNCH_FNC_ERROR) or the associated WriteData operations (if parameter DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR) or the associated SenderReceiver interfaces (if parameter DcmDspDataUsePort is set to USE_DATA_SENDER_RECEIVER or to USE_DATA_SENDER_RECEIVER_AS_SERVICE) 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 (DcmDspDataType is different than UINT8_DYN) or from the diagnostic request length if the data has dynamic length (DcmDspDataType is set to UINT8_DYN)). ⌋*()*

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

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

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

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

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

4b) Otherwise (on any NvM failure) the Dcm module shall trigger a negative response with NRC 0x72 (GeneralProgrammingFailure). ⌋*()*

[SWS_Dcm_CONSTR_6039] Signals with variable datalength ⌈Only the last signal (DcmDspDidSignal) of a DID can have variable datalength (DcmDspDataType is set to UINT8_DYN). ⌋*()*

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

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

[SWS_Dcm_CONSTR_6018] ⌈DcmDspData elements used in service 0x2E shall not have DcmDspDataUsePorts set to USE_ECU_SIGNAL. ⌋*()*

[SWS_Dcm_CONSTR_6073] Dependency for DcmDspDataWriteFnc ⌈ DcmDspDataWriteFnc shall be only present if:

- DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC_ERROR

⌋*()*

7.5.2.13 Service 0x2F - InputOutputControlByIdentifier

[SWS_Dcm_00256] ⌈The Dcm module shall implement the UDS Service InputOutputControlByIdentifier (0x2F). ⌋*()*

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

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

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

[SWS_Dcm_00579] ⌈ The Dcm shall support InputOutputControlParameter definitions according to Table 7.29.⌋ *()*

Hex	Description
00	returnControlToECU
01	resetToDefault
02	freezeCurrentState
03	shortTermAdjustment

Table 7.29: InputOutputControlParameter definitions

[SWS_Dcm_00563] ⌈ On reception of the UDS Service InputOutputControlByIdentifier (0x2F), the Dcm module shall check if the DID is supported (see configuration parameter DcmDspDid) If not, the Dcm module shall send NRC 0x31 (Request out of range). ⌋ *()*

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

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

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

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

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

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

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

[SWS_Dcm_01272] ⌈ If the configuration parameter DcmDspDidControlMask is set to DCM_CONTROLMASK_EXTERNAL, the control enable mask record shall be forwarded within each interface. ⌋*()*

[SWS_Dcm_01273] ⌈ If the configuration parameter DcmDspDidControlMask is set to DCM_CONTROLMASK_EXTERNAL or DCM_CONTROLMASK_INTERNAL, or the DcmDspData element used in service 0x2F has DcmDspDataUsePorts set to USE_DATA_SENDER_RECEIVER or to USE_DATA_SENDER_RECEIVER_AS_SERVICE, the Dcm shall reject request without included control enable mask record with the NRC 0x13 (incorrectMessageLengthOrInvalidFormat). ⌋*()*

[SWS_Dcm_01274] ⌈ If the configuration parameter DcmDspDidControlMask is set to DCM_CONTROLMASK_NO, the Dcm shall reject request with included control enable mask record with the NRC 0x13 (incorrectMessageLengthOrInvalidFormat). ⌋*()*

[SWS_Dcm_CONSTR_6049] Limitation to one data element ⌈ In case DcmDspDidControlMask is set to DCM_CONTROLMASK_EXTERNAL, or the DcmDspData element used in service 0x2F has DcmDspDataUsePorts set to

USE_DATA_SENDER_RECEIVER ||USE_DATA_SENDER_RECEIVER_AS_SERVICE, the upper multiplicity DcmDspDidSignal is limited to 1. ⌋*()*

[SWS_Dcm_CONSTR_6050] ⌈ In case DcmDspDidControlMask is set to DCM_CONTROLMASK_EXTERNAL, or the DcmDspData element used in service 0x2F has DcmDspDataUsePorts set to USE_DATA_SENDER_RECEIVER ||USE_DATA_SENDER_RECEIVER_AS_SERVICE, the parameter DcmDspDidControlMaskSize shall be present with a value greater than zero. ⌋*()*

[SWS_Dcm_00680] Mapping of internal ControlEnableMaskRecord to DID data elements ⌈ If DcmDspDidControlMask is set to DCM_CONTROLMASK_INTERNAL,the ControlEnableMaskRecord shall be mapped to the DID data elements by applying the following mapping :

- The most significant bit of the first byte of the ControlEnableMask shall correspond to the first DID data element
- The second most significant bit of the first byte of the ControlEnableMask shall correspond to the second DID data element and continuing on in this fashion utilizing as many ControlEnableMask bytes as necessary to map all DID data elements.

⌋*(SRS_Diag_04218)*

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

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

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

USE_DATA_ASYNCH_CLIENT_SERVER_ERROR) for every data of the DID received in the request. ⌋*()*

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

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

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

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

abled: call the configured function xxx_ReturnControlToECU (see parameter DcmDspDataReturnControlToEcuFnc)

- For every DID signals with DcmDspDataUsePort set to USE_DATA_SENDER_RECEIVER or to USE_DATA_SENDER_RECEIVER_AS_SERVICE and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: update the data element IOOperationRequest with inputOutputControlParameter = 0x00, the controlEnableMask = 0xFFFFFFFF[1] and data element underControl = 0x00. There is no need to analyse the data element IOOperationResponse wherefore the result shall be ignored.

⌋()

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

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

- For every DID signals with DcmDspDataUsePort set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR 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 or USE_DATA_ASYNCH_FNC_ERROR and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: call the configured function xxx_ReturnControlToECU (see parameter DcmDspDataReturnControlToEcuFnc)

- For every DID signals with DcmDspDataUsePort set to USE_DATA_SENDER_RECEIVER or to USE_DATA_SENDER_RECEIVER_AS_SERVICE and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: update the data element IOOperationRequest with inputOutputControlParameter = 0x00, the controlEnableMask = 0xFFFFFFFF[2] and data element underControl = 0x00. There is no need to analyse the data element IOOperationResponse wherefore the result shall be ignored.

⌋()

[1] The size of the mask depends on the parameter DcmDspDidControlMaskSize

[2] The size of the mask depends on the parameter DcmDspDidControlMaskSize

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

- For every DID signals with DcmDspDataUsePort set to USE_ECU_SIGNAL and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: call to IoHwAb_Dcm_<symbolic name of ECU signal (parameter DcmDspDataEcuSignal)>() with 'action' parameter set to returnControlToEcu.
- For every DID signals with DcmDspDataUsePort set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR 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 or USE_DATA_ASYNCH_FNC_ERROR and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: call the configured function xxx_ReturnControlToECU (see parameter DcmDspDataReturnControlToEcuFnc
- For every DID signals with DcmDspDataUsePort set to USE_DATA_SENDER_RECEIVER or to USE_DATA_SENDER_RECEIVER_AS_SERVICE and either DcmDspDidFreezeCurrentState or DcmDspDidResetToDefault or DcmDspDidShortTermAdjustment enabled: update the data element IOOperationRequest with inputOutputControlParameter = 0x00, the controlEnableMask = 0xFFFFFFFF[3] and data element underControl = 0x00. There is no need to analyse the data element IOOperationResponse wherefore the result shall be ignored.

⌋*()*

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

[SWS_Dcm_00682] ⌈ The controlState in the ControlStatusRecord for positive response message of IoControl service shall be retrieved using the associated ReadData operation/function/SenderReceiver after application processing on the IO control request is positively finalized. ⌋*()*

Beside the Client/Server interface, the Dcm provides a SenderReceiver interface IOControlRequest_{Data} for I/O control service. Within this interface the Dcm will main-

[3] The size of the mask depends on the parameter DcmDspDidControlMaskSize

tain a bit-mask underControl with a state for each data element identical to the controlEnableMask. With this state the application could directly derives the control enable status without the need to maintain internal states.

The bit-mask underControl contains the status if any data elements of this particular I/O is currently under diagnostic control. The normal operation state could be derived if the value of underControl is set to 0x00 (which is the initial value). Each bit which is set is under diagnostic control via 'freezeCurrentState', 'resetToDefault' or 'shortTermAdjustment',

With each I/O Control request a command IOOperationRequest is provided to the application to update the input or the respective output. IOOperationRequest contains the inputOutputControlParameter, the controlEnableMask and in case of 'shortTermAdjustment' the controlState. The controlState could be processed by a data transmformer to provide a composite type to the application.

The value 0xFF of the inputOutputControlParameter of the command IOOperationRequest is the 'idle' state. The values 0x00 (returnControlToECU), 0x01 (resetToDefault), 0x02 (freezeCurrentState) or 0x03 (shortTermAdjustment) start the request processing and include the control option:

inputOutputControlParameter |controlEnableMask |controlState (optional)

The application needs to update their output values and finalizes the request with the response message IOOperationResponse to the Dcm. The possible values are:

- 0x00 positive response (similar to E_OK)
- 0x10 generalReject
- 0x21 busyRepeatRequest
- 0x22 conditionsNotCorrect
- 0x26 FailurePreventsExecutionOfRequestedAction
- 0x31 requestOutOfRange
- 0x78 ResponsePending (similar to E_PENDING)
- 0xFF Idle - no request present

Based on this respoinse message the Dcm will:

1. wait for final processing (0x78)
2. send a positive response message (0x00)
3. send a negative response message (all other values, except 0xFF)

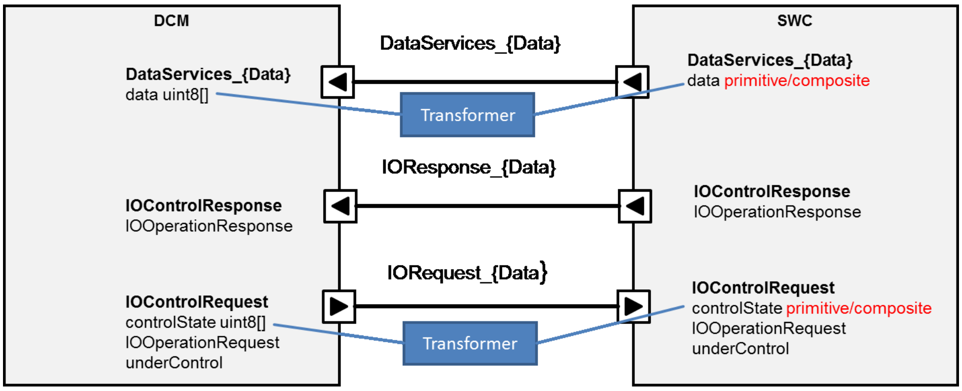


Figure 7.9: IO-Control with Sender/Receiver interfaces

0x2F D001 02 0x80 (freezeCurrentState) //Diagnostic request message
-> IOControlResponse_{Data} //R-Port of interface IOControlResponse
IOOperationResponse = 0xFF
-> IOControlRequest_{Data} //P-Port of interface IOControlRequest_{Data}
IOOperationRequest.inputOutputControlParameter = 0x02
IOOperationRequest.controlEnableMask = 0x80
-> ...
-> IOControlResponse_{Data} // R-Port of interface IOControlResponse_{Data}
IOOperationResponse == 0x00
-> IOControlRequest_{Data} //P-Port of interface IOControlRequest_{Data}
IOOperationRequest.inputOutputControlParameter = 0xFF
underControl = 0x80
-> controlState = DataServices_{Data}
.data //Read current data
0x6F D001 02 <controlState> // Diagnostic response message

0x2F D001 03 <data> 0x40 (shortTermAdjustment) Diagnostic request message
-> IOControlResponse_{Data} //R-Port of interface IOControlResponse
IOOperationResponse = 0xFF
-> IOControlRequest_{Data} //P-Port of interface IOControlRequest_{Data}
IOControlState = <data> //could be transformed to a complex type
-> IOControlRequest_{Data} //P-Port of interface IOControlRequest_{Data}
IOOperationRequest.inputOutputControlParameter = 0x03
IOOperationRequest.controlEnableMask = 0x40
-> ...
-> IOControlResponse_{Data} // R-Port of interface IOControlResponse_{Data}
IOOperationResponse == 0x78
-> ...
-> IOControlResponse_{Data} // R-Port of interface IOControlResponse_{Data}
IOOperationResponse == 0x00
-> IOControlRequest_{Data} //P-Port of interface IOControlRequest_{Data}
IOOperationRequest.inputOutputControlParameter = 0xFF

underControl = 0xC0
-> controlState = DataServices_{Data}
.data //Read current data
0x6F D001 03 <controlState> // Diagnostic response message

Application pseudo-code

```
IOcheck
{
  If (underControl & 0x80)
  {
    signal_1 = signal_1.overruleValue;
  } else {
    signal_1 = signal_1.currentValue;
  }
  If (underControl & 0x40)
  {
    signal_2 = signal_2.overruleValue;
  } else {
    signal_2 = signal_2.currentValue;
  }

  switch(IOOperationRequest.inputOutputControlParameter)
  {
    case 0x00: // returnControlToECU
      IOOperationResponse = 0x00;
    break;
    case 0x01: // resetToDefault
      if (IOOperationRequest.controlEnableMask & 0x80)
        signal_1.overruleValue = defaultValue_signal_1;
      if (IOOperationRequest.controlEnableMask & 0x40)
        signal_2.overruleValue = defaultValue_signal_2;
      IOOperationResponse = 0x00;
    break;
    case 0x02: // freezeCurrentState
      if (IOOperationRequest.controlEnableMask & 0x80)
        signal_1.overruleValue = currentValue;
      if (IOOperationRequest.controlEnableMask & 0x40)
        signal_2.overruleValue = currentValue;
      IOOperationResponse = 0x00;
    break;
    case 0x03: // shortTermAdjustment
      if (IOOperationRequest.controlEnableMask & 0x80)
        signal_1.overruleValue = controlState.signal_1;
      if (IOOperationRequest.controlEnableMask & 0x40)
        signal_2.overruleValue = controlState.signal_2;
      IOOperationResponse = 0x00;
    break;
    default: // Idle
    break;
   }
}
```

[SWS_Dcm_01275] ⌈ On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to returnControlToEcu, if all verifications have been successfully done (see [SWS_Dcm_00563], [SWS_Dcm_00565], [SWS_Dcm_00566], [SWS_Dcm_00567]), if parameter DcmDspDataUsePort set to USE_DATA_SENDER_RECEIVER or to USE_DATA_SENDER_RECEIVER_AS_SERVICE, the Dcm module shall update the IOOperationRequest command with inputOutputControlParameter = 0x00, the controlEnableMask = controlEnableMaskRecord of the request message and the IOOperationResponse with 0xFF (Idle). ⌋*()*

[SWS_Dcm_01276] ⌈ On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to resetToDefault or freezeCurrentState, if all verifications have been successfully done (see [SWS_Dcm_00563], [SWS_Dcm_00565], [SWS_Dcm_00566], [SWS_Dcm_00567]), if parameter DcmDspDataUsePort set to USE_DATA_SENDER_RECEIVER or to USE_DATA_SENDER_RECEIVER_AS_SERVICE, the Dcm module shall update the IOOperationRequest command with inputOutputControlParameter = InputOutputControlParameter of the request message, the controlEnableMask = controlEnableMaskRecord of the request message and the IOOperationResponse with 0xFF (Idle). ⌋*()*

[SWS_Dcm_01277] ⌈ On reception of a request for UDS Service InputOutputControlByIdentifier (0x2F) with InputOutputControlParameter equal to shortTermAdjustment, if all verifications have been successfully done (see [SWS_Dcm_00563], [SWS_Dcm_00565], [SWS_Dcm_00566], [SWS_Dcm_00567]), if parameter DcmDspDataUsePort set to USE_DATA_SENDER_RECEIVER or to USE_DATA_SENDER_RECEIVER_AS_SERVICE, the Dcm module shall

1. Update the data element controlState in the corresponding port IOControlRequest_{Data} with controlState of the request message.
2. Update the data element IOOperationRequest in the corresponding port IOControlRequest_{Data} with inputOutputControlParameter = InputOutputControlParameter of the request message, controlEnableMask = controlEnableMaskRecord of the request message and the IOOperationResponse with 0xFF (Idle).

⌋*()*

Note: The controlState is a separate data elment that it can be optionally processed by a data transformer to transform the byte stream into a composite type (see Figure 7.9: IO-Control with Sender/Receiver interfaces).

[SWS_Dcm_01278] ⌈ While a command is processed in the application (inputOutputControlParameter different from 0xFF), the Dcm shall poll the data element IOOperationResponse in the corresponding port IOControlResponse_{Data} until a final state is reached (IOOperationResponse having other value than 0x78 and 0xFF). ⌋*()*

For more information on possible values of IOOperationResponse, see the note under [SWS_Dcm_00682].

[SWS_Dcm_01280] ⌈ For FreezeCurrentState, ShortTermAdjustment and ResetToDefault, after the final response message (of [SWS_Dcm_01278]) is triggered, the data element IOOperationRequest in the corresponding port IOControlRequest_{Data} shall be updated with inputOutputControlParameter = 0xFF (Idle) to reset the command interface and the positive bits of the current controlEnableMask shall be added to underControl. ⌋*(SRS_Diag_04218)*

[SWS_Dcm_01337] ⌈ For ReturnControlToECU, after the final response message (of [SWS_Dcm_01278]) is triggered, the data element IOOperationRequest in the corresponding port IOControlRequest_{Data} shall be updated with inputOutputControlParameter = 0xFF (Idle) to reset the command interface and the positive bits of the current controlEnableMask shall be removed from underControl. ⌋*(SRS_Diag_04218)*

[SWS_Dcm_CONSTR_6030] ⌈ The ReturnControlToEcu functionnality is existing if at least one of the following parameters are activated : DcmDspDidFreezeCurrentState in ECUC_Dcm_00624 : or DcmDspDidResetToDefault in ECUC_Dcm_00623 : or DcmDspDidShortTermAdjustment in ECUC_Dcm_00625 : . ⌋*()*

[SWS_Dcm_CONSTR_6059] Dependency for DcmDspDataFreezeCurrentStateFnc ⌈ DcmDspDataFreezeCurrentStateFnc shall be only present if:

- DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC_ERROR

⌋*()*

[SWS_Dcm_CONSTR_6063] Dependency for DcmDspDataResetToDefaultFnc ⌈ DcmDspDataResetToDefaultFnc shall be only present if:

- DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC_ERROR

⌋*()*

[SWS_Dcm_CONSTR_6064] Dependency for DcmDspDidControlMaskSize ⌈ DcmDspDidControlMaskSize shall be only present if DcmDspDidControlMask is equal to DCM_CONTROLMASK_EXTERNAL or DCM_CONTROLMASK_INTERNAL. ⌋*()*

[SWS_Dcm_CONSTR_6081] Dependency for DcmDspDidControlMaskBitPosition ⌈ The value configured for DcmDspDidControlMaskBitPosition shall be lower than DcmDspDidControlMaskSize * 8. ⌋*()*

[SWS_Dcm_CONSTR_6065] Dependency for DcmDspDataReturnControlToEcuFnc ⌈ DcmDspDataReturnControlToEcuFnc shall be only present if:

- DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC_ERROR

⌋*()*

[SWS_Dcm_CONSTR_6066] Dependency for DcmDspDataShortTermAdjustmentFnc ⌈ DcmDspDataShortTermAdjustmentFnc shall be only present if:

- DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC or
- DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC_ERROR

⌋*()*

7.5.2.14 Service 0x31 - RoutineControl

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

A tester can use UDS Service 0x31 to start, stop or obtain the results of a routine identified by a 2-byte routineIdentifier. The Dcm module configuration contains a list of the routineIdentifiers (see configuration parameter DcmDspRoutineIdentifier) supported by the DCM. For each routineIdentifier, the Dcm configuration specifies:

- The function Xxx_Start() associated with this routineIdentifier (see configuration parameters DcmDspStartRoutineFnc and DcmDspRoutineUsePort)
- List of signal available in the request and in the response (see configuration parameters DcmDspStartRoutineIn and DcmDspStartRoutineOut)
- The function Xxx_Stop() associated with this routineIdentifier (see configuration parameters DcmDspStopRoutineFnc and DcmDspRoutineUsePort)
- List of signal available in the request and in the response (see configuration parameters DcmDspStopRoutineIn and DcmDspStopRoutineOut)
- The function Xxx_RequestResults() associated with this routineIdentifier (see configuration parameters DcmDspRequestRoutineResultsFnc and DcmDspRoutineUsePort)
- List of signal available in the response (see configuration parameter DcmDspRequestRoutineResultsOut)

[SWS_Dcm_00568] ⌈ On reception of the UDS Service RoutineControl (0x31), the Dcm module shall check if the Routine is supported (see configuration parameter DcmDspRoutine) If not, the Dcm module shall send NRC 0x31 (Request out of range). ⌋ *()*

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

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

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

[SWS_Dcm_00869] ⌈ On reception of the UDS Service RoutineControl (0x31), the Dcm module shall check if the SubFunction to the corresponding Routine is supported (see existence of configuration container DcmDspStopRoutine for SubFunction 0x02; DcmDspRequestRoutineResults for SubFunction 0x03). If not, the Dcm module shall send NRC 0x12 (SubFunction not supported). ⌋ *()*

[SWS_Dcm_01169] ⌈ On reception of the UDS Service RoutineControl (0x31) with SubFunction startRoutine, the Dcm module shall check if the Routine can be executed in the current mode condition (see configuration parameter DcmDspStartRoutineCommonAuthorizationRef). If not, the Dcm module shall send the calculated negative response code of the referenced DcmModeRule. ⌋ *()*

[SWS_Dcm_01170] ⌈ On reception of the UDS Service RoutineControl (0x31) with SubFunction stopRoutine, the Dcm module shall check if the Routine can be executed in the current mode condition (see configuration parameter DcmDspStopRoutineCommonAuthorizationRef). If not, the Dcm module shall send the calculated negative response code of the referenced DcmModeRule. ⌋ *()*

[SWS_Dcm_01171] ⌈ On reception of the UDS Service RoutineControl (0x31) with SubFunction requestRoutineResults, the Dcm module shall check if the Routine can be executed in the current mode condition (see configuration parameter DcmDspRequestRoutineResultsCommonAuthorizationRef). If not, the Dcm module shall send the calculated negative response code of the referenced DcmModeRule. ⌋ *()*

Routines have different input and output parameters depending on the routine configuration (e.g. DcmDspStartRoutineIn for input parameter for the routine start service). The signature of the called routine operations Xxx_Start, Xxx_Stop and Xxx_RequestResults is depending on this configuration. The defined parameters for input and output routine data are optional, and marked in brackets '[]' in the definition in [SWS_Dcm_01203], [SWS_Dcm_01204] and [SWS_Dcm_91013].

[SWS_Dcm_01360] ⌈ For each configured routine input signal in DcmDspStartRoutineInSignal or DcmDspStopRoutineInSignal with a signal type unequal to VARIABLE_LENGTH, the optional parameter 'DcmDspRoutineSignalType dataIn_n' shall be provided in the corresponding operations in [SWS_Dcm_01203] or [SWS_Dcm_01204]. ⌋*()*

[SWS_Dcm_01361] ⌈ For a configured routine input signal in DcmDspStartRoutineInSignal or DcmDspStopRoutineInSignal with a signal type equal to VARIABLE_LENGTH the optional parameter const 'uint8 * dataInVar' shall be provided in the corresponding operations in [SWS_Dcm_01203] or [SWS_Dcm_01204]. ⌋*()*

[SWS_Dcm_01362] ⌈ For each configured routine output signal in DcmDspStartRoutineOutSignal, DcmDspStopRoutineOutSignal or DcmDspRequestRoutineResultsOutSignal with a signal type unequal to VARIABLE_LENGTH the optional parameter 'DcmDspRoutineSignalType dataOut_n' shall be provided in the corresponding operations in [SWS_Dcm_01203], [SWS_Dcm_01204] or [SWS_Dcm_91013]. ⌋*()*

[SWS_Dcm_01363] ⌈ For a configured routine output signal in DcmDspStartRoutineOutSignal, DcmDspStopRoutineOutSignal or DcmDspRequestRoutineResultsOutSignal with a signal type equal to VARIABLE_LENGTH the optional parameter const 'uint8 * dataOutVar' shall be provided in the corresponding operations in [SWS_Dcm_01203], [SWS_Dcm_01204] or [SWS_Dcm_91013]. ⌋*()*

[SWS_Dcm_01364] ⌈ The optional in/out parameter variableDataLength in [SWS_Dcm_01203], [SWS_Dcm_01204] or [SWS_Dcm_91013] is always present if at least one of the routine input signal data or routine output signal data have a signal with routine type 'VARIABLE_LENGTH'. ⌋*()*

Note: The 'variableDataLength' parameter as in/out parameter contains the data length in bytes of the 'dataInVar' while calling the operation and it returns the length in bytes of the 'dataOutVar'. As 'dataInVar' and 'dataOutVar' are optional, 'variableDataLength' is only present if at least one of this optional parameter is used.

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

[SWS_Dcm_00400] ⌈ When receiving a request for UDS Service RoutineControl (0x31) with subfunction startRoutine, if all verifications have been successfully done (see [SWS_Dcm_00568], [SWS_Dcm_00570], [SWS_Dcm_00571]), the Dcm module shall call the configured Xxx_Start() function passing the dataIn, calculated from routineControlOptionRecord (see [SWS_Dcm_00590]), and the dataOut reference according of the list of output signal configured for this routine (see configuration parameter DcmDspStartRoutineOut). ⌋()

[SWS_Dcm_00401] ⌈ Upon completing [SWS_Dcm_00400], when Xxx_Start() returns E_OK, 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)). ⌋()

[SWS_Dcm_00402] ⌈ When receiving a request for UDS Service RoutineControl (0x31) with subfunction stopRoutine, if all verifications have been successfully done (see [SWS_Dcm_00568], [SWS_Dcm_00570], [SWS_Dcm_00571]), the Dcm module shall call the configured Xxx_Stop() function passing the dataIn, calculated from routineControlOptionRecord (see [SWS_Dcm_00590]), and the dataOut reference according of the list of output signal configured for this routine (see configuration parameter DcmDspStopRoutineOut). ⌋()

[SWS_Dcm_00403] ⌈ Upon completing [SWS_Dcm_00402], when Xxx_Stop() returns E_OK, 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)). ⌋()

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

[SWS_Dcm_00405] ⌈ Upon completing [SWS_Dcm_00404], when Xxx_RequestResults() returns E_OK, 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 DcmDspRequestRoutineResultsOut)). ⌋()

[SWS_Dcm_00641] ⌈ To serialize the required AUTOSAR data types (signed- and unsigned integer) from the request message / into the response message of UDS Service RoutineControl, the target endianness configured in DcmDspRoutineSignalEndianness shall be considered for DcmDspRoutine signals having set to fixed length (DcmDspRoutineSignalType set to other value than VARIABLE_LENGTH). ⌋()

[SWS_Dcm_CONSTR_6072] Dependency for DcmDspRoutineSignalEndianness ⌈ In case DcmDspRoutineSignalEndianness is not present, the DcmDspDataDefaultEndianness shall be used instead. ⌋()

[SWS_Dcm_01139] ⌈ The Dcm shall follow the NRC handling for RoutineControlService according to ISO 14229-1 [2]. ⌋()

[SWS_Dcm_01140] ⌈ On reception of the UDS Service RoutineControl (0x31), the Dcm module shall check the overall length of the request. If length of the request is wrong, the Dcm module shall send NRC 0x13 (Incorrect message length or invalid format) to the tester. ⌋*()*

[SWS_Dcm_01141] ⌈ The Dcm shall call the appropriate routine functions of the SWC after having performed the total length check and the Mode rules, security level and session checks (DcmDspStartRoutineCommonAuthorizationRef, DcmDspStopRoutineCommonAuthorizationRef and DcmDspRequestRoutineResultsCommonAuthorizationRef). ⌋*()*

Note: Subsequent checks have to be performed by the SWC.

[SWS_Dcm_01194] ⌈ On reception of the UDS Service RoutineControl (0x31), for every requested RID inside the OBD range (E000-E0FF), the Dcm shall implicitly allow sub-function StartRoutine. ⌋*()*

[SWS_Dcm_00701] ⌈ On reception of the UDS Service RoutineControl (0x31), for every requested RID inside the OBD range (E000-E0FF) and usage of UDS interface, the Dcm module shall use the routineInfo byte value from the configuration (see ECUC_Dcm_01063) in the response to the tester. ⌋*()*

[SWS_Dcm_01330] ⌈ If DcmDspEnableObdMirror is set to true, an explicitly configured RID inside the OBD range (E000-E0FF) shall use the UDS interface. ⌋*()*

[SWS_Dcm_01331] ⌈ If DcmDspEnableObdMirror is set to false, all requests within the OBD RID range shall use the UDS interface. ⌋*()*

[SWS_Dcm_01332] ⌈ On reception of the UDS Service RoutineControl (0x31), for every requested RID inside the OBD range (E000-E0FF), the Dcm module shall handle the RID as defined for OBD Service $08 (see [SWS_Dcm_00418], [SWS_Dcm_00947], [SWS_Dcm_00419], [SWS_Dcm_00420], [SWS_Dcm_00948], [SWS_Dcm_01192]) if DcmDspEnableObdMirror is set to true and RID not explicitly configured. ⌋*()*

[SWS_Dcm_01333] ⌈ On reception of the UDS Service RoutineControl (0x31), for every requested RID inside the OBD range (E000-E0FF) and usage of OBD interface, the Dcm shall use the routineInfo byte value from the configuration (see ECUC_Dcm_01078) in the response to the tester. ⌋*()*

If DcmDspEnableObdMirror is set to FALSE or the RID is explicitly configured inside the OBD TestId range (E000-E0FF), the access to the OBD data shall be given in the following way:

[SWS_Dcm_01390] ⌈ On reception of an UDS Service RoutineControl (0x31) request with one or more "availability OBDTestIds" as parameter, the Dcm module shall respond with the corresponding supported (=configured) RIDs. ⌋*()*

[SWS_Dcm_01391] ⌈ On reception of an UDS Service RoutineControl (0x31) request "availability OBDTestIds" together with other OBDTestIds as parameter, the Dcm module shall ignore the request. ⌋*()*

[SWS_Dcm_01392] ⌈ On reception of an UDS Service RoutineControl (0x31) request with a OBDTestIds that is not an "availability OBDTestIds", the Dcm module shall invoke the configured Xxx_Start() function. ⌋*()*

[SWS_Dcm_01393] ⌈ As specified in [3, SAE J1979], unused data bytes shall be filled with $00. ⌋*()*

[SWS_Dcm_01394] ⌈ If Xxx_Start() doesn't return E_OK, the Dcm shall return NRC 0x22. ⌋*()*

[SWS_Dcm_00668] ⌈ If the operation Start() returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to ErrorCode parameter value. ⌋*()*

[SWS_Dcm_00669] ⌈ If the operation Start() returns value DCM_E_FORCE_RCRRP, the Dcm module shall start the transmission of NRC 0x78. ⌋*()*

[SWS_Dcm_00670] ⌈ If the operation Stop() returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to ErrorCode parameter value. ⌋*()*

[SWS_Dcm_00671] ⌈ If the operation Stop() returns value DCM_E_FORCE_RCRRP, the Dcm module shall start the transmission of NRC 0x78. ⌋*()*

[SWS_Dcm_00672] ⌈ If the operation RequestResults() returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to ErrorCode parameter value. ⌋*()*

[SWS_Dcm_00673] ⌈ If the operation RequestResults () returns value DCM_E_FORCE_RCRRP, the Dcm module shall start the transmission of NRC 0x78. ⌋*()*

[SWS_Dcm_CONSTR_6071] Dependency for DcmDspStartRoutineFnc, DcmDspStopRoutineFnc, DcmDspRequestRoutineResultsFnc, DcmDspStartRoutineConfirmationFnc, DcmDspStopRoutineConfirmationFnc ⌈ The following configuration parameters shall only be present if DcmDspRoutineUsePort is set to FALSE.

- DcmDspStartRoutineFnc
- DcmDspStopRoutineFnc
- DcmDspRequestRoutineResultsFnc
- DcmDspStartRoutineConfirmationFnc
- DcmDspStopRoutineConfirmationFnc

⌋*()*

7.5.2.15 Service 0x3E - Tester Present

[SWS_Dcm_00251] ⌈ The Dcm module shall implement the Tester Present (service 0x3E, diagnostic communication and security) of the Unified Diagnostic Services for the subfunction values 0x00 and 0x80. ⌋*()*

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

7.5.2.16 Service 0x3D - WriteMemoryByAddress

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

This service is used to write data using a physical memory address. **[SWS_Dcm_00855]** ⌈ On reception of the UDS Service WriteMemoryByAddress (0x3D), the Dcm shall check if the requested AddressAndLengthFormatIdentifier is supported (refer to configuration parameter DcmDspSupportedAddressAndLengthFormatIdentifier), Otherwise the NRC 0x31 (requestOutOfRange) shall be responded. In case the container AddressAndLengthFormatIdentifier is not present, the Dcm shall accept all possible AddressAndLengthFormatIdentifiers. ⌋*()*

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

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

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

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

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

[SWS_Dcm_01056] ⌈ The configured ranges of memory address (DcmDspReadMemoryRangeHigh and DcmDspReadMemoryRangeLow or DcmDspReadMemoryRangeByLabelHigh and DcmDspReadMemoryRangeByLabelLow) shall not overlap each other. ⌋*()*

[SWS_Dcm_01358] ⌈ 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 session (see DcmDspWriteMemoryRangeSessionLevelRef). If the session is not correct, the Dcm module shall send NRC 0x31 (RequestOutOfRange). ⌋*()*

7.5.2.17 Service 0x23 - ReadMemoryByAddress

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

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

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

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

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

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

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

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

[SWS_Dcm_01158] ⌈ The configured ranges of memory address (DcmDspReadMemoryRangeHigh and DcmDspReadMemoryRangeLow or DcmDspReadMemoryRangeByLabelHigh and DcmDspReadMemoryRangeByLabelLow) shall not overlap each other. ⌋()

[SWS_Dcm_01359] ⌈ 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 read in the current session (see DcmDspReadMemoryRangeSessionLevelRef). If the session is not correct, the Dcm module shall send NRC 0x31 (RequestOutOfRange). ⌋()

[SWS_Dcm_00644] ⌈ If the operation Dcm_ReadMemory returns DCM_READ_FAILED, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value. ⌋()

[SWS_Dcm_00839] ⌈ If the call to Dcm_ReadMemory returns DCM_READ_FORCE_RCRRP, the Dcm shall invoke the transmit request for RCR-RP (NRC 0x78 transmission) and the Dcm shall not realize further invocation of the operation till RCR-RP is transmitted. ⌋()

[SWS_Dcm_00840] ⌈ After transmit confirmation of a RCR-RP transmitted on the context of [SWS_Dcm_00839], the Dcm calls, from Dcm_MainFunction (due to call context), Dcm_ReadMemory again with OpStatus = DCM_FORCE_RCRRP_OK. ⌋()

7.5.2.18 Service 0x34 - RequestDownload

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

[SWS_Dcm_00496] ⌈ The Dcm module shall implement the RequestDownload (service 0x34) of the Unified Diagnostic Services. ⌋*(SRS_Diag_04033)*

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

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

[SWS_Dcm_01132] ⌈ NRC described in Table 7.30 shall be the responsibility of the callout function.⌋*()*

NRC	Use Case
0x31 requestOutOfRange	The specified dataFormatIdentifier is not valid.
0x70 uploadDownloadNotAccepted	An attempt to download to a server's memory cannot be accomplished due to some fault conditions. Note: this NRC will be handled by the callout only if mode rule is not used for this case

Table 7.30: NRC managed by callout function for service 0x34

Note: the callout function can, if needed, return also other NRC but the ones above won't be treated by the Dcm module.

[SWS_Dcm_00757] ⌈ If the operation Dcm_ProcessRequestDownload returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value. ⌋*()*

[SWS_Dcm_01417] ⌈ Upon calling Dcm_ProcessRequestDownload, the Dcm shall write the maximum possible buffer size into the BlockLength parameter. ⌋*(SRS_Diag_04033)*

[SWS_Dcm_01418] ⌈ If the function call Dcm_ProcessRequestDownload returns a requested buffer length larger than the supported buffer length of the current protocol connection, the Dcm shall report the Det error DCM_E_INTERFACE_BUFFER_OVERFLOW. ⌋*(SRS_Diag_04033)*

[SWS_Dcm_01419] ⌈ If the function call Dcm_ProcessRequestDownload returns a requested buffer length smaller or equal than the supported buffer length of the current protocol connection, the Dcm shall return the BlockLength value within the maxNumberOfBlockLength parameter of the positive response. ⌋*(SRS_Diag_04033)*

7.5.2.19 Service 0x35 - RequestUpload

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

[SWS_Dcm_00499] ⌈ The Dcm module shall implement the RequestUpload (service 0x35) of the Unified Diagnostic Services. ⌋*(SRS_Diag_04033)*

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

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

[SWS_Dcm_01133] ⌈ NRC described in Table 7.31 shall be the responsibility of the callout function.⌋()

NRC	Use Case
0x31 - requestOutOfRange	The specified dataFormatIdentifier is not valid.
0x70 - uploadDownloadNotAccepted	An attempt to download to a server's memory cannot be accomplished due to some fault conditions. Note: this NRC will be handled by the callout only if mode rule is not used for this case

Table 7.31: NRC managed by callout function for service 0x35

Note: the callout function can, if needed, return also other NRC but the ones above won't be treated by the Dcm module.

[SWS_Dcm_00758] ⌈ If the operation Dcm_ProcessRequestUpload returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value. ⌋()

[SWS_Dcm_01420] ⌈ Upon calling Dcm_ProcessRequestUpload, the Dcm shall write the maximum possible buffer size into the BlockLength parameter. ⌋ *(SRS_Diag_04033)*

[SWS_Dcm_01421] ⌈ If the function call Dcm_ProcessRequestUpload returns a requested buffer length larger than the supported buffer length of the current protocol connection, the Dcm shall report the Det error DCM_E_INTERFACE_BUFFER_OVERFLOW. ⌋*(SRS_Diag_04033)*

[SWS_Dcm_01422] ⌈ If the function call Dcm_ProcessRequestUpload returns a requested buffer length smaller or equal than the supported buffer length of the current protocol connection, the Dcm shall return the BlockLength value within the maxNumberOfBlockLength parameter of the positive response. ⌋*(SRS_Diag_04033)*

7.5.2.20 Service 0x36 - TransferData

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

[SWS_Dcm_00502] ⌈ The Dcm module shall implement the TransferData (service 0x36) of the Unified Diagnostic Services. ⌋*(SRS_Diag_04033)*

[SWS_Dcm_00503] ⌈ On reception of the UDS Service TransferData (0x36), if a download process is running (RequestDownload service has been previously received) and the request format is correct, the Dcm module shall call the callout Dcm_WriteMemory. ⌋*(SRS_Diag_04033)*

[SWS_Dcm_00504] ⌈ On reception of the UDS Service TransferData (0x36), if an upload process is running (RequestUpload service has been previously received) and the request format is correct, the Dcm module shall call the callout Dcm_ReadMemory. ⌋ *(SRS_Diag_04033)*

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

[SWS_Dcm_01173] ⌈ NRCs described in Table 7.32 shall be the responsibility of the callout function.⌋*()*

NRC	Use Case
0x24 - requestSequenceError	only for the following conditions: If the RequestDownload or RequestUpload service is active, but the server has already received all data as determined by the memorySize parameter in the active RequestDownload or RequestUpload service
0x31 - requestOutOfRange	Only for the following conditions: The transferRequestParameterRecord contains additional control parameters (e.g. additional address information) and this control information is invalid. The transferRequestParameterRecord is not consistent with the server's memory alignment constraints
0x71 - transferDataSuspended	The data transfer operation was halted due to some fault.
0x72 - generalProgrammingFailure	If the server detects an error when finalizing the data transfer between the client and server (e.g., via an integrity check).
0x92 - voltageTooHigh	The voltage measured is higher than the maximum acceptable voltage for downloading data.
0x93 - voltageTooLow	The voltage measured is under the minimum acceptable voltage for downloading data.

Table 7.32: NRC managed by callout function for service 0x36

Note: the callout function can, if needed, return also other NRCs but the ones above won't be treated by the Dcm module.

7.5.2.21 Service 0x37 - RequestTransferExit

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

[SWS_Dcm_00505] ⌈ The Dcm module shall implement the RequestTransferExit (service 0x37) of the Unified Diagnostic Services. ⌋*(SRS_Diag_04033)*

[SWS_Dcm_01395] ⌈ If the transferRequestParameterRecord is not available in the 0x37 request, the Dcm shall call Dcm_ProcessRequestTransferExit with transferRequestParameterRecordSize=0. In this case, the parameter transferRequestParameterRecord is undefined. ⌋*(SRS_Diag_04033)*

[SWS_Dcm_01396] ⌈ If the OUT parameter transferResponseParameterRecordSize of the API Dcm_ProcessRequestTransferExit is equal to 0, the Dcm shall not evaluate the parameter transferResponseParameterRecord and the positive response transferResponseParameterRecord shall be empty. ⌋*(SRS_Diag_04033)*

[SWS_Dcm_01134] ⌈ NRC described in Table 7.33 shall be the responsibility of the callout function.⌋*()*

NRC	**Use Case**
0x13 - incorrectMessageLengthOrInvalidFormat	If the length of the message is wrong.
0x24 - requestSequenceError	The programming process is not completed when a request for this service is received.
0x31 - requestOutOfRange	If the transferRequestParameterRecord contains invalid data
0x72 - generalProgrammingFailure	If the server detects an error when finalizing the data transfer between the client and server (e.g., via an integrity check).

Table 7.33: NRC managed by callout function for service 0x37

Note: the callout function can, if needed, return also other NRC but the ones above won't be treated by the Dcm module.

[SWS_Dcm_00759] ⌈ If the operation Dcm_ProcessRequestTransferExit returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value. ⌋*()*

[SWS_Dcm_00643] ⌈ If the operation Dcm_WriteMemory returns DCM_WRITE_FAILED, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value. ⌋*()*

[SWS_Dcm_00837] ⌈ If the call to Dcm_WriteMemory returns DCM_WRITE_FORCE_RCRRP, the Dcm shall invoke the transmit request for RCR-RP (NRC 0x78 transmission) and the Dcm shall not realize further invocation of the operation till RCR-RP is transmitted. ⌋*()*

[SWS_Dcm_00838] ⌈ After transmit confirmation of a RCR-RP transmitted on the context of [SWS_Dcm_00837], the Dcm calls, from Dcm_MainFunction (due to call context), Dcm_WriteMemory again with OpStatus = DCM_FORCE_RCRRP_OK. ⌋*()*

7.5.2.22 Service 0x38 - RequestFileTransfer

[SWS_Dcm_01083] ⌈ The Dcm module shall implement the RequesFileTransfer (service 0x38) of the Unified Diagnostic Services by calling a callout. ⌋ *()*

This service is used to request the start of a file transfer process according to ISO-14229-1 [2].

[SWS_Dcm_01084] ⌈ The Dcm shall process RequestFileTransfer accoding to [14] and call XXX_ProcessRequestFileTransfer after the Full length check. Further checks shall be performed in the callout. ⌋ *()*

[SWS_Dcm_01085] ⌈ The Dcm shall pass the parameter of the RequestFileTransfer request to the parameter with the same name of the XXX_ProcessRequestFileTransfer operation. ⌋ *()*

[SWS_Dcm_01086] ⌈ If the parameters in the RequestFileTransfer request are smaller than the according paramters in the XXX_ProcessRequestFileTransfer operation, the Dcm shall adapt the size to the size expected by XXX_ProcessRequestFileTransfer. ⌋ *()*

[SWS_Dcm_01087] ⌈ If the parameters in the RequestFileTransfer request are greater than the according parameters in the XXX_ProcessRequestFileTransfer operation, the Dcm shall send a negative response with NRC 0x31 (RequestOutOfRange). ⌋ *()*

[SWS_Dcm_01088] ⌈ If the operation ProcessRequestFileTransfer returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value. ⌋ *()*

[SWS_Dcm_01089] ⌈ If the parameter is not part of the request, the Dcm shall set the according value to 0 in the XXX_ProcessRequestFileTransfer operation ⌋ *()*

[SWS_Dcm_01090] ⌈ The configuration parameter DcmRequestFileTransferFileSizeParameterLength shall define the size of the parameter filePathAndName of the XXX_ProcessRequestFileTransfer operation. ⌋ *()*

[SWS_Dcm_01091] ⌈ The configuration parameter DcmRequestFileTransferLengthFormatIdentifier shall define the size of the parameter filePathAndName of the XXX_ProcessRequestFileTransfer operation ⌋ *()*

7.5.2.23 Service 0x85 - ControlDTCSetting

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 sub-function 0x01 ("ON") or 0x02 ("OFF").

[SWS_Dcm_00249] ⌈ The Dcm module shall implement UDS Service ControlDTCSetting (0x85) to enable or disable the storage of DTCs in the ECUs error memory. ⌋ *(SRS_Diag_04159)*

[SWS_Dcm_01399] ⌈ If the Dcm receives a ControlDTCSetting (0x85) service with DTCSettingControlOptionRecord != 0xFFFFFF, the Dcm shall send a NRC 0x31 (RequestOutOfRange).⌋*(SRS_Diag_04159)*

[SWS_Dcm_01063] ⌈ On reception of UDS Service 0x85 with subfunction 0x01 (DTCSettingType "ON"), the Dcm shall call Dem_EnableDTCSetting() with ClientId = Client Id for this Dcm instance (see DcmDemClientRef). ⌋*(SRS_Diag_04115)*

[SWS_Dcm_00783] ⌈ In case of Dem_EnableDTCSetting returns E_OK (see [SWS_Dcm_01063]), the Dcm shall invoke a mode switch of the ModeDeclarationGroupPrototype DcmControlDTCSetting by calling SchM_Switch_<bsnp>_DcmControlDTCSetting (RTE_MODE_DcmControlDTCSetting_ENABLEDTCSETTING). ⌋*()*

[SWS_Dcm_00406] ⌈ On reception of UDS Service 0x85 with subfunction 0x02 (DTCSettingType "OFF"), the Dcm shall call Dem_DisableDTCSetting() with ClientId = Client Id for this Dcm instance (see DcmDemClientRef). ⌋*(SRS_Diag_04115)*

[SWS_Dcm_00784] ⌈ In case of Dem_DisableDTCSetting returns E_OK (see [SWS_Dcm_00406]), the Dcm shall invoke a mode switch of the ModeDeclarationGroupPrototype DcmControlDTCSetting by calling SchM_Switch_<bsnp>_DcmControlDTCSetting (RTE_MODE_DcmControlDTCSetting_DISABLEDTCSETTING). ⌋*()*

[SWS_Dcm_00751] ⌈ In case the DTCSetting is disabled and a transitions to default session or upon any diagnostic session change where the new session does not support UDS Service ControlDTCsetting anymore, the Dcm module shall call Dem_EnableDTCSetting() with the following parameters

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef)

and switch the mode DcmControlDTCSetting to DCM_ENABLEDTCSETTING. ⌋*()*

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

[SWS_Dcm_00752] ⌈ In case the DTCSetting is disabled and at least one referenced arbitrary ModeDeclarationGroupPrototypes (see configuration parameter DcmDspControlDTCSettingReEnableModeRuleRef) for service ControlDTCSetting (0x85) with DTCSettingType "OFF" (0x02) are not fulfilled anymore, the Dcm module shall call Dem_EnableDTCSetting() with the following parameters:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef)

and switch the mode DcmControlDTCSetting to DCM_ENABLEDTCSETTING⌋*()*

Note: If at least one ModeDeclarationGroupPrototypes is configured (see configuration parameter DcmDspControlDTCSettingReEnableModeRuleRef) for service ControlDTCSetting (0x85) with DTCSettingType "OFF" (0x02), it is recommended to activate the condition check for the according sub-function 0x02(see configuration parameter DcmDsdSubServiceModeRuleRef).

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

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

[SWS_Dcm_00852] ⌈ If the configuration parameter DcmSupportDTCSettingControlOptionRecord is set to false and the request contains any data after the sub-function, the Dcm shall return NRC 0x13 (Incorrect message length or invalid format). ⌋*()*

7.5.2.24 Service 0x87 - LinkControl

This service is used to gain bus bandwidth for diagnostic purposes.

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

Therefore the service LinkControl needs to be implemented project specific as external service (refer to Chapter 8.8 Dcm as Service-Component).

7.5.3 OBD Services

7.5.3.1 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 7.34: Support for OBD services in the DCM

7.5.3.2 General behavior

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

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

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

From the Basic Software point of view Dependent / Secondary ECUs doesn't need any specific OBD functionality. In Dependent / Secondary ECUs OBD-relevant information will not be stored in the Basic Software (e.g. no direct communication with the scan tool). The respective OBD functionality might be handled in Dependent / Secondary ECUs by a SWC.

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

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

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

Table 7.35: Overview about OBD functionality in different OBD ECUs

[SWS_Dcm_00077] ⌈ When calling the DEM module for OBD services, the Dcm module shall use the following values for the parameter DTCOrigin:
Service $0A uses DEM_DTC_ORIGIN_PERMANENT_MEMORY
All other services use DEM_DTC_ORIGIN_OBD_RELEVANT_MEMORY ⌋ *(SRS_Diag_04058)*

7.5.3.3 Service $01 - Request Current Powertrain Diagnostic Data

[SWS_Dcm_00243] ⌈ The Dcm module shall implement the OBD service $01 (Request Current Powertrain diagnostic Data) in compliance to all provisions of the OBD standard. ⌋ *()*

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 DcmDspPidDataByteSize)
  - The position of the data in the PID (see configuration parameter DcmDspPidByteOffset)
  - The reference to the supported information container (see configuration parameter DcmDspPidDataSupportInfo)
  - The Xxx_ReadData() function that the Dcm must call to obtain the current value of the data or the name of the port that the Dcm uses to obtain the current value through the RTE from a SW-C (see configuration parameters DcmDspPidDataReadFnc and DcmDspPidDataUsePort)

[SWS_Dcm_00407] ⌈ On reception of an OBD Service $01 request with only "availability PIDs" as parameter, the Dcm shall respond with the corresponding supported (=configured) PIDs encoded according to the OBD standard. ⌋*()*

To obtain the value for a PID, the Dcm uses the configured Xxx_ReadData() functions for every data of the PID. To provide OBD Service $01, the Dcm relies on external functions that allow it to obtain the value of the PIDs. There is one such function per data of every PID that is needed by the DCM.

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

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

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

The entity providing the actual implementation of the Xxx_ReadData() function for a specific signal of a PID might be a SW-C or another basic software module. The origin of the function is not known to the Dcm but is part of the Dcm configuration. Some PIDs are provided by the DEM. These PIDs are also explicitly configured in the Dcm configuration and it is the responsibility of a correct Dcm configuration to make the Xxx_ReadData() function point to the correct function provided by the DEM.

[SWS_Dcm_CONSTR_6069] Dependency for DcmDspPidDataReadFnc ⌈ DcmDspPidDataReadFnc shall be only present if DcmDspPidDataUsePort is set to USE_DATA_SYNCH_FNC. ⌋()

For certain PIDs, the Dem provides the function to obtain the PID value. Which PIDs come from the Dem are part of the Dcm configuration.

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

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

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

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

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

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

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

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

[SWS_Dcm_00718] ⌈ To serialize the required AUTOSAR data types (signed- and unsigned integer) into the response message of OBD Service $01 responses the target endianness configured in DcmDspPidDataEndianness shall be considered for DcmDspPidData elements having DcmDspPidDataUsePort set to USE_DATA_SENDER_RECEIVER or USE_DATA_SENDER_RECEIVER_AS_SERVICE. In case DcmDspPidDataEndianness is not present, the DcmDspDataDefaultEndianness shall be used instead. ⌋*()*

[SWS_Dcm_CONSTR_6068] Dependency for DcmDspPidDataEndianness ⌈ In case DcmDspPidDataEndianness is not present, the DcmDspDataDefaultEndianness shall be used instead. ⌋*()*

7.5.3.4 Service $02 - Request Power Train FreezeFrame Data

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

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

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

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

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

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

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

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

7.5.3.4.1 Service $02 - PID$02

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

[SWS_Dcm_00279] ⌈ On reception of a request for Service $02 with PID $02, the Dcm shall call Dem_DcmGetDTCOfOBDFreezeFrame() with FrameNumber set to 0x00 to get the DTC number. ⌋*(SRS_Diag_04058)*

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

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

7.5.3.4.2 Service $02 - availability PID

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

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

7.5.3.4.3 Service $02 - other PIDs

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

[SWS_Dcm_00286] ⌈ On reception of a service $02 request with a PID that is not an "availability PID" and is not $02, the Dcm shall call Dem_DcmReadDataOfOBDFreezeFrame() for every data of the PID with the following parameter values:

- PID = the PID received in the OBD request
- DestBuffer = a buffer in which the callee can write the value of the PID
- BufSize = the size of the DestBuffer, this must be at least equal to the size needed to store the value of the PID as configured in the DCM
- DataElementIndexOfPid = implicit index (from 0 to n) of the DataElement calculated by Dcm according to the order of the DataElement positions in the PID (see parameter DcmDspPidByteOffset)

⌋*()*

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

[SWS_Dcm_00287] ⌈ Upon the completion of [SWS_Dcm_00286], the Dcm shall generate a response message including the respective PID, FreezeFrame Number and the associated data record for the requested FreezeFrame number. ⌋*()*

[SWS_Dcm_01252] ⌈ If Dem_DcmReadDataOfOBDFreezeFrame() returns E_NOT_OK and a single PID is requested, the Dcm shall not provide any answer. ⌋*()*

[SWS_Dcm_01253] ⌈ If Dem_DcmReadDataOfOBDFreezeFrame() returns E_NOT_OK and all PIDs from the requested multiple PID(s) are not supported, the Dcm shall not provide any answer. ⌋*()*

[SWS_Dcm_01254] ⌈ If Dem_DcmReadDataOfOBDFreezeFrame() returns E_NOT_OK and at least one PID from the requested multiple PID(s) is supported, the Dcm shall send a positive response including the data of the supported PID(s). ⌋*()*

7.5.3.5 Service $03 $07 $0A - Obtaining DTCs

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

[SWS_Dcm_00410] ⌈ The Dcm module shall implement OBD Service $07 (Request Emission-Related Diagnostic Trouble Codes Detected during Current or Last Completed Driving Cycle) in compliance to all provisions of the OBD standard. ⌋*()*

[SWS_Dcm_00411] ⌈ The Dcm module shall implement OBD Service $0A (Request Emission-Related Diagnostic Trouble Codes with Permanent Status) in compliance to all provisions of the OBD standard. ⌋*()*

An external test tool can request an emission-related ECU to report all stored, pending or permanent emission-related DTCs by sending the request $03, $07, $0A respectively.

[SWS_Dcm_00289] ⌈ When receiving a request for OBD Service $03, the Dcm module shall obtain from the DEM all DTCs in primary memory and with a "confirmed" status using the functions Dem_SetDTCFilter() and Dem_GetNextFilteredDTC(). ⌋*()*

Note: The Dcm module can get an indication of 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.

[SWS_Dcm_00412] ⌈ When receiving a request for OBD Service $07, the Dcm module shall obtain from the DEM module all DTCs in primary memory with a "pending" status using the functions Dem_SetDTCFilter() and Dem_GetNextFilteredDTC(). ⌋*()*

Note: The Dcm module can get an indication of 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.

[SWS_Dcm_00330] ⌈ When receiving a request for OBD Service $0A, the Dcm module shall obtain from the DEM all DTCs stored in permanent memory using the functions Dem_SetDTCFilter() and Dem_GetNextFilteredDTC(). ⌋*()*

Note: The Dcm module can get an indication of 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 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
ClientId	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)	Client Id for this Dcm instance (see DcmDemClientRef)
DTCStatusMask	0x08 (confirmed bit set)	0x04(pending bit set)	0x00
DTCFormat	DEM_DTC_FORMAT_OBD	DEM_DTC_FORMAT_OBD	DEM_DTC_FORMAT_OBD
DTCOrigin	DEM_DTC_ORIGIN_OBD_RELEVANT_MEMORY	DEM_DTC_ORIGIN_OBD_RELEVANT_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
FilterForFaultDetectionCounter	DEM_FILTER_FOR_FDC_NO	DEM_FILTER_FOR_FDC_NO	DEM_FILTER_FOR_FDC_NO

Table 7.36: Dem_SetDTCFilter Parameters

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

[SWS_Dcm_01227] ⌈ Dem_GetNextFilteredDTC returns DEM_NO_SUCH_ELEMENT and at least one matching element could be retrieved before, the Dcm shall send a positive response including these data elements and the number of DTCs. ⌋*()*

[SWS_Dcm_01228] ⌈ If Dem_GetNextFilteredDTC returns DEM_NO_SUCH_ELEMENT and no matching element could be retrieved before, the Dcm shall send a positive response with the number of DTCs set to 0x00. ⌋*()*

7.5.3.6 Service $04 - Clear/reset emission-related diagnostic information

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

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

[SWS_Dcm_00004] ⌈ When receiving a request for OBD Service $04, the Dcm module shall call the interface Dem_SelectDTC with the following parameter values:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef)
- DTC = DEM_DTC_GROUP_ALL_DTCS
- DTCFormat = DEM_DTC_FORMAT_OBD
- DTCOrigin = DEM_DTC_ORIGIN_OBD_RELEVANT_MEMORY

⌋*(SRS_Diag_04058)*

Note: this interface will always return E_OK.

[SWS_Dcm_01401] ⌈ After calling Dem_SelectDTC, the Dcm shall call the interface Dem_ClearDTC() with the following parameter value:

- ClientId: Client Id for this Dcm instance (see DcmDemClientRef)

⌋*()*

[SWS_Dcm_00413] ⌈ In case Dem_ClearDTC() returns E_OK, the Dcm module shall send a positive response. ⌋*()*

[SWS_Dcm_00703] ⌈ In case Dem_ClearDTC() returns DEM_PENDING, the Dcm shall invoke Dem_ClearDTC() on next Dcm_MainFunction call again. It is up to the Dcm to send NRC 78 (ResponsePending) to respect the response behaviour ⌋*()*

[SWS_Dcm_00704] ⌈ In case Dem_ClearDTC() returns DEM_CLEAR_FAILED, the Dcm shall send a negative response 0x22 (conditionsNotCorrect). ⌋*()*

[SWS_Dcm_00967] ⌈ In case Dem_ClearDTC() returns DEM_CLEAR_BUSY, the Dcm shall send a negative response 0x22 (ConditionsNotCorrect). ⌋*()*

[SWS_Dcm_01067] ⌈ In case Dem_ClearDTC() returns DEM_CLEAR_MEMORY_ERROR, the Dcm module shall send a negative response 0x22 (ConditionNotCorrect). ⌋*()*

7.5.3.7 Service $06 - Request On-Board Monitoring Test-results for Specific Monitored Systems

7.5.3.7.1 General requirements

[SWS_Dcm_00414] ⌈ The Dcm module shall implement OBD Service $06 (Request On-Board Monitoring Test-results for Specific Monitored Systems) in compliance to all provisions of the OBD standard. ⌋*()*

Using Service $06, an external test tool can request an emission-related ECU to return the DTR's associated with the OBDMID or to return the supported OBDMIDs. OBD reserves certain OBDMIDs for the special purpose of obtaining the list of supported OBDMIDs in a certain range. These OBDMIDs are called "availability OBDMIDs" and are $00, $20, $40, $60, $80, $A0, $C0 an $E0.

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

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

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

7.5.3.7.2 Test results obtained via Dem interaction

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

[SWS_Dcm_00957] ⌈ On reception of an OBD Service $06 request with only "availability OBDMID(s)" as parameter(s), the Dcm module shall obtain the supported OBDMIDs by calling the Dem interface Dem_DcmGetAvailableOBDMIDs()for each "availability OBDMID ($00, $20, ...)" contained within the request and concatenate the results within the response message according to ISO-15031-5 [1]. ⌋*()*

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

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

[SWS_Dcm_00417] ⌈ The Dcm module shall implement OBD Service $08 (Request Control of On-Board System, Test or Component) in compliance to all provisions of the OBD standard. ⌋*()*

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

The Dcm module's configuration defines the TIDs that are available on the ECU for the purpose of OBD Service $08. The configuration defines for each such TID (see configuration parameter DcmDspRequestControlTestId):

- the name of the port the Dcm uses to access the RequestControlServices interface (see configuration parameter DcmDspRequestControl)
- the number of bytes this function takes as input (see configuration parameter DcmDspRequestControlInBufferSize)
- the number of bytes this function writes as output (see configuration parameter DcmDspRequestControlOutBufferSize)

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

[SWS_Dcm_00418] ⌈ On reception of an OBD Service $08 request with one or more "availability TIDs" as parameter, the Dcm module shall respond with the corresponding supported (=configured) TIDs. ⌋*()*

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

[SWS_Dcm_00419] ⌈ On reception of an OBD Service $08 request with a TID that is not an "availability TID", the Dcm module shall invoke the configured Xxx_RequestControl() function with the following parameters values: InBuffer: data contained in the OBD request (the size of which must correspond to the size configured in the Dcm module's configuration) OutBuffer: space in which the RequestControl function can store its result (the size of the buffer is taken from the Dcm module's configuration) ⌋*()*

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

[SWS_Dcm_00948] ⌈ As specified in [3, SAE J1979], unused data bytes shall be filled with $00. ⌋*()*

[SWS_Dcm_01192] ⌈ If Xxx_RequestControl() doesn't return E_OK, the Dcm shall return NRC 0x22. ⌋*()*

7.5.3.9 Service $09 - Request Vehicle Information

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

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

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

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

To provide OBD Service $09, the Dcm relies on external functions that allow it to obtain the value of an InfoType data. There is one such function per InfoType data that is needed by the DCM.

When invoking a Xxx_GetInfotypeValueData() function, the Dcm module provides a buffer of the correct size in which the value of the InfoType data can be stored. The entity providing the actual implementation of the Xxx_GetInfotypeValueData() function for a specific InfoType data might be a SW-C or another basic software module. The origin of the function is part of the Dcm module's configuration.

Certain InfoTypes needed by the Dcm to provide Service $09 are provided by the DEM. This is handled in the Dcm configuration.

[SWS_Dcm_00422] ⌈ On reception of an OBD Service $09 request with one or more "availability InfoTypes" as parameter, the Dcm module shall respond with the corresponding supported (=configured) InfoTypes. ⌋*()*

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

[SWS_Dcm_00423] ⌈ On reception of an OBD Service $09 request for an InfoType that is not an "availability InfoType", the Dcm module shall obtain the value of this InfoType by invoking all the configured Xxx_GetInfotypeValueData() function for every data of this InfoType and shall return the value as response to Service $09 ⌋*()*

[SWS_Dcm_00684] ⌈ In case DcmDspVehInfoNODIProvResp is set to FALSE, in addition to collect the available InfoType value contributions from the individual SW-C, the Dcm shall compute the data byte NofDataItems in the diagnostic response, which defines the number of DataItems included in one InfoType. ⌋*()*

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

[SWS_Dcm_01167] ⌈ In case DcmDspVehInfoNODIProvResp is set to TRUE, the Dcm shall take over the value returned by the provider and report it as NofDataItems in the diagnostic response. ⌋*()*

[SWS_Dcm_CONSTR_6045] ⌈ In case the responsibility is on provider side (DcmDspVehInfoNODIProvResp is set to TRUE), only one DcmDspVehInfoData container shall be allowed. ⌋*()*

[SWS_Dcm_CONSTR_6046] ⌈ In case DcmDspVehInfoDataUsePort is set to FALSE and DcmDspVehInfoDataReadFnc is set to either Dem_DcmGetInfoTypeValue08 or Dem_DcmGetInfoTypeValue0B then DcmDspVehInfoNODIProvResp shall be set to TRUE. ⌋*()*

Note : The integrator has to make sure that the buffer determined by the DcmDspVehInfoDataSize is sufficiently sized to receive the data returned by the provider of the data.

[SWS_Dcm_01191] ⌈ If Xxx_GetInfoTypeValueData() doesn't return E_OK or E_PENDING, the Dcm shall return NRC 0x12. ⌋*()*

7.5.4 Interaction usecases

The Dcm shall be able to manage a jump to the bootloader / jump due to ECUReset request. Due to the diversity of possibility to realize this jump, this will be done using callout call.

7.5.4.1 Jump to Bootloader

4 different use cases have been identified for the jump to the bootloader, if all preconditions are fulfilled and assuming the 'suppressPosRspMsgIndicationBit ' flag is set to 'false':

1. The application immediately sends a final positive response and then jumps to the bootloader
2. The application first sends a NRC 0x78 response, then the final positive response and afterwards jumps to the bootloader

3. The application immediately jumps to the bootloader and the bootloader sends the final positive response
4. The application first sends a NRC 0x78 response, then jumps to the bootloader and the bootloader sends the final positive response

Note: In case the 'suppressPosRspMsgIndicationBit' flag is set to 'true', use case '1' and use case '3' will not issue a positive response.

[SWS_Dcm_00532] ⌈ On reception of service DiagnosticSessionControl if the provided session is used to jump to OEM bootloader (parameter DcmDspSessionForBoot set to DCM_OEM_BOOT or DCM_OEM_BOOT_RESPAPP) the Dcm shall prepare the jump to the OEM bootloader (see [SWS_Dcm_00535]) by triggering the mode switch of ModeDeclarationGroupPrototype DcmEcuReset to JUMPTOBOOTLOADER. ⌋*(SRS_Diag_04098)*

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

[SWS_Dcm_00592] ⌈ On reception of service DiagnosticSessionControl if the provided session is used to jump to System Supplier bootloader (parameter DcmDspSessionForBoot set to DCM_SYS_BOOT or DCM_SYS_BOOT_RESPAPP) the Dcm shall prepare the jump to the System Supplier bootloader (see [SWS_Dcm_00535]) by triggering the mode switch of ModeDeclarationGroupPrototype DcmEcuReset to JUMPTOSYSSUPPLIERBOOTLOADER ⌋*(SRS_Diag_04098)*

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

[SWS_Dcm_01164] ⌈ In case the service DiagnosticSessionControl implies an ECU reset, the Dcm shall ignore all further requests while that reset is being processed. ⌋*()*

[SWS_Dcm_00654] ⌈ In case the ModeDeclarationGroupPrototype DcmEcuReset is switched to mode JUMPTOBOOTLOADER or JUMPTOSYSSUPPLIERBOOTLOADER and the configuration parameter DcmSendRespPendOnRestart is set to TRUE, the Dcm shall trigger transmission of NRC 0x78 - RCR-RP. ⌋*(SRS_Diag_04098)*

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

[SWS_Dcm_01175] ⌈ In case the ModeDeclarationGroupPrototype DcmEcuReset can not be switched JUMPTOBOOTLOADER or JUMPTOSYSSUPPLIERBOOTLOADER, the Dcm shall answer negatively to the request with NRC 0x22 (Conditions not correct). ⌋*()*

[SWS_Dcm_00535] ⌈ If the jump to bootloader is requested (see [SWS_Dcm_00532], [SWS_Dcm_00592], the configuration parameter DcmSendRespPendOnRestart is set to TRUE (see [SWS_Dcm_00654]) and the configuration parameter DcmDspSessionForBoot is set to DCM_OEM_BOOT or DCM_SYS_BOOT, the Dcm shall call Dcm_SetProgConditions after a successful transmission of NRC 0x78 (Response pending). ⌋*(SRS_Diag_04098)*

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

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

[SWS_Dcm_01163] ⌈ In the context of a request to jump to the bootloader (see [SWS_Dcm_00532] and [SWS_Dcm_00592]), after Dcm_SetProgConditions returns E_OK according to [SWS_Dcm_00535], the Dcm shall trigger the mode switch of the ModeDeclarationGroupPrototype DcmEcuReset to EXECUTE. ⌋ *(SRS_Diag_04098)*

[SWS_Dcm_01177] ⌈ If the jump to bootloader is requested (see [SWS_Dcm_00532], [SWS_Dcm_00592], the configuration parameter DcmSendRespPendOnRestart is set to TRUE (see [SWS_Dcm_00654]), and the configuration parameter DcmDspSessionForBoot is set to DCM_OEM_BOOT_RESPAPP or DCM_SYS_BOOT_RESPAPP, the Dcm shall initiate the final response after a successful transmission of NRC 0x78 (Response pending). ⌋ *(SRS_Diag_04098)*

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

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

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

[SWS_Dcm_01178] ⌈ In case the ModeDeclarationGroupPrototype DcmEcuReset is switched to mode JUMPTOBOOTLOADER or JUMPTOSYSSUPPLIERBOOTLOADER, the configuration parameter DcmSendRespPendOnRestart is set to FALSE and the configuration parameter DcmDspSessionForBoot is set to DCM_OEM_BOOT_RESPAPP or DCM_SYS_BOOT_RESPAPP , the Dcm shall initiate the final response ⌋ *()*

[SWS_Dcm_01179] ⌈ In case the final response has been successfully sent according to [SWS_Dcm_01177] or [SWS_Dcm_01178], the Dcm shall call Dcm_SetProgConditions ⌋ *()*

[SWS_Dcm_01180] ⌈ If Dcm_SetProgConditions returns E_OK according to [SWS_Dcm_01179], the Dcm shall trigger the mode switch of the ModeDeclarationGroupPrototype DcmEcuReset to EXECUTE. ⌋ *()*

[SWS_Dcm_01181] ⌈ If Dcm_SetProgConditions returns E_NOT_OK according to [SWS_Dcm_01179], the Dcm shall not request any reset, shall not perform the jump to bootloader, and shall not switch the ModeDeclarationGroupPrototype DcmEcuReset to EXECUTE. ⌋ *()*

[SWS_Dcm_00720] ⌈ In case the ModeDeclarationGroupPrototype DcmEcuReset is switched to mode JUMPTOBOOTLOADER or JUMPTOSYSSUPPLIERBOOTLOADER, the configuration parameter DcmSendRespPendOnRestart is set to FALSE and the configuration parameter DcmDspSessionForBoot it set to DCM_OEM_BOOT or DCM_SYS_BOOT, the Dcm shall call Dcm_SetProgConditions immediately. (see [SWS_Dcm_00532] and [SWS_Dcm_00592]) ⌋*()*

[SWS_Dcm_00719] ⌈ If Dcm_SetProgConditions returns E_OK according to [SWS_Dcm_00720], the Dcm shall shall trigger the mode switch of the ModeDeclarationGroupPrototype DcmEcuReset to EXECUTE without sending a NRC 0x78 (Response pending). ⌋*()*

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

[SWS_Dcm_00715] ⌈ If the jump to bootloader is requested (see [SWS_Dcm_00532] and [SWS_Dcm_00592]) and if the call to Dcm_SetProgConditions returns E_NOT_OK (see [SWS_Dcm_00535] and [SWS_Dcm_00720]), no further action shall be taken by the Dcm and negative response NRC 0x22 (Conditions not correct) shall be returned. ⌋*()*

7.5.4.2 Jump due to ECUReset

On reception of an ECUReset Service 0x11 request, if the configuration parameter DcmResponseToEcuReset is set to AFTER_RESET, the Dcm will set the ResponseRequired flag by calling Dcm_SetProgConditions.

[SWS_Dcm_01423] Answer to ECUReset request ⌈ On reception of an ECUReset Service 0x11 request, if DcmResponseToEcuReset is set to AFTER_RESET, the Dcm shall answer to EcuReset service after the reset. ⌋*(SRS_Diag_04098)*

[SWS_Dcm_01424] Answer to ECUReset request ⌈ On reception of an ECUReset Service 0x11 request, if DcmResponseToEcuReset is set to BEFORE_RESET, the Dcm shall answer to EcuReset service before the reset. ⌋*(SRS_Diag_04098)*

[SWS_Dcm_01425] Answer to ECUReset request ⌈ If the Dcm intiates a reset and DcmSendRespPendOnRestart is set to TRUE, the Dcm shall trigger transmission of NRC 0x78 (Response pending) before the reset. ⌋*(SRS_Diag_04098)*

7.5.4.3 Jump from Bootloader / ECUReset

[SWS_Dcm_00536] ⌈ At Dcm initialization, the Dcm shall call Dcm_GetProgConditions to know if the initialization is the consequence of a jump from the bootloader / ECUReset. ⌋*(SRS_Diag_04098)*

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

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

[SWS_Dcm_00767] ⌈ When the ComM reports full communication to the Dcm, the Dcm shall send the Response to the Service Id passed in the Dcm_ProgConditionsType. ⌋*(SRS_Diag_04098)*

[SWS_Dcm_00768] ⌈ If the initialization of the Dcm is the consequence of a jump from the bootloader (see [SWS_Dcm_00536] and the application is updated by an FLASH download (Dcm_ProgConditionsType.ApplUpdated == True), the Dcm shall call BswM_Dcm_ApplicationUpdated() to notify the BswM that the application was updated. ⌋*()*

7.5.4.4 Flags management

7.5.4.4.1 Jump to Bootloader

[SWS_Dcm_01182] ⌈ On reception of a UDS Service 0x10 request (Diagnostic Session Control) with subfunction 0x02 (Start Programming Session), the Dcm shall set the ReprogramingRequest flag and, if indicated for this service, the ResponseRequired flag by calling Dcm_SetProgConditions. ⌋*()*

[SWS_Dcm_01183] ⌈ Depending on configuration parameter DcmDspSessionForBoot, either the application shall send the positive response (if suppressPosRspMsgIndicationBit = FALSE) or after an ECU reset, when the bootloader is started, it shall send a response and clear the ResponseRequired flag. In either case, the bootloader shall clear the ReprogramingRequest flag. ⌋*()*

[SWS_Dcm_01185] ⌈ In case that, during jump to Bootloader, the Dcm_SetProgConditions API returns E_NOT_OK, a DET error shall be reported (DCM_E_SET_PROG_CONDITIONS_FAIL) and normal functionality shall resume. ⌋*()*

7.5.4.4.2 Jump from Bootloader

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

After an ECU reset, when the newly programmed application is started for the first time, the Dcm will read the *ApplUpdated* and *ResponseRequired* flag by calling

Dcm_GetProgConditions. During this function call the *ApplUpdated* and *ResponseRequired* flags are cleared by the integration code.

7.6 Error notification

The Default 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).

7.7 Debugging

7.8 Synchronous and Asynchronous implementation

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

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

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 canceling a pending request, see [SWS_Dcm_00686]).

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

[SWS_Dcm_01187] ⌈ If an asynchronous interface is used, the Dcm shall consider the Output data (OUT) only valid after the last call to the interface that returns E_OK. ⌋*()*

[SWS_Dcm_01188] ⌈ If an asynchronous interface is used, the Dcm shall consider the OUT-parameter ErrorCode only valid after the last call to the interface that returns E_NOT_OK ⌋*()*

Note : The "last call" to the interface is the call that returns with a value that indicates that processing has finished, i.e. E_OK or E_NOT_OK.

Note : INOUT parameter are a combination of the requirements above, i.e. on each call of the interface the parameters shall have a valid in-value, and the Dcm considers the out-value valid only after the last call of the interface.

[SWS_Dcm_01189] ⌈ If an asynchronous interface is used, the Dcm shall provide the values originating from the request for the Input data (IN) on every call to the interface. ⌋*()*

Note: Requirements [SWS_Dcm_01187], [SWS_Dcm_01188] and [SWS_Dcm_01189] do not apply for functions where a deviant behaviour is explicitly specified.

7.9 DID configuration

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

- The individual DID configuration,which required one connection (either via a port or a c-function) per configured data element of the respective DID to access to the data (reading, writing and controlling). The interface DataServices should be used for each DID in this case.
- The DID range configuration, used to handle a set of DIDs sharing the same behavior uniformly in one SW-C with only one port-connection. The interface DataServices_DIDRange_{Range} should be used in this case. Using this configuration allows an interface optimization.The following parameters shall be configured in order to use the DIDRange optimization: DcmDspDidRangeIdentifierLowerLimit and DcmDspDidRangeIdentifierUpperLimit which delimited the range of the DIDs. DcmDspDidRangeMaxDataLength and DcmDspDidRangeHasGaps.

[SWS_Dcm_01174] ⌈ If DcmVinRef is configured then the VIN shall be fetched once by the Dcm during startup by calling Dcm_GetVin. ⌋*()*

7.9.1 Individual DID

The individual DID can be configured in ECUC_Dcm_00601: DcmDspDid. A unique DID identifier is configured on 2 bytes in ECUC_Dcm_00602: DcmDspDidIdentifier. In case the DID refers to other DIDs, the link between them can be configured in ECUC_Dcm_00606 : DcmDspDidRef[4]. Each DID allows to access to signal data values (by reading and/or writing). The signal reference (to DcmDspData) and the position of the data in the diagnostic answer (for reading) or request (for writing) can be configured in ECUC_Dcm_00813: DcmDspDidSignal.

[4] Overview of DcmDspDid container is described in chapter 10.3.5.3.1

The configuration of the data belonging to the DID can be provided in the container DcmDspData (ECUC_Dcm_00869). This container collects the following information:

- The Data endianness of the data belonging to the DID (ECUC_Dcm_00986: DcmDspDataEndianness)
- The length and the type of the data (ECUC_Dcm_00605: DcmDspDataByteSize, ECUC_Dcm_00985 : DcmDspDataType)
- The interface to be used to access to the data (ECUC_Dcm_00713: DcmDspDataUsePort)
- The NRAM blockId to access the data (ECUC_Dcm_00809 : DcmDspDataBlockIdRef)
- The interfaces to the application in order to :
  - Check if the DID can be accessed in the current conditions. This can be achieved by checking for each DataElement if the conditions to read the data are satisfied (ECUC_Dcm_00677: DcmDspDataConditionCheckReadFnc and ECUC_Dcm_00955: DcmDspDataConditionCheckReadFncUsed)
  - Request to freeze the current state of an IOControl (ECUC_Dcm_00674: DcmDspDataFreezeCurrentStateFnc)
  - Get the scaling information of the DID. This can be achieved by getting the scaling information for each DataElement (ECUC_Dcm_00676: DcmDspDataGetScalingInfoFnc)
  - Request the data length of a DataElement (ECUC_Dcm_00671: DcmDspDataReadDataLengthFnc)
  - Read a certain ECU signal (ECUC_Dcm_00824: DcmDspDataReadEcuSignal).
  - Access in reading or writing to the data (ECUC_Dcm_00669 : DcmDspDataReadFnc, ECUC_Dcm_00670: DcmDspDataWriteFnc)
  - Request to reset an IOControl to default value (ECUC_Dcm_00673 : DcmDspDataResetToDefaultFnc)
  - Request to return control to ECU of an IOControl (ECUC_Dcm_00672 : DcmDspDataReturnControlToEcuFnc)
  - Request to adjust the IO signal (ECUC_Dcm_00675 : DcmDspDataShortTermAdjustmentFnc)

It is also possible to configure an alternative diagnosis representation via ECUC_Dcm_00993: DcmDspDiagnosisScaling.

The following example shows how to configure the containers DcmDspDid and DcmDspData for an individual DID 0xF080. This configuration allows access to a byte of data via synchronous C APIs WriteDID_F080 (for writing) and ReadDID_F080 (for reading).

- DcmDspDidIdentifier=0xF080
- DcmDspDidByteOffset=0
- DcmDspDidDataRef=DcmDspData_F080
- DcmDspDataByteSize=8
- DcmDspDataType=UINT8_N
- DcmDspDataUsePort=USE_DATA_SYNCH_FNC
- DcmDspDataWriteFnc=WriteDID_F080
- DcmDspDataReadFnc=ReadDID_F080

[SWS_Dcm_CONSTR_6067] Dependency for DcmDspDataBlockIdRef ⌈ DcmDspDataBlockIdRef shall be only present if DcmDspDataUsePort is set to USE_BLOCK_ID. ⌋*()*

7.9.2 DID ranges

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

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

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

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

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

ReadDidRangeDataLength operation allows to request the application to return the data length of a DID Range.

[SWS_Dcm_CONSTR_6020] Definition of allowed DID access ⌈ Any defined range shall only reference via DcmDspDidRangeInfoRef. The sub-containers DcmDspDidControl and DcmDspDidDefineinDcmDspDidInfo shall not be used] . ⌋*()*

[SWS_Dcm_CONSTR_6021] DID ranges cannot be mapped on DDDIDs, because service 0x2C DDDID does not support the range feature. Practically DcmDspDidRangeIdentifierLowerLimit and DcmDspDidRangeIdentifierUpperLimit should not include DIDs of the range 0xF200 till 0xF3FF. ⌈Any defined range shall only reference DcmDspDidInfo via DcmDspDidRangeInfoRef, having set DcmDspDidDynamicallyDefined == False. ⌋*()*

7.10 Startup behavior

[SWS_Dcm_00334] ⌈ Dcm_Init shall initialize all Dcm global variables with the values of the configuration ⌋*()*

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

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

[SWS_Dcm_00333] ⌈

Module	***Imported Type***
ComStack_Types	BufReq_ReturnType NetworkHandleType PduIdType PduInfoType PduLengthType RetryInfoType TPParameterType
Dem	Dem_DTCFormatType Dem_DTCOriginType Dem_DTCRequestType Dem_DTCSeverityType Dem_DTCTranslationFormatType

	Dem_UdsStatusByteType
GENERIC TYPES	<EcuSignalDataType>
NvM	NvM_BlockIdType
SchM	SchM_ReturnType
Std_Types	Std_ReturnType Std_VersionInfoType
UNDEFINED TYPES	DcmDspRoutineSignalType bit

Table 8.1: Dcm_ImportedTypes

⌋*()*

8.2 Type definitions

The Dcm module shall ensure that implementation-specific types are not "visible" outside of Dcm. Otherwise, the complete architecture would be corrupted.

This section lists the types which are defined by the Dcm SWS.

8.2.1 Dcm_StatusType

[SWS_Dcm_00976] ⌈

Name:	Dcm_StatusType		
Type:	uint8		
Range:	DCM_E_OK	0x00	This value is representing a successful operation.
	DCM_E_ROE_NOT_ACCEPTED	0x06	ResponseOnOneEvent request is not accepted by DCM (e.g. old ResponseOnOneEvent is not finished) (used at API: Dcm_ResponseOnOneEvent())
	DCM_E_PERIODICID_NOT_ACCEPTED	0x07	Periodic transmission request is not accepted by DCM (e.g. old Periodic transmission is not finished) (used at API: Dcm_ResponseOnOneDataByPeriodicId ())
Description:	Base item type to transport status information.		

Table 8.2: Dcm_StatusType

⌋()

8.2.2 Dcm_CommunicationModeType

[SWS_Dcm_00981] ⌈

Name:	Dcm_CommunicationModeType		
Type:	uint8		
Range:	DCM_ENABLE_RX_TX_NORM	0x00	Enable the Rx and Tx for normal communication
	DCM_ENABLE_RX_DISABLE_TX_NORM	0x01	Enable the Rx and disable the Tx for normal communication
	DCM_DISABLE_RX_ENABLE_TX_NORM	0x02	Disable the Rx and enable the Tx for normal communication
	DCM_DISABLE_RX_TX_NORMAL	0x03	Disable Rx and Tx for normal communication
	DCM_ENABLE_RX_TX_NM	0x04	Enable the Rx and Tx for network management communication
	DCM_ENABLE_RX_DISABLE_TX_NM	0x05	Enable Rx and disable the Tx for network management communication
	DCM_DISABLE_RX_ENABLE_TX_NM	0x06	Disable the Rx and enable the Tx for network management communication
	DCM_DISABLE_RX_TX_NM	0x07	Disable 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:	–		

Table 8.3: Dcm_CommunicationModeType

⌋()

8.2.3 Dcm_ConfigType

[SWS_Dcm_00982] ⌈

Name:	Dcm_ConfigType
Type:	Structure

Range:	Implementation specific		–
Description:	This type defines a data structure for the post build parameters of the DCM . At initialization the DCM gets a pointer to a structure of this type to get access to its configuration data, which is necessary for initialization.		

Table 8.4: Dcm_ConfigType

⌋()

8.2.4 Dcm_ReturnReadMemoryType

[SWS_Dcm_00985] ⌈

Name:	Dcm_ReturnReadMemoryType		
Type:	uint8		
Range:	DCM_READ_OK	0x00	Reading has been done
	DCM_READ_PENDING	0x01	Reading is pending, another call is request to finalize the reading
	DCM_READ_FAILED	0x02	Reading has failed
	DCM_READ_FORCE_RCRRP	0x03	Reading is pending, the Response pending transmission starts immediately
Description:	Return values of Callout Dcm_ReadMemory		

Table 8.5: Dcm_ReturnReadMemoryType

⌋()

8.2.5 Dcm_ReturnWriteMemoryType

[SWS_Dcm_00986] ⌈

Name:	Dcm_ReturnWriteMemoryType		
Type:	uint8		
Range:	DCM_WRITE_OK	0x00	Writing has been done
	DCM_WRITE_PENDING	0x01	Writing is pending, another called is requested
	DCM_WRITE_FAILED	0x02	The writing has failed
	DCM_WRITE_FORCE_RCRRP	0x03	Writing is pending, the Response pending transmission starts immediately
Description:	Return type of callout Dcm_WriteMemory		

Table 8.6: Dcm_ReturnWriteMemoryType

⌋()

8.2.6 Dcm_EcuStartModeType

[SWS_Dcm_00987] ⌈

Name:	`Dcm_EcuStartModeType`		
Type:	`uint8`		
Range:	`DCM_COLD_START`	`0x00`	The ECU starts normally
	`DCM_WARM_START`	`0x01`	The ECU starts from a bootloader jump
Description:	Allows the DCM to know if a diagnostic response shall be sent in the case of a jump from bootloader		

Table 8.7: Dcm_EcuStartModeType

⌋*()*

8.2.7 Dcm_ProgConditionsType

[SWS_Dcm_00988] ⌈

Name:	`Dcm_ProgConditionsType`		
Type:	`Structure`		
Element:	`uint16`	`ConnectionId`	Unique id of the connection on which the request has been received
	`uint16`	`TesterAddress`	Source address of the received request if meta data is enabled, otherwise the value as configured in DcmDslProtocolRxTesterSourceAddr
	`uint8`	`Sid`	Service identifier of the received request
	`uint8`	`SubFncId`	Identifier of the received subfonction
	`boolean`	`Reprograming Request`	Set to true in order to request reprogramming of the ECU.
	`boolean`	`ApplUpdated`	Indicate whether the application has been updated or not.
	`boolean`	`ResponseRequired`	Set to true in case the flashloader or application shall send a response.
Description:	Used in Dcm_SetProgConditions() to allow the integrator to store relevant information prior to jumping to bootloader / jump due to ECUReset request.		

Table 8.8: Dcm_ProgConditionsType

⌋*()*

8.2.8 Dcm_MsgItemType

[SWS_Dcm_00989] ⌈

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

Table 8.9: Dcm_MsgItemType

⌋*()*

8.2.9 Dcm_MsgType

[SWS_Dcm_00990] ⌈

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

Table 8.10: Dcm_MsgType

⌋*()*

8.2.10 Dcm_MsgLenType

[SWS_Dcm_00991] ⌈

Name:	`Dcm_MsgLenType`
Type:	`uint32`
Description:	Length of diagnostic message (request, positive or negative response). The maximum length is dependent of the underlying transport protocol/media.

Table 8.11: Dcm_MsgLenType

⌋*()*

8.2.11 Dcm_MsgAddInfoType

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

[SWS_Dcm_00992] ⌈

Name:	`Dcm_MsgAddInfoType`
Type:	`Structure`

Element:	`bit`	`reqType`	(Pos LSB+0) 0 = physical request 1 = functional request
	`bit`	`suppressPos Response`	Position LSB+1 0 = no (do not suppress) 1 = yes (no positive response will be sent)
Description:	Additional information on message request. Datastructure: Bitfield		

Table 8.12: Dcm_MsgAddInfoType

⌋*()*

8.2.12 Dcm_IdContextType

[SWS_Dcm_00993] ⌈

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

Table 8.13: Dcm_IdContextType

⌋*()*

8.2.13 Dcm_MsgContextType

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

[SWS_Dcm_00994] ⌈

Name:	`Dcm_MsgContextType`		
Type:	`Structure`		
Element:	`Dcm_MsgType`	`reqData`	Request data, starting directly after service identifier (which is not part of this data)
	`Dcm_MsgLenType`	`reqDataLen`	Request data length (excluding service identifier)
	`Dcm_MsgType`	`resData`	Positive response data, starting directly after service identifier (which is not part of this data).
	`Dcm_MsgLenType`	`resDataLen`	Positive response data length (excluding service identifier)

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

Description:	This data structure contains all information which is necessary to process a diagnostic message from request to response and response confirmation.

Table 8.14: Dcm_MsgContextType

⌋*()*

8.2.14 Dcm_ExtendedOpStatusType

[SWS_Dcm_91015] ⌈

Name:	`Dcm_ExtendedOpStatusType`		
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 Variation
	`DCM_POS_RESPONSE_SENT`	`0x04`	Indicates that a positive response has been sent successfully
	`DCM_POS_RESPONSE_FAILED`	`0x05`	Indicates that a positive response has not been sent successfully
	`DCM_NEG_RESPONSE_SENT`	`0x06`	Indicates that a negative response has been sent successfully
	`DCM_NEG_RESPONSE_FAILED`	`0x07`	Indicates that a negative response has not been sent successfully
Description:	–		

Table 8.15: Dcm_ExtendedOpStatusType

⌋*()*

8.3 Function definitions

This section defines the functions provided for other modules.

8.3.1 Functions provided for other BSW components

8.3.1.1 Dcm_Init

[SWS_Dcm_00037] ⌈

Service name:	Dcm_Init	

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

Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfigPtr	Pointer to configuration set in Variant Post-Build.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Service for basic initialization of DCM module.	

Table 8.16: Dcm_Init

⌋*(SRS_BSW_00438, SRS_BSW_00101, SRS_BSW_00438)*

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

8.3.1.2 Dcm_GetVersionInfo

[SWS_Dcm_00065] ⌈

Service name:	Dcm_GetVersionInfo	

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

Service ID[hex]:	0x24	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versionInfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Returns the version information of this module	

Table 8.17: Dcm_GetVersionInfo

⌋*(SRS_BSW_00407, SRS_BSW_00482, SRS_BSW_00003)*

8.3.1.3 Dcm_DemTriggerOnDTCStatus

[SWS_Dcm_00614] ⌈

Service name:	Dcm_DemTriggerOnDTCStatus	

Syntax:	`Std_ReturnType Dcm_DemTriggerOnDTCStatus(`
	`uint32 DTC,`
	`Dem_UdsStatusByteType DTCStatusOld,`
	`Dem_UdsStatusByteType DTCStatusNew`
	`)`	

Service ID[hex]:	0x2B	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	DTC DTCStatusOld DTCStatusNew	This is the DTC the change trigger is assigned to. DTC status before change 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 status byte. Allows to trigger on ROE Event for subservice OnDTCStatusChanged.	

Table 8.18: Dcm_DemTriggerOnDTCStatus

⌋*()*

8.3.1.4 Dcm_GetVin

[SWS_Dcm_00950] ⌈

Service name:	Dcm_GetVin	

Syntax:	`Std_ReturnType Dcm_GetVin(`
	`uint8* Data`
	`)`	

Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	Data	Pointer to where to store the VIN
Return value:	Std_ReturnType	E_OK: The Data pointer has been filled with valid VIN E_NOT_OK: The default VIN will be used in the DoIP
Description:	Function to get the VIN (as defined in SAE J1979-DA)	

Table 8.19: Dcm_GetVin

⌋*()*

Note: After fetching the VIN, the Dcm can offer the data to all users without worrying that the data is unavailable if a user asks for it. This is necessary because the VIN could not be fetched synchronously for all settings of DcmDspDidDataUsePort.

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.

8.3.2.1 Dcm_GetSecurityLevel

[SWS_Dcm_00338] ⌈

Service name:	Dcm_GetSecurityLevel	

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

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

Table 8.20: Dcm_GetSecurityLevel

⌋*(SRS_Diag_04005, SRS_Diag_04011)*

8.3.2.2 Dcm_GetSesCtrlType

[SWS_Dcm_00339] ⌈

Service name:	Dcm_GetSesCtrlType

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

Service ID[hex]:	0x06
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None

Parameters (inout):	None	
Parameters (out):	SesCtrlType	Active Session Control Type value Content is according to "diagnosticSessionType" parameter of DiagnosticSessionControl request (see [11])
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	This function provides the active session control type value.	

Table 8.21: Dcm_GetSesCtrlType

⌋*(SRS_Diag_04006, SRS_Diag_04011)*

8.3.2.3 Dcm_GetActiveProtocol

[SWS_Dcm_00340] ⌈

Service name:	Dcm_GetActiveProtocol	

Syntax:	`Std_ReturnType Dcm_GetActiveProtocol(`
	`Dcm_ProtocolType* ActiveProtocolType,`
	`uint16* ConnectionId,`
	`uint16* TesterSourceAddress`
	`)`	

Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	ActiveProtocolType ConnectionId TesterSourceAddress	Active protocol type value Unique connection identifier source address of the tester
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	This function returns the active UDS protocol details	

Table 8.22: Dcm_GetActiveProtocol

⌋*(SRS_Diag_04011)*

8.3.2.4 Dcm_ResetToDefaultSession

[SWS_Dcm_00520] ⌈

Service name:	Dcm_ResetToDefaultSession

Syntax:	`Std_ReturnType Dcm_ResetToDefaultSession(`
	`void`
	`)`

Service ID[hex]:	0x2a
Sync/Async:	Synchronous
Reentrancy:	Reentrant

Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	The call to this function allows the application to reset the current session to Default session. Example: Automatic termination of an extended diagnostic session upon exceeding of a speed limit.	

Table 8.23: Dcm_ResetToDefaultSession

⌋*()*

8.3.2.5 Dcm_TriggerOnEvent

[SWS_Dcm_00521] ⌈

Service name:	Dcm_TriggerOnEvent	

Syntax:	`Std_ReturnType Dcm_TriggerOnEvent(`
	`uint8 RoeEventId`
	`)`	

Service ID[hex]:	0x2D	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	RoeEventId	Identifier of the event that is triggered
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: RoeEventId value is valid E_NOT_OK: RoeEventId value is not valid
Description:	The call to this function allows to trigger an event linked to a ResponseOnEvent request. On the function call, the DCM will execute the associated service if the corresponding Mode of the RoeEventId is 'ROE started'.	

Table 8.24: Dcm_TriggerOnEvent

⌋*()*

8.3.2.6 Dcm_ SetActiveDiagnostic

[SWS_Dcm_01068] ⌈

Service name:	Dcm_SetActiveDiagnostic

Syntax:	`Std_ReturnType Dcm_SetActiveDiagnostic(`
	`boolean active`
	`)`

Service ID[hex]:	0x56
Sync/Async:	Synchronous
Reentrancy:	Reentrant

Parameters (in):	active	If false Dcm shall not call ComM_DCM_ActiveDiagnostic(). If true Dcm will call ComM_DCM_ActiveDiagnostic().
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: this value is always returned.
Description:	Allows to activate and deactivate the call of ComM_DCM_ActiveDiagnostic() function.	

Table 8.25: Dcm_SetActiveDiagnostic

⌋()

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

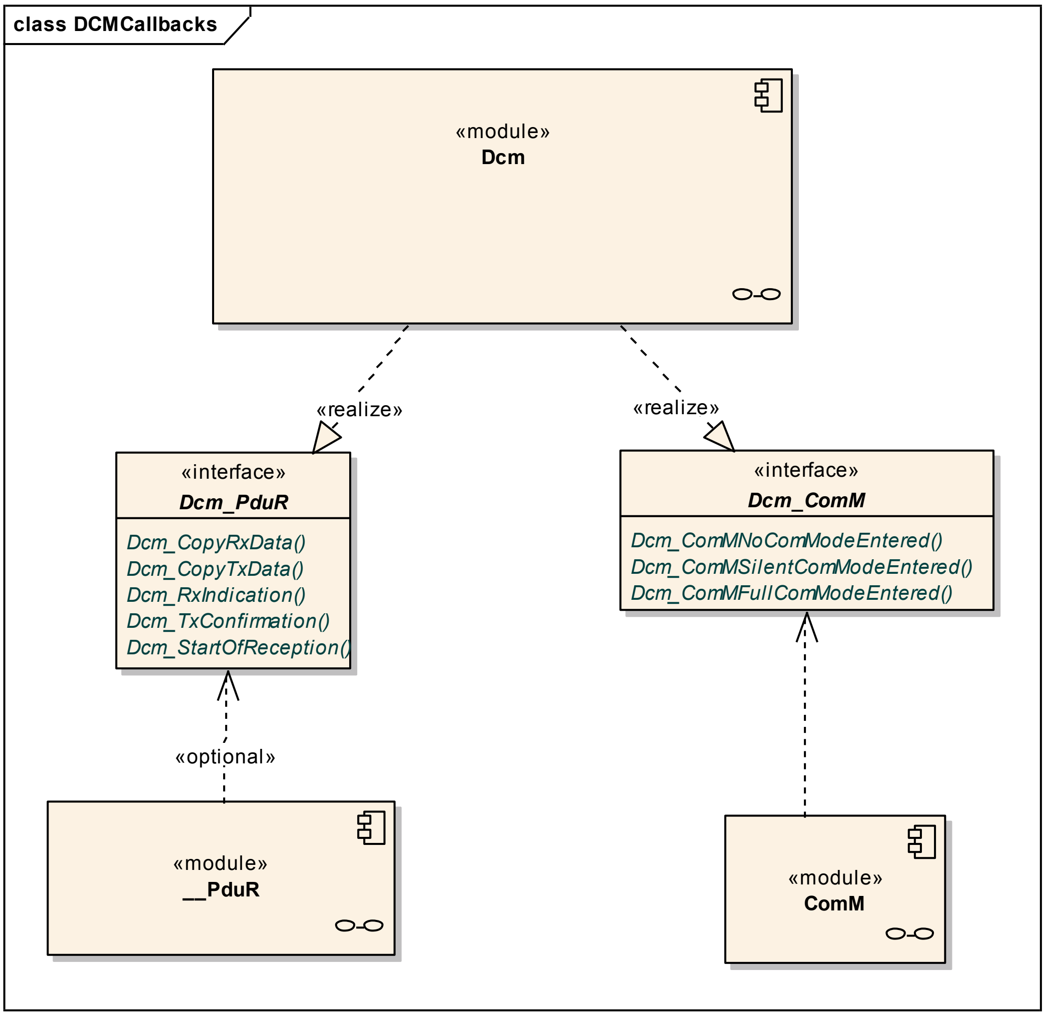


Figure 8.1: Overview of the callbacks provided by the DCM

8.4.1 Dcm_StartOfReception

[SWS_Dcm_00094] ⌈

Service name:	Dcm_StartOfReception	

Syntax:	`BufReq_ReturnType Dcm_StartOfReception(`
	`PduIdType id,`
	`const PduInfoType* info,`
	`PduLengthType TpSduLength,`
	`PduLengthType* bufferSizePtr`
	`)`	

Service ID[hex]:	0x46	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	Identification of the I-PDU.
	info	Pointer to a PduInfoType structure containing the payload data (without protocol information) and payload length of the first frame or single frame of a transport protocol I-PDU reception, and the MetaData related to this PDU. If neither first/single frame data nor MetaData are available, this parameter is set to NULL_PTR.
	TpSduLength	Total length of the N-SDU to be received.
Parameters (inout):	None	
Parameters (out):	bufferSizePtr	Available receive buffer in the receiving module. This parameter will be used to compute the Block Size (BS) in the transport protocol module.
Return value:	BufReq_ReturnType	BUFREQ_OK: Connection has been accepted. bufferSizePtr indicates the available receive buffer; reception is continued. If no buffer of the requested size is available, a receive buffer size of 0 shall be indicated by bufferSizePtr. BUFREQ_E_NOT_OK: Connection has been rejected; reception is aborted. bufferSizePtr remains unchanged. BUFREQ_E_OVFL: No buffer of the required length can be provided; reception is aborted. bufferSizePtr remains unchanged.
Description:	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSduLength equal to 0.	

Table 8.26: Dcm_StartOfReception

⌋*()*

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.

8.4.2 Dcm_CopyRxData

[SWS_Dcm_00556] ⌈

Service name:	Dcm_CopyRxData	

Syntax:	`BufReq_ReturnType Dcm_CopyRxData(`
	`PduIdType id,`
	`const PduInfoType* info,`
	`PduLengthType* bufferSizePtr`
	`)`	

Service ID[hex]:	0x44	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id info	Identification of the received I-PDU. Provides the source buffer (SduDataPtr) and the number of bytes to be copied (SduLength). An SduLength of 0 can be used to query the current amount of available buffer in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
Parameters (inout):	None	
Parameters (out):	bufferSizePtr	Available receive buffer after data has been copied.
Return value:	BufReq_ReturnType	BUFREQ_OK: Data copied successfully BUFREQ_E_NOT_OK: Data was not copied because an error occurred.
Description:	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining buffer is written to the position indicated by bufferSizePtr.	

Table 8.27: Dcm_CopyRxData

⌋*()*

8.4.3 Dcm_TpRxIndication

[SWS_Dcm_00093] ⌈

Service name:	Dcm_TpRxIndication	

Syntax:	`void Dcm_TpRxIndication(`
	`PduIdType id,`
	`Std_ReturnType result`
	`)`	

Service ID[hex]:	0x45	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id result	Identification of the received I-PDU. Result of the reception.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	

Description:	Called after an I-PDU has been received via the TP API, the result indicates whether the transmission was successful or not.

Table 8.28: Dcm_TpRxIndication

⌋*()*

8.4.4 Dcm_CopyTxData

[SWS_Dcm_00092] ⌈

Service name:	Dcm_CopyTxData	

Syntax:	`BufReq_ReturnType Dcm_CopyTxData(`
	`PduIdType id,`
	`const PduInfoType* info,`
	`const RetryInfoType* retry,`
	`PduLengthType* availableDataPtr`
	`)`	

Service ID[hex]:	0x43	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	Identification of the transmitted I-PDU.
	info	Provides the destination buffer (SduDataPtr) and the number of bytes to be copied (SduLength). If not enough transmit data is available, no data is copied by the upper layer module and BUFREQ_E_BUSY is returned. The lower layer module may retry the call. An SduLength of 0 can be used to indicate state changes in the retry parameter or to query the current amount of available data in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.

	retry	This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems. If the retry parameter is a NULL_PTR, it indicates that the transmit data can be removed from the buffer immediately after it has been copied. Otherwise, the retry parameter must point to a valid RetryInfoType element. If TpDataState indicates TP_CONFPENDING, the previously copied data must remain in the TP buffer to be available for error recovery. TP_DATACONF indicates that all data that has been copied before this call is confirmed and can be removed from the TP buffer. Data copied by this API call is excluded and will be confirmed later. TP_DATARETRY indicates that this API call shall copy previously copied data in order to recover from an error. In this case TxTpDataCnt specifies the offset in bytes from the current data copy position.
Parameters (inout):	None	
Parameters (out):	availableDataPtr	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
Return value:	BufReq_ReturnType	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_BUSY: Request could not be fulfilled, because the required amount of Tx data is not available. The lower layer module may retry this call later on. No data has been copied. BUFREQ_E_NOT_OK: Data has not been copied. Request failed.
Description:	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.	

Table 8.29: Dcm_CopyTxData

⌋*()*

8.4.5 Dcm_TpTxConfirmation

[SWS_Dcm_00351] ⌈

Service name:	Dcm_TpTxConfirmation

Syntax:	`void Dcm_TpTxConfirmation(`
	`PduIdType id,`
	`Std_ReturnType result`
	`)`	

Service ID[hex]:	0x48	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id result	Identification of the transmitted I-PDU. Result of the transmission of the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.	

Table 8.30: Dcm_TpTxConfirmation

⌋*()*

8.4.6 Dcm_TxConfirmation

[SWS_Dcm_01092] ⌈

Service name:	Dcm_TxConfirmation	

Syntax:	`void Dcm_TxConfirmation(`
	`PduIdType TxPduId,`
	`Std_ReturnType result`
	`)`	

Service ID[hex]:	0x40	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId result	ID of the PDU that has been transmitted. E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	

Table 8.31: Dcm_TxConfirmation

⌋*()*

8.4.7 Dcm_ComM_NoComModeEntered

[SWS_Dcm_00356] ⌈

Service name:	Dcm_ComM_NoComModeEntered	

Syntax:	`void Dcm_ComM_NoComModeEntered(`
	`uint8 NetworkId`
	`)`	

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

Table 8.32: Dcm_ComM_NoComModeEntered

⌋*()*

8.4.8 Dcm_ComM_SilentComModeEntered

[SWS_Dcm_00358] ⌈

Service name:	Dcm_ComM_SilentComModeEntered	

Syntax:	`void Dcm_ComM_SilentComModeEntered(`
	`uint8 NetworkId`
	`)`	

Service ID[hex]:	0x22	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	NetworkId	Identifier of the network concerned by the mode change
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This call informs the Dcm module about a ComM mode change to COMM_SILENT_COMMUNICATION.	

Table 8.33: Dcm_ComM_SilentComModeEntered

⌋*()*

8.4.9 Dcm_ComM_FullComModeEntered

[SWS_Dcm_00360] ⌈

Service name:	Dcm_ComM_FullComModeEntered

Syntax:	`void Dcm_ComM_FullComModeEntered(`
	`uint8 NetworkId`
	`)`	

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

Table 8.34: Dcm_ComM_FullComModeEntered

⌋*()*

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 which identified a memory block.

According to that, the Dcm is not able to fully support the implementation of ISO14229-1 [2]services which 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

[SWS_Dcm_00539] ⌈

Service name:	Dcm_ReadMemory

Syntax:	`Dcm_ReturnReadMemoryType Dcm_ReadMemory(`
	`Dcm_OpStatusType OpStatus,`
	`uint8 MemoryIdentifier,`
	`uint32 MemoryAddress,`
	`uint32 MemorySize,`
	`uint8* MemoryData,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x26	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL DCM_PENDING DCM_CANCEL DCM_FORCE_RCRRP_OK
	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)
	ErrorCode	If the operation Dcm_ReadMemory returns value DCM_READ_FAILED, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Dcm_ReturnReadMemoryType	DCM_READ_OK: read was successful DCM_READ_FAILED: read was not successful DCM_READ_PENDING: read is not yet finished DCM_READ_FORCE_RCRRP: reading is pending, the Response pending transmission starts immediately
Description:	The Dcm_ReadMemory callout is used to request memory data identified by the parameter memoryAddress and memorySize from the UDS request message. This service is needed for the implementation of UDS services: - ReadMemoryByAddress - RequestUpload - ReadDataByIdentifier (in case of Dynamical DID defined by memory address) - TransferData	

Table 8.35: Dcm_ReadMemory

⌋*()*

8.5.2 Dcm_WriteMemory

[SWS_Dcm_00540] ⌈

Service name:	Dcm_WriteMemory

Syntax:	`Dcm_ReturnWriteMemoryType Dcm_WriteMemory(`
	`Dcm_OpStatusType OpStatus,`
	`uint8 MemoryIdentifier,`
	`uint32 MemoryAddress,`
	`uint32 MemorySize,`
	`const uint8* MemoryData,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x27	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL DCM_PENDING DCM_CANCEL DCM_FORCE_RCRRP_OK
	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):	ErrorCode	If the operation Dcm_WriteMemory returns value DCM_WRITE_FAILED, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Dcm_ReturnWrite MemoryType	DCM_WRITE_OK: write was successful DCM_WRITE_FAILED: write was not successful DCM_WRITE_PENDING: write is not yet finished DCM_WRITE_FORCE_RCRRP: writing is pending, the Response pending transmission starts immediately
Description:	The Dcm_WriteMemory callout is used to write memory data identified by the parameter memoryAddress and memorySize. This service is needed for the implementation of UDS services : - WriteMemoryByAddress - RequestDownload - TransferData	

Table 8.36: Dcm_WriteMemory

⌋*()*

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

8.5.3 Dcm_SetProgConditions

[SWS_Dcm_00543] ⌈

Service name:	Dcm_SetProgConditions	

Syntax:	`Std_ReturnType Dcm_SetProgConditions(`
	`Dcm_OpStatusType OpStatus,`
	`const Dcm_ProgConditionsType* ProgConditions`
	`)`	

Service ID[hex]:	0x61	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	OpStatus DCM_INITIAL DCM_PENDING DCM_CANCEL DCM_FORCE_RCRRP_OK
	ProgConditions	Conditions on which the jump to bootloader has been requested
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Conditions have correctly been set E_NOT_OK: Conditions cannot be set DCM_E_PENDING: Conditions set is in progress, a further call to this API is needed to end the setting DCM_E_FORCE_RCRRP: Application requests the transmission of a response Response Pending (NRC 0x78)
Description:	The Dcm_SetProgConditions callout allows the integrator to store relevant information prior to jumping to bootloader / jump due to ECUReset request. The context parameter are defined in Dcm_ProgConditionsType.	

Table 8.37: Dcm_SetProgConditions

⌋*()*

Note: In case the SecurityAccess AttemptCounter needs to be shared between application and bootloader in addition to the ProgConditionStructure the current value can be retrieved via the `API Xxx_GetSecurityAttemptCounter` (see chapter 7.5.4 Interaction)

8.5.4 Dcm_GetProgConditions

[SWS_Dcm_00544] ⌈

Service name:	Dcm_GetProgConditions

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

Service ID[hex]:	0x62
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. The context parameters are defined in Dcm_ProgConditionsType.	

Table 8.38: Dcm_GetProgConditions

⌋()

8.5.5 Dcm_ProcessRequestTransferExit

[SWS_Dcm_00755] ⌈

Service name:	Dcm_ProcessRequestTransferExit	

Syntax:	`Std_ReturnType Dcm_ProcessRequestTransferExit(`
	`Dcm_OpStatusType OpStatus,`
	`const uint8* transferRequestParameterRecord,`
	`uint32 transferRequestParameterRecordSize,`
	`uint8* transferResponseParameterRecord,`
	`uint32* transferResponseParameterRecordSize,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x32	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL DCM_PENDING DCM_CANCEL
	transferRequestParameterRecord	Pointer to vehicle-manufacturer-specific data
	transferRequestParameterRecordSize	Length of ParameterRecord in bytes
Parameters (inout):	transferResponseParameterRecordSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in transferResponseParameterRecord.
Parameters (out):	transferResponseParameterRecord	Pointer to vehicle-manufacturer-specific data
	ErrorCode	see below
Return value:	Std_ReturnType	E_OK: Transfer was successful E_NOT_OK: Transfer was not successful or the response buffer is too small DCM_E_PENDING: Transfer is not yet finished

Description:	Callout function. DCM shall call this callout function to terminate a download or upload process. This callout is needed for the implementation of UDS service RequestTransferExit.

Table 8.39: Dcm_ProcessRequestTransferExit

⌋*()*

8.5.6 Dcm_ProcessRequestUpload

[SWS_Dcm_00756] ⌈

Service name:	Dcm_ProcessRequestUpload	

Syntax:	`Std_ReturnType Dcm_ProcessRequestUpload(`
	`Dcm_OpStatusType OpStatus,`
	`uint8 DataFormatIdentifier,`
	`uint32 MemoryAddress,`
	`uint32 MemorySize,`
	`uint32* BlockLength,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x31	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL DCM_PENDING DCM_CANCEL DCM_FORCE_RCRRP_OK
	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):	BlockLength	Max. Number of bytes for one Dcm_ReadMemory
Parameters (out):	ErrorCode	If the operation Dcm_ProcessRequestUpload returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful E_NOT_OK: Request was not successful DCM_E_PENDING: Request is not yet finished
Description:	Callout function. DCM shall call this callout function to start an upload process. This service is needed for the implementation of UDS service RequestUpload.	

Table 8.40: Dcm_ProcessRequestUpload

⌋()

8.5.7 Dcm_ProcessRequestDownload

[SWS_Dcm_00754] ⌈

Service name:	Dcm_ProcessRequestDownload	

Syntax:	`Std_ReturnType Dcm_ProcessRequestDownload(`
	`Dcm_OpStatusType OpStatus,`
	`uint8 DataFormatIdentifier,`
	`uint32 MemoryAddress,`
	`uint32 MemorySize,`
	`uint32* BlockLength,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x30	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL DCM_PENDING DCM_CANCEL
	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):	BlockLength	Max. Number of bytes for one Dcm_WriteMemory
Parameters (out):	ErrorCode	If the operation Dcm_ProcessRequestDownload returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful E_NOT_OK: Request was not successful DCM_E_PENDING: Request is not yet finished
Description:	Callout function. DCM shall call this callout function to start a download process. This service is needed for the implementation of UDS service RequestDownload.	

Table 8.41: Dcm_ProcessRequestDownload

⌋()

8.5.8 Dcm_ ProcessRequestFileTransfer

[SWS_Dcm_01120] ⌈

Service name:	Dcm_ProcessRequestFileTransfer	

Syntax:	`Std_ReturnType Dcm_ProcessRequestFileTransfer(`
	`Dcm_OpStatusType OpStatus,`
	`uint8 modeofOperation,`
	`uint16 fileSizeParameterLength,`
	`const uint8* filePathAndName,`
	`uint8 dataFormatIdentifier,`
	`uint8* fileSizeUncompressedOrDirInfoLength,`
	`uint8* fileSizeCompressed,`
	`uint32* BlockLength,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x57	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL DCM_PENDING DCM_CANCEL DCM_FORCE_RCRRP_OK
	modeofOperation	This data-parameter defines the type of operation to be applied to the file or directory indicated in the filePathAndName parameter.
	fileSizeParameter Length	Defines the length in byte for the parameter filePath.
	filePathAndName	Defines the file system location of the server where the file which shall be added, deleted, replaced or read from depending on the parameter modeOfOperation parameter. In addition this parameter includes the file name of the file which shall be added, deleted, replaced or read as part of the file path.
	dataFormatIdentifier	Defines the length (number of bytes) of the maxNumberOfBlockLength parameter.
Parameters (inout):	fileSizeUncompressedOrDirInfo Length	Defines the size of the uncompressed file to be uploaded or the length of the directory information to be read in bytes.
	fileSizeCompressed	Defines the size of the compressed file in bytes.
Parameters (out):	BlockLength	Max. Number of bytes for one Dcm_WriteMemory
	ErrorCode	If the operation Dcm_ ProcessRequestFileTransfer returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful E_NOT_OK: Request was not successful DCM_E_PENDING: Request is not yet finished
Description:	Callout function. DCM shall call this callout function to start a RequestFileTransferprocess. This service is needed for the implementation of UDS service RequestFileTransfer.	

Table 8.42: Dcm_ProcessRequestFileTransfer

⌋*()*

8.6 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.6.1 Dcm_MainFunction

[SWS_Dcm_00053] ⌈

Service name:	Dcm_MainFunction

Syntax:	`void Dcm_MainFunction(`
	`void`
	`)`

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

Table 8.43: Dcm_MainFunction

⌋*(SRS_BSW_00424, SRS_BSW_00373)*

8.7 Expected interfaces

In this chapter 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.

[SWS_Dcm_91001] ⌈

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 a PDU.

Table 8.44: Dcm Mandatory Interfaces

⌋*()*

8.7.2 Optional interfaces

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

[SWS_Dcm_91002] ⌈

API function	*Description*
BswM_Dcm_ApplicationUpdated	This function is called by the DCM in order to report an updated application.
BswM_Dcm_CommunicationMode_CurrentState	Function called by DCM to inform the BswM about the current state of the communication mode.
Dem_ClearDTC	Clears single DTCs, as well as groups of DTCs.
Dem_DcmGetAvailableOBDMIDs	Reports the value of a requested "availability-OBDMID" to the DCM upon a Service $06 request. Derived from that the tester displays the supported tests a mechanic can select from. API is needed in OBD-relevant ECUs only. API Availability: This API will be available only if ({ecuc(Dem/DemGeneral.DemOBDSupport)} != DEM_OBD_NO_OBD_SUPPORT)
Dem_DcmGetDTCOfOBDFreezeFrame	Gets DTC by freeze frame record number. API is needed in OBD-relevant ECUs only. API Availability: This API will be available only if ({ecuc(Dem/DemGeneral.DemOBDSupport)} != DEM_OBD_NO_OBD_SUPPORT)
Dem_DcmGetDTRData	Reports a DTR data along with TID-value, UaSID, test result with lower and upper limit. API is needed in OBD-relevant ECUs only. API Availability: This API will be available only if ({ecuc(Dem/DemGeneral.DemOBDSupport)} != DEM_OBD_NO_OBD_SUPPORT)
Dem_DcmGetNumTIDsOfOBDMID	Gets the number of TIDs per (functional) OBDMID. This can be used by the DCM to iteratively request for OBD/TID result data within a loop from 0....numberOfTIDs-1 API is needed in OBD-relevant ECUs only. API Availability: This API will be available only if ({ecuc(Dem/DemGeneral.DemOBDSupport)} != DEM_OBD_NO_OBD_SUPPORT)
Dem_DcmReadDataOfOBDFreezeFrame	Gets data element per PID and index of the most important freeze frame being selected for the output of service $02. The function stores the data in the provided DestBuffer. API is needed in OBD-relevant ECUs only. API Availability: This API will be available only if ({ecuc(Dem/DemGeneral.DemOBDSupport)} != DEM_OBD_NO_OBD_SUPPORT)
Dem_DisableDTCRecordUpdate	Disables the event memory update of a specific DTC (only one at one time).
Dem_DisableDTCSetting	Disables the DTC setting for all DTCs assigned to the DemEventMemorySet of the addressed client.
Dem_EnableDTCRecordUpdate	Enables the event memory update of the DTC disabled by Dem_DisableDTCRecordUpdate() before.
Dem_EnableDTCSetting	(Re)-Enables the DTC setting for all DTCs assigned to the DemEventMemorySet of the addressed client.

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_GetDTCSeverityAvailabilityMask	Gets the DTC Severity availability mask.
Dem_GetDTCStatusAvailabilityMask	Gets the DTC Status availability mask.
Dem_GetFunctionalUnitOfDTC	Gets the functional unit of the requested DTC.
Dem_GetNextExtendedDataRecord	Gets extended data record for the DTC selected by Dem_SelectExtendedDataRecord. The function stores the data in the provided DestBuffer.
Dem_GetNextFilteredDTC	Gets the next filtered DTC matching the filter criteria. For UDS services, the interface has an asynchronous behavior, because a large number of DTCs has to be processed.
Dem_GetNextFilteredDTCAndFDC	Gets the next filtered DTC and its associated Fault Detection Counter (FDC) matching the filter criteria. The interface has an asynchronous behavior, because a large number of DTCs has to be processed and the FDC might be received asynchronously from a SW-C, too.
Dem_GetNextFilteredDTCAndSeverity	Gets the next filtered DTC and its associated Severity matching the filter criteria. The interface has an asynchronous behavior, because a large number of DTCs has to be processed.
Dem_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_GetNextFreezeFrameData	Gets freeze frame data by the DTC selected by Dem_SelectFreezeFrameData. The function stores the data in the provided DestBuffer.
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_GetSizeOfExtendedDataRecordSelection	Gets the size of Extended Data Record by DTC selected by the call of Dem_SelectExtendedDataRecord.
Dem_GetSizeOfFreezeFrameSelection	Gets the size of freeze frame data by DTC selected by the call of Dem_SelectFreezeFrameData.
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). The DTCs of OBD Events Suppression shall be reported as Dem_WRONG_DTC.
Dem_GetTranslationType	Gets the supported DTC formats of the ECU.The supported formats are configured via DemTypeOfDTCSupported.

Dem_SetDTCFilter	Sets the DTC Filter. The server shall perform a bit-wise logical AND-ing operation between the parameter DTCStatusMask and the current UDS status in the server. In addition to the DTCStatusAvailabilityMask, the server shall return all DTCs for which the result of the AND-ing operation is non-zero [i.e. (statusOfDTC & DTCStatusMask) != 0]. The server shall process only the DTC Status bits that it is supporting. OBD Events Suppression shall be ignored for this computation. If no DTCs within the server match the masking criteria specified in the client&apos;s request, no DTC or status information shall be provided following the DTCStatusAvailabilityMask byte in the positive response message (((statusOfDTC & DTCStatusMask) != 0) && ((severity & DTCSeverityMask) != 0)) == TRUE
Dem_SetFreezeFrameRecordFilter	Sets a freeze frame record filter.
Det_ReportError	Service to report development errors.
IoHwAb_Dcm_<EcuSignalName>	This function provides control access to a certain ECU Signal to the DCM module (<EcuSignalname> is the symbolic name of an ECU Signal). The ECU signal can be locked and unlocked by this function. Locking ’freezes’ the ECU signal to the current value, the configured default value or a value given by the parameter ’signal’.
IoHwAb_Dcm_Read<EcuSignal Name>	This function provides read access to a certain ECU Signal to the DCM module (<EcuSignalname> is the symbolic name of an ECU Signal).
NvM_ReadBlock	Service to copy the data of the NV block to its corresponding RAM block.
NvM_SetBlockLockStatus	Service for setting the lock status of a permanent RAM block or of the explicit synchronization of a NVRAM block.
NvM_SetRamBlockStatus	Service for setting the RAM block status of a permanent RAM block or the status of the explicit synchronization of a NVRAM block.
NvM_WriteBlock	Service to copy the data of the RAM block to its corresponding NV block.
PduR_DcmCancelReceive	Requests cancellation of an ongoing reception of a PDU in a lower layer transport protocol module.
PduR_DcmCancelTransmit	Requests cancellation of an ongoing transmission of a PDU in a lower layer communication module.
PduR_DcmChangeParameter(obsolete)	Request to change a specific transport protocol parameter (e.g. block size).
SchM_ActMainFunction_Dcm	Invokes the SchM_ActMainFunction function to trigger the activation of a corresponding main processing function.

Table 8.45: Dcm Optional Interfaces

⌋*()*

Dem_DcmReadDataOfOBDFreezeFrame 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

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

8.7.3.1.1 GetSeed

If DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR, the following definition is used:

If DcmDspSecurityADRSize is present:

[SWS_Dcm_01151] ⌈

Service name:	Xxx_GetSeed	

Syntax:	`Std_ReturnType Xxx_GetSeed(`
	`const uint8* SecurityAccessDataRecord,`
	`Dcm_OpStatusType OpStatus,`
	`uint8* Seed,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x44	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SecurityAccessDataRecord	This data record contains additional data to calculate the seed value; the size of this parameter is DcmDspSecurityADRSize which is at least "1".
	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	Seed	Pointer for provided seed
	ErrorCode	If the operation Xxx_GetSeed returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	Request to application for asynchronous provision of seed value	

Table 8.46: Xxx_GetSeedAsynchAdr

⌋()

If DcmDspSecurityADRSize is not present:

[SWS_Dcm_91003] ⌈

Service name:	Xxx_GetSeed	

Syntax:	`Std_ReturnType Xxx_GetSeed(`
	`Dcm_OpStatusType OpStatus,`
	`uint8* Seed,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x45	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	Seed ErrorCode	Pointer for provided seed If the operation Xxx_GetSeed returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	Request to application for asynchronous provision of seed value	

Table 8.47: Xxx_GetSeedAsynch

⌋()

8.7.3.1.2 CompareKey

[SWS_Dcm_91004] ⌈

Service name:	Xxx_CompareKey	

Syntax:	`Std_ReturnType Xxx_CompareKey(`
	`const uint8* Key,`
	`Dcm_OpStatusType OpStatus,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x47	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Key OpStatus	Key, which needs to be compared Status of the current operation
Parameters (inout):	None	

Parameters (out):	ErrorCode	NRC to be sent if E_NOT_OK is returned
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish. DCM_E_COMPARE_KEY_FAILED: Key did not match.
Description:	Request to application for asynchronous comparing key (DcmDspSecurityUsePort = USE_ASYNCH_CLIENT_SERVER)	

Table 8.48: Xxx_CompareKeyAsynch

⌋()

8.7.3.1.3 GetSecurityAttemptCounter

[SWS_Dcm_01152] ⌈

Service name:	Xxx_GetSecurityAttemptCounter	

Syntax:	`Std_ReturnType Xxx_GetSecurityAttemptCounter(`
	`Dcm_OpStatusType OpStatus,`
	`uint8* AttemptCounter`
	`)`	

Service ID[hex]:	0x59	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL DCM_PENDING DCM_CANCEL
Parameters (inout):	None	
Parameters (out):	AttemptCounter	The attempt counter for this security level
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	Read the attempt counter for a specific security level from the application	

Table 8.49: Xxx_GetSecurityAttemptCounter

⌋()

Note: In case the Security Access AttemptCounter needs to be shared between application and bootloader, the application needs to consider this in the API-call `Xxx_GetSecurityAttemptCounter` (see chapter 7.5.4 Interaction). Further this has also impact on the security delay timer which needs to be considered.

8.7.3.1.4 SetSecurityAttemptCounter

[SWS_Dcm_01153] ⌈

Service name:	Xxx_SetSecurityAttemptCounter	

Syntax:	`Std_ReturnType Xxx_SetSecurityAttemptCounter(`
	`Dcm_OpStatusType OpStatus,`
	`uint8 AttemptCounter`
	`)`	

Service ID[hex]:	0x5a	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL DCM_PENDING DCM_CANCEL
	AttemptCounter	The attempt counter for this security level
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	Set the attempt counter for a specific security level in the application	

Table 8.50: Xxx_SetSecurityAttemptCounter

⌋*()*

8.7.3.2 DataServices

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

Note : The OpStatus parameter shall only exist for asynchronous operations (if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR or USE_DATA_ASYNCH_FNC or USE_DATA_ASYNCH_FNC_ERROR). In case of synchronous operations (DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC), the OpStatus parameter shall not exist.

8.7.3.2.1 ReadData

If DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used: **[SWS_Dcm_00793]** ⌈

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.	

Table 8.51: Xxx_ReadData1

⌋*()*

If DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC, the following definition is used:

[SWS_Dcm_91006] ⌈

Service name:	Xxx_ReadData	

Syntax:	`Std_ReturnType Xxx_ReadData(`
	`Dcm_OpStatusType OpStatus,`
	`uint8* Data`
	`)`	

Service ID[hex]:	0x3b	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	Data	Buffer where the requested data shall be copied to
Return value:	Std_ReturnType	E_OK: Request was successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application a data value of a DID/PID if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER.	

Table 8.52: Xxx_ReadData2

⌋*()*

If DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER_ERROR or USE_DATA_ASYNCH_FNC_ERROR, the following definition is used: **[SWS_Dcm_91005]** ⌈

Service name:	Xxx_ReadData

Syntax:	`Std_ReturnType Xxx_ReadData(`
	`Dcm_OpStatusType OpStatus,`
	`uint8* Data,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`

Service ID[hex]:	0x58
Sync/Async:	Asynchronous
Reentrancy:	Non Reentrant

Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	Data ErrorCode	Buffer where the requested data shall be copied to If the operation Xxx_ReadData returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application a data value of a DID/PID if DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER.	

Table 8.53: Xxx_ReadData3

⌋*()*

8.7.3.2.2 WriteData

If DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used:

If DcmDspDataType is NOT set to UINT8_DYN, the following definition is used:

[SWS_Dcm_00794] ⌈

Service name:	Xxx_WriteData	

Syntax:	`Std_ReturnType Xxx_WriteData(`
	`const uint8* Data,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x51	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Data	Buffer containing the data to be written
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_WriteData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests the application to write a data value of a DID.	

Table 8.54: Xxx_WriteData1

⌋*()*

If DcmDspDataType is set to UINT8_DYN, the following definition is used:
[SWS_Dcm_91007] ⌈

Service name:	Xxx_WriteData	

Syntax:	`Std_ReturnType Xxx_WriteData(`
	`const uint8* Data,`
	`uint16 DataLength,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x52	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Data DataLength	Buffer containing the data to be written Length in byte of the data to be written
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_WriteData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests the application to write a data value of a DID.	

Table 8.55: Xxx_WriteData2

⌋()

If DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR USE_DATA_ASYNCH_FNC_ERROR, the following definition is used:

If DcmDspDataType is NOT set to UINT8_DYN, the following definition is used:

[SWS_Dcm_91008] ⌈

Service name:	Xxx_WriteData	

Syntax:	`Std_ReturnType Xxx_WriteData(`
	`const uint8* Data,`
	`Dcm_OpStatusType OpStatus,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x35	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Data OpStatus	Buffer containing the data to be written Status of the current operation
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_WriteData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests the application to write a data value of a DID.	

Table 8.56: Xxx_WriteData3

⌋()

If DcmDspDataType is set to UINT8_DYN, the following definition is used:

[SWS_Dcm_91009] ⌈

Service name:	Xxx_WriteData	

Syntax:	`Std_ReturnType Xxx_WriteData(`
	`const uint8* Data,`
	`uint16 DataLength,`
	`Dcm_OpStatusType OpStatus,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x3e	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Data DataLength OpStatus	Buffer containing the data to be written Length in byte of the data to be written Status of the current operation
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_WriteData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests the application to write a data value of a DID.	

Table 8.57: Xxx_WriteData4

⌋()

8.7.3.2.3 ReadDataLength

If DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used:

[SWS_Dcm_00796] ⌈

Service name:	Xxx_ReadDataLength	

Syntax:	`Std_ReturnType Xxx_ReadDataLength(`
	`uint16* DataLength`
	`)`	

Service ID[hex]:	0x36	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	DataLength	Length in byte of the data to be read
Return value:	Std_ReturnType	E_OK: this value is always returned.

Description:	This function requests the application to return the data length in byte of a Data.

Table 8.58: Xxx_ReadDataLength1

⌋()

If DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR or USE_DATA_ASYNCH_FNC or USE_DATA_ASYNCH_FNC_ERROR, the following definition is used:

[SWS_Dcm_91010] ⌈

Service name:	Xxx_ReadDataLength	

Syntax:	`Std_ReturnType Xxx_ReadDataLength(`
	`Dcm_OpStatusType OpStatus,`
	`uint16* DataLength`
	`)`	

Service ID[hex]:	0x4c	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	DataLength	Length in byte of the data to be read
Return value:	Std_ReturnType	E_OK: this value is always returned. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests the application to return the data length in byte of a Data.	

Table 8.59: Xxx_ReadDataLength2

⌋()

8.7.3.2.4 ConditionCheckRead

If DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used:

[SWS_Dcm_00797] ⌈

Service name:	Xxx_ConditionCheckRead

Syntax:	`Std_ReturnType Xxx_ConditionCheckRead(`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`

Service ID[hex]:	0x49
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None

Parameters (out):	ErrorCode	If the operation Xxx_ConditionCheckRead returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests to the application if the conditions to read the Data are correct.	

Table 8.60: Xxx_ConditionCheckRead1

⌋()

If DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR USE_DATA_ASYNCH_FNC_ERROR, the following definition is used:

[SWS_Dcm_91011] ⌈

Service name:	Xxx_ConditionCheckRead	

Syntax:	`Std_ReturnType Xxx_ConditionCheckRead(`
	`Dcm_OpStatusType OpStatus,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x37	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ConditionCheckRead returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application if the conditions to read the Data are correct.	

Table 8.61: Xxx_ConditionCheckRead2

⌋()

8.7.3.2.5 GetScalingInformation

This function requests to the application for the scaling information of a Data (scalingByte and scalingByteExtension).

If DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC, the following definition is used: **[SWS_Dcm_00798]** ⌈

Service name:	Xxx_GetScalingInformation	

Syntax:	`Std_ReturnType Xxx_GetScalingInformation(`
	`uint8* ScalingInfo,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x4b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	ScalingInfo ErrorCode	Scaling information (scalingByte and scalingByte-Extension) If the operation Xxx_GetScalingInformation returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests to the application for the scaling information of a Data.	

Table 8.62: Xxx_GetScalingInformation1

⌋*()*

If DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR USE_DATA_ASYNCH_FNC_ERROR, the following definition is used:

[SWS_Dcm_91012] ⌈

Service name:	Xxx_GetScalingInformation	

Syntax:	`Std_ReturnType Xxx_GetScalingInformation(`
	`Dcm_OpStatusType OpStatus,`
	`uint8* ScalingInfo,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x38	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	ScalingInfo ErrorCode	Scaling information (scalingByte and scalingByte-Extension) If the operation Xxx_GetScalingInformation returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.

Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application for the scaling information of a Data.	

Table 8.63: Xxx_GetScalingInformation2

⌋*()*

8.7.3.2.6 ReturnControlToECU

[SWS_Dcm_01285] ⌈

Service name:	Xxx_ReturnControlToECU	

Syntax:	`Std_ReturnType Xxx_ReturnControlToECU(`
	`[Dcm_ControlMask_{Data}Type controlMask,]`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x4f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Variation:	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)}== (USE_DATA_SYNCH_FNC || USE_DATA_ASYNCH_FNC ||USE_DATA_ASYNCH_FNC_ERROR) && {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE) || ({ecuc(Dcm/Dc	
Parameters (in):	controlMask	– Variation: ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_SYNCH_FNC || USE_DATA_ASYNCH_FNC ||USE_DATA_ASYNCH_FNC_ERROR) && {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE)|| ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidResetToDefault)} == TRUE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)}== TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ReturnControlToECU returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.

Description:	This function requests to the application to return control to ECU of an IOControl.

Table 8.64: Xxx_ReturnControlToECU1

⌋*()*

Note: Square brackets [] indicate that an argument is optional.

8.7.3.2.7 ResetToDefault

8.7.3.2.7.1 Synchronous interface

[SWS_Dcm_01286] ⌈

Service name:	Xxx_ResetToDefault	

Syntax:	`Std_ReturnType Xxx_ResetToDefault(`
	`[Dcm_ControlMask_{Data}Type controlMask,]`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x4d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	controlMask	– Variation: ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_SYNCH_FNC)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidResetToDefault)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ResetToDefault returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests to the application to reset an IOControl to default value.	

Table 8.65: Xxx_ResetToDefault1

⌋*()*

Note: Square brackets [] indicate that an argument is optional.

8.7.3.2.7.2 Asynchronous interface

[SWS_Dcm_01314] ⌈

Service name:	Xxx_ResetToDefault	

Syntax:	`Std_ReturnType Xxx_ResetToDefault(`
	`Dcm_OpStatusType OpStatus,`
	`[Dcm_ControlMask_{Data}Type controlMask,]`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x3c	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus controlMask	Status of the current operation – Variation: ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_FNC || USE_DATA_ASYNCH_FNC_ERROR))&& ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidResetToDefault)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef -> DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ResetToDefault returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application to reset an IOControl to default value.	

Table 8.66: Xxx_ResetToDefault2

⌋*()*

Note: Square brackets [] indicate that an argument is optional.

8.7.3.2.8 FreezeCurrentState

8.7.3.2.8.1 Synchronous interface

[SWS_Dcm_01290] ⌈

Service name:	Xxx_FreezeCurrentState

Syntax:	`Std_ReturnType Xxx_FreezeCurrentState(`
	`[Dcm_ControlMask_{Data}Type controlMask,]`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x4a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	controlMask	– Variation: ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_FNC)&& ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef -> DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_FreezeCurrentState returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests to the application to freeze the current state of an IOControl.	

Table 8.67: Xxx_FreezeCurrentState1

⌋()

Note: Square brackets [] indicate that an argument is optional.

8.7.3.2.8.2 Asynchronous interface

[SWS_Dcm_01315] ⌈

Service name:	Xxx_FreezeCurrentState	

Syntax:	`Std_ReturnType Xxx_FreezeCurrentState(`
	`Dcm_OpStatusType OpStatus,`
	`[Dcm_ControlMask_{Data}Type controlMask,]`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x3a	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus controlMask	Status of the current operation – Variation:

		({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_FNC || USE_DATA_ASYNCH_FNC_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef -> DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_FreezeCurrentState returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application to freeze the current state of an IOControl.	

Table 8.68: Xxx_FreezeCurrentState2

⌋*()*

8.7.3.2.9 ShortTermAdjustment

8.7.3.2.9.1 Synchronous interface

[SWS_Dcm_00802] ⌈

Service name:	Xxx_ShortTermAdjustment	

Syntax:	`Std_ReturnType Xxx_ShortTermAdjustment(`
	`const uint8* ControlStateInfo,`
	`[Dcm_ControlMask_{Data}Type controlMask,]`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x50	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControlStateInfo	ControlState information contained in the ControlOptionRecord parameter of the InputOutputControlByIdentifier diagnostic request
	controlMask	– Variation:

		({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_FNC) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} != UINT8_DYN) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef -> DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ShortTermAdjustment returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function requests to the application to adjust the IO signal.	

Table 8.69: Xxx_ShortTermAdjSynchFixed

⌋*()*

Note: Square brackets [] indicate that an argument is optional.

8.7.3.2.9.2 Asynchronous interface

[SWS_Dcm_01316] ⌈

Service name:	Xxx_ShortTermAdjustment	

Syntax:	`Std_ReturnType Xxx_ShortTermAdjustment(`
	`const uint8* ControlStateInfo,`
	`uint16 DataLength,`
	`Dcm_OpStatusType OpStatus,`
	`[Dcm_ControlMask_{Data}Type controlMask,]`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x55	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControlStateInfo	ControlState information contained in the ControlOptionRecord parameter of the InputOutputControlByIdentifier diagnostic request
	DataLength	Length in byte of the data to be written
	OpStatus	Status of the current operation
	controlMask	–
		Variation:

		({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_FNC || USE_DATA_ASYNCH_FNC_ERROR))&& ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} == UINT8_DYN) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef -> DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)
Parameters (inout):	None	
Parameters (out):	ErrorCode	NRC to be sent in the negative response in case of failure (E_NOT_OK)
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application to adjust the IO signal.	

Table 8.70: Xxx_ShortTermAdjAsynchNonFixed

⌋*()*

8.7.3.3 DataServices_DIDRange

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 or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR USE_DATA_ASYNCH_FNC_ERROR). In case of synchronous operations (DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER or USE_DATA_SYNCH_FNC), the OpStatus parameter should not be used.

8.7.3.3.1 IsDidAvailable

[SWS_Dcm_00803] ⌈

Service name:	Xxx_IsDidAvailable	

Syntax:	`Std_ReturnType Xxx_IsDidAvailable(`
	`uint16 DID,`
	`Dcm_OpStatusType OpStatus,`
	`Dcm_DidSupportedType* supported`
	`)`	

Service ID[hex]:	0x53	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DID OpStatus	DID value Status of the current operation

Parameters (inout):	None	
Parameters (out):	supported	Indicate if the DID is available within the range. Returning DCM_DID_SUPPORTED means it is supported within the range, Returning DCM_DID_NOT_SUPPORTED means it is not supported within the range
Return value:	Std_ReturnType	E_OK: This value is returned when the Did is finally available. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests if a specific DID is available within the range or not.	

Table 8.71: Xxx_IsDidAvailableAsynch

⌋()

8.7.3.3.2 ReadDidData

[SWS_Dcm_00804] ⌈

Service name:	Xxx_ReadDidData	

Syntax:	`Std_ReturnType Xxx_ReadDidData(`
	`uint16 DID,`
	`uint8* Data,`
	`Dcm_OpStatusType OpStatus,`
	`uint16 DataLength,`
	`Dcm_NegativeResponseCodeType ErrorCode`
	`)`	

Service ID[hex]:	0x40	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DID OpStatus	Data ID value Status of the current operation
Parameters (inout):	None	
Parameters (out):	Data DataLength ErrorCode	Buffer where the requested data shall be copied to Length of the data to be read If the operation Xxx_ReadDidData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests to the application a data value of a DID	

Table 8.72: Xxx_ReadDidData

⌋()

8.7.3.3.3 WriteDidData

[SWS_Dcm_00805] ⌈

Service name:	Xxx_WriteDidData	

Syntax:	`Std_ReturnType Xxx_WriteDidData(`
	`uint16 DID,`
	`const uint8* Data,`
	`Dcm_OpStatusType OpStatus,`
	`uint16 DataLength,`
	`Dcm_NegativeResponseCodeType ErrorCode`
	`)`	

Service ID[hex]:	0x41	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DID Data OpStatus DataLength	Data ID value Buffer containing the data to be written Status of the current operation Length of the data to be written
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_WriteDidData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests the application to write a data value of a DID.	

Table 8.73: Xxx_WriteDidData

⌋*()*

8.7.3.3.4 ReadDidRangeDataLength

ReadDidRangeDataLength requests the application to return the data length of a DID range. This interface is used for UDS Service ReadDataByIdentifier.

[SWS_Dcm_01271] ⌈

Service name:	Xxx_ReadDidRangeDataLength	

Syntax:	`Std_ReturnType Xxx_ReadDidRangeDataLength(`
	`uint16 DID,`
	`Dcm_OpStatusType OpStatus,`
	`uint16* DataLength`
	`)`	

Service ID[hex]:	0x5e	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DID	Data ID value

	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	DataLength	Length of the data to be written/read
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	This function requests the application to return the data length of a DID Range.	

Table 8.74: Xxx_ReadDidRangeDataLength

⌋*()*

8.7.3.4 InfoTypesServices

8.7.3.4.1 GetInfotypeValueData

[SWS_Dcm_91014] ⌈

Service name:	Xxx_GetInfotypeValueData	

Syntax:	`Std_ReturnType Xxx_GetInfotypeValueData(`
	`Dcm_OpStatusType OpStatus,`
	`uint8* DataValueBuffer,`
	`uint8* DataValueBufferSize`
	`)`	

Service ID[hex]:	0x60	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	DataValueBuffer Size	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The callee fills in the number of written data bytes in DataValueBuffer.
Parameters (out):	DataValueBuffer	Buffer containing the Infotype information
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description:	The function provides the data related to the requested Infotype.	

Table 8.75: Xxx_GetInfotypeValueData

⌋*()*

8.7.3.5 RoutineServices

The operations mentioned in the following sub-chapters are only general examples, because the number of In and OUT parameters can be variable from 0 to an arbitrary number.It is therefore not possible to list all variations of operation prototypes.

8.7.3.5.1 Xxx_Start Operation

[SWS_Dcm_01203] ⌈

Service name:	Xxx_Start	

Syntax:	`Std_ReturnType Xxx_Start(`
	`[DcmDspRoutineSignalType dataIn_1,] ...`
	`[DcmDspRoutineSignalType dataIn_n,]`
	`[const uint8* dataInVar,]`
	`Dcm_OpStatusType OpStatus,`
	`[DcmDspRoutineSignalType dataOut_1,] ...`
	`[DcmDspRoutineSignalType dataOut_n,]`
	`[uint8* dataOutVar,]`
	`[uint16* variableDataLength,]`
	`Dcm_NegativeResponseCodeType ErrorCode`
	`)`	

Service ID[hex]:	0x5b	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	dataIn_1	Fixed-length input data provided in the routine control request
	...	...
	dataIn_n	Fixed-length input data provided in the routine control request
	dataInVar	Variable-length input data provided in the routine control request
	OpStatus	Status of the current operation
Parameters (inout):	variableDataLength	If variable length routine input data is used, this parameter contains the length in bytes of the dataInVar array. If variable length routine output data is used, this parameter contains the length in bytes of the dataOutVar parameter.
Parameters (out):	dataOut_1	Fixed-length output data to provide in the routine control response
	...	...
	dataOut_n	Fixed-length output data to provide in the routine control response
	dataOutVar	Variable-length output data to provide in the routine control response
	ErrorCode	If the operation Xxx_Start returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.

Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish. DCM_E_FORCE_RCRRP: application requests the transmission of a response Pending (NRC 0x78)
Description:	This function requests to the application to start the execution of a routine.	

Table 8.76: Xxx_Start

⌋*()*

Note: Square brackets [] indicate that an argument is optional.

8.7.3.5.2 Xxx_StartConfirmation Operation

[SWS_Dcm_91016] ⌈

Service name:	Xxx_StartConfirmation	

Syntax:	`Std_ReturnType Xxx_StartConfirmation(`
	`Dcm_ConfirmationStatusType ConfirmationStatus`
	`)`	

Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfirmationStatus	Confirmation status of a StartRoutine request
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function indicates the transmission of a response to a StartRoutine request	

Table 8.77: Xxx_StartConfirmation

⌋*()*

Note: Square brackets [] indicate that an argument is optional.

8.7.3.5.3 Xxx_Stop Operation

[SWS_Dcm_01204] ⌈

Service name:	Xxx_Stop

Syntax:	Std_ReturnType Xxx_Stop([DcmDspRoutineSignalType dataIn_1,] ... [DcmDspRoutineSignalType dataIn_n,] [const uint8* dataInVar,] [DcmDspRoutineSignalType dataOut_1,] ... [DcmDspRoutineSignalType dataOut_n,] [uint8* dataOutVar,] [uint16* variableDataLength,] Dcm_NegativeResponseCodeType* ErrorCode)	
Service ID[hex]:	0x5c	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	dataIn_1	Fixed-length input data provided in the routine control request
	...	...
	dataIn_n	Fixed-length input data provided in the routine control request
	dataInVar	Variable-length input data provided in the routine control request
Parameters (inout):	variableDataLength	If variable length routine input data is used, this parameter contains the length in bytes of the dataInVar array. If variable length routine output data is used, this parameter contains the length in bytes of the dataOutVar parameter.
Parameters (out):	dataOut_1	Fixed-length output data to provide in the routine control response
	...	...
	dataOut_n	Fixed-length output data to provide in the routine control response
	dataOutVar	Variable-length output data to provide in the routine control response
	ErrorCode	If the operation Xxx_Stop returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish DCM_E_FORCE_RCRRP: application requests the transmission of a response Pending (NRC 0x78)
Description:	This function requests to the application to stop the execution of a routine	

Table 8.78: Xxx_Stop

⌋*()*

Note: Square brackets [] indicate that an argument is optional.

8.7.3.5.4 Xxx_StopConfirmation Operation

[SWS_Dcm_91017] ⌈

Service name:	Xxx_StopConfirmation	

Syntax:	`Std_ReturnType Xxx_StopConfirmation(`
	`Dcm_ConfirmationStatusType ConfirmationStatus`
	`)`	

Service ID[hex]:	0x69	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfirmationStatus	Dcm_ConfirmationStatus Confirmation status of a StopRoutine request
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function indicates the transmission of a response to a StopRoutine request	

Table 8.79: Xxx_StopConfirmation

⌋*()*

Note: Square brackets [] indicate that an argument is optional.

8.7.3.5.5 Xxx_RequestResults Operation

[SWS_Dcm_91013] ⌈

Service name:	Xxx_RequestResults	

Syntax:	`Std_ReturnType Xxx_RequestResults(`
	`Dcm_OpStatusType OpStatus,`
	`[DcmDspRoutineSignalType* dataOut_1,] ...`
	`[DcmDspRoutineSignalType* dataOut_n,]`
	`[uint8* dataOutVar,]`
	`[uint16* variableDataLength,]`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x5d	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	OpStatus	Status of the current operation
Parameters (inout):	None	
Parameters (out):	dataOut_1 ... dataOut_n dataOutVar variableDataLength	Fixed-length Output data to provide in the routine control response ... Fixed-length Output data to provide in the routine control response Variable-length Output data to provide in the routine control response Length in bytes of the dataOutVar parameter.

	ErrorCode	If the operation Xxx_RequestResults returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish DCM_E_FORCE_RCRRP: application requests the transmission of a response Pending (NRC 0x78)
Description:	This function requests to the application the result of a routine execution	

Table 8.80: Xxx_RequestResults

⌋*()*

Note: Square brackets [] indicate that an argument is optional.

8.7.3.5.6 Xxx_RequestResultsConfirmation Operation

[SWS_Dcm_91018] ⌈

Service name:	Xxx_RequestResultsConfirmation	

Syntax:	`Std_ReturnType Xxx_RequestResultsConfirmation(`
	`Dcm_ConfirmationStatusType ConfirmationStatus`
	`)`	

Service ID[hex]:	0x70	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfirmationStatus	Confirmation status of a RequestRoutineResults request
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function indicates the transmission of a response to a RequestRoutineResults request	

Table 8.81: Xxx_RequestResultsConfirmation

⌋*()*

Note: Square brackets [] indicate that an argument is optional.

8.7.3.6 RequestControlServices

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

8.7.3.6.1 RequestControl callout

[SWS_Dcm_01338] ⌈

Service name:	Xxx_RequestControl	

Syntax:	`Std_ReturnType Xxx_RequestControl(`
	`uint8* OutBuffer,`
	`const uint8* InBuffer`
	`)`	

Service ID[hex]:	0x63	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	OutBuffer InBuffer	Output buffer in which the RequestControl function can store its result Input buffer containing the data of the OBD Service 0x08 request
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	Invokes a TID-specific function taking a configured number of bytes as input and returing a fixed number of bytes as output. This is typically used to implement OBD Service $08	

Table 8.82: Xxx_RequestControl

⌋*()*

8.7.3.7 CallbackDCMRequestServices

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

8.7.3.7.1 StartProtocol

[SWS_Dcm_01339] ⌈

Service name:	Xxx_StartProtocol	

Syntax:	`Std_ReturnType Xxx_StartProtocol(`
	`Dcm_ProtocolType ProtocolType,`
	`uint16 TesterSourceAddress,`
	`uint16 ConnectionId`
	`)`	

Service ID[hex]:	0x67	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ProtocolType TesterSourceAddress ConnectionId	Type of the protocol to be started source address of the tester Unique connection identifier
Parameters (inout):	None	

Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. E_PROTOCOL_NOT_ALLOWED: Protocol not allowed
Description:	This function allows the application to examine the environment conditions and enable/disable further processing of the protocol.	

Table 8.83: Xxx_StartProtocol

⌋*()*

8.7.3.7.2 StopProtocol

[SWS_Dcm_01340] ⌈

Service name:	Xxx_StopProtocol	

Syntax:	`Std_ReturnType Xxx_StopProtocol(`
	`Dcm_ProtocolType ProtocolType,`
	`uint16 TesterSourceAddress,`
	`uint16 ConnectionId`
	`)`	

Service ID[hex]:	0x64	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ProtocolType TesterSourceAddress ConnectionId	Type of the protocol to be stopped source address of the tester Unique connection identifier
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function informs the application of the protocol stop.	

Table 8.84: Xxx_StopProtocol

⌋*()*

8.7.3.8 ServiceRequestNotification

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

8.7.3.8.1 Indication

[SWS_Dcm_01341] ⌈

Service name:	Xxx_Indication	

Syntax:	`Std_ReturnType Xxx_Indication(`
	`uint8 SID,`
	`const uint8* RequestData,`
	`uint16 DataSize,`
	`uint8 ReqType,`
	`uint16 ConnectionId,`
	`Dcm_NegativeResponseCodeType* ErrorCode,`
	`Dcm_ProtocolType ProtocolType,`
	`uint16 TesterSourceAddress`
	`)`	

Service ID[hex]:	0x65	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SID	–
	RequestData	Complete request data (diagnostic buffer), except the service ID
	DataSize	Number of valid bytes in the RequestData parameter
	ReqType	Addressing type of the request(0=physical request,1=functional request)
	ConnectionId	Unique connection identifier
	ProtocolType	Type of the protocol to be indicated
	TesterSourceAddress	source address of the tester
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_Indication returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. E_REQUEST_NOT_ACCEPTED : Request not accepted
Description:	This function indicates to the application that a service is about to be executed and allows the application to reject the execution of the service request	

Table 8.85: Xxx_Indication

⌋*()*

8.7.3.8.2 Confirmation

[SWS_Dcm_01342] ⌈

Service name:	Xxx_Confirmation

Syntax:	`Std_ReturnType Xxx_Confirmation(`
	`uint8 SID,`
	`uint8 ReqType,`
	`uint16 ConnectionId,`
	`Dcm_ConfirmationStatusType ConfirmationStatus,`
	`Dcm_ProtocolType ProtocolType,`
	`uint16 TesterSourceAddress`
	`)`	

Service ID[hex]:	0x66	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SID	Value of service identifier
	ReqType	Addressing type of the request(0=physical request,1=functional request)
	ConnectionId	Unique connection identifier
	ConfirmationStatus	Confirmation of a successful transmission or a transmission error of a diagnostic service.
	ProtocolType	Type of Dcm Protocol
	TesterSourceAddress	source address of the tester
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description:	This function confirms to the application the successful transmission or a transmission error of a diagnostic service.	

Table 8.86: Xxx_Confirmation

⌋*()*

8.7.3.9 ClearDTCCheckFnc

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

[SWS_Dcm_01270] ⌈

Service name:	Xxx_ClearDTCCheckFnc	

Syntax:	`Std_ReturnType Xxx_ClearDTCCheckFnc(`
	`uint32 GoDTC,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x5f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	GoDTC	requested groupOfDTC
Parameters (inout):	None	
Parameters (out):	ErrorCode	If the operation Xxx_ClearDTCCheckFnc returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.

Return value:	Std_ReturnType	E_OK: application allows to clear the requested groupOfDTC E_NOT_OK: application does not allow to clear the requested groupOfDTC. Dcm shall send a negative response with the NRC returned in the ErrorCode
Description:	Callout function for condition check, manufacturer / supplier specific checks on the groupOfDTC, which is requested to clear.	

Table 8.87: Xxx_ClearDTCCheckFnc

⌋()

8.8 Dcm as Service-Component

8.8.1 Implementation Data Types

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

8.8.1.1 Dcm_OpStatusType

[SWS_Dcm_00984] ⌈

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

Table 8.88: Implementation Data Type Dcm_OpStatusType

⌋()

8.8.1.2 Dcm_ConfirmationStatusType

[SWS_Dcm_00983] ⌈

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

Table 8.89: Implementation Data Type Dcm_ConfirmationStatusType

⌋*()*

8.8.1.3 Dcm_SecLevelType

[SWS_Dcm_00977] ⌈

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

Table 8.90: Implementation Data Type Dcm_SecLevelType

⌋*()*

8.8.1.4 Dcm_SesCtrlType

[SWS_Dcm_00978] ⌈

Name	Dcm_SesCtrlType		
Kind	Type		
Derived from	uint8		
Description	Session type definition. 0, 127 and all values above 127 are reserved by ISO.		
Range	DCM_DEFAULT_SESSION	0x01	--
	DCM_PROGRAMMING_SESSION	0x02	--
	DCM_EXTENDED_DIAGNOSTIC_SESSION	0x03	--
	DCM_SAFETY_SYSTEM_DIAGNOSTIC_SESSION	0x04	--

	configuration dependent	0x40...0x7E	(according to "diagnosticSessionType" parameter of DiagnosticSessionControl request)
Variation	--		

Table 8.91: Implementation Data Type Dcm_SesCtrlType

⌋()

8.8.1.5 Dcm_ProtocolType

[SWS_Dcm_00979] ⌈

Name	Dcm_ProtocolType		
Kind	Type		
Derived from	uint8		
Description	Protocol type definition		
Range	DCM_OBD_ON_CAN	0x00	OBD on CAN (ISO15765-4; ISO15031-5)
	DCM_OBD_ON_FLEXRAY	0x01	(OBD on Flexray (Manufacturer specific; ISO15031-5))
	DCM_OBD_ON_IP	0x02	(OBD on Internet Protocol (Manufacturer specific; ISO15031-5))
	DCM_UDS_ON_CAN	0x03	UDS on CAN (ISO15765-3; ISO14229-1)
	DCM_UDS_ON_FLEXRAY	0x04	UDS on FlexRay (Manufacturer specific; ISO14229-1)
	DCM_UDS_ON_IP	0x05	(UDS on Internet Protocol (Manufacturer specific; ISO14229-1))
	DCM_ROE_ON_CAN	0x06	Response On Event on CAN
	DCM_ROE_ON_FLEXRAY	0x07	Response On Event on FlexRay
	DCM_ROE_ON_IP	0x08	(Response on Event on Internet Protocol)
	DCM_PERIODICTRANS_ON_CAN	0x09	Periodic Transmission on CAN
	DCM_PERIODICTRANS_ON_FLEXRAY	0x0A	Periodic Transmission on FlexRay
	DCM_PERIODICTRANS_ON_IP	0x0B	(Periodic Transmission on Internet Protocol)
	DCM_NO_ACTIVE_PROTOCOL	0x0C	No protocol has been started
	DCM_UDS_ON_LIN	0x0D	UDS on LIN (ISO14229-1; ISO14229-7)
	Reserved for further AUTOSAR implementation	0x0E..0xEF	--

	DCM_SUPPLIER_1	0xF0	Reserved for SW supplier specific.
	DCM_SUPPLIER_2	0xF1	Reserved for SW supplier specific.
	DCM_SUPPLIER_3	0xF2	Reserved for SW supplier specific.
	DCM_SUPPLIER_4	0xF3	Reserved for SW supplier specific.
	DCM_SUPPLIER_5	0xF4	Reserved for SW supplier specific.
	DCM_SUPPLIER_6	0xF5	Reserved for SW supplier specific.
	DCM_SUPPLIER_7	0xF6	Reserved for SW supplier specific.
	DCM_SUPPLIER_8	0xF7	Reserved for SW supplier specific.
	DCM_SUPPLIER_9	0xF8	Reserved for SW supplier specific.
	DCM_SUPPLIER_10	0xF9	Reserved for SW supplier specific.
	DCM_SUPPLIER_11	0xFA	Reserved for SW supplier specific.
	DCM_SUPPLIER_12	0xFB	Reserved for SW supplier specific.
	DCM_SUPPLIER_13	0xFC	Reserved for SW supplier specific.
	DCM_SUPPLIER_14	0xFD	Reserved for SW supplier specific.
	DCM_SUPPLIER_15	0xFE	Reserved for SW supplier specific.
Variation	--		

Table 8.92: Implementation Data Type Dcm_ProtocolType

⌋*()*

8.8.1.6 Dcm_NegativeResponseCodeType

[SWS_Dcm_00980] ⌈

Name	Dcm_NegativeResponseCodeType		
Kind	Type		
Derived from	uint8		
Description	This Table of available Negative Response Codes represents the allowed Response Codes an AUTOSAR SW Component shall return after a function call. For the allowed NRC of the executed Service ID please refer to the specification of the service in ISO14229-1 (UDS) and ISO15031-5 (OBD/CARB) (see chapter 4.2.4 Response code parameter definition Table 12).		
Range	DCM_POS_RESP	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_FAILUREPREVENTSEXECUTIONOFREQUESTEDACTION	0x26	FPEORA
	range of values 0x27..0x30 reserved by ISO 14229	0x27..0x30	ISOSAERESRVD
	DCM_E_REQUESTOUTOFRANGE	0x31	ROOR
	value 0x32 reserved by ISO 14229	0x32	ISOSAERESRVD
	DCM_E_SECURITYACCESSDENIED	0x33	SAD
	value 0x34 reserved by ISO 14229	0x34	ISOSAERESRVD
	DCM_E_INVALIDKEY	0x35	IK
	DCM_E_EXCEEDNUMBEROFATTEMPTS	0x36	ENOA
	DCM_E_REQUIREDTIMEDELAYNOTEXPIRED	0x37	RTDNE
	range of values 0x38..0x4F reserved by ISO 15764	0x38..0x4F	RBEDLSD
	range of values 0x50..0x6F reserved by ISO 14229	0x50..0x6F	ISOSAERESRVD
	DCM_E_UPLOADDOWNLOADNOTACCEPTED	0x70	UDNA
	DCM_E_TRANSFERDATASUSPENDED	0x71	TDS
	DCM_E_GENERALPROGRAMMINGFAILURE	0x72	GPF
	DCM_E_WRONGBLOCKSEQUENCECOUNTER	0x73	WBSC
	range of values 0x74..0x77 reserved by ISO 14229	0x74..0x77	ISOSAERESRVD

	range of values 0x79..0x7D reserved by ISO 14229	0x79..0x7D	ISOSAERESRVD
	DCM_E_SUBFUNCTIONNOTSUPPORTEDINACTIVESESSION	0x7E	SFNSIAS
	DCM_E_SERVICENOTSUPPORTEDINACTIVESESSION	0x7F	SNSIAS
	value 0x80 reserved by ISO 14229	0x80	ISOSAERESRVD
	DCM_E_RPMTOOHIGH	0x81	RPMTH
	DCM_E_RPMTOOLOW	0x82	RPMTL
	DCM_E_ENGINEISRUNNING	0x83	EIR
	DCM_E_ENGINEISNOTRUNNING	0x84	EINR
	DCM_E_ENGINERUNTIMETOOLOW	0x85	ERTTL
	DCM_E_TEMPERATURETOOHIGH	0x86	TEMPTH
	DCM_E_TEMPERATURETOOLOW	0x87	TEMPTL
	DCM_E_VEHICLESPEEDTOOHIGH	0x88	VSTH
	DCM_E_VEHICLESPEEDTOOLOW	0x89	VSTL
	DCM_E_THROTTLE_PEDALTOOHIGH	0x8A	TPTH
	DCM_E_THROTTLE_PEDALTOOLOW	0x8B	TPTL
	DCM_E_TRANSMISSIONRANGENOTINNEUTRAL	0x8C	TRNIN
	DCM_E_TRANSMISSIONRANGENOTINGEAR	0x8D	TRNIG
	value 0x8E reserved by ISO 14229	0x8E	ISOSAERESRVD
	DCM_E_BRAKESWITCH_NOTCLOSED	0x8F	BSNC
	DCM_E_SHIFTERLEVERNOTINPARK	0x90	SLNIP
	DCM_E_TORQUECONVERTERCLUTCHLOCKED	0x91	TCCL
	DCM_E_VOLTAGETOOHIGH	0x92	VTH
	DCM_E_VOLTAGETOOLOW	0x93	VTL
	range of values 0x94..0xEF reserved by ISO 14229	0x94..0xEF	RFSCNC
	DCM_E_VMSCNC_0	0xF0	VMSCNC
	DCM_E_VMSCNC_1	0xF1	VMSCNC1
	DCM_E_VMSCNC_2	0xF2	VMSCNC2
	DCM_E_VMSCNC_3	0xF3	VMSCNC3
	DCM_E_VMSCNC_4	0xF4	VMSCNC4
	DCM_E_VMSCNC_5	0xF5	VMSCNC5
	DCM_E_VMSCNC_6	0xF6	VMSCNC6
	DCM_E_VMSCNC_7	0xF7	VMSCNC7
	DCM_E_VMSCNC_8	0xF8	VMSCNC8
	DCM_E_VMSCNC_9	0xF9	VMSCNC9
	DCM_E_VMSCNC_A	0xFA	VMSCNCA
	DCM_E_VMSCNC_B	0xFB	VMSCNCB

	DCM_E_VMSCNC_C	0xFC	VMSCNCC
	DCM_E_VMSCNC_D	0xFD	VMSCNCD
	DCM_E_VMSCNC_E	0xFE	VMSCNCE
	value 0xFF reserved by ISO 14229	0xFF	ISOSAERESRVD
Variation	--		

Table 8.93: Implementation Data Type Dcm_NegativeResponseCodeType

⌋*()*

8.8.1.7 Dcm_DataArrayTypeUint8_{Data}Type

[SWS_Dcm_01121] ⌈

Name	Dcm_DataArrayTypeUint8_{Data}Type		
Kind	Array	***Element type***	uint8
Size	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataByteSize)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.DcmDspPidDataByteSize)})) Elements		
Description	--		
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT8_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT8_DYN) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT8_N) || Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})		

Table 8.94: Implementation Data Type Dcm_DataArrayTypeUint8_{Data}Type

⌋*()*

8.8.1.8 Dcm_DataArrayTypeUint16_{Data}Type

[SWS_Dcm_01122] ⌈

Name	Dcm_DataArrayTypeUint16_{Data}Type		
Kind	Array	***Element type***	uint16
Size	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataByteSize)}/2) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.DcmDspPidDataByteSize)}/2)) Elements		
Description	--		

Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT16_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT16_N)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})

Table 8.95: Implementation Data Type Dcm_DataArrayTypeUint16_{Data}Type

⌋*()*

8.8.1.9 Dcm_DataArrayTypeUint32_{Data}Type

[SWS_Dcm_01123] ⌈

Name	Dcm_DataArrayTypeUint32_{Data}Type		
Kind	Array	***Element type***	uint32
Size	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataByteSize)}/4) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.DcmDspPidDataByteSize)}/4)) Elements		
Description	--		
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT32_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT32_N)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})		

Table 8.96: Implementation Data Type Dcm_DataArrayTypeUint32_{Data}Type

⌋*()*

8.8.1.10 Dcm_DataArrayTypeSint8_{Data}Type

[SWS_Dcm_01124] ⌈

Name	Dcm_DataArrayTypeSint8_{Data}Type		
Kind	Array	***Element type***	sint8
Size	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataByteSize)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.DcmDspPidDataByteSize)})) Elements		
Description	--		

Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == SINT8_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == SINT8_N)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})

Table 8.97: Implementation Data Type Dcm_DataArrayTypeSint8_{Data}Type

⌋*()*

8.8.1.11 Dcm_DataArrayTypeSint16_{Data}Type

[SWS_Dcm_01125] ⌈

Name	Dcm_DataArrayTypeSint16_{Data}Type		
Kind	Array	***Element type***	sint16
Size	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataByteSize)}/2) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.DcmDspPidDataByteSize)}/2)) Elements		
Description	--		
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == SINT16_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == SINT16_N)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})		

Table 8.98: Implementation Data Type Dcm_DataArrayTypeSint16_{Data}Type

⌋*()*

8.8.1.12 Dcm_DataArrayTypeSint32_{Data}Type

[SWS_Dcm_01126] ⌈

Name	Dcm_DataArrayTypeSint32_{Data}Type		
Kind	Array	***Element type***	sint32
Size	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataByteSize)}/4) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.DcmDspPidDataByteSize)}/4)) Elements		
Description	--		

Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == SINT32_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == SINT32_N)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})

Table 8.99: Implementation Data Type Dcm_DataArrayTypeSint32_{Data}Type

⌋()

8.8.1.13 Dcm_RangeArray_{Range}Type

[SWS_Dcm_01012] ⌈

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

Table 8.100: Implementation Data Type Dcm_RangeArray_{Range}Type

⌋()

8.8.1.14 Dcm_InfoTypeServicesArray_{VehInfoData}Type

[SWS_Dcm_01013] ⌈

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

Table 8.101: Implementation Data Type Dcm_InfoTypeServicesArray_{VehInfoData}Type

⌋()

8.8.1.15 Dcm_RequestControlServicesInArray_{Tid}Type

[SWS_Dcm_01014] ⌈

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

Table 8.102: Implementation Data Type Dcm_RequestControlServicesInArray_{Tid}Type

⌋*()*

8.8.1.16 Dcm_RequestControlServicesOutArray_{Tid}Type

[SWS_Dcm_01015] ⌈

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

Table 8.103: Implementation Data Type Dcm_RequestControlServicesOutArray_{Tid}Type

⌋*()*

8.8.1.17 Dcm_ScalingInfoArray_{Data}Type

[SWS_Dcm_01017] ⌈

Name	Dcm_ScalingInfoArray_{Data}Type		
Kind	Array	***Element type***	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDataInfo.DcmDspDataScalingInfoSize)} Elements		
Description	--		

Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData->DcmDspDataInfoRef.DcmDspDataScalingInfoSize)} != NULL) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}

Table 8.104: Implementation Data Type Dcm_ScalingInfoArray_{Data}Type

⌋*()*

8.8.1.18 Dcm_RequestDataOut_{Routine}_{Signal}PrimitivType

[SWS_Dcm_01018] ⌈

Name	Dcm_RequestDataOut_{Routine}_{Signal}PrimitivType	
Kind	Type	
Derived from	*Base Type*	*Variation*
	sint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT16
	sint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT32
	sint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT8
	uint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT16

	uint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT32
	uint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT8
Description	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

Table 8.105: Implementation Data Type Dcm_RequestDataOut_{Routine}_{Signal}PrimitivType

⌋*()*

8.8.1.19 Dcm_RequestDataOut_{Routine}_{Signal}Type

[SWS_Dcm_91040] ⌈

Name	Dcm_RequestDataOut_{Routine}_{Signal}Type	
Kind	Type	
Derived from	*Base Type*	*Variation*
	Dcm_RequestDataOut_{Routine}_{Signal}PrimitivType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32])
	Dcm_RequestDataOut_{Routine}_{Signal}ArrayType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]_N)
Description	--	

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}

Table 8.106: Implementation Data Type Dcm_RequestDataOut_{Routine}_{Signal}Type

⌋*()*

8.8.1.20 Dcm_RequestDataOut_{Routine}_{Signal}ArrayType

[SWS_Dcm_91041] ⌈

Name	Dcm_RequestDataOut_{Routine}_{Signal}ArrayType	
Kind	Array	
Derived from	*Base Type*	*Variation*
	sint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT16_N)
	sint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT32_N)
	sint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT8_N)
	uint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT16_N)
	uint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT32_N)

	uint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/ DcmDspRequestRoutineResults/ DcmDspRequestRoutineResultsOut/ DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT8_N)
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineParameterSize)} Elements	
Description	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspRequestRoutineResults/ DcmDspRequestRoutineResultsOut/ DcmDspRequestRoutineResultsOutSignal. DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]_N) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspRequestRoutineResults/ DcmDspRequestRoutineResultsOut/ DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine. SHORT-NAME)}	

Table 8.107: Implementation Data Type Dcm_RequestDataOut_{Routine}_{Signal}ArrayType

⌋*()*

8.8.1.21 Dcm_RequestFlexibleOutArrayData_{Routine}_{Signal}Type

[SWS_Dcm_01019] ⌈

Name	Dcm_RequestFlexibleOutArrayData_{Routine}_{Signal}Type		
Kind	Array	***Element type***	uint8
Size	{(ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineParameterSize} Elements		
Description	--		

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspRequestRoutineResults/ DcmDspRequestRoutineResultsOut/ DcmDspRequestRoutineResultsOutSignal. DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspRequestRoutineResults/ DcmDspRequestRoutineResultsOut/ DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine. SHORT-NAME)}

Table 8.108: Implementation Data Type Dcm_RequestFlexibleOutArrayData_{Routine}_{Signal}Type

⌋*()*

8.8.1.22 Dcm_StartDataIn_{Routine}_{Signal}PrimitivType

[SWS_Dcm_01020] ⌈

Name	Dcm_StartDataIn_{Routine}_{Signal}PrimitivType	
Kind	Type	
Derived from	*Base Type*	*Variation*
	sint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStartRoutine/ DcmDspStartRoutineIn/ DcmDspStartRoutineInSignal. DcmDspRoutineSignalType)} == SINT16
	sint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStartRoutine/ DcmDspStartRoutineIn/ DcmDspStartRoutineInSignal. DcmDspRoutineSignalType)} == SINT32
	sint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStartRoutine/ DcmDspStartRoutineIn/ DcmDspStartRoutineInSignal. DcmDspRoutineSignalType)} == SINT8
	uint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStartRoutine/ DcmDspStartRoutineIn/ DcmDspStartRoutineInSignal. DcmDspRoutineSignalType)} == UINT16
	uint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStartRoutine/ DcmDspStartRoutineIn/ DcmDspStartRoutineInSignal. DcmDspRoutineSignalType)} == UINT32

	uint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == UINT8
Description	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

Table 8.109: Implementation Data Type Dcm_StartDataIn_{Routine}_{Signal}PrimitivType

⌋*()*

8.8.1.23 Dcm_StartDataIn_{Routine}_{Signal}Type

[SWS_Dcm_91042] ⌈

Name	Dcm_StartDataIn_{Routine}_{Signal}Type	
Kind	Type	
Derived from	*Base Type*	*Variation*
	Dcm_StartDataIn_{Routine}_{Signal}PrimitivType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32])
	Dcm_StartDataIn_{Routine}_{Signal}ArrayType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]_N)
Description	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

Table 8.110: Implementation Data Type Dcm_StartDataIn_{Routine}_{Signal}Type

⌋*()*

8.8.1.24 Dcm_StartDataIn_{Routine}_{Signal}ArrayType

[SWS_Dcm_91043] ⌈

Name	Dcm_StartDataIn_{Routine}_{Signal}ArrayType	
Kind	Array	
Derived from	*Base Type*	*Variation*
	sint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == SINT16_N)
	sint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == SINT32_N)
	sint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == SINT8_N)
	uint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == UINT16_N)
	uint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == UINT32_N)
	uint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == UINT8_N)
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineParameterSize)} Elements	
Description	--	

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]_N) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}

Table 8.111: Implementation Data Type Dcm_StartDataIn_{Routine}_{Signal}ArrayType

⌋*()*

8.8.1.25 Dcm_StartDataOut_{Routine}_{Signal}PrimitivType

[SWS_Dcm_01021] ⌈

Name	Dcm_StartDataOut_{Routine}_{Signal}PrimitivType	
Kind	Type	
Derived from	*Base Type*	*Variation*
	sint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT16
	sint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT32
	sint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT8
	uint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT16
	uint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT32
	uint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT8

Description	--
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}

Table 8.112: Implementation Data Type Dcm_StartDataOut_{Routine}_{Signal}PrimitivType

⌋*()*

8.8.1.26 Dcm_StartDataOut_{Routine}_{Signal}Type

[SWS_Dcm_91044] ⌈

Name	Dcm_StartDataOut_{Routine}_{Signal}Type	
Kind	Type	
Derived from	*Base Type*	*Variation*
	Dcm_StartDataOut_{Routine}_{Signal}PrimitivType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32])
	Dcm_StartDataOut_{Routine}_{Signal}ArrayType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]_N)
Description	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

Table 8.113: Implementation Data Type Dcm_StartDataOut_{Routine}_{Signal}Type

⌋*()*

8.8.1.27 Dcm_StartDataOut_{Routine}_{Signal}ArrayType

[SWS_Dcm_91045] ⌈

Name	Dcm_StartDataOut_{Routine}_{Signal}ArrayType	
Kind	Array	
Derived from	*Base Type*	*Variation*
	sint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT16_N)
	sint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT32_N)
	sint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT8_N)
	uint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT16_N)
	uint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT32_N)
	uint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT8_N)
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineParameterSize)} Elements	
Description	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]_N) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

Table 8.114: Implementation Data Type Dcm_StartDataOut_{Routine}_{Signal}ArrayType

⌋()

8.8.1.28 Dcm_StartFlexibleInArrayData_{Routine}_{Signal}Type

[SWS_Dcm_01022] ⌈

Name	Dcm_StartFlexibleInArrayData_{Routine}_{Signal}Type		
Kind	Array	***Element type***	uint8
Size	{(ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineParameterSize} Elements		
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		

Table 8.115: Implementation Data Type Dcm_StartFlexibleInArrayData_{Routine}_{Signal}Type

⌋()

8.8.1.29 Dcm_StartFlexibleOutArrayData_{Routine}_{Signal}Type

[SWS_Dcm_01023] ⌈

Name	Dcm_StartFlexibleOutArrayData_{Routine}_{Signal}Type		
Kind	Array	***Element type***	uint8
Size	{(ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineParameterSize} Elements		
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		

Table 8.116: Implementation Data Type Dcm_StartFlexibleOutArrayData_{Routine}_{Signal}Type

⌋()

8.8.1.30 Dcm_StopDataIn_{Routine}_{Signal}PrimitivType

[SWS_Dcm_01024] ⌈

Name	Dcm_StopDataIn_{Routine}_{Signal}PrimitivType	
Kind	Type	
Derived from	*Base Type*	*Variation*
	sint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT16
	sint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT32
	sint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT8
	uint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == UINT16
	uint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == UINT32
	uint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == UINT8
Description	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

Table 8.117: Implementation Data Type Dcm_StopDataIn_{Routine}_{Signal}PrimitivType

⌋()

8.8.1.31 Dcm_StopDataIn_{Routine}_{Signal}Type

[SWS_Dcm_91046] ⌈

Name	Dcm_StopDataIn_{Routine}_{Signal}Type	
Kind	Type	
Derived from	*Base Type*	*Variation*
	Dcm_StopDataIn_{Routine}_{Signal}PrimitivType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32])
	Dcm_StopDataIn_{Routine}_{Signal}ArrayType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]_N)
Description	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

Table 8.118: Implementation Data Type Dcm_StopDataIn_{Routine}_{Signal}Type

⌋*()*

8.8.1.32 Dcm_StopDataIn_{Routine}_{Signal}ArrayType

[SWS_Dcm_91047] ⌈

Name	Dcm_StopDataIn_{Routine}_{Signal}ArrayType	
Kind	Array	
Derived from	*Base Type*	*Variation*
	sint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT16_N)
	sint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT32_N)

	sint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT8_N)
	uint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == UINT16_N)
	uint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == UINT32_N)
	uint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == UINT8_N)
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineParameterSize)} Elements	
Description	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]_N) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

Table 8.119: Implementation Data Type Dcm_StopDataIn_{Routine}_{Signal}ArrayType

⌋*()*

8.8.1.33 Dcm_StopDataOut_{Routine}_{Signal}PrimitivType

[SWS_Dcm_01025] ⌈

Name	Dcm_StopDataOut_{Routine}_{Signal}PrimitivType	
Kind	Type	
Derived from	*Base Type*	*Variation*

	sint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. DcmDspRoutineSignalType)} == SINT16
	sint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. DcmDspRoutineSignalType)} == SINT32
	sint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. DcmDspRoutineSignalType)} == SINT8
	uint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. DcmDspRoutineSignalType)} == UINT16
	uint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. DcmDspRoutineSignalType)} == UINT32
	uint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. DcmDspRoutineSignalType)} == UINT8
Description	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

Table 8.120: Implementation Data Type Dcm_StopDataOut_{Routine}_{Signal}PrimitivType

⌋*()*

8.8.1.34 Dcm_StopDataOut_{Routine}_{Signal}Type

[SWS_Dcm_91048] ⌈

Name	Dcm_StopDataOut_{Routine}_{Signal}Type	
Kind	Type	
Derived from	*Base Type*	*Variation*

	Dcm_StopDataOut_{Routine}_{Signal}PrimitivType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32])
	Dcm_StopDataOut_{Routine}_{Signal}ArrayType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]_N)
Description	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	

Table 8.121: Implementation Data Type Dcm_StopDataOut_{Routine}_{Signal}Type

⌋*()*

8.8.1.35 Dcm_StopDataOut_{Routine}_{Signal}ArrayType

[SWS_Dcm_91049] ⌈

Name	Dcm_StopDataOut_{Routine}_{Signal}ArrayType	
Kind	Array	
Derived from	*Base Type*	*Variation*
	sint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == SINT16_N)
	sint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == SINT32_N)
	sint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == SINT8_N)

	uint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. DcmDspRoutineSignalType)} == UINT16_N)
	uint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. DcmDspRoutineSignalType)} == UINT32_N)
	uint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. DcmDspRoutineSignalType)} == UINT8_N)
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal. DcmDspRoutineParameterSize)} Elements	
Description	--	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32]_N) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine. SHORT-NAME)}	

Table 8.122: Implementation Data Type Dcm_StopDataOut_{Routine}_{Signal}ArrayType

⌋*()*

8.8.1.36 Dcm_StopFlexibleInArrayData_{Routine}_{Signal}Type

[SWS_Dcm_01026] ⌈

Name	Dcm_StopFlexibleInArrayData_{Routine}_{Signal}Type		
Kind	Array	***Element type***	uint8
Size	{(ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineParameterSize} Elements		
Description	--		

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}

Table 8.123: Implementation Data Type Dcm_StopFlexibleInArrayData_{Routine}_{Signal}Type

⌋*()*

8.8.1.37 Dcm_StopFlexibleOutArrayData_{Routine}_{Signal}Type

[SWS_Dcm_01027] ⌈

Name	Dcm_StopFlexibleOutArrayData_{Routine}_{Signal}Type		
Kind	Array	***Element type***	uint8
Size	{(ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineParameterSize} Elements		
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		

Table 8.124: Implementation Data Type Dcm_StopFlexibleOutArrayData_{Routine}_{Signal}Type

⌋*()*

8.8.1.38 Dcm_KeyArray_{SecurityLevel}Type

[SWS_Dcm_01028] ⌈

Name	Dcm_KeyArray_{SecurityLevel}Type		
Kind	Array	***Element type***	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityKeySize)} Elements		
Description	--		

Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)} == USE_ASYNCH_CLIENT_SERVER SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}

Table 8.125: Implementation Data Type Dcm_KeyArray_{SecurityLevel}Type

⌋*()*

8.8.1.39 Dcm_SeedArray_{SecurityLevel}Type

[SWS_Dcm_01029] ⌈

Name	Dcm_SeedArray_{SecurityLevel}Type		
Kind	Array	***Element type***	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecuritySeedSize)} Elements		
Description	--		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)} == USE_ASYNCH_CLIENT_SERVER SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}		

Table 8.126: Implementation Data Type Dcm_SeedArray_{SecurityLevel}Type

⌋*()*

8.8.1.40 Dcm_SecurityAccessDataRecordArray_{SecurityLevel}Type

[SWS_Dcm_01159] ⌈

Name	Dcm_SecurityAccessDataRecordArray_{SecurityLevel}Type		
Kind	Array	***Element type***	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp/DcmDspSecurity/DcmDspSecurityRow/DcmDspSecurityADRSize)} Elements		
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)} == USE_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityADRSize)} != NULL) SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}		

Table 8.127: Implementation Data Type Dcm_SecurityAccessDataRecordArray_{SecurityLevel}Type

⌋*()*

8.8.1.41 Dcm_RequestDataArrayType

[SWS_Dcm_01165] ⌈

Name	Dcm_RequestDataArrayType		
Kind	Array	***Element type***	uint8
Size	(MAX({ecuc(Dcm/DcmDsl/DcmDslProtocol/DcmDslProtocolRow/ DcmDslProtocolRxBufferID->DcmDslBuffer.DcmDslBufferSize)}) -1) Elements		
Description	--		
Variation	--		

Table 8.128: Implementation Data Type Dcm_RequestDataArrayType

⌋*()*

8.8.1.42 Dcm_ControlMask_{Data}Type

[SWS_Dcm_01320] ⌈

Name	Dcm_ControlMask_{Data}Type	
Kind	Type	
Derived from	*Base Type*	*Variation*
	uint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspDid/ DcmDspDidInfoRef->DcmDspDidInfo/ DcmDspDidControl/ DcmDspDidControlMaskSize)} == 0x02)
	uint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspDid/ DcmDspDidInfoRef->DcmDspDidInfo/ DcmDspDidControl/ DcmDspDidControlMaskSize)} >= 0x03)
	uint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspDid/ DcmDspDidInfoRef->DcmDspDidInfo/ DcmDspDidControl/ DcmDspDidControlMaskSize)} == 0x01)
Description	--	

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_SYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlEnableMask)} == NULL) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})

Table 8.129: Implementation Data Type Dcm_ControlMask_{Data}Type

⌋*()*

8.8.1.43 ControlMask types

[SWS_Dcm_01317] ⌈

Name	Dcm_ControlMask_8Type
Kind	Type
Derived from	uint8
Description	--
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlEnableMask)} == NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} == 0x01)

Table 8.130: Implementation Data Type Dcm_ControlMask_8Type

⌋*()*

[SWS_Dcm_01318] ⌈

Name	Dcm_ControlMask_16Type
Kind	Type
Derived from	uint16
Description	--

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlEnableMask)} == NULL) &&({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} == 0x02)

Table 8.131: Implementation Data Type Dcm_ControlMask_16Type

⌋()

[SWS_Dcm_01319] ⌈

Name	Dcm_ControlMask_32Type
Kind	Type
Derived from	uint32
Description	--
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlEnableMask)} == NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} >= 0x03)

Table 8.132: Implementation Data Type Dcm_ControlMask_32Type

⌋()

8.8.1.44 Dcm_inputOutputControlParameterType

[SWS_Dcm_01305] ⌈

Name	Dcm_InputOutputControlParameterType		
Kind	Type		
Derived from	uint8		
Description	--		
Range	DCM_RETURN_CONTROL_TO_ECU	0x00	returnControlToECU
	DCM_RESET_TO_DEFAULT	0x01	resetToDefault
	DCM_FREEZE_CURRENT_STATE	0x02	freezeCurrentState
	DCM_SHORT_TERM_ADJUSTMENT	0x03	shortTermAdjustment
	DCM_IDLE	0xff	Idle state, no request in processing (initial value)

Variation	--

Table 8.133: Implementation Data Type Dcm_InputOutputControlParameterType

⌋()

8.8.1.45 Dcm_IOOperationRequest_{Data}Type

[SWS_Dcm_01306] ⌈

Name	Dcm_IOOperationRequest_{Data}Type		
Kind	Structure		
Elements	inputOutputControlParameter	Dcm_InputOutputControlParameterType	--
	controlEnableMask	Dcm_ControlMask_8Type	--
	Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} == 0x01)	
	controlEnableMask	Dcm_ControlMask_16Type	--
	Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} == 0x02)	
	controlEnableMask	Dcm_ControlMask_32Type	--
	Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} >= 0x03)	
Description	--		

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl)} != NULL) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})

Table 8.134: Implementation Data Type Dcm_IOOperationRequest_{Data}Type

⌋*()*

8.8.1.46 Dcm_IOOperationResponseType

[SWS_Dcm_01307] ⌈

Name	Dcm_IOOperationResponseType		
Kind	Type		
Derived from	uint8		
Description	--		
Range	DCM_POSITIVE_RESPONSE	0x00	positive response (similar to E_OK)
	DCM_GENERAL_REJECT	0x10	NRC generalReject
	DCM_BUSY_REPEAT_REQUEST	0x21	NRC busyRepeatRequest
	DCM_CONDITIONS_NOT_CORRECT	0x22	NRC conditionsNotCorrect
	DCM_FAILURE_PREVENTS_EXECUTION	0x26	NRC FailurePreventsExecutionOfRequestedAction
	DCM_REQUEST_OUT_OF_RANGE	0x31	NRC requestOutOfRange
	DCM_RESPONSE_PENDING	0x78	ResponsePending (similar to E_PENDING)
	DCM_IDLE	0xFF	Idle - no request present (initial value)
Variation	--		

Table 8.135: Implementation Data Type Dcm_IOOperationResponseType

⌋*()*

8.8.1.47 Dcm_DidSupportedType

[SWS_Dcm_01138] ⌈

Name	Dcm_DidSupportedType
Kind	Type
Derived from	uint8
Description	--

Range	DCM_DID_SUPPORTED	0x00	--
	DCM_DID_NOT_ SUPPORTED	0x01	--
Variation	--		

Table 8.136: Implementation Data Type Dcm_DidSupportedType

⌋*()*

8.8.2 Sender-Receiver-Interfaces

Using the concepts of the SW-C template, the interface is defined as follows if Sender-Receiver interface is used (DcmDspDataUsePort set to USE_DATA_SENDER_RECEIVER ||USE_DATA_SENDER_RECEIVER_AS_SERVICE):

8.8.2.1 DataServices_{Data}

[SWS_Dcm_00687] ⌈

Name	DataServices_{Data}	
Comment	--	
IsService	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/ DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER_AS_SERVICE) || ({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidService01/DcmDspPidDataUsePort)} == USE_DATA_SENDER_RECEIVER_AS_SERVICE))	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/ DcmDspDataUsePort)} == (USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE)) ||({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidService01/DcmDspPidDataUsePort)} == (USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData. SHORT-NAME})	
Data Elements	data	
	Type	boolean
	Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData. DcmDspDataType)} == BOOLEAN) || ({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidService01.DcmDspPidDataType)} == BOOLEAN))
	data	
	Type	sint8
	Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData. DcmDspDataType)} == SINT8) || ({ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidService01.DcmDspPidDataType)} == SINT8))

	data	
	Type	sint16
	Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == SINT16) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == SINT16))
	data	
	Type	sint32
	Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == SINT32) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == SINT32))
	data	
	Type	uint32
	Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT32) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT32))
	data	
	Type	uint8
	Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT8) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT8))
	data	
	Type	uint16
	Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT16) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT16))
	data	
	Type	Dcm_DataArrayTypeUint8_{Data}Type
	Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT8_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT8_N)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})
	data	
	Type	Dcm_DataArrayTypeUint16_{Data}Type
	Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT16_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT16_N)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})
	data	
	Type	Dcm_DataArrayTypeUint32_{Data}Type

	Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT32_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT32_N)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})
	data	
	Type	Dcm_DataArrayTypeSint8_{Data}Type
	Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == SINT8_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == SINT8_N)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})
	data	
	Type	Dcm_DataArrayTypeSint16_{Data}Type
	Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == SINT16_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == SINT16_N)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})
	data	
	Type	Dcm_DataArrayTypeSint32_{Data}Type
	Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT32_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT32_N)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})

Table 8.137: Service Interface DataServices_{Data}

⌋*()*

8.8.2.2 IOControlRequest_{Data}

[SWS_Dcm_01308] ⌈

Name	IOControlRequest_{Data}
Comment	Attention: controlState is only valid in case of IOOperationRequest is set to shortTermAdjustment. The DCM provides a byte stream which could be transformed via transformer into an complex type.
IsService	true

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl)} != NULL) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})	
Data Elements	underControl	
	Type	Dcm_ControlMask_8Type
	Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} == 0x01)
	underControl	
	Type	Dcm_ControlMask_16Type
	Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} == 0x02)
	underControl	
	Type	Dcm_ControlMask_32Type
	Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) &&({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} >= 0x03)
	IOOperationRequest	
	Type	Dcm_IOOperationRequest_{Data}Type
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
	controlState	
	Type	Dcm_DataArrayTypeUint8_{Data}Type
	Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == True) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})

Table 8.138: Service Interface IOControlRequest_{Data}

⌋*()*

8.8.2.3 IOControlResponse

[SWS_Dcm_01309] ⌈

Name	IOControlResponse

Comment	--	
IsService	true	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/ DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspDid/ DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl)} != NULL)	
Data Elements	IOOperationResponse	
	Type	Dcm_IOOperationResponseType
	Variation	--

Table 8.139: Service Interface IOControlResponse

⌋()

8.8.3 Client-Server-Interfaces

8.8.3.1 SecurityAccess_{SecurityLevel}

Provides functions required for the UDS Service SecurityAccess (see [SWS_Dcm_00323],[SWS_Dcm_00862] and [SWS_Dcm_00863]).

Using the concepts of the SW-C template, the interface is defined as follows if ClientServer interface is used (DcmDspSecurityUsePort set to or USE_ASYNCH_CLIENT_SERVER):

[SWS_Dcm_00685] ⌈

Name	SecurityAccess_{SecurityLevel}	
Comment	--	
IsService	true	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/ DcmDspSecurityRow.DcmDspSecurityUsePort)} == USE_ASYNCH_CLIENT_SERVER SecurityLevel = {ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow. SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	10	DCM_E_PENDING
	11	DCM_E_COMPARE_KEY_FAILED

Table 8.140: Service Interface SecurityAccess_{SecurityLevel}

Operations

CompareKey			
Comments	--		
Variation	--		
Parameters	Key	Comment	Key, which needs to be compared
		Type	Dcm_KeyArray_{SecurityLevel}Type

		Variation	SecurityLevel = {ecuc(Dcm/ DcmConfigSet/DcmDsp/ DcmDspSecurity/ DcmDspSecurityRow. SHORT-NAME)}
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	ErrorCode	Comment	return Error Code
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful.	
	E_NOT_OK	Request was not successful.	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	
	DCM_E_COMPARE_ KEY_FAILED	Key did not match.	

Table 8.141: Operation CompareKey

GetSecurityAttemptCounter			
Comments	Restore the attempt counter from the application		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/ DcmDspSecurityRow.DcmDspSecurityAttemptCounterEnabled)} == TRUE		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	AttemptCounter	Comment	--
		Type	uint8
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful.	
	E_NOT_OK	Request was not successful.	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.142: Operation GetSecurityAttemptCounter

GetSeed			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/ DcmDspSecurityRow.DcmDspSecurityADRSize)} == NULL)		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	Seed	Comment	Pointer for provided seed
		Type	Dcm_SeedArray_{SecurityLevel} Type

		Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}
		Direction	OUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful.	
	E_NOT_OK	Request was not successful.	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.143: Operation GetSeed

GetSeed			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityADRSize)} != NULL)		
Parameters	SecurityAccessDataRecord	Comment	--
		Type	Dcm_SecurityAccessDataRecordArray_{SecurityLevel}Type
		Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	Seed	Comment	Pointer for provided seed
		Type	Dcm_SeedArray_{SecurityLevel}Type
		Variation	SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}
		Direction	OUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful.	
	E_NOT_OK	Request was not successful.	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.144: Operation GetSeed

SetSecurityAttemptCounter			
Comments	Store the attempt counter in the application		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/ DcmDspSecurityRow.DcmDspSecurityAttemptCounterEnabled)} == TRUE		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	AttemptCounter	Comment	--
		Type	uint8
		Variation	--
		Direction	IN
Possible Errors	E_OK	Request was successful.	
	E_NOT_OK	Request was not successful.	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.145: Operation SetSecurityAttemptCounter

⌋()

8.8.3.2 DataServices_{Data}

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 or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR):

[SWS_Dcm_00686] ⌈

Name	DataServices_{Data}	
Comment	--	
IsService	true	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData. DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData. DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) || ({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER_ERROR) || ({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/ DcmDspPidService01.DcmDspPidDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) Data = ({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData. SHORT-NAME)})	
Possible Errors	0	E_OK
	1	E_NOT_OK
	10	DCM_E_PENDING

Table 8.146: Service Interface DataServices_{Data}

Operations

ConditionCheckRead			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidRead)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataConditionCheckReadFncUsed)} == TRUE)		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.147: Operation ConditionCheckRead

ConditionCheckRead			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidRead)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataConditionCheckReadFncUsed)} == TRUE)		
Parameters	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.148: Operation ConditionCheckRead

FreezeCurrentState	
Comments	--

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData. DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/ DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/ DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/ DcmDspDidControlMask)} != DCM_CONTROLMASK_EXTERNAL)		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.149: Operation FreezeCurrentState

FreezeCurrentState			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData. DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/ DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/ DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/ DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	controlMask	Comment	--
		Type	Dcm_ControlMaskType
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspData. SHORT-NAME)})
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.150: Operation FreezeCurrentState

FreezeCurrentState			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} != DCM_CONTROLMASK_EXTERNAL)		
Parameters	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.151: Operation FreezeCurrentState

FreezeCurrentState			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)		
Parameters	controlMask	Comment	--
		Type	Dcm_ControlMaskType
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.152: Operation FreezeCurrentState

GetScalingInformation			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataInfoRef->DcmDspDataScalingInfoSize)} != NULL)		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--

		Direction	IN
	ScalingInfo	Comment	--
		Type	Dcm_ScalingInfoArray_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	OUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.153: Operation GetScalingInformation

GetScalingInformation			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataInfoRef->DcmDspDataScalingInfoSize)} != NULL)		
Parameters	ScalingInfo	Comment	--
		Type	Dcm_ScalingInfoArray_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	OUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.154: Operation GetScalingInformation

ReadData			
Comments	The server is not allowed to return E_NOT_OK, but shall always provide a valid data value (e.g. a default/replacement value in an error-case) to Dcm/Dem nevertheless the signature of the operation includes E_NOT_OK to ensure compatibility between server runnable and RTE Call API, since the RTE may return negative Std_Return values in certain cases (e.g. partition of server stopped)		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidRead)} != NULL)		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--

		Direction	IN
	Data	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.155: Operation ReadData

ReadData			
Comments	The server is not allowed to return E_NOT_OK, but shall always provide a valid data value (e.g. a default/replacement value in an error-case) to Dcm/Dem nevertheless the signature of the operation includes E_NOT_OK to ensure compatibility between server runnable and RTE Call API, since the RTE may return negative Std_Return values in certain cases (e.g. partition of server stopped)		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER_ERROR) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidRead)} != NULL)		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	Data	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})
		Direction	OUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.156: Operation ReadData

ReadData			
Comments	The server is not allowed to return E_NOT_OK, but shall always provide a valid data value (e.g. a default/replacement value in an error-case) to Dcm/Dem nevertheless the signature of the operation includes E_NOT_OK to ensure compatibility between server runnable and RTE Call API, since the RTE may return negative Std_Return values in certain cases (e.g. partition of server stopped)		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidRead)} != NULL) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER))		
Parameters	Data	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.157: Operation ReadData

ReadDataLength			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidRead)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} == UINT8_DYN)		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.158: Operation ReadDataLength

ReadDataLength			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidRead)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} == UINT8_DYN)		
Parameters	DataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.159: Operation ReadDataLength

ResetToDefault			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidResetToDefault)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} != DCM_CONTROLMASK_EXTERNAL)		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.160: Operation ResetToDefault

ResetToDefault	
Comments	--
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidResetToDefault)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)

Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	controlMask	Comment	--
		Type	Dcm_ControlMaskType
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspData. SHORT-NAME)})
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.161: Operation ResetToDefault

ResetToDefault			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData. DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/ DcmDspDidControl/DcmDspDidResetToDefault)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/ DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/ DcmDspDidControlMask)} != DCM_CONTROLMASK_EXTERNAL)		
Parameters	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.162: Operation ResetToDefault

ResetToDefault			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData. DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/ DcmDspDidControl/DcmDspDidResetToDefault)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/ DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/ DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)		
Parameters	controlMask	Comment	--
		Type	Dcm_ControlMaskType
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspData. SHORT-NAME)})

		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.163: Operation ResetToDefault

ReturnControlToECU			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData. DcmDspDataUsePort)} == (USE_DATA_SYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/ DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/ DcmDspDidControl/DcmDspDidResetToDefault)} == TRUE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/ DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/ DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/ DcmDspDidControlMask)} != DCM_CONTROLMASK_EXTERNAL)		
Parameters	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.164: Operation ReturnControlToECU

ReturnControlToECU			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/ DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/ DcmDspDidControl/DcmDspDidResetToDefault)} == TRUE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/ DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/ DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/ DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)		
Parameters	controlMask	Comment	--
		Type	Dcm_ControlMaskType
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspData. SHORT-NAME)})
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--

		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.165: Operation ReturnControlToECU

ShortTermAdjustment			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} != UINT8_DYN) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} != DCM_CONTROLMASK_EXTERNAL)		
Parameters	ControlStateInfo	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.166: Operation ShortTermAdjustment

ShortTermAdjustment	
Comments	--

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} != UINT8_DYN) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)		
Parameters	ControlStateInfo	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	controlMask	Comment	--
		Type	Dcm_ControlMaskType
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.167: Operation ShortTermAdjustment

ShortTermAdjustment			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} == UINT8_DYN) &&({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} != DCM_CONTROLMASK_EXTERNAL)		
Parameters	ControlStateInfo	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data}Type

		Variation	Data = ({ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspData. SHORT-NAME)})
		Direction	IN
	DataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.168: Operation ShortTermAdjustment

ShortTermAdjustment			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData. DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/ DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData. DcmDspDataType} == UINT8_DYN) && ({ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/ DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)		
Parameters	ControlStateInfo	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data} Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspData. SHORT-NAME)})
		Direction	IN
	DataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	controlMask	Comment	--
		Type	Dcm_ControlMaskType

		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.169: Operation ShortTermAdjustment

ShortTermAdjustment			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} != UINT8_DYN) &&({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} != DCM_CONTROLMASK_EXTERNAL)		
Parameters	ControlStateInfo	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.170: Operation ShortTermAdjustment

ShortTermAdjustment			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} != UINT8_DYN) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)		
Parameters	ControlStateInfo	Comment	--

		Type	Dcm_DataArrayTypeUint8_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	controlMask	Comment	--
		Type	Dcm_ControlMaskType
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.171: Operation ShortTermAdjustment

ShortTermAdjustment			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} == UINT8_DYN) &&({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} != DCM_CONTROLMASK_EXTERNAL)		
Parameters	ControlStateInfo	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	DataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.172: Operation ShortTermAdjustment

ShortTermAdjustment

Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} == UINT8_DYN) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)		
Parameters	ControlStateInfo	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	DataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	controlMask	Comment	--
		Type	Dcm_ControlMaskType
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.173: Operation ShortTermAdjustment

WriteData			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} != UINT8_DYN)		
Parameters	Data	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType

		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.174: Operation WriteData

WriteData			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData. DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/ DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspData.DcmDspDataType} == UINT8_DYN)		
Parameters	Data	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data} Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspData. SHORT-NAME)})
		Direction	IN
	DataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.175: Operation WriteData

WriteData	
Comments	--

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} != UINT8_DYN)		
Parameters	Data	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.176: Operation WriteData

WriteData			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType} == UINT8_DYN)		
Parameters	Data	Comment	--
		Type	Dcm_DataArrayTypeUint8_{Data}Type
		Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})
		Direction	IN
	DataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Request was successful	
	E_NOT_OK	Request was not successful	

Table 8.177: Operation WriteData

⌋*()*

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

8.8.3.2.1 ReadData

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

This interface is used for UDS Service ReadDataByIdentifier and for UDS Service ReadDataByPeriodicIdentifier (0x2A) and for UDS Service InputOutputControlByIdentifier (0x2F).

The ReadData interface can be defined as synchronous or asynchronous according to configuration parameter DcmDspDataUsePort. The synchronous mechanism of the ReadData interface is compatible to the related DEM interface to allow the provider to use the same interface for both Dcm and DEM.

8.8.3.2.2 WriteData

WriteData requests the application to write a data value of a DID. The Data specific type is an array of uint8 which represent either the fix length of this Data or the maximum possible length of this Data. A WriteData interface is defined for every data of each DID with write access. This interface is used for the UDS Service WriteDataByIdentifier (0x2E).

8.8.3.2.3 ReadDataLength

ReadDataLength requests the application to return the data length of a Data. A ReadDataLength interface is defined for every data of each DID with variable data length. This interface is used for UDS Service ReadDataByIdentifier and for UDS Service ReadDataByPeriodicIdentifier (0x2A).

8.8.3.2.4 ConditionCheckRead

ConditionCheckRead requests to the application if the conditions (System state,...) to read the Data are correct. This operation is called for all requested DIDs before requesting the data of each DID. A ConditionCheckRead interface is defined for every data of each DID with read access.

8.8.3.2.5 GetScalingInformation

Request to the application for the scaling information of a Data (see [SWS_Dcm_00394]).

8.8.3.2.6 ReturnControlToEcu

Request to the application to return control to ECU of an IOControl (see [SWS_Dcm_00396]).

8.8.3.2.7 ResetToDefault

Request to the application to reset an IOControl to default value (see [SWS_Dcm_00397]).

8.8.3.2.8 FreezeCurrentState

Request to the application to freeze the current state of an IOControl (see [SWS_Dcm_00398]).

8.8.3.2.9 ShortTermAdjustment

Request to the application to adjust the IO signal (see [SWS_Dcm_00399]).

8.8.3.3 DataServices_DIDRange_{Range}

The following interface defines an operation needed to get the DID range. Using the concepts of the SW-C template, the interface is defined as follows:

[SWS_Dcm_00769] ⌈

Name	DataServices_DIDRange_{Range}	
Comment	--	
IsService	true	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRange. DcmDspDidRangeUsePort)} == TRUE Range = ({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspDidRange.SHORT-NAME)})	
Possible Errors	0	E_OK
	1	E_NOT_OK
	10	DCM_E_PENDING

Table 8.178: Service Interface DataServices_DIDRange_{Range}

Operations

IsDidAvailable			
Comments	--		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRange. DcmDspDidRangeHasGaps)} == TRUE		
Parameters	DID	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	supported	Comment	--
		Type	Dcm_DidSupportedType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.179: Operation IsDidAvailable

ReadDidData			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRange/ DcmDspDidRangeInfoRef->DcmDspDidRead)} != NULL)		
Parameters	DID	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	Data	Comment	--
		Type	Dcm_RangeArray_{Range}Type
		Variation	Range = ({ecuc(Dcm/ DcmConfigSet/DcmDsp/ DcmDspDidRange. SHORT-NAME)})
		Direction	OUT
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	OUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.180: Operation ReadDidData

ReadDidRangeDataLength			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRange/ DcmDspDidRangeInfoRef->DcmDspDidRead)} != NULL)		
Parameters	DID	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.181: Operation ReadDidRangeDataLength

WriteDidData			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRange/ DcmDspDidRangeInfoRef->DcmDspDidWrite)} != NULL)		
Parameters	DID	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	Data	Comment	--
		Type	Dcm_RangeArray_{Range}Type
		Variation	Range = ({ecuc(Dcm/ DcmConfigSet/DcmDsp/ DcmDspDidRange. SHORT-NAME)})
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT

Possible Errors	E_OK	Operation successful
	E_NOT_OK	Request was not successful
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.

Table 8.182: Operation WriteDidData

⌋()

8.8.3.4 InfotypeServices_{VehInfoData}

The following interface defines an operation needed to get data from one or several SW-C in order to supply OBD Service $09 (see [SWS_Dcm_00423]).

Using the concepts of the SW-C template, the interface is defined as follows: **[SWS_Dcm_00688]** ⌈

Name	InfotypeServices_{VehInfoData}	
Comment	--	
IsService	true	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspVehInfo/DcmDspVehInfoData/DcmDspVehInfoDataUsePort)}==TRUE VehInfoData = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspVehInfo/DcmDspVehInfoData.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	10	DCM_E_PENDING

Table 8.183: Service Interface InfotypeServices_{VehInfoData}

Operations

GetInfotypeValueData			
Comments	--		
Variation	--		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataValueBuffer	Comment	--
		Type	Dcm_InfoTypeServicesArray_{VehInfoData}Type
		Variation	VehInfoData = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspVehInfo/DcmDspVehInfoData.SHORT-NAME)}
		Direction	OUT

	DataValueBufferSize	Comment	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in DataValueBuffer
		Type	uint8
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

Table 8.184: Operation GetInfotypeValueData

⌋*()*

8.8.3.5 RoutineServices_{RoutineName}

The following interface defines operations needed for the UDS Service RoutineControl (0x31) (see [SWS_Dcm_00400], [SWS_Dcm_00401], [SWS_Dcm_00402], [SWS_Dcm_00403], [SWS_Dcm_00404], [SWS_Dcm_00405]).

Using the concepts of the SW-C template, the interface is defined as follows: **[SWS_Dcm_00690]** ⌈

Name	RoutineServices_{RoutineName}	
Comment	--	
IsService	true	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineUsePort)} == TRUE RoutineName = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	10	DCM_E_PENDING
	12	DCM_E_FORCE_RCRRP

Table 8.185: Service Interface RoutineServices_{RoutineName}

Operations

RequestResults			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH)		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--

		Direction	IN
	DataOut_{Signal}	Comment	--
		Type	Dcm_RequestDataOut_{Routine}_{Signal}Type

		Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)}
			Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}

		Direction	OUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)	

Table 8.186: Operation RequestResults

RequestResults			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH)		
Parameters	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataOut_{Signal}	Comment	--
		Type	Dcm_RequestDataOut_{Routine}_{Signal}Type

		Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
		Direction	OUT
	DataOut_{Signal}	Comment	--
		Type	Dcm_RequestFlexibleOutArrayData_{Routine}_{Signal}Type
		Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
		Direction	OUT
	currentDataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	OUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	

	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)

Table 8.187: Operation RequestResults

RequestResultsConfirmation			
Comments	This operation indicates the transmission of a response to a RequestResultsRoutine request		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsConfirmationEnabled)}==TRUE)		
Parameters	ConfirmationStatus	Comment	Confirmation status of a RequestResultsRoutinerequest
		Type	Dcm_ConfirmationStatusType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	

Table 8.188: Operation RequestResultsConfirmation

Start			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH)		
Parameters	DataIn_{Signal}	Comment	--
		Type	Dcm_StartDataIn_{Routine}_{Signal}Type
		Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataOut_{Signal}	Comment	--
		Type	Dcm_StartDataOut_{Routine}_{Signal}Type

		Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
		Direction	OUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)	

Table 8.189: Operation Start

Start			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH)		
Parameters	DataIn_{Signal}	Comment	--
		Type	Dcm_StartDataIn_{Routine}_{Signal}Type
		Variation	Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)} Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)}
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataOut_{Signal}	Comment	--
		Type	Dcm_StartDataOut_{Routine}_{Signal}Type

		Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)
		Direction	OUT
	DataOut_{Signal}	Comment	--
		Type	Dcm_StartFlexibleOutArrayData_{Routine}_{Signal}Type
		Variation	--
		Direction	OUT
	currentDataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	OUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)	

Table 8.190: Operation Start

Start			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH)		
Parameters	DataIn_{Signal}	Comment	--
		Type	Dcm_StartDataIn_{Routine}_{Signal}Type
		Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
		Direction	IN
	DataIn_{Signal}	Comment	--

		Type	Dcm_StartFlexibleInArrayData_{Routine}_{Signal}Type
		Variation	--
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataOut_{Signal}	Comment	--
		Type	Dcm_StartDataOut_{Routine}_{Signal}Type
		Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)
		Direction	OUT
	currentDataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)	

Table 8.191: Operation Start

Start			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH)		
Parameters	DataIn_{Signal}	Comment	--
		Type	Dcm_StartDataIn_{Routine}_{Signal}Type

		Variation	{ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine/ DcmDspStartRoutine/ DcmDspStartRoutineIn/ DcmDspStartRoutineInSignal. DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine/ DcmDspStartRoutine/ DcmDspStartRoutineIn/ DcmDspStartRoutineInSignal. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine. SHORT-NAME)}
		Direction	IN
	DataIn_{Signal}	Comment	--
		Type	Dcm_StartFlexibleInArrayData_{Routine}_{Signal}Type
		Variation	{ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine/ DcmDspStartRoutine/ DcmDspStartRoutineIn/ DcmDspStartRoutineInSignal. DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine/ DcmDspStartRoutine/ DcmDspStartRoutineIn/ DcmDspStartRoutineInSignal. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine. SHORT-NAME)}
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataOut_{Signal}	Comment	--
		Type	Dcm_StartDataOut_{Routine}_{Signal}Type

		Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
		Direction	OUT
	DataOut_{Signal}	Comment	--
		Type	Dcm_StartFlexibleOutArrayData_{Routine}_{Signal}Type
		Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
		Direction	OUT
	currentDataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	INOUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)	

Table 8.192: Operation Start

StartConfirmation

Comments	This operation indicates the transmission of a response to a StartRoutine request		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspStartRoutine/ DcmDspStartRoutineConfirmationEnabled)}==TRUE		
Parameters	ConfirmationStatus	Comment	Confirmation status of a StartRoutine request
		Type	Dcm_ConfirmationStatusType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	

Table 8.193: Operation StartConfirmation

Stop			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/DcmDspStopRoutineIn/ DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH)		
Parameters	DataIn_{Signal}	Comment	--
		Type	Dcm_StopDataIn_{Routine}_{ Signal}Type
		Variation	Signal = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/ DcmDspStopRoutineIn/ DcmDspStopRoutineInSignal. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine. SHORT-NAME)}
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataOut_{Signal}	Comment	--
		Type	Dcm_StopDataOut_{Routine}_{ Signal}Type
		Variation	Signal = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine. SHORT-NAME)}
		Direction	OUT

	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	
	DCM_E_FORCE_ RCRRP	application request the transmission of a response Response Pending (NRC 0x78)	

Table 8.194: Operation Stop

Stop			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/DcmDspStopRoutineIn/ DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/ DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH)		
Parameters	DataIn_{Signal}	Comment	--
		Type	Dcm_StopDataIn_{Routine}_{ Signal}Type
		Variation	Signal = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/ DcmDspStopRoutineIn/ DcmDspStopRoutineInSignal. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine. SHORT-NAME)}
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataOut_{Signal}	Comment	--
		Type	Dcm_StopDataOut_{Routine}_{ Signal}Type
		Variation	Signal = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine. SHORT-NAME)
		Direction	OUT
	DataOut_{Signal}	Comment	--
		Type	Dcm_StopFlexibleOutArrayData_{ Routine}_{Signal}Type

		Variation	--
		Direction	OUT
	currentDataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	OUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	
	DCM_E_FORCE_ RCRRP	application request the transmission of a response Response Pending (NRC 0x78)	

Table 8.195: Operation Stop

Stop			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH)		
Parameters	DataIn_{Signal}	Comment	--
		Type	Dcm_StopDataIn_{Routine}_{Signal}Type
		Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
		Direction	IN
	DataIn_{Signal}	Comment	--
		Type	Dcm_StopFlexibleInArrayData_{Routine}_{Signal}Type
		Variation	--
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataOut_{Signal}	Comment	--
		Type	Dcm_StopDataOut_{Routine}_{Signal}Type

		Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)
		Direction	OUT
	currentDataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)	

Table 8.196: Operation Stop

Stop			
Comments	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH)		
Parameters	DataIn_{Signal}	Comment	--
		Type	Dcm_StopDataIn_{Routine}_{Signal}Type
		Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}

		Direction	IN
	DataIn_{Signal}	Comment	--
		Type	Dcm_StopFlexibleInArrayData_{Routine}_{Signal}Type
		Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
		Direction	IN
	OpStatus	Comment	--
		Type	Dcm_OpStatusType
		Variation	--
		Direction	IN
	DataOut_{Signal}	Comment	--
		Type	Dcm_StopDataOut_{Routine}_{Signal}Type
		Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
		Direction	OUT
	DataOut_{Signal}	Comment	--
		Type	Dcm_StopFlexibleOutArrayData_{Routine}_{Signal}Type

		Variation	{ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/ DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal. SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspRoutine. SHORT-NAME)}
		Direction	OUT
	currentDataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	INOUT
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Request was not successful	
	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.	
	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)	

Table 8.197: Operation Stop

StopConfirmation			
Comments	This operation indicates the transmission of a response to a StopRoutine request		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/ DcmDspStopRoutine/ DcmDspStopRoutineConfirmationEnabled)}==TRUE)		
Parameters	ConfirmationStatus	Comment	Confirmation status of a StopRoutine request
		Type	Dcm_ConfirmationStatusType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	

Table 8.198: Operation StopConfirmation

⌋*()*

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

8.8.3.6 RequestControlServices_{Tid}

The following interface allows the Dcm to provide OBD Service $08 (see [SWS_Dcm_00419]).

Using the concepts of the SW-C template, the interface is defined as follows: **[SWS_Dcm_00691]** ⌈

Name	RequestControlServices_{Tid}	
Comment	--	
IsService	true	
Variation	Tid = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestControl.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK

Table 8.199: Service Interface RequestControlServices_{Tid}

Operations

RequestControl			
Comments	--		
Variation	--		
Parameters	OutBuffer	Comment	--
		Type	Dcm_RequestControlServicesOutArray_{Tid}Type
		Variation	Tid = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestControl.SHORT-NAME)}
		Direction	OUT
	InBuffer	Comment	--
		Type	Dcm_RequestControlServicesInArray_{Tid}Type
		Variation	Tid = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestControl.SHORT-NAME)}
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	

Table 8.200: Operation RequestControl

⌋*()*

8.8.3.7 CallbackDCMRequestServices

The following interface provides information on the status of the protocol communication and allows the Application to disallow a protocol (see [SWS_Dcm_00036], [SWS_Dcm_00144], [SWS_Dcm_00145], [SWS_Dcm_00146]; [SWS_Dcm_00147], [SWS_Dcm_00459]).

Using the concepts of the SW-C template, the interface is defined as follows: **[SWS_Dcm_00692]** ⌈

Name	CallbackDCMRequestServices	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	5	E_PROTOCOL_NOT_ALLOWED

Table 8.201: Service Interface CallbackDCMRequestServices

Operations

StartProtocol			
Comments	--		
Variation	--		
Parameters	ProtocolType	Comment	--
		Type	Dcm_ProtocolType
		Variation	--
		Direction	IN
	TesterSourceAddress	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	ConnectionId	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	E_PROTOCOL_NOT_ALLOWED	conditions in application allows no further procession of protocol	

Table 8.202: Operation StartProtocol

StopProtocol			
Comments	--		
Variation	--		
Parameters	ProtocolType	Comment	--
		Type	Dcm_ProtocolType
		Variation	--
		Direction	IN
	TesterSourceAddress	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
	ConnectionId	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	

Table 8.203: Operation StopProtocol

⌋()

8.8.3.8 ServiceRequestNotification

The following interface indicates to the Application that a service is about to be executed and allows the Application to reject the execution of the service request (see [SWS_Dcm_00218], [SWS_Dcm_00462], [SWS_Dcm_00463]).

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

[SWS_Dcm_00694] ⌈

Name	ServiceRequestNotification	
Comment	--	
IsService	true	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsd/ DcmDsdServiceRequestManufacturerNotification)} != NULL)||({ecuc(Dcm/DcmConfigSet/DcmDsd/ DcmDsdRequestSupplierNotification)} != NULL)	
Possible Errors	0	E_OK
	1	E_NOT_OK
	8	E_REQUEST_NOT_ACCEPTED

Table 8.204: Service Interface ServiceRequestNotification

Operations

Confirmation			
Comments	--		
Variation	--		
Parameters	SID	Comment	Value of service identifier
		Type	uint8
		Variation	--
		Direction	IN
	ReqType	Comment	Addressing type of the request(0=physical request, 1=functional request)
		Type	uint8
		Variation	--
		Direction	IN
	ConnectionId	Comment	Unique connection identifier
		Type	uint16
		Variation	--
		Direction	IN
	ConfirmationStatus	Comment	Confirmation of a successful transmission or a transmission error of a diagnostic service.
		Type	Dcm_ConfirmationStatusType

		Variation	--
		Direction	IN
	ProtocolType	Comment	--
		Type	Dcm_ProtocolType
		Variation	--
		Direction	IN
	TesterSourceAddress	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	

Table 8.205: Operation Confirmation

Indication			
Comments	--		
Variation	--		
Parameters	SID	Comment	Value of service identifier
		Type	uint8
		Variation	--
		Direction	IN
	RequestData	Comment	This parameter contains the complete request data (diagnostic buffer), except the service ID
		Type	Dcm_RequestDataArrayType
		Variation	--
		Direction	IN
	DataSize	Comment	This parameter defines how many bytes in the RequestData parameter are valid
		Type	uint16
		Variation	--
		Direction	IN
	ReqType	Comment	Addressing type of the request(0=physical request, 1=functional request
		Type	uint8
		Variation	--
		Direction	IN
	ConnectionId	Comment	Unique connection identifier
		Type	uint16
		Variation	--
		Direction	IN
	ErrorCode	Comment	--
		Type	Dcm_NegativeResponseCodeType
		Variation	--
		Direction	OUT
	ProtocolType	Comment	--
		Type	Dcm_ProtocolType
		Variation	--
		Direction	IN
	TesterSourceAddress	Comment	--
		Type	uint16
		Variation	--

		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	E_REQUEST_NOT_ACCEPTED	no response will be sent	

Table 8.206: Operation Indication

⌋()

8.8.3.9 DCMServices

[SWS_Dcm_00698] ⌈

Name	DCMServices	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK

Table 8.207: Service Interface DCMServices

Operations

GetActiveProtocol			
Comments	--		
Variation	--		
Parameters	ActiveProtocolType	Comment	--
		Type	Dcm_ProtocolType
		Variation	--
		Direction	OUT
	ConnectionId	Comment	--
		Type	uint16
		Variation	--
		Direction	OUT
	TesterSourceAddress	Comment	--
		Type	uint16
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	

Table 8.208: Operation GetActiveProtocol

GetSecurityLevel			
Comments	--		
Variation	--		
Parameters	SecLevel	Comment	--
		Type	Dcm_SecLevelType
		Variation	--
		Direction	OUT

Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	

Table 8.209: Operation GetSecurityLevel

GetSesCtrlType			
Comments	--		
Variation	--		
Parameters	SesCtrlType	Comment	--
		Type	Dcm_SesCtrlType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	

Table 8.210: Operation GetSesCtrlType

ResetToDefaultSession		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

Table 8.211: Operation ResetToDefaultSession

SetActiveDiagnostic			
Comments	Allows to activate and deactivate the call of ComM_DCM_ActiveDiagnostic() function.		
Variation	--		
Parameters	active	Comment	If false Dcm shall not call ComM_DCM_ActiveDiagnostic(). If true Dcm will call ComM_DCM_ActiveDiagnostic().
		Type	boolean
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	

Table 8.212: Operation SetActiveDiagnostic

⌋*()*

8.8.3.10 DCM_Roe

The RoeEventId shall be a Portdefined argument value.

[SWS_Dcm_00699] ⌈

Name	DCM_Roe

Comment	--	
IsService	true	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoe/ DcmDspRoeEvent)}	
Possible Errors	0	E_OK
	1	E_NOT_OK

Table 8.213: Service Interface DCM_Roe

Operations

TriggerOnEvent		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

Table 8.214: Operation TriggerOnEvent

⌋*()*

8.8.4 Ports

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 \ARTechTermRef{Dcm} {

  //the presence and name of this port is configuration-independent
  ProvidePort DCMServices DCMServices;

  //see configuration parameter \AREcucContainer{DcmDspSecurityRow}
  RequirePort SecurityAccess_\{SecurityLevel\} SecurityAccess_\{
      SecurityLevel\};
  ...


  //see configuration parameter \AREcucContainer{DcmDspData}
RequirePort DataServices_\{Data\} DataServices_\{Data\};
ProvidePort DataServices_\{Data\} DataServices_\{Data\}; // Only if
        the data can be written and \AREcucParameter{DcmDspDataUsePort
   }{DcmDspData} is set to USE_DATA_SENDER_RECEIVER or to
   USE_DATA_SENDER_RECEIVER_AS_SERVICE
```

```
 ...


 //see configuration parameter \AREcucContainer{DcmDspVehInfoData}
 RequirePort InfotypeServices_{VehInfoData}
    InfotypeServices_{VehInfoData}
 ...


 //see configuration parameter \AREcucContainer{DcmDspRoutine}
 RequirePort RoutineServices_{RoutineName}
  RoutineServices_{RoutineName};
 ...

 //see configuration parameter \AREcucContainer{DcmDspRequestControl}
 RequirePort RequestControlServices_\{Tid\}
  RequestControlServices_\{Tid\};
 ...

 //see configuration parameter \AREcucContainer{
   DcmDslCallbackDCMRequestService}
 RequirePort CallbackDCMRequestServices
  CallbackDCMRequestServices_\_SWC>;
 ...

 //see configuration parameter
   DcmDsdServiceRequestManufacturerNotication
 RequirePort ServiceRequestNotification
  ServiceRequestManufacturerNotification_{Name};
 ...

//see configuration parameter DcmDsdServiceRequestSupplierNotication
 RequirePort ServiceRequestNotification
  ServiceRequestSupplierNotification_\_SWC>;
 ...

//Interface containing the DEM operations DcmClearDTC, //
  Dem_DisableDTCSetting and Dem_EnableDTCSetting
RequirePort Dem/DcmIf Dcm;

//see configuration parameter \AREcucContainer{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 \
  ARTechTermRef{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 \ARApiRef{Dcm_MainFunction}"
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣canbeInvokedConcurrently␣=␣FALSE
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣SSCP␣=␣port␣CBFaultDetectCtrDummy,␣
    GetFaultDetectionCounter

Connector␣from␣CBFaultDetectCtrDummy␣to␣Dem/GeneralEvtInfo
}
```

8.8.4.1 Dcm_CallbackDCMRequestServices_{Name}

[SWS_Dcm_01033] ⌈

Name	CallbackDCMRequestServices_{Name}		
Kind	RequiredPort	***Interface***	CallbackDCMRequest Services
Description	--		
Variation	Name = {ecuc(Dcm/DcmConfigSet/DcmDsl/ DcmDslCallbackDCMRequestService.SHORT-NAME)}		

Table 8.215: Port CallbackDCMRequestServices_{Name}

⌋*()*

8.8.4.2 DataServices_DIDRange_{Range}

[SWS_Dcm_01034] ⌈

Name	DataServices_DIDRange_{Range}		
Kind	RequiredPort	***Interface***	DataServices_DIDRange_{ Range}
Description	--		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRange. DcmDspDidRangeUsePort)} == TRUE Range = ({ecuc(Dcm/ DcmConfigSet/DcmDsp/DcmDspDidRange.SHORT-NAME)})		

Table 8.216: Port DataServices_DIDRange_{Range}

⌋*()*

8.8.4.3 DataServices_{Data}

[SWS_Dcm_01035] ⌈

Name	DataServices_{Data}		
Kind	RequiredPort	***Interface***	DataServices_{Data}

Description	--
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR || USE_DATA_SYNCH_CLIENT_SERVER)) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDspPidDataUsePort)} == (USE_DATA_SYNCH_CLIENT_SERVER || USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE)) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)} || {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)}

Table 8.217: Port DataServices_{Data}

⌋*()*

[SWS_Dcm_01310] ⌈

Name	DataServices_{Data}		
Kind	RequiredPort	***Interface***	DataServices_{Data}
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == (USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidWrite)} == NULL) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidRead)} != NULL) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl)} != NULL)) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}		

Table 8.218: Port DataServices_{Data}

⌋*()*

[SWS_Dcm_01031] ⌈

Name	DataServices_{Data}		
Kind	ProvidedPort	***Interface***	DataServices_{Data}
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == (USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidRead)} == NULL) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}		

Table 8.219: Port DataServices_{Data}

⌋()

[SWS_Dcm_01311] ⌈

Name	DataServices_{Data}		
Kind	ProvidedRequiredPort	***Interface***	DataServices_{Data}
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == (USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidRead)} != NULL) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}		

Table 8.220: Port DataServices_{Data}

⌋()

8.8.4.4 IOControlRequest_{Data}

[SWS_Dcm_01312] ⌈

Name	IOControlRequest_{Data}		
Kind	ProvidedPort	***Interface***	IOControlRequest_{Data}
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl)} != NULL) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})		

Table 8.221: Port IOControlRequest_{Data}

⌋()

8.8.4.5 IOControlResponse_{Data}

[SWS_Dcm_01313] ⌈

Name	IOControlResponse_{Data}		
Kind	ProvidedRequiredPort	***Interface***	IOControlResponse

Description	--
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER || USE_DATA_SENDER_RECEIVER_AS_SERVICE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl)} != NULL) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})

Table 8.222: Port IOControlResponse_{Data}

⌋*()*

8.8.4.6 DCM_Roe_{RoeName}

[SWS_Dcm_01032] ⌈

Name	DCM_Roe_{RoeName}		
Kind	ProvidedPort	***Interface***	DCM_Roe
Description	--		
Port Defined Argument Value(s)	Type	uint8	
	Value	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoe/DcmDspRoeEvent/DcmDspRoeEventId.value)}	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoe/DcmDspRoeEvent)} RoeName = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoe/DcmDspRoeEvent.SHORT-NAME)}		

Table 8.223: Port DCM_Roe_{RoeName}

⌋*()*

8.8.4.7 DCMServices

[SWS_Dcm_01030] ⌈

Name	DCMServices		
Kind	ProvidedPort	***Interface***	DCMServices
Description	--		
Variation	--		

Table 8.224: Port DCMServices

⌋*()*

8.8.4.8 InfotypeServices_{VehInfoData}

[SWS_Dcm_01037] ⌈

Name	InfotypeServices_{VehInfoData}		
Kind	RequiredPort	**Interface**	InfotypeServices_{VehInfoData}
Description	--		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspVehInfo/DcmDspVehInfoData/DcmDspVehInfoDataUsePort)}==TRUE VehInfoData = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspVehInfo/DcmDspVehInfoData.SHORT-NAME)}		

Table 8.225: Port InfotypeServices_{VehInfoData}

⌋()

8.8.4.9 RequestControlServices_{Tid}

[SWS_Dcm_01038] ⌈

Name	RequestControlServices_{Tid}		
Kind	RequiredPort	**Interface**	RequestControlServices_{Tid}
Description	--		
Variation	Tid = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestControl.SHORT-NAME)}		

Table 8.226: Port RequestControlServices_{Tid}

⌋()

8.8.4.10 ServiceRequestManufacturerNotification_{Name}

[SWS_Dcm_01039] ⌈

Name	ServiceRequestManufacturerNotification_{Name}		
Kind	RequiredPort	**Interface**	ServiceRequestNotification
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsd/DcmDsdServiceRequestManufacturerNotification)} != NULL) Name = {ecuc(Dcm/DcmConfigSet/DcmDsd/DcmDsdServiceRequestManufacturerNotification.SHORT-NAME)}		

Table 8.227: Port ServiceRequestManufacturerNotification_{Name}

⌋()

8.8.4.11 ServiceRequestSupplierNotification_{Name}

[SWS_Dcm_01042] ⌈

Name	ServiceRequestSupplierNotification_{Name}		
Kind	RequiredPort	***Interface***	ServiceRequestNotification
Description	--		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsd/DcmDsdServiceRequestSupplierNotification)} != NULL) Name = {ecuc(Dcm/DcmConfigSet/DcmDsd/DcmDsdServiceRequestSupplierNotification.SHORT-NAME)}		

Table 8.228: Port ServiceRequestSupplierNotification_{Name}

⌋*()*

8.8.4.12 RoutineServices_{RoutineName}

[SWS_Dcm_01040] ⌈

Name	RoutineServices_{RoutineName}		
Kind	RequiredPort	***Interface***	RoutineServices_{RoutineName}
Description	--		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineUsePort)} == TRUE RoutineName = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		

Table 8.229: Port RoutineServices_{RoutineName}

⌋*()*

8.8.4.13 SecurityAccess_{SecurityLevel}

[SWS_Dcm_01041] ⌈

Name	SecurityAccess_{SecurityLevel}		
Kind	RequiredPort	***Interface***	SecurityAccess_{SecurityLevel}
Description	--		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)} == USE_ASYNCH_CLIENT_SERVER SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}		

Table 8.230: Port SecurityAccess_{SecurityLevel}

⌋*()*

8.8.4.14 Dcm_DiagnosticSessionControlModeSwitchInterface

[SWS_Dcm_91033] ⌈

Name	DiagnosticSessionControlModeSwitchInterface		
Kind	ProvidedPort	***Interface***	Dcm_DiagnosticSessionControlModeSwitchInterface
Description	A ModeSwitchInterface PPortPrototype used to notify SW-Cs about the current Diagnostic Session		
Variation	--		

Table 8.231: Port DiagnosticSessionControlModeSwitchInterface

⌋*()*

8.8.4.15 Dcm_EcuResetModeSwitchInterface

[SWS_Dcm_91034] ⌈

Name	EcuResetModeSwitchInterface		
Kind	ProvidedPort	***Interface***	Dcm_EcuResetModeSwitchInterface
Description	A ModeSwitchInterface PPortPrototype used to notify SW-Cs about an upcoming ECU Reset and its type		
Variation	--		

Table 8.232: Port EcuResetModeSwitchInterface

⌋*()*

8.8.4.16 Dcm_ModeRapidPowerShutDownModeSwitchInterface

[SWS_Dcm_91035] ⌈

Name	ModeRapidPowerShutDownModeSwitchInterface		
Kind	ProvidedPort	***Interface***	Dcm_ModeRapidPowerShutDownModeSwitchInterface
Description	A ModeSwitchInterface PPortPrototype used to notify SW-Cs about the rapid power shut down mode		
Variation	--		

Table 8.233: Port ModeRapidPowerShutDownModeSwitchInterface

⌋*()*

8.8.4.17 Dcm_CommunicationControlModeSwitchInterface_{ComMChannelName}

[SWS_Dcm_91036] ⌈

Name	CommunicationControlModeSwitchInterface_{ComMChannelName}		
Kind	ProvidedPort	***Interface***	CommunicationControlModeSwitchInterface_{ComMChannelName}
Description	A ModeSwitchInterface PPortPrototype used to notify SW-Cs about the communication control of the indicated ComM channel		
Variation	ComMChannelName = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspComControl/DcmDspComControlAllChannel/DcmDspAllComMChannelRef->ComMChannel.SHORT-NAME)} or {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspComControl/DcmDspComControlSpecificChannel/DcmDspSpecificComMChannelRef->ComMChannel.SHORT-NAME)}		

Table 8.234: Port CommunicationControlModeSwitchInterface_{ComMChannelName}

⌋*()*

8.8.4.18 Dcm_ControlDTCSettingModeSwitchInterface

[SWS_Dcm_91037] ⌈

Name	ControlDTCSettingModeSwitchInterface		
Kind	ProvidedPort	***Interface***	Dcm_ControlDTCSettingModeSwitchInterface
Description	A ModeSwitchInterface PPortPrototype used to notify SW-Cs about the DTC Setting mode		
Variation	--		

Table 8.235: Port ControlDTCSettingModeSwitchInterface

⌋*()*

8.8.4.19 Dcm_ResponseOnEventModeSwitchInterface_{RoeEventID}

[SWS_Dcm_91038] ⌈

Name	ResponseOnEventModeSwitchInterface_{RoeEventID}		
Kind	ProvidedPort	***Interface***	ResponseOnEvent_{RoeEventID}
Description	A ModeSwitchInterface PPortPrototype used to notify SW-Cs about the mode of the indicated Response On Event		
Variation	RoeEventID = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoe/DcmDspRoeEvent.SHORT-NAME)}		

Table 8.236: Port ResponseOnEventModeSwitchInterface_{RoeEventID}

⌋()

8.8.4.20 Dcm_SecurityAccessModeSwitchInterface

[SWS_Dcm_91039] ⌈

Name	SecurityAccessModeSwitchInterface		
Kind	ProvidedPort	***Interface***	Dcm_SecurityAccessModeSwitchInterface
Description	A ModeSwitchInterface PPortPrototype used to notify SW-Cs about the current Security Level		
Variation	--		

Table 8.237: Port SecurityAccessModeSwitchInterface

⌋()

8.8.5 ModeDeclarationGroups

8.8.5.1 DcmDiagnosticSessionControl

[SWS_Dcm_91019] ⌈

Name	DcmDiagnosticSessionControl	
Kind	ModeDeclarationGroup	
Category	EXPLICIT_ORDER	
Initial mode	DCM_DEFAULT_SESSION	
On transition value	255	
Modes	DCM_DEFAULT_SESSION	0
	DCM_PROGRAMMING_SESSION	1
	DCM_EXTENDED_DIAGNOSTIC_SESSION	2
	DCM_SAFETY_SYSTEM_DIAGNOSTIC_SESSION	3
Description	ModeDeclarationGroup representing the different diagnostic sessions Further modes to be added: {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSession/DcmDspSessionRow.SHORT-NAME)}	

Table 8.238: Mode Declaration Group DcmDiagnosticSessionControl

⌋()

Note: According [SWS_Dcm_CONSTR_6001] there are standardized mode declaration which are part of the standardized AUTOSAR interface. Note: Refer [ecuc_sws_2108] defining the symbolic name prefix

8.8.5.2 DcmEcuReset

[SWS_Dcm_91021] ⌈

Name	DcmEcuReset	
Kind	ModeDeclarationGroup	
Category	EXPLICIT_ORDER	
Initial mode	DCM_NONE	
On transition value	255	
Modes	DCM_NONE	0
	DCM_HARD	1
	DCM_KEYONOFF	2
	DCM_SOFT	3
	DCM_JUMPTOBOOTLOADER	4
	DCM_JUMPTOSYSSUPPLIERBOOTLOADER	5
	DCM_EXECUTE	6
Description	ModeDeclarationGroup representing the different ECU reset types	

Table 8.239: Mode Declaration Group DcmEcuReset

⌋*()*

8.8.5.3 DcmModeRapidPowerShutDown

[SWS_Dcm_91023] ⌈

Name	DcmModeRapidPowerShutDown	
Kind	ModeDeclarationGroup	
Category	EXPLICIT_ORDER	
Initial mode	DCM_ENABLE_RAPIDPOWERSHUTDOWN	
On transition value	255	
Modes	DCM_ENABLE_RAPIDPOWERSHUTDOWN	0
	DCM_DISABLE_RAPIDPOWERSHUTDOWN	1
Description	ModeDeclarationGroup representing the enable/disable state of rapid power shutdown	

Table 8.240: Mode Declaration Group DcmModeRapidPowerShutDown

⌋*()*

8.8.5.4 DcmCommunicationControl

[SWS_Dcm_91025] ⌈

Name	DcmCommunicationControl

Kind	ModeDeclarationGroup	
Category	EXPLICIT_ORDER	
Initial mode	DCM_ENABLE_RX_TX_NORM_NM	
On transition value	255	
Modes	DCM_ENABLE_RX_TX_NORM	0
	DCM_ENABLE_RX_DISABLE_TX_NORM	1
	DCM_DISABLE_RX_ENABLE_TX_NORM	2
	DCM_DISABLE_RX_TX_NORMAL	3
	DCM_ENABLE_RX_TX_NM	4
	DCM_ENABLE_RX_DISABLE_TX_NM	5
	DCM_DISABLE_RX_ENABLE_TX_NM	6
	DCM_DISABLE_RX_TX_NM	7
	DCM_ENABLE_RX_TX_NORM_NM	8
	DCM_ENABLE_RX_DISABLE_TX_NORM_NM	9
	DCM_DISABLE_RX_ENABLE_TX_NORM_NM	10
	DCM_DISABLE_RX_TX_NORM_NM	11
Description	ModeDeclarationGroup representing the different communication control states	

Table 8.241: Mode Declaration Group DcmCommunicationControl

⌋*()*

8.8.5.5 DcmControlDTCSetting

[SWS_Dcm_91027] ⌈

Name	DcmControlDTCSetting	
Kind	ModeDeclarationGroup	
Category	EXPLICIT_ORDER	
Initial mode	DCM_ENABLEDTCSETTING	
On transition value	255	
Modes	DCM_ENABLEDTCSETTING	0
	DCM_DISABLEDTCSETTING	1
Description	ModeDeclarationGroup representing the enable/disable state for DTC storage	

Table 8.242: Mode Declaration Group DcmControlDTCSetting

⌋*()*

8.8.5.6 DcmResponseOnEvent

[SWS_Dcm_91029] ⌈

Name	DcmResponseOnEvent	
Kind	ModeDeclarationGroup	
Category	EXPLICIT_ORDER	
Initial mode	DCM_EVENT_CLEARED	
On transition value	255	
Modes	DCM_EVENT_STARTED	0
	DCM_EVENT_STOPPED	1
	DCM_EVENT_CLEARED	2
Description	ModeDeclarationGroup representing the state of a Response On Event	

Table 8.243: Mode Declaration Group DcmResponseOnEvent

⌋*()*

8.8.5.7 DcmSecurityAccess

[SWS_Dcm_91031] ⌈

Name	DcmSecurityAccess	
Kind	ModeDeclarationGroup	
Category	EXPLICIT_ORDER	
Initial mode	DCM_SEC_LEV_LOCKED	
On transition value	255	
Modes	DCM_SEC_LEV_LOCKED	0
Description	ModeDeclarationGroup representing the different diagnostic security levels Further modes to be added: {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT NAME)}	

Table 8.244: Mode Declaration Group DcmSecurityAccess

⌋*()*

8.8.6 Mode-Switch-Interfaces

8.8.6.1 Dcm_DiagnosticSessionControlModeSwitchInterface

[SWS_Dcm_91020] ⌈

Name	Dcm_DiagnosticSessionControlModeSwitchInterface
Comment	A SW-C that wants to get informed about the current Diagnostic Session requires the ModeSwitchInterface Dcm_DiagnosticSessionControlModeSwitchInterface
IsService	true

Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSession/ DcmDspSessionRow.SHORT-NAME)}	
ModeGroup	diagnosticSession	DcmDiagnosticSessionControl

Table 8.245: Service Interface Dcm_DiagnosticSessionControlModeSwitchInterface

⌋()

8.8.6.2 Dcm_EcuResetModeSwitchInterface

[SWS_Dcm_91022] ⌈

Name	Dcm_EcuResetModeSwitchInterface	
Comment	A SW-C that wants to get informed about an upcoming ECU Reset requires the ModeSwitchInterface Dcm_EcuResetModeSwitchInterface	
IsService	true	
Variation	--	
ModeGroup	ecuReset	DcmEcuReset

Table 8.246: Service Interface Dcm_EcuResetModeSwitchInterface

⌋()

8.8.6.3 Dcm_ModeRapidPowerShutDownModeSwitchInterface

[SWS_Dcm_91024] ⌈

Name	Dcm_ModeRapidPowerShutDownModeSwitchInterface	
Comment	A SW-C that wants to get informed about the rapid power shut down mode requires the ModeSwitchInterface Dcm_ModeRapidPowerShutDownModeSwitchInterface	
IsService	true	
Variation	--	
ModeGroup	modeRapidPowerShutDown	DcmModeRapidPowerShutDown

Table 8.247: Service Interface Dcm_ModeRapidPowerShutDownModeSwitchInterface

⌋()

8.8.6.4 Dcm_CommunicationControlModeSwitchInterface

[SWS_Dcm_91026] ⌈

Name	Dcm_CommunicationControlModeSwitchInterface

Comment	A SW-C that wants to get informed about the communication control of a ComM channel requires the ModeSwitchInterface Dcm_CommunicationControlModeSwitchInterface	
IsService	true	
Variation	ComMChannelName = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspComControl/DcmDspComControlAllChannel/DcmDspAllComMChannelRef->ComMChannel.SHORT-NAME)} || {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspComControl/DcmDspComControlSpecificChannel/DcmDspSpecificComMChannelRef->	
ModeGroup	communicationControl	DcmCommunicationControl

Table 8.248: Service Interface Dcm_CommunicationControlModeSwitchInterface

⌋()

8.8.6.5 Dcm_ControlDTCSettingModeSwitchInterface

[SWS_Dcm_91028] ⌈

Name	Dcm_ControlDTCSettingModeSwitchInterface	
Comment	A SW-C that wants to get informed about the DTC Setting mode requires the ModeSwitchInterface Dcm_ControlDTCSettingModeSwitchInterface	
IsService	true	
Variation	--	
ModeGroup	controlDTCSetting	DcmControlDTCSetting

Table 8.249: Service Interface Dcm_ControlDTCSettingModeSwitchInterface

⌋()

8.8.6.6 Dcm_ResponseOnEventModeSwitchInterface

[SWS_Dcm_91030] ⌈

Name	Dcm_ResponseOnEventModeSwitchInterface	
Comment	A SW-C that wants to get informed about a Response On Event mode requires the ModeSwitchInterface Dcm_ResponseOnEventModeSwitchInterface	
IsService	true	
Variation	RoeEventID = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoe/DcmDspRoeEvent.SHORT-NAME)}	
ModeGroup	responseOnEvent	DcmResponseOnEvent

Table 8.250: Service Interface Dcm_ResponseOnEventModeSwitchInterface

⌋()

8.8.6.7 Dcm_SecurityAccessModeSwitchInterface

[SWS_Dcm_91032] ⌈

Name	Dcm_SecurityAccessModeSwitchInterface	
Comment	A SW-C that wants to get informed about the current Security Level requires the ModeSwitchInterface Dcm_SecurityAccessModeSwitchInterface	
IsService	true	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/ DcmDspSecurityRow.SHORT-NAME)}	
ModeGroup	securityAccess	DcmSecurityAccess

Table 8.251: Service Interface Dcm_SecurityAccessModeSwitchInterface

⌋*()*

8.9 External diagnostic service processing

The following chapter applies only to external processed diagnostic services.

8.9.1 <Module>_<DiagnosticService>

[SWS_Dcm_00763] ⌈

Service name:	<Module>_<DiagnosticService>	

Syntax:	`Std_ReturnType <Module>_<DiagnosticService>(`
	`Dcm_ExtendedOpStatusType OpStatus,`
	`Dcm_MsgContextType* pMsgContext,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x32	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL DCM_PENDING DCM_CANCEL DCM_FORCE_RCRRP_OK DCM_POS_RESPONSE_SENT DCM_POS_RESPONSE_FAILED DCM_NEG_RESPONSE_SENT DCM_NEG_RESPONSE_FAILED
Parameters (inout):	pMsgContext	Message-related information for one diagnostic protocol identifier. The pointers in pMsgContext shall point behind the SID.
Parameters (out):	ErrorCode	If the operation <Module>_<DiagnosticService> returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.

Return value:	Std_ReturnType	E_OK: Request was successful E_NOT_OK: Request was not successful DCM_E_PENDING: Request is not yet finished DCM_E_FORCE_RCRRP: Application requests the transmission of a response Response Pending (NRC 0x78)
Description:	Callout function. The 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 that it is reentrant. Caller is responsible for the lifetime of the argument pMsgContext. The name of the callout is defined within parameter DcmDsdSidTabFnc	

Table 8.252: <Module>_<DiagnosticService>

⌋*()*

8.9.2 <Module>_<DiagnosticService>_<SubService>

[SWS_Dcm_00764] ⌈

Service name:	<Module>_<DiagnosticService>_<SubService>	

Syntax:	`Std_ReturnType <Module>_<DiagnosticService>_<SubService>(`
	`Dcm_ExtendedOpStatusType OpStatus,`
	`Dcm_MsgContextType* pMsgContext,`
	`Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID[hex]:	0x33	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	OpStatus	DCM_INITIAL DCM_PENDING DCM_CANCEL DCM_FORCE_RCRRP_OK DCM_POS_RESPONSE_SENT DCM_POS_RESPONSE_FAILED DCM_NEG_RESPONSE_SENT DCM_NEG_RESPONSE_FAILED
Parameters (inout):	pMsgContext	Message-related information for one diagnostic protocol identifier. The pointers in pMsgContext shall point behind the SID.
Parameters (out):	ErrorCode	If the operation <Module>_<DiagnosticService>_<SubService> returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.

Return value:	Std_ReturnType	E_OK: Request was successful E_NOT_OK: Request was not successful DCM_E_PENDING: Request is not yet finished DCM_E_FORCE_RCRRP: Application requests the transmission of a response Response Pending (NRC 0x78)
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.	

Table 8.253: <Module>_<DiagnosticService>_<SubService>

⌋*()*

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 ConnectionId
Dcm_ConfirmationStatusType status)
```

This function confirms the successful transmission or a transmission error of a diagnostic service. This is the right time to perform any application state transitions.

This API is also called if the response to a diagnostic service is suppressed.

8.10.4 DslInternal_ResponseOnOneEvent

```
Dcm_StatusType
DslInternal_ResponseOnOneEvent(const Dcm_MsgType MsgPtr,
Dcm_MsgLenType MsgLen,
uint16 ConnectionId)
```

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

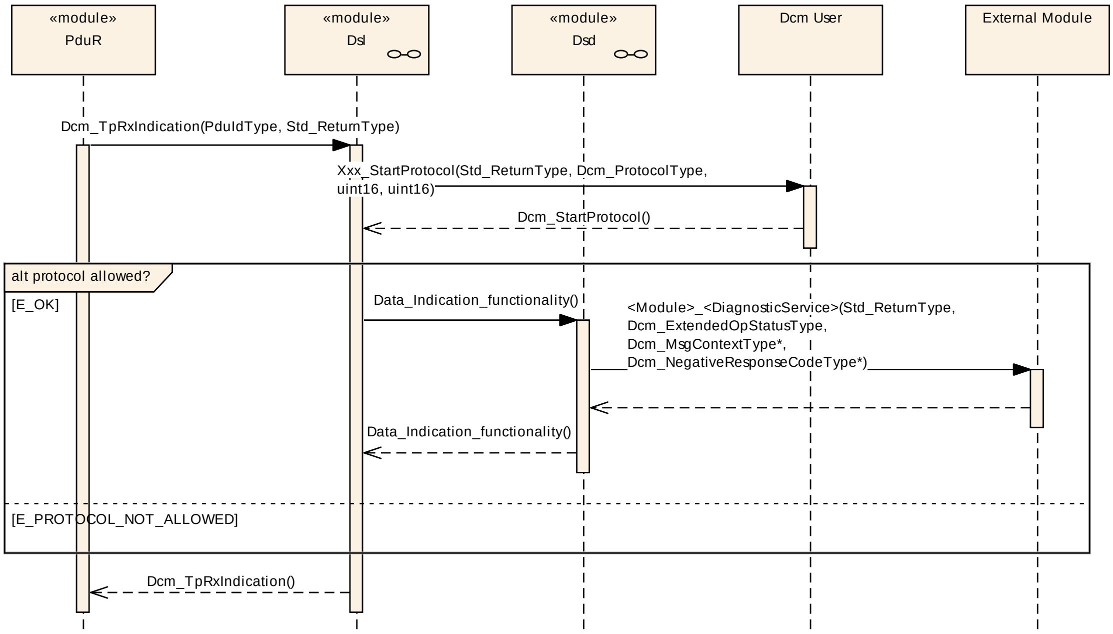

«module»
PduR
«module»
Dsl
«module»
Dsd
Dcm User
External Module
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Xxx_StartProtocol(Std_ReturnType, Dcm_ProtocolType, uint16, uint16)
Dcm_StartProtocol()
alt protocol allowed?
[E_OK]
Data_Indication_functionality()
<Module>_<DiagnosticService>(Std_ReturnType, Dcm_ExtendedOpStatusType, Dcm_MsgContextType*, Dcm_NegativeResponseCodeType*)
Data_Indication_functionality()
[E_PROTOCOL_NOT_ALLOWED]
Dcm_TpRxIndication()

Figure 9.1

9.2.2 Process Busy behavior

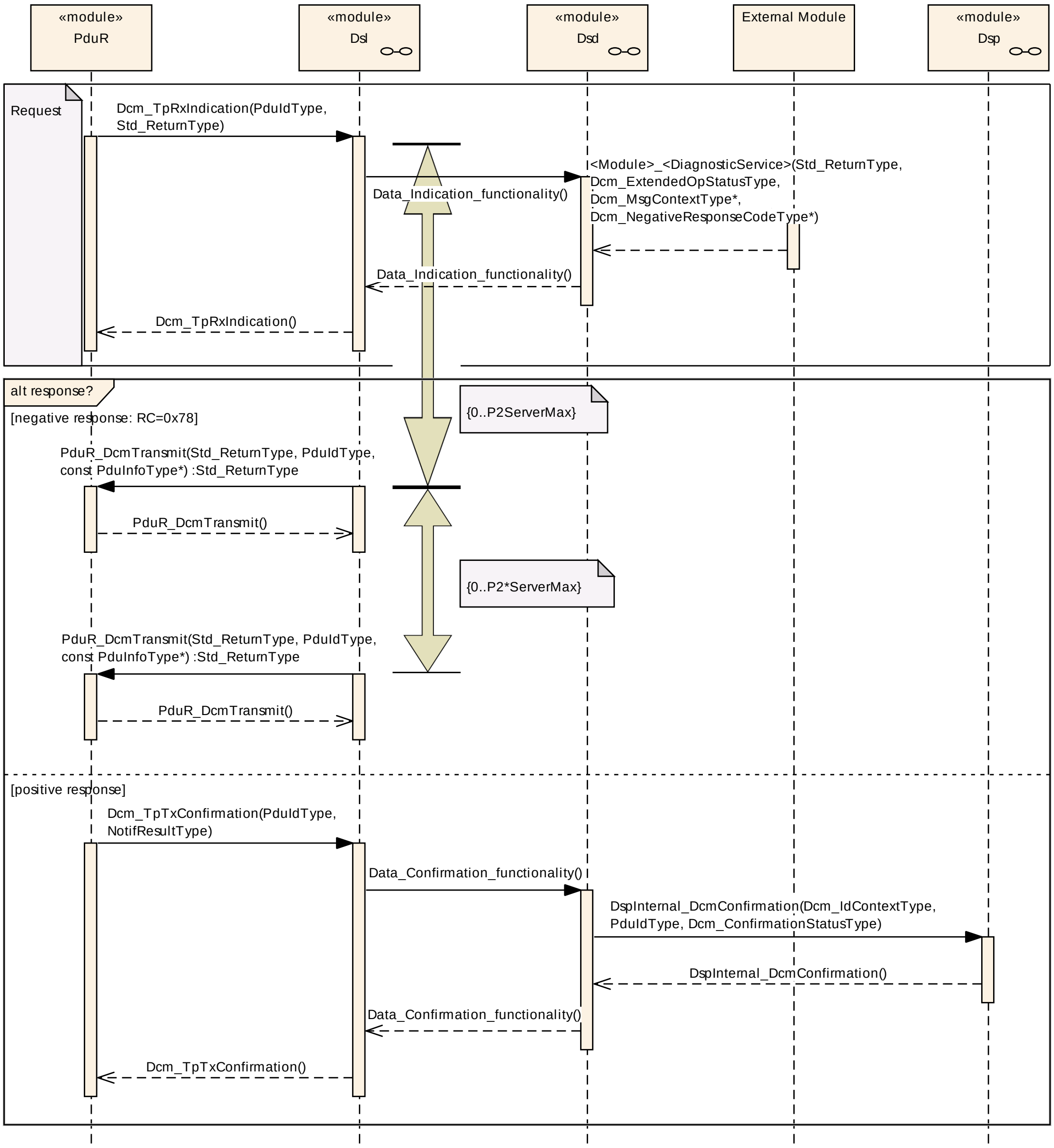

«module»
PduR
«module»
Dsl
«module»
Dsd
External Module
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
<Module>_<DiagnosticService>(Std_ReturnType, Dcm_ExtendedOpStatusType, Dcm_MsgContextType*, Dcm_NegativeResponseCodeType*)
Data_Indication_functionality()
Dcm_TpRxIndication()
alt response?
[negative response: RC=0x78]
{0..P2ServerMax}
PduR_DcmTransmit(Std_ReturnType, PduIdType, const PduInfoType*) :Std_ReturnType
PduR_DcmTransmit()
{0..P2*ServerMax}
PduR_DcmTransmit(Std_ReturnType, PduIdType, const PduInfoType*) :Std_ReturnType
PduR_DcmTransmit()
[positive response]
Dcm_TpTxConfirmation(PduIdType, NotifResultType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TpTxConfirmation()

Figure 9.2

Internally, the DSL submodule calculates the time to response the tester. In the case that the external module processing the request doesn't close the request by returning E_OK or E_NOT_OK to <Module>_<DiagnosticService>() or <Module>_<DiagnosticService>_<SubService>() APIs call (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

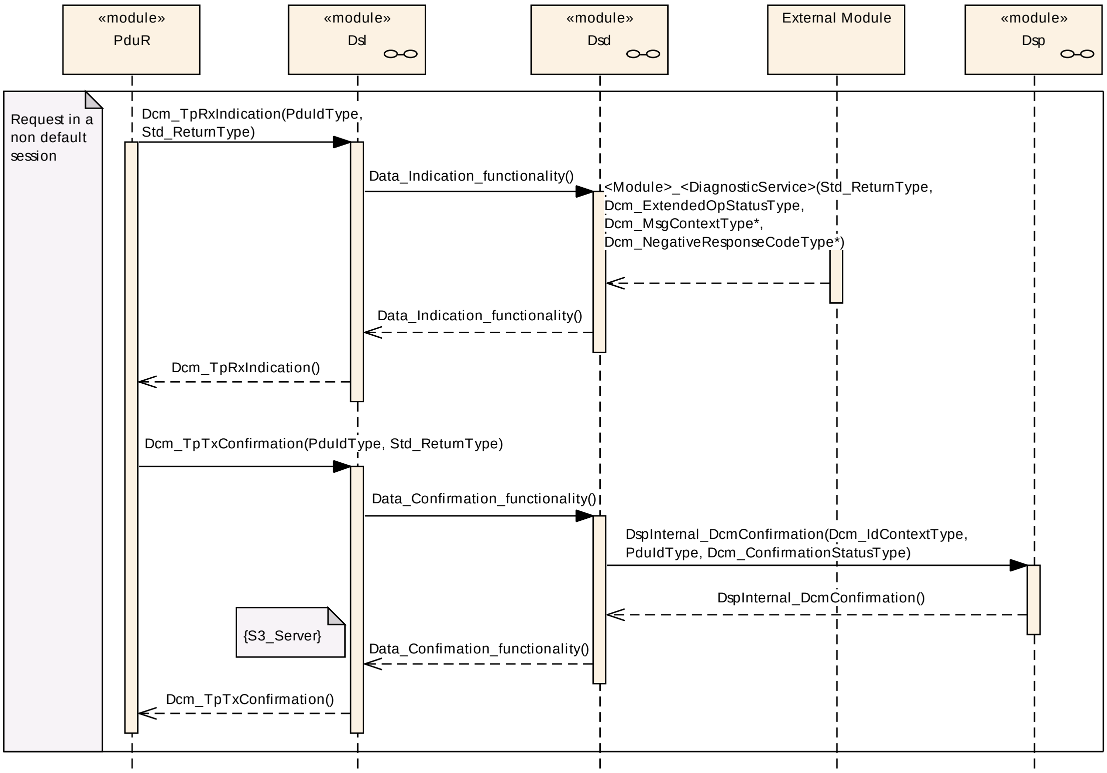

«module»
PduR
«module»
Dsl
«module»
Dsd
External Module
«module»
Dsp
Request in a non default session
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
<Module>_<DiagnosticService>(Std_ReturnType, Dcm_ExtendedOpStatusType, Dcm_MsgContextType*, Dcm_NegativeResponseCodeType*)
Data_Indication_functionality()
Dcm_TpRxIndication()
Dcm_TpTxConfirmation(PduIdType, Std_ReturnType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
{S3_Server}
Data_Confimation_functionality()
Dcm_TpTxConfirmation()

Figure 9.3

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

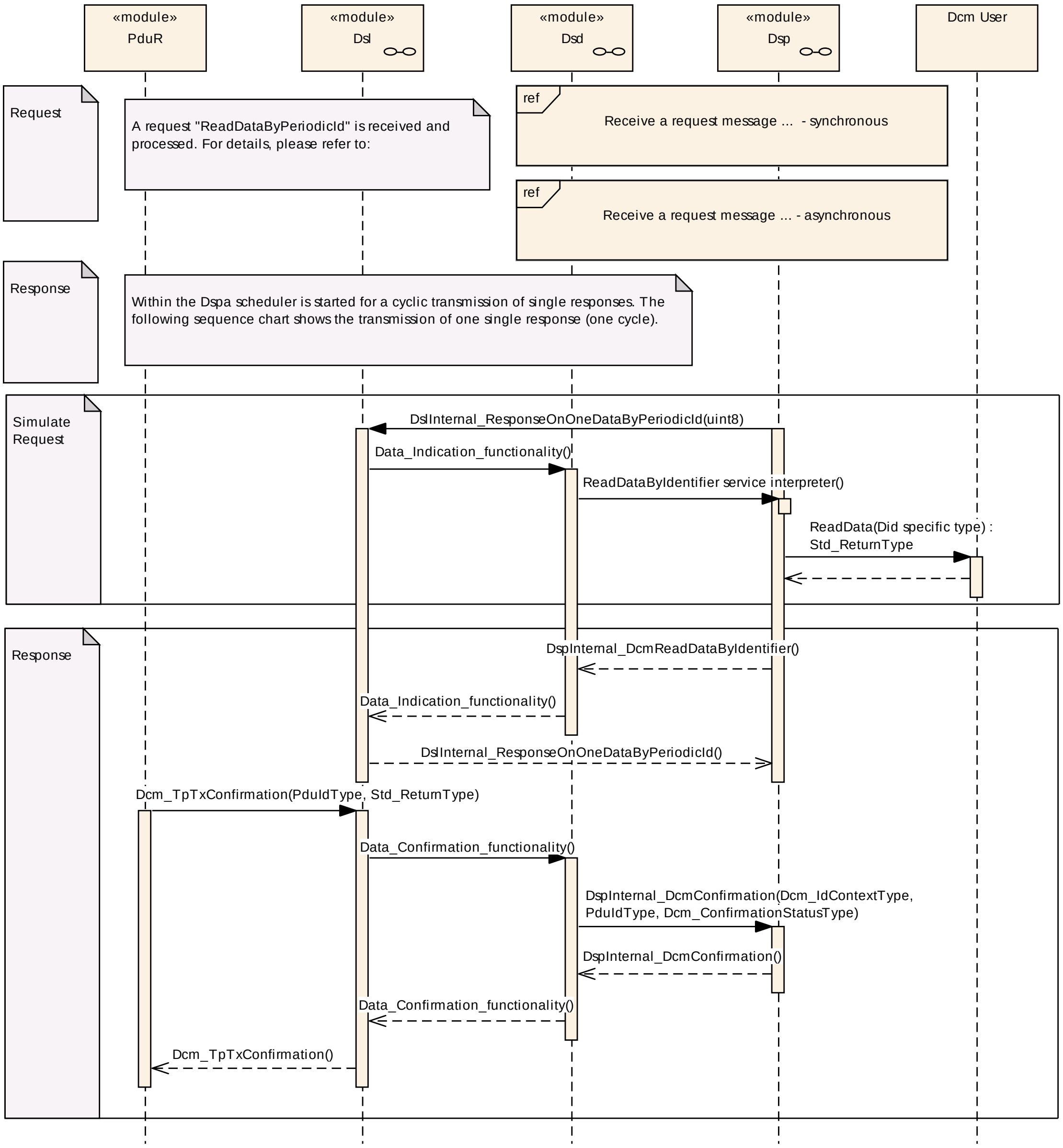

«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
Dcm User
Request
A request "ReadDataByPeriodicId" is received and processed. For details, please refer to:
ref
Receive a request message ... - synchronous
ref
Receive a request message ... - asynchronous
Response
Within the Dspa scheduler is started for a cyclic transmission of single responses. The following sequence chart shows the transmission of one single response (one cycle).
Simulate Request
DslInternal_ResponseOnOneDataByPeriodicId(uint8)
Data_Indication_functionality()
ReadDataByIdentifier service interpreter()
ReadData(Did specific type) : Std_ReturnType
Response
DspInternal_DcmReadDataByIdentifier()
Data_Indication_functionality()
DslInternal_ResponseOnOneDataByPeriodicId()
Dcm_TpTxConfirmation(PduIdType, Std_ReturnType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TpTxConfirmation()

Figure 9.4

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 [15]) 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

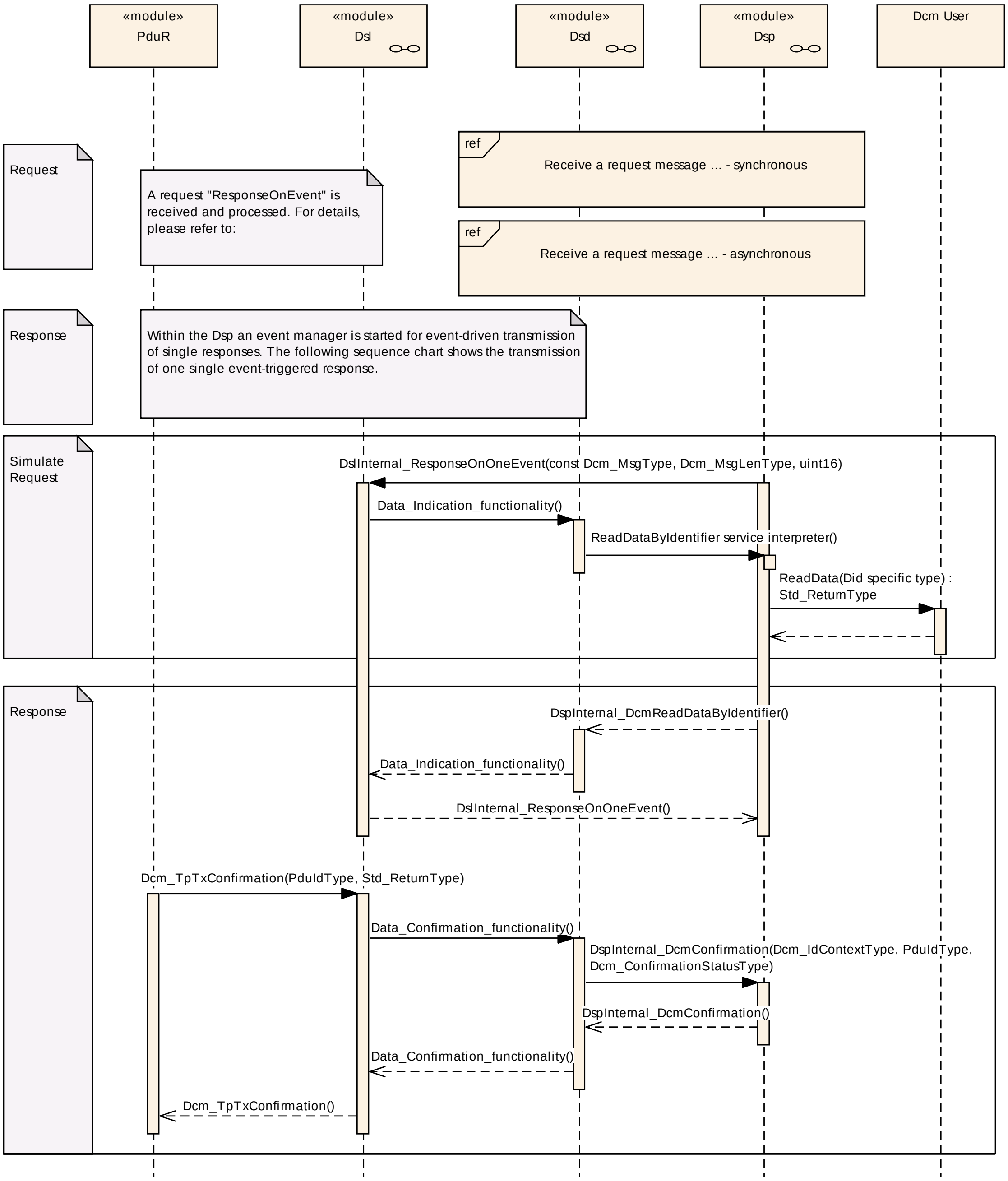

«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
Dcm User
ref
Receive a request message ... - synchronous
ref
Receive a request message ... - asynchronous
Request
A request "ResponseOnEvent" is received and processed. For details, please refer to:
Response
Within the Dsp an event manager is started for event-driven transmission of single responses. The following sequence chart shows the transmission of one single event-triggered response.
Simulate Request
DslInternal_ResponseOnOneEvent(const Dcm_MsgType, Dcm_MsgLenType, uint16)
Data_Indication_functionality()
ReadDataByIdentifier service interpreter()
ReadData(Did specific type) : Std_ReturnType
Response
DspInternal_DcmReadDataByIdentifier()
Data_Indication_functionality()
DslInternal_ResponseOnOneEvent()
Dcm_TpTxConfirmation(PduIdType, Std_ReturnType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TpTxConfirmation()

Figure 9.5

This sequence diagram shows an example for ResponseOnEvent. ResponseOnEvent is setup and started for onDTCStatusChange. Event changes are reported to the Dcm which will trigger a serviceToRespondTo.

9.2.6 Process concurrent requests

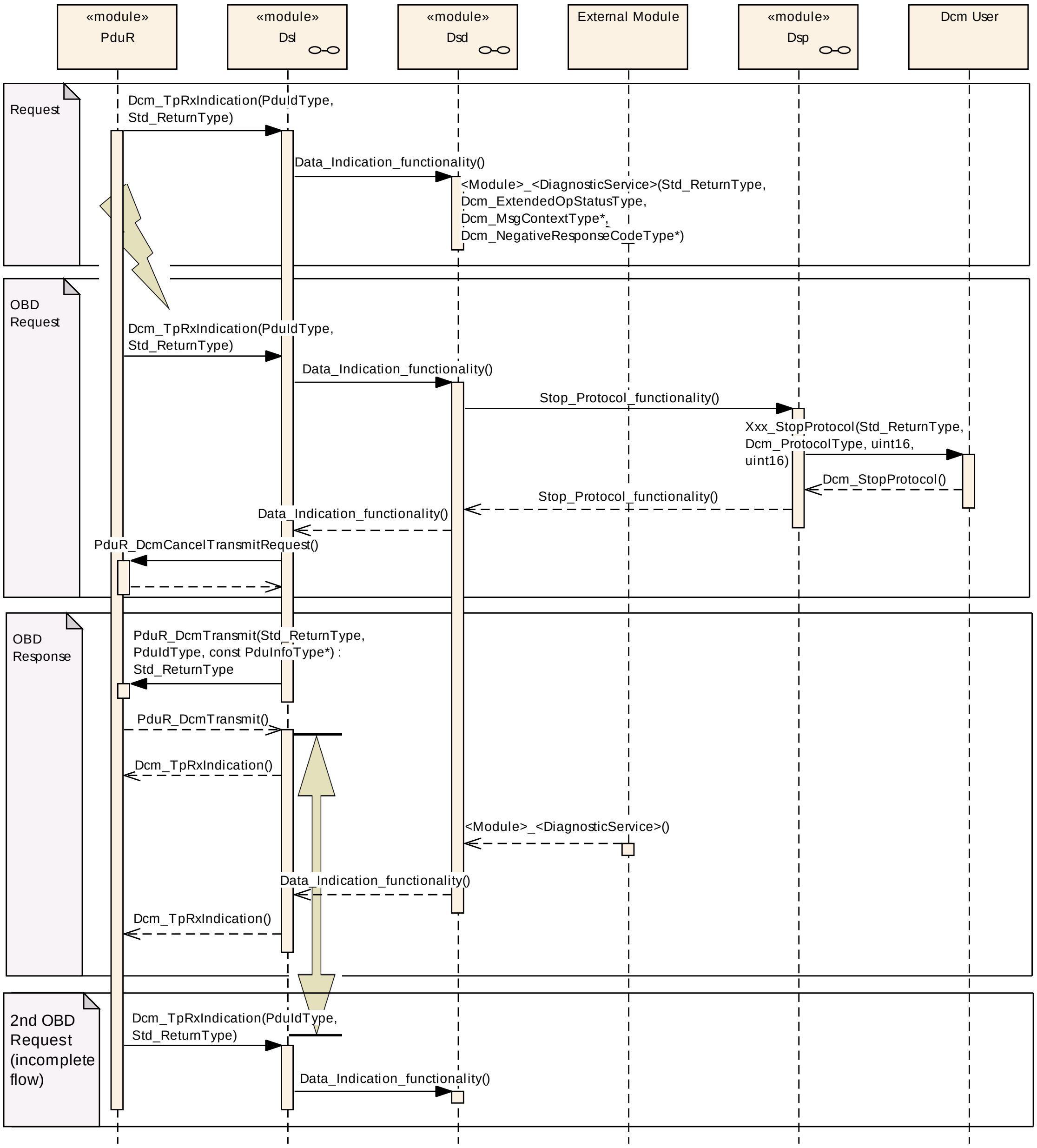

«module» PduR
«module» Dsl
«module» Dsd
External Module
«module» Dsp
Dcm User
Request
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
<Module>_<DiagnosticService>(Std_ReturnType, Dcm_ExtendedOpStatusType, Dcm_MsgContextType*, Dcm_NegativeResponseCodeType*)
OBD Request
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
Stop_Protocol_functionality()
Xxx_StopProtocol(Std_ReturnType, Dcm_ProtocolType, uint16, uint16)
Dcm_StopProtocol()
Stop_Protocol_functionality()
Data_Indication_functionality()
PduR_DcmCancelTransmitRequest()
OBD Response
PduR_DcmTransmit(Std_ReturnType, PduIdType, const PduInfoType*) : Std_ReturnType
PduR_DcmTransmit()
Dcm_TpRxIndication()
<Module>_<DiagnosticService>()
Data_Indication_functionality()
Dcm_TpRxIndication()
2nd OBD Request (incomplete flow)
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()

Figure 9.6

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.

As long as the external module processing the request is not finished (finish is indicated by returning E_OK or E_NOT_OK to <Module>_<DiagnosticService>()/<Module>_<DiagnosticService>_<SubService>() API call) or no timeout occurs, the DSL submodule responses with negative response "BusyRepeatRequest".

With receiving E_OK or E_NOT_OK from the external module to <Module>_<DiagnosticService>()/<Module>_<DiagnosticService>_<SubService>() API call, 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 external module processing the request never returns E_OK or E_NOT_OK to <Module>_<DiagnosticService>()/<Module>_<DiagnosticService>_<SubService>() API call, 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

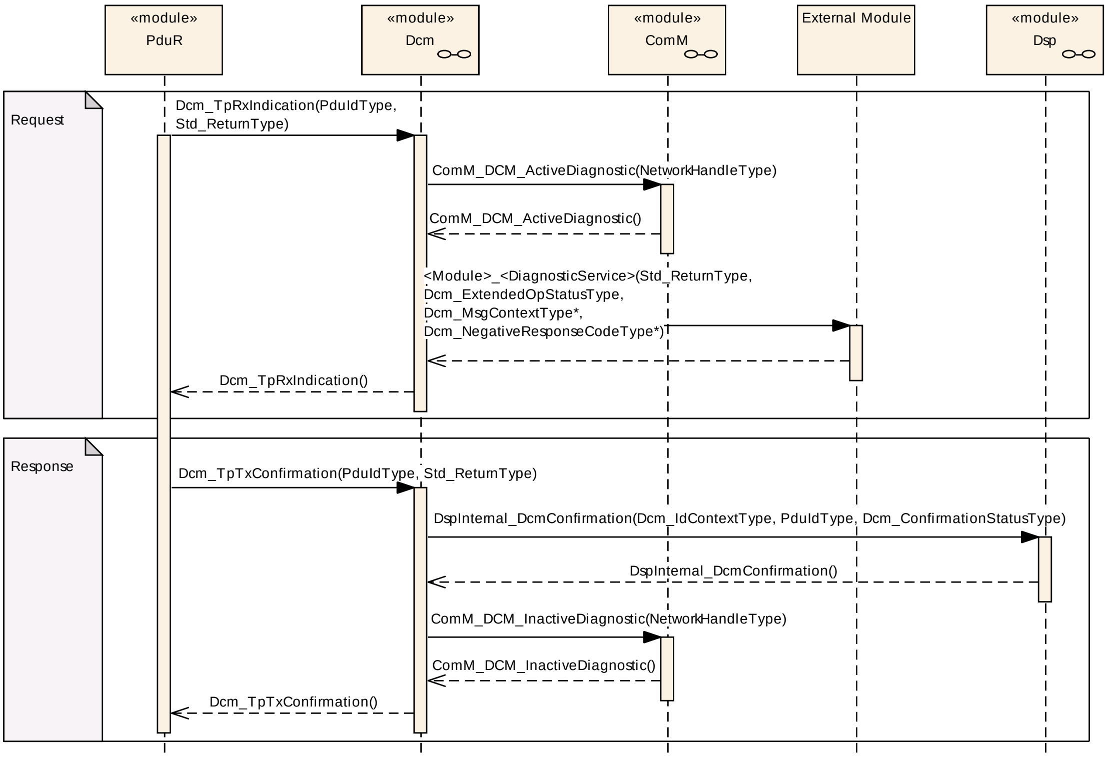

«module»
PduR
«module»
Dcm
«module»
ComM
External Module
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
ComM_DCM_ActiveDiagnostic(NetworkHandleType)
ComM_DCM_ActiveDiagnostic()
<Module>_<DiagnosticService>(Std_ReturnType, Dcm_ExtendedOpStatusType, Dcm_MsgContextType*, Dcm_NegativeResponseCodeType*)
Dcm_TpRxIndication()
Response
Dcm_TpTxConfirmation(PduIdType, Std_ReturnType)
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
ComM_DCM_InactiveDiagnostic(NetworkHandleType)
ComM_DCM_InactiveDiagnostic()
Dcm_TpTxConfirmation()

Figure 9.7

9.2.7.2 Handling in Non-Default Session

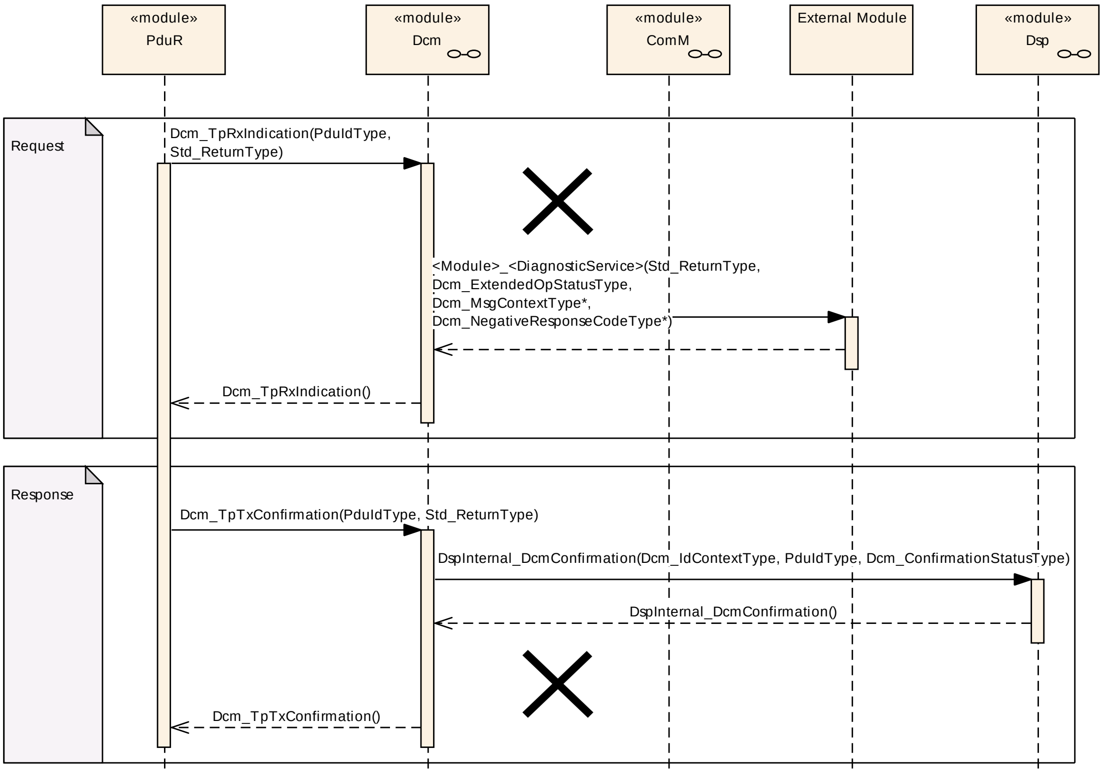

«module»
PduR
«module»
Dcm
«module»
ComM
External Module
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType,
Std_ReturnType)
<Module>_<DiagnosticService>(Std_ReturnType,
Dcm_ExtendedOpStatusType,
Dcm_MsgContextType*,
Dcm_NegativeResponseCodeType*)
Dcm_TpRxIndication()
Response
Dcm_TpTxConfirmation(PduIdType, Std_ReturnType)
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Dcm_TpTxConfirmation()

Figure 9.8

9.2.7.3 Session transitions

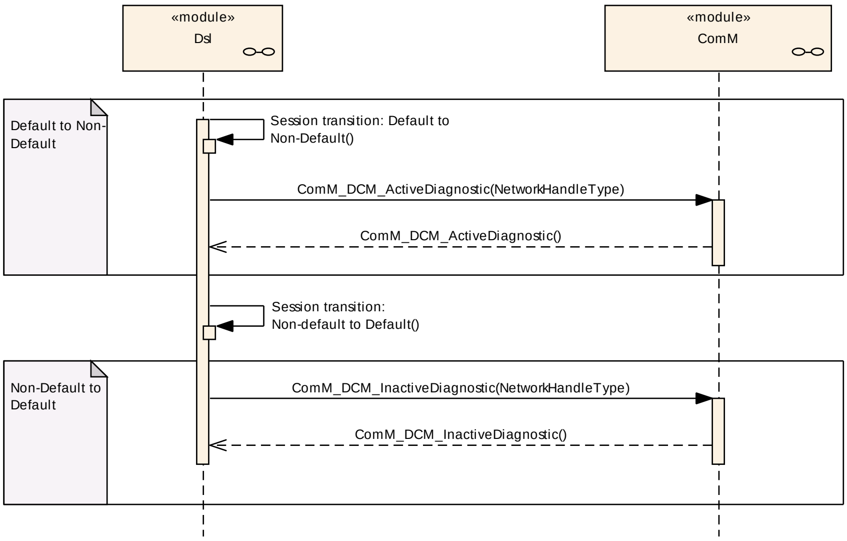

«module»
Dsl
«module»
ComM
Default to Non-Default
Session transition: Default to
Non-Default()
ComM_DCM_ActiveDiagnostic(NetworkHandleType)
ComM_DCM_ActiveDiagnostic()
Session transition:
Non-default to Default()
Non-Default to Default
ComM_DCM_InactiveDiagnostic(NetworkHandleType)
ComM_DCM_InactiveDiagnostic()

Figure 9.9

9.2.7.4 Communication States

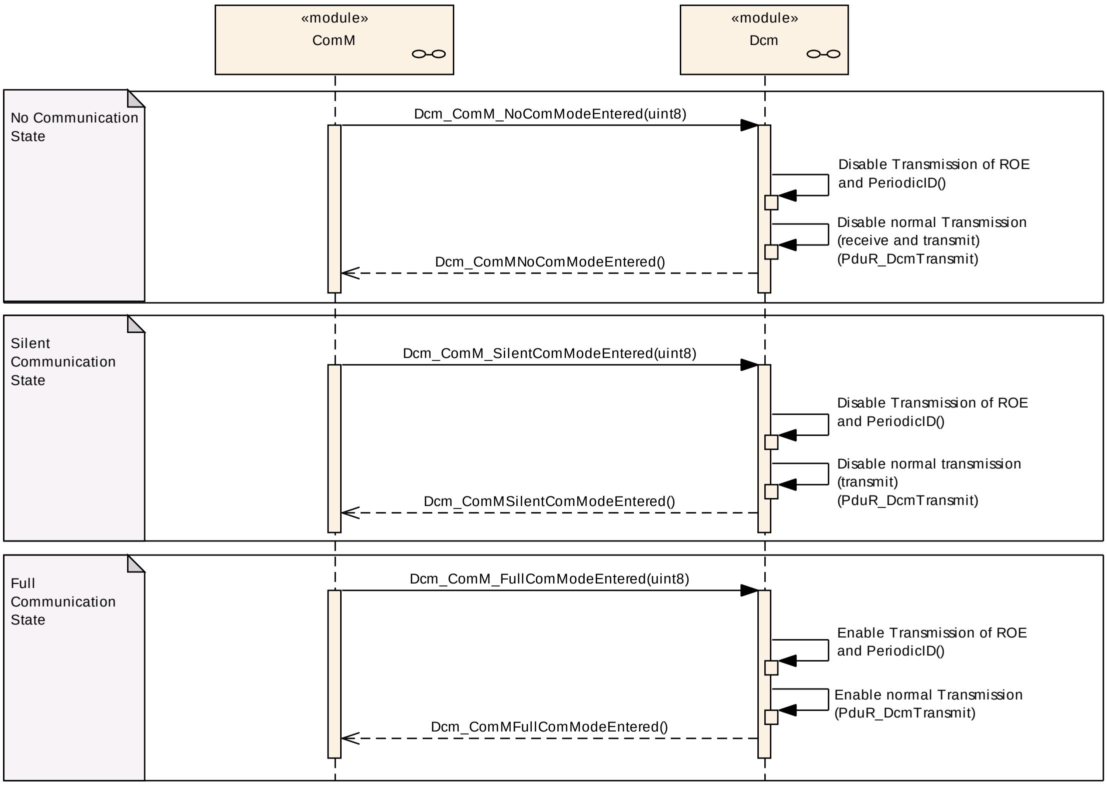

«module»
ComM
«module»
Dcm
No Communication State
Dcm_ComM_NoComModeEntered(uint8)
Disable Transmission of ROE and PeriodicID()
Disable normal Transmission (receive and transmit) (PduR_DcmTransmit)
Dcm_ComMNoComModeEntered()
Silent Communication State
Dcm_ComM_SilentComModeEntered(uint8)
Disable Transmission of ROE and PeriodicID()
Disable normal transmission (transmit) (PduR_DcmTransmit)
Dcm_ComMSilentComModeEntered()
Full Communication State
Dcm_ComM_FullComModeEntered(uint8)
Enable Transmission of ROE and PeriodicID()
Enable normal Transmission (PduR_DcmTransmit)
Dcm_ComMFullComModeEntered()

Figure 9.10

DSD (Diagnostic Service Dispatcher) Receive a request message and transmit a positive response message - synchronous transmission

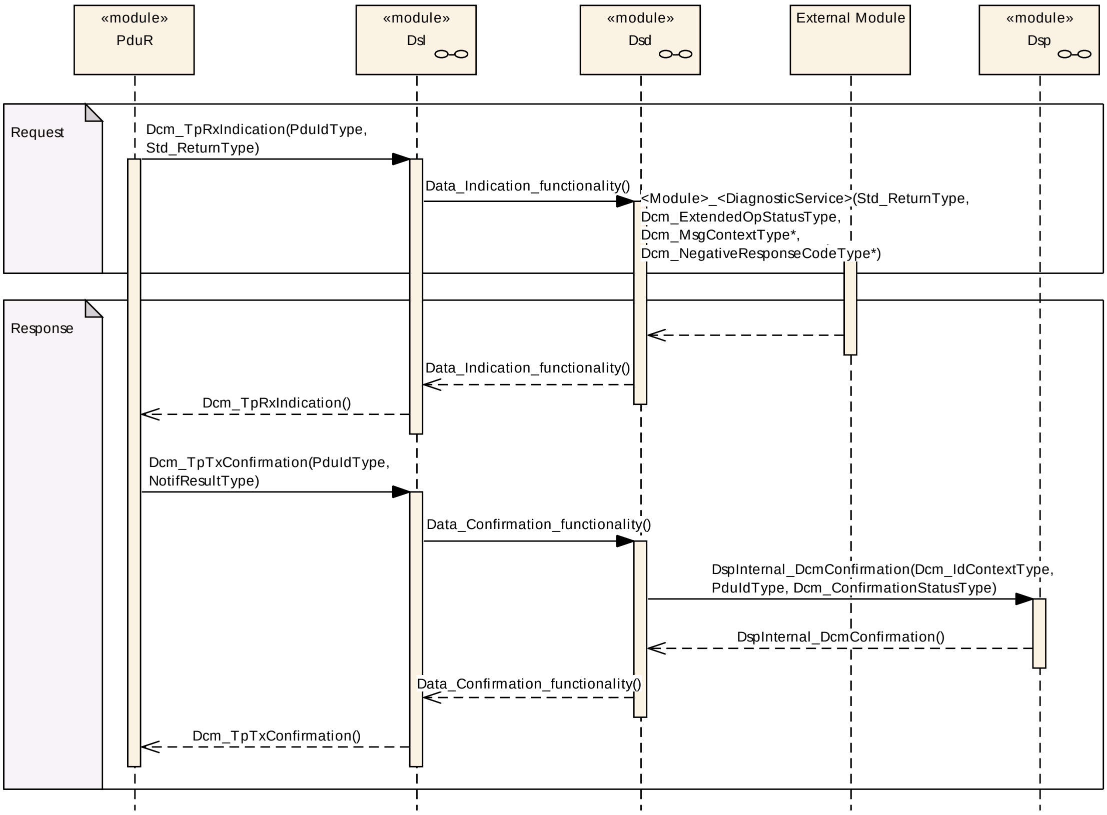

«module»
PduR
«module»
Dsl
«module»
Dsd
External Module
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType,
Std_ReturnType)
Data_Indication_functionality()
<Module>_<DiagnosticService>(Std_ReturnType,
Dcm_ExtendedOpStatusType,
Dcm_MsgContextType*,
Dcm_NegativeResponseCodeType*)
Response
Data_Indication_functionality()
Dcm_TpRxIndication()
Dcm_TpTxConfirmation(PduIdType,
NotifResultType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType,
PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TpTxConfirmation()

Figure 9.11

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

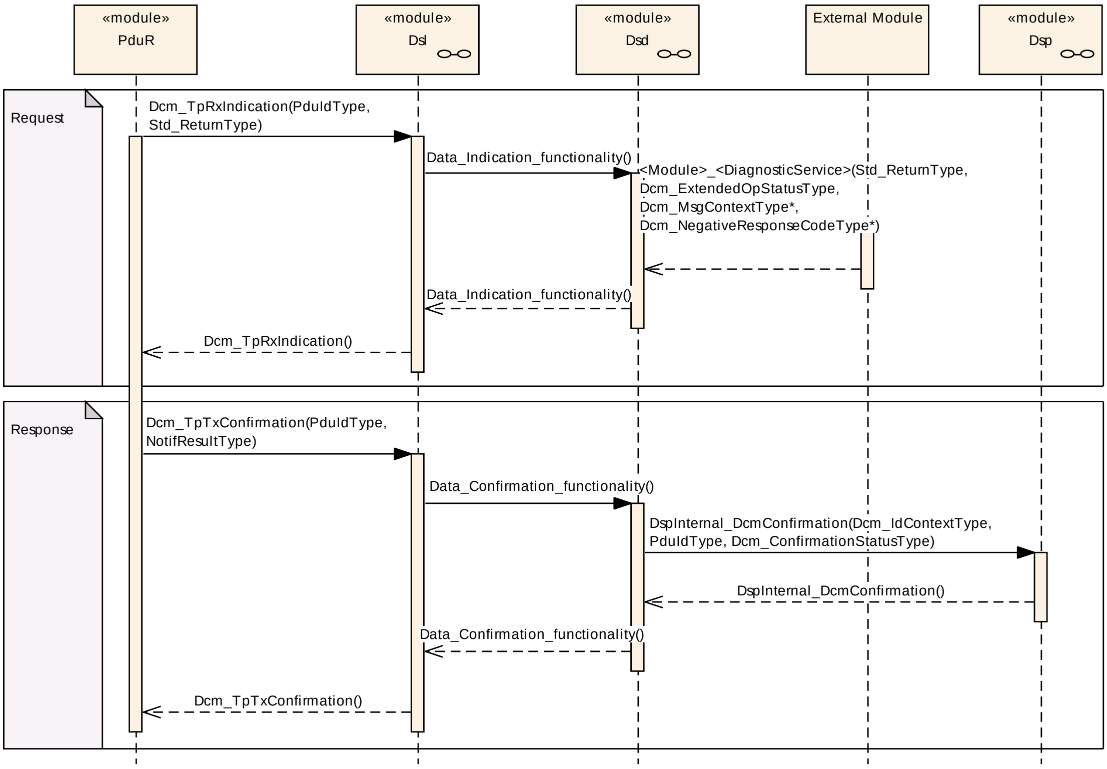

«module»
PduR
«module»
Dsl
«module»
Dsd
External Module
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType,
Std_ReturnType)
Data_Indication_functionality()
<Module>_<DiagnosticService>(Std_ReturnType,
Dcm_ExtendedOpStatusType,
Dcm_MsgContextType*,
Dcm_NegativeResponseCodeType*)
Data_Indication_functionality()
Dcm_TpRxIndication()
Response
Dcm_TpTxConfirmation(PduIdType,
NotifResultType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType,
PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TpTxConfirmation()

Figure 9.12

Receive a request message and suppress a positive response

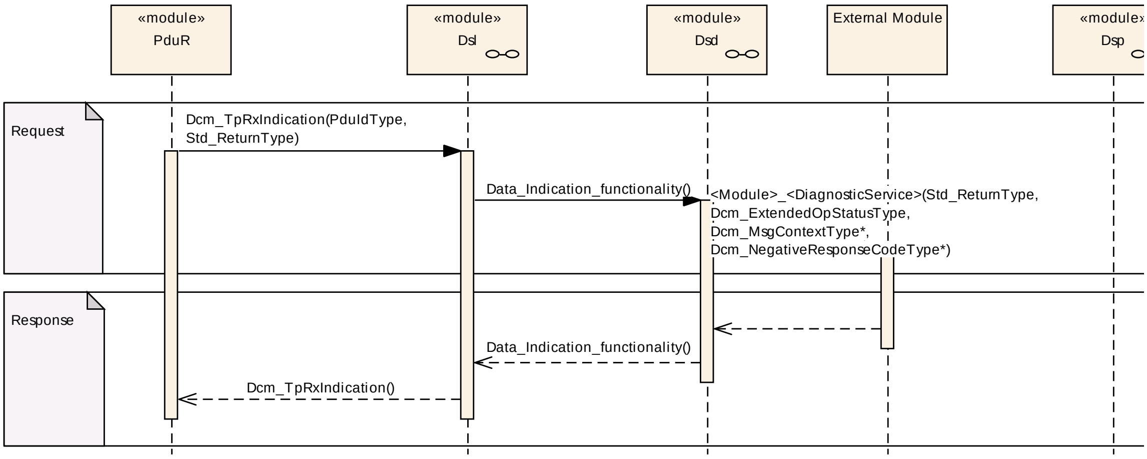

«module»
PduR
«module»
Dsl
«module»
Dsd
External Module
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType,
Std_ReturnType)
Data_Indication_functionality()
<Module>_<DiagnosticService>(Std_ReturnType,
Dcm_ExtendedOpStatusType,
Dcm_MsgContextType*,
Dcm_NegativeResponseCodeType*)
Response
Data_Indication_functionality()
Dcm_TpRxIndication()

Figure 9.13

9.2.8 Receive request message and transmit negative response message

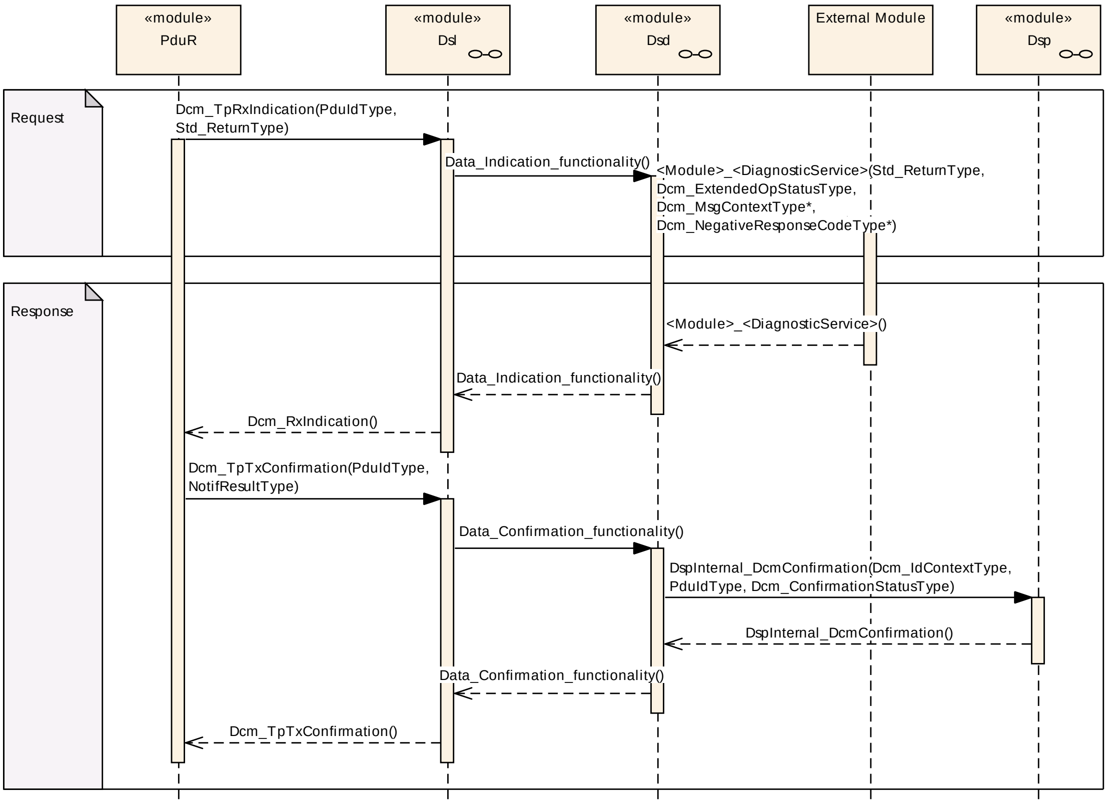

«module»
PduR
«module»
Dsl
«module»
Dsd
External Module
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType,
Std_ReturnType)
Data_Indication_functionality()
<Module>_<DiagnosticService>(Std_ReturnType,
Dcm_ExtendedOpStatusType,
Dcm_MsgContextType*,
Dcm_NegativeResponseCodeType*)
Response
<Module>_<DiagnosticService>()
Data_Indication_functionality()
Dcm_RxIndication()
Dcm_TpTxConfirmation(PduIdType,
NotifResultType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType,
PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TpTxConfirmation()

Figure 9.14

9.2.9 Process Service Request with paged-buffer

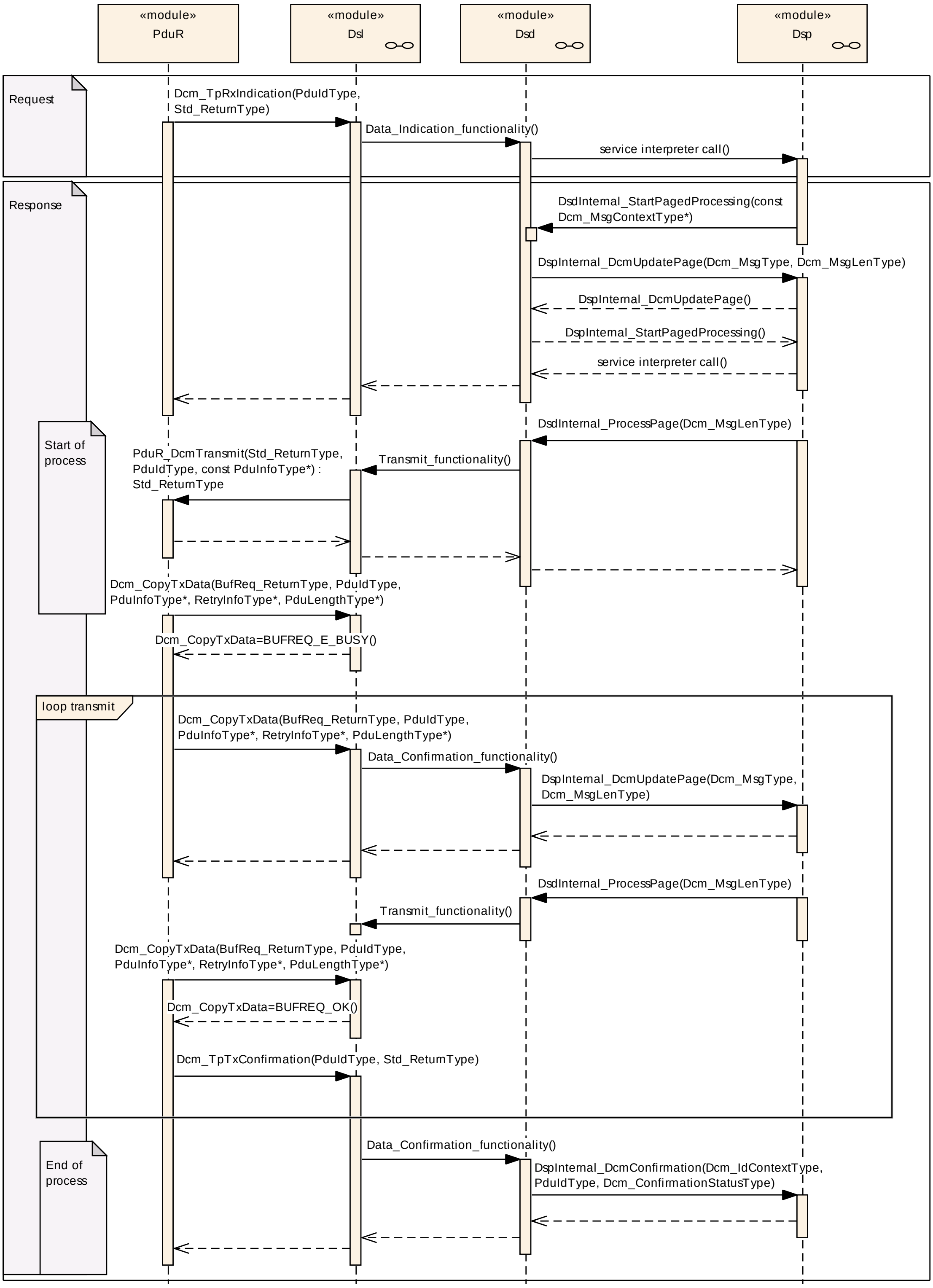

«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
service interpreter call()
Response
DsdInternal_StartPagedProcessing(const Dcm_MsgContextType*)
DspInternal_DcmUpdatePage(Dcm_MsgType, Dcm_MsgLenType)
DspInternal_DcmUpdatePage()
DspInternal_StartPagedProcessing()
service interpreter call()
Start of process
DsdInternal_ProcessPage(Dcm_MsgLenType)
PduR_DcmTransmit(Std_ReturnType, PduIdType, const PduInfoType*) : Std_ReturnType
Transmit_functionality()
Dcm_CopyTxData(BufReq_ReturnType, PduIdType, PduInfoType*, RetryInfoType*, PduLengthType*)
Dcm_CopyTxData=BUFREQ_E_BUSY()
loop transmit
Dcm_CopyTxData(BufReq_ReturnType, PduIdType, PduInfoType*, RetryInfoType*, PduLengthType*)
Data_Confirmation_functionality()
DspInternal_DcmUpdatePage(Dcm_MsgType, Dcm_MsgLenType)
DsdInternal_ProcessPage(Dcm_MsgLenType)
Transmit_functionality()
Dcm_CopyTxData(BufReq_ReturnType, PduIdType, PduInfoType*, RetryInfoType*, PduLengthType*)
Dcm_CopyTxData=BUFREQ_OK()
Dcm_TpTxConfirmation(PduIdType, Std_ReturnType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
End of process

Figure 9.15

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()), and getting a negative Tx confirmation from underlying Transport Layers, the 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.

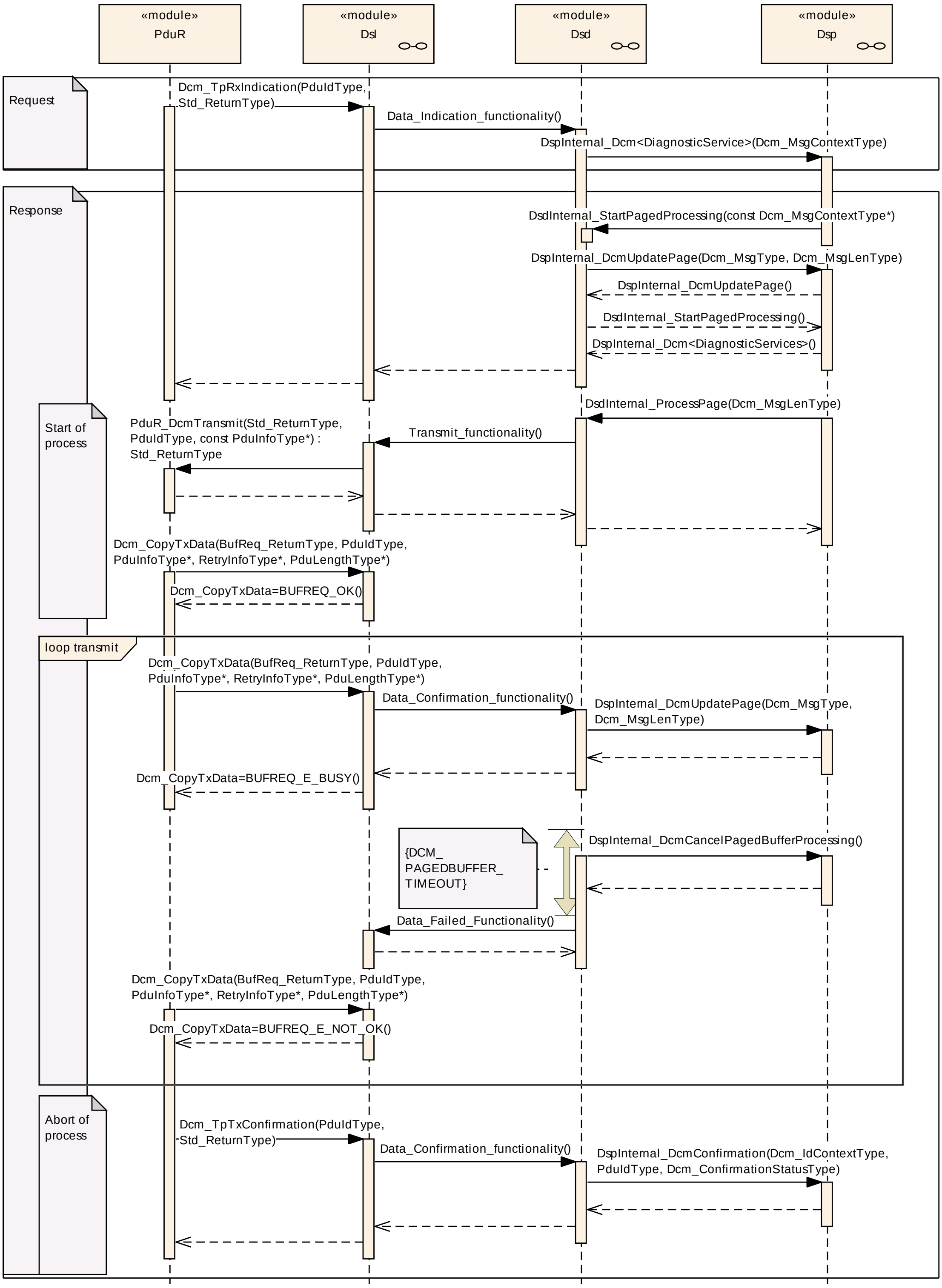

«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
Request
Dcm_TpRxIndication(PduIdType,
Std_ReturnType)
Data_Indication_functionality()
DspInternal_Dcm<DiagnosticService>(Dcm_MsgContextType)
Response
DsdInternal_StartPagedProcessing(const Dcm_MsgContextType*)
DspInternal_DcmUpdatePage(Dcm_MsgType, Dcm_MsgLenType)
DspInternal_DcmUpdatePage()
DsdInternal_StartPagedProcessing()
DspInternal_Dcm<DiagnosticServices>()
DsdInternal_ProcessPage(Dcm_MsgLenType)
Start of
process
PduR_DcmTransmit(Std_ReturnType,
PduIdType, const PduInfoType*) :
Std_ReturnType
Transmit_functionality()
Dcm_CopyTxData(BufReq_ReturnType, PduIdType,
PduInfoType*, RetryInfoType*, PduLengthType*)
Dcm_CopyTxData=BUFREQ_OK()
loop transmit
Dcm_CopyTxData(BufReq_ReturnType, PduIdType,
PduInfoType*, RetryInfoType*, PduLengthType*)
Data_Confirmation_functionality()
DspInternal_DcmUpdatePage(Dcm_MsgType,
Dcm_MsgLenType)
Dcm_CopyTxData=BUFREQ_E_BUSY()
{DCM_
PAGEDBUFFER_
TIMEOUT}
DspInternal_DcmCancelPagedBufferProcessing()
Data_Failed_Functionality()
Dcm_CopyTxData(BufReq_ReturnType, PduIdType,
PduInfoType*, RetryInfoType*, PduLengthType*)
Dcm_CopyTxData=BUFREQ_E_NOT_OK()
Abort of
process
Dcm_TpTxConfirmation(PduIdType,
Std_ReturnType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType,
PduIdType, Dcm_ConfirmationStatusType)

Figure 9.16

9.2.10 Process copy data in reception

Please refer to Figure 9 "CanTp I-PDU reception" in [9, SWS PduR].

9.2.11 Process copy data in transmission

Please refer to Figure 14 "CanTp I-PDU transmission" in [9, SWS PduR].

9.3 DSP (Diagnostic Service Processing)

9.3.1 Interface DSP - DEM (service 0x19, 0x14, 0x85)

Please refer to Section 9 in [11, SWS Dem].

9.3.2 Interface special services

9.3.2.1 Process Diagnostic Session Control

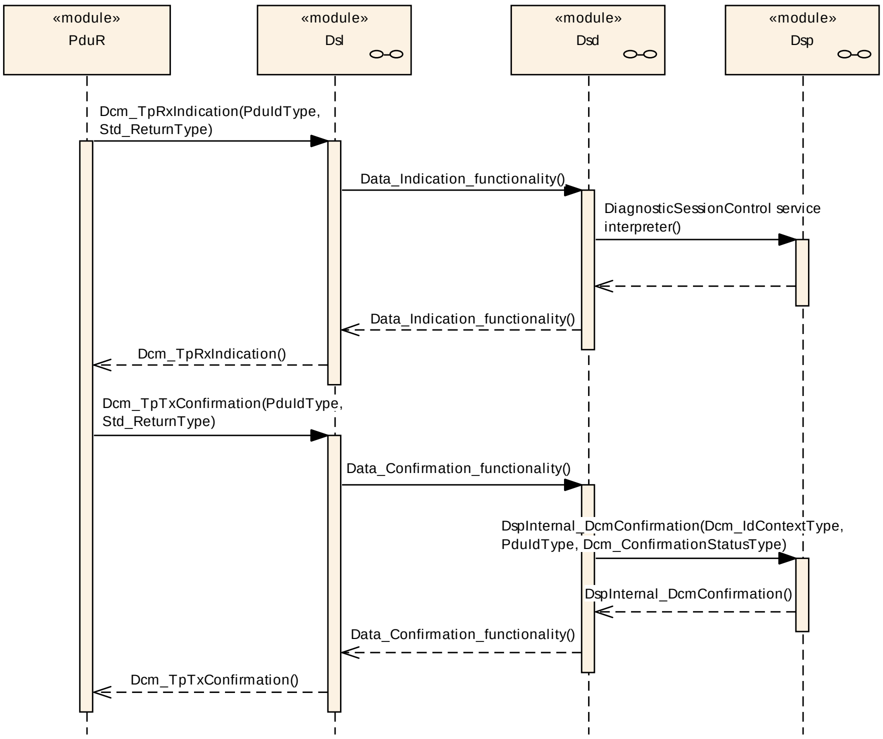

«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
DiagnosticSessionControl service interpreter()
Data_Indication_functionality()
Dcm_TpRxIndication()
Dcm_TpTxConfirmation(PduIdType, Std_ReturnType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TpTxConfirmation()

Figure 9.17

Above sequence diagram shows processing of Diagnostic Session Control request from a tester. Note that the new diagnostic session and timing parameters only apply after the transmission confirmation of the server positive response

9.3.2.2 Process Tester Present

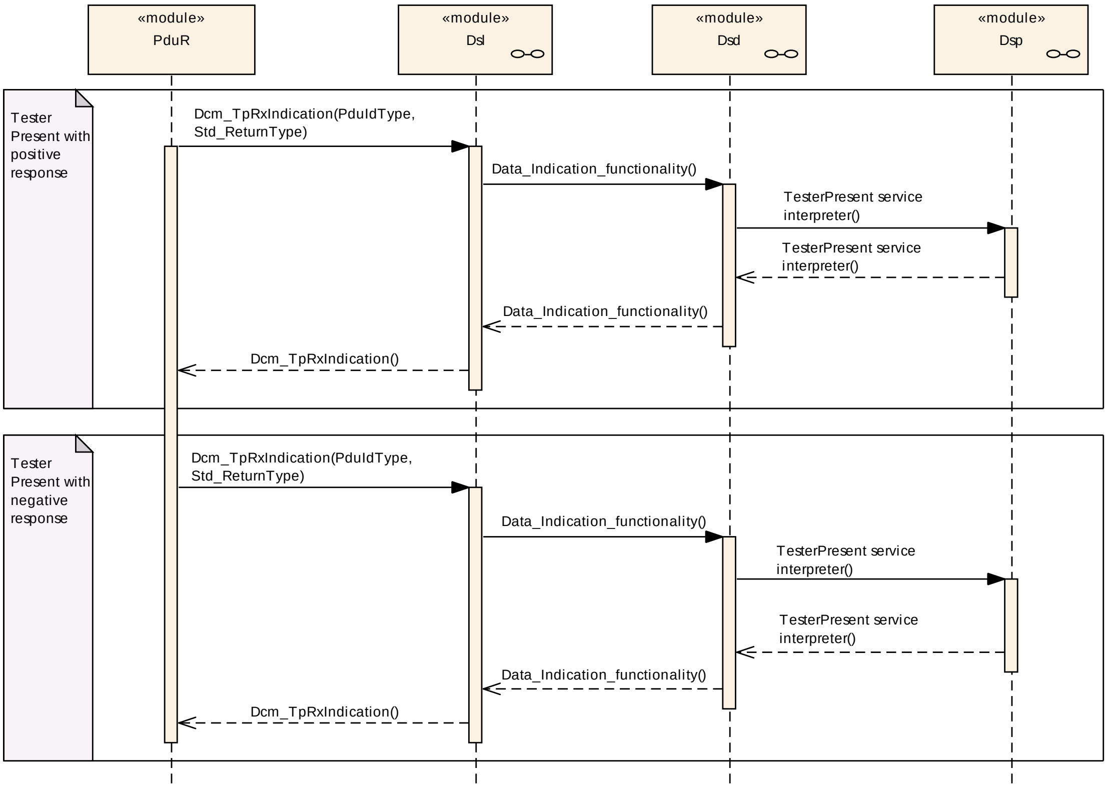

«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
Tester Present with positive response
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
TesterPresent service interpreter()
TesterPresent service interpreter()
Data_Indication_functionality()
Dcm_TpRxIndication()
Tester Present with negative response
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
TesterPresent service interpreter()
TesterPresent service interpreter()
Data_Indication_functionality()
Dcm_TpRxIndication()

Figure 9.18

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 7.3.4.3 Concurrent "TesterPresent keep alive logic").

All the other TesterPresent commands are processed in the following way: On a command TesterPresent the DSD submodule calls the DSP submodule with the function TesterPresent(). The sequence chart also shows the case when an error occurs and a negative response is sent.

9.3.2.3 Process Security Access

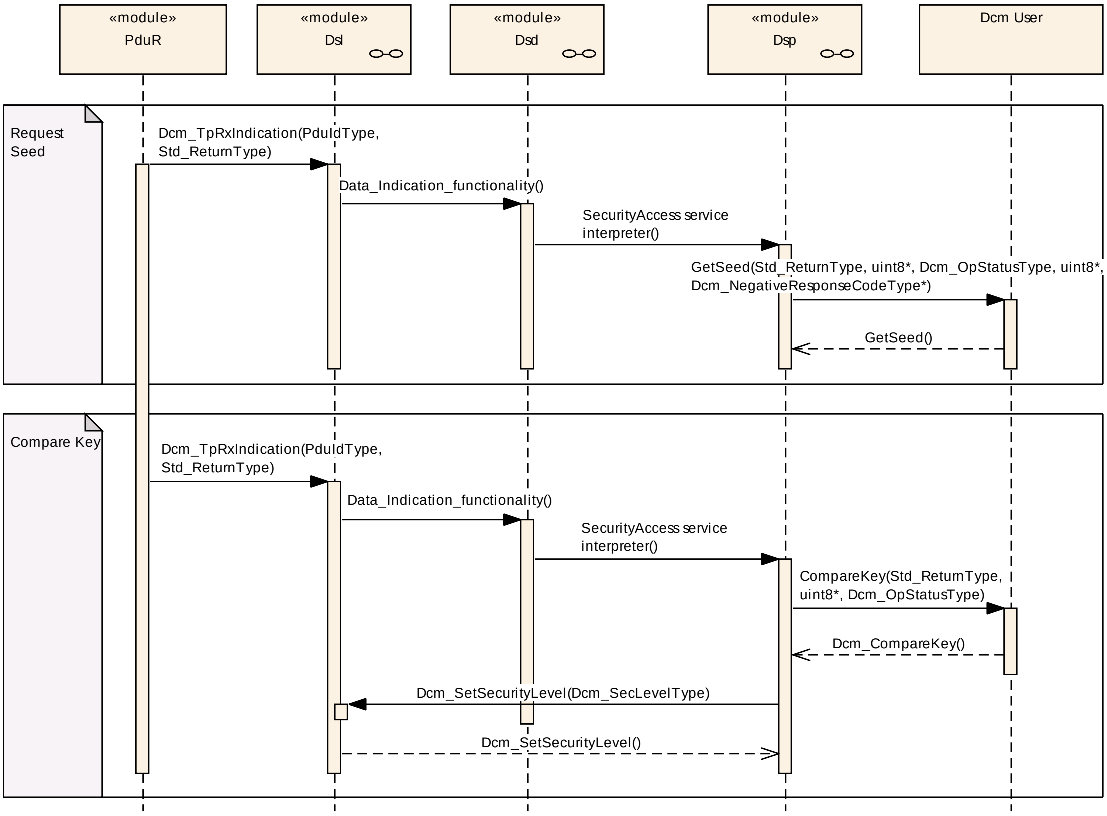

«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
Dcm User
Request Seed
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
SecurityAccess service interpreter()
GetSeed(Std_ReturnType, uint8*, Dcm_OpStatusType, uint8*, Dcm_NegativeResponseCodeType*)
GetSeed()
Compare Key
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
SecurityAccess service interpreter()
CompareKey(Std_ReturnType, uint8*, Dcm_OpStatusType)
Dcm_CompareKey()
Dcm_SetSecurityLevel(Dcm_SecLevelType)
Dcm_SetSecurityLevel()

Figure 9.19

To get the security access, the DSD submodule has to call the DSP submodule to get the seed value from the application. If no error is detected, the seed value is sent in the positive response.

In a second step, the DSP submodule gets the key calculated by the tester and requests the application to compare this key with the internal calculated key. If no error occurs, the new access type is set in the DSL submodule and a positive response is sent.

9.3.2.4 Process ResponseOnEvent OnDtcChange

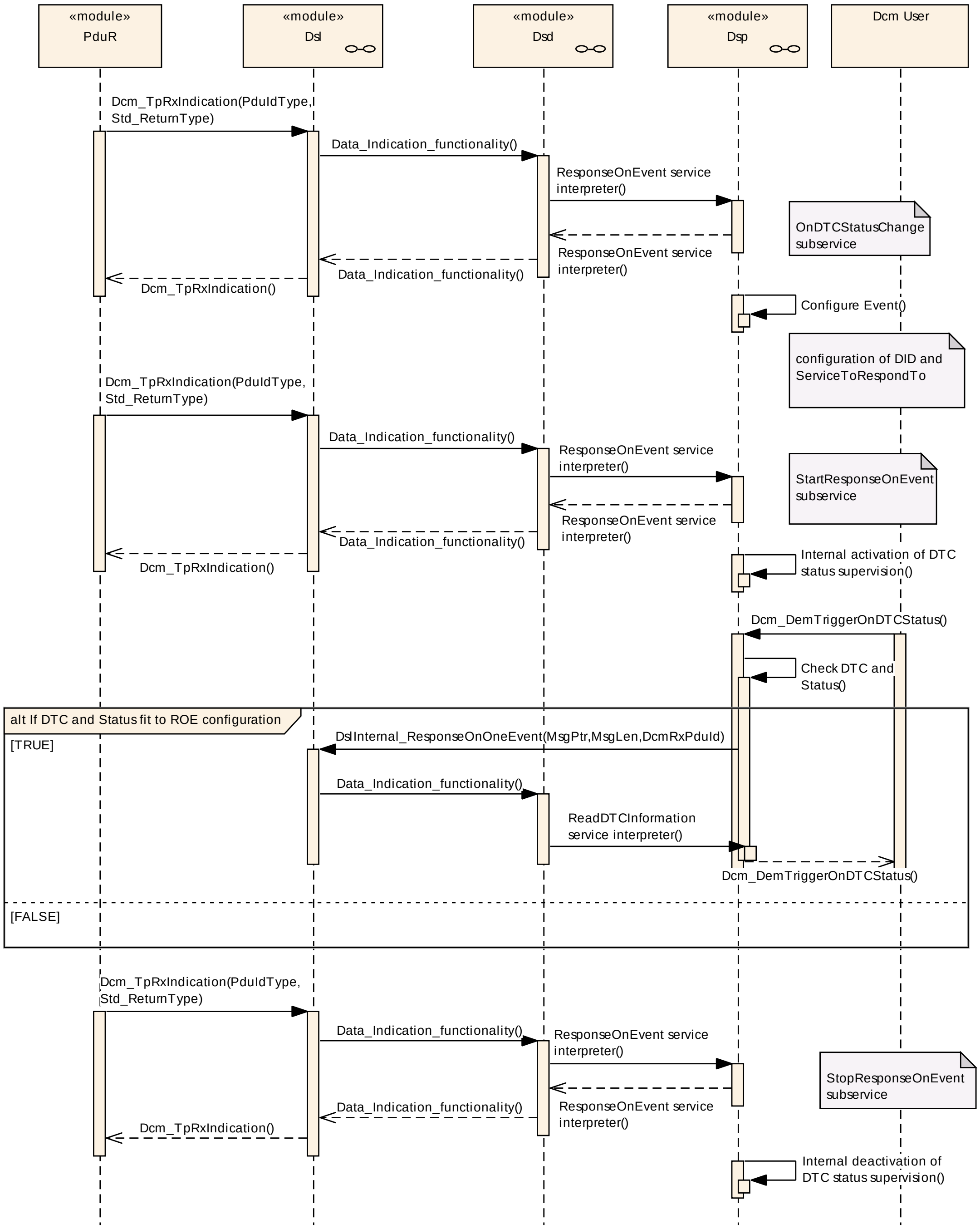


Figure 9.20

Above sequence diagram shows processing of ResponseOnEvent service for subservice OnDtcChange.

After configuration and activation of the event by the service ResponseOnEvent, the Dcm checks the status of the configured DTC on every call to interface Dcm_DemTriggerOnDTCStatus in order to identify if the event shall be trigger. This interface is called by DEM for any DTC status change and independing of the activation/unactivation of ResponseOnEvent.

9.3.2.5 Process ResponseOnEvent OnChangeOfDataIdentifier

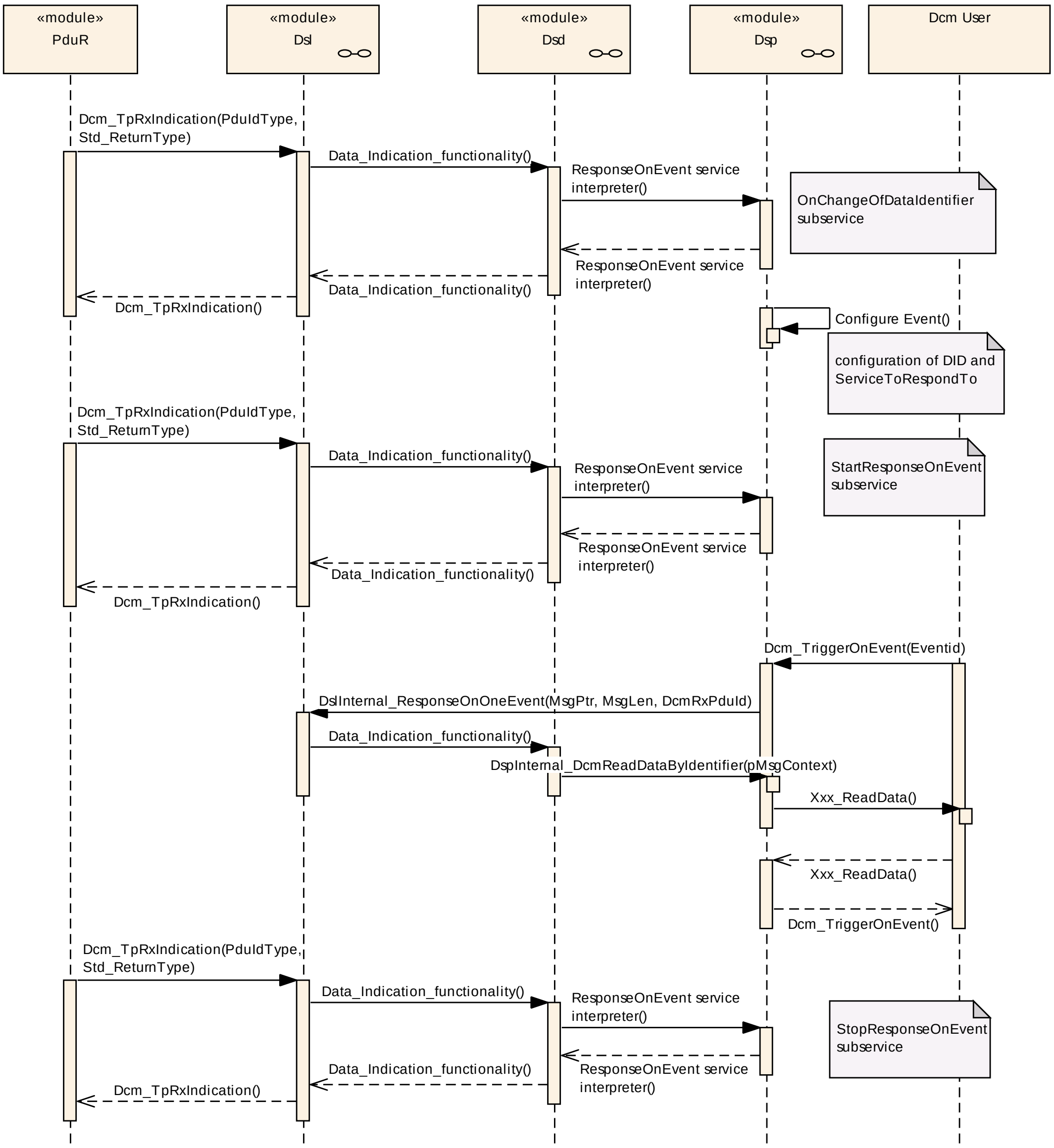

«module»
PduR
«module»
Dsl
«module»
Dsd
«module»
Dsp
Dcm User
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
ResponseOnEvent service interpreter()
OnChangeOfDataIdentifier subservice
ResponseOnEvent service interpreter()
Data_Indication_functionality()
Dcm_TpRxIndication()
Configure Event()
configuration of DID and ServiceToRespondTo
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
ResponseOnEvent service interpreter()
StartResponseOnEvent subservice
ResponseOnEvent service interpreter()
Data_Indication_functionality()
Dcm_TpRxIndication()
Dcm_TriggerOnEvent(Eventid)
DslInternal_ResponseOnOneEvent(MsgPtr, MsgLen, DcmRxPduId)
Data_Indication_functionality()
DspInternal_DcmReadDataByIdentifier(pMsgContext)
Xxx_ReadData()
Xxx_ReadData()
Dcm_TriggerOnEvent()
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
ResponseOnEvent service interpreter()
StopResponseOnEvent subservice
Data_Indication_functionality()
ResponseOnEvent service interpreter()
Dcm_TpRxIndication()

Figure 9.21

Above sequence diagram shows processing of ResponseOnEvent service for subservice OnChangeOfDataIdentifier in the case the event is externally managed (The event can be internally managed, but is not describe in this diagram).

After configuration and etxernal activation of the event by the service ResponseOnEvent, the Dcm wait to be trigger by the external module managing this DID.

9.3.2.6 Process Jump to Bootloader

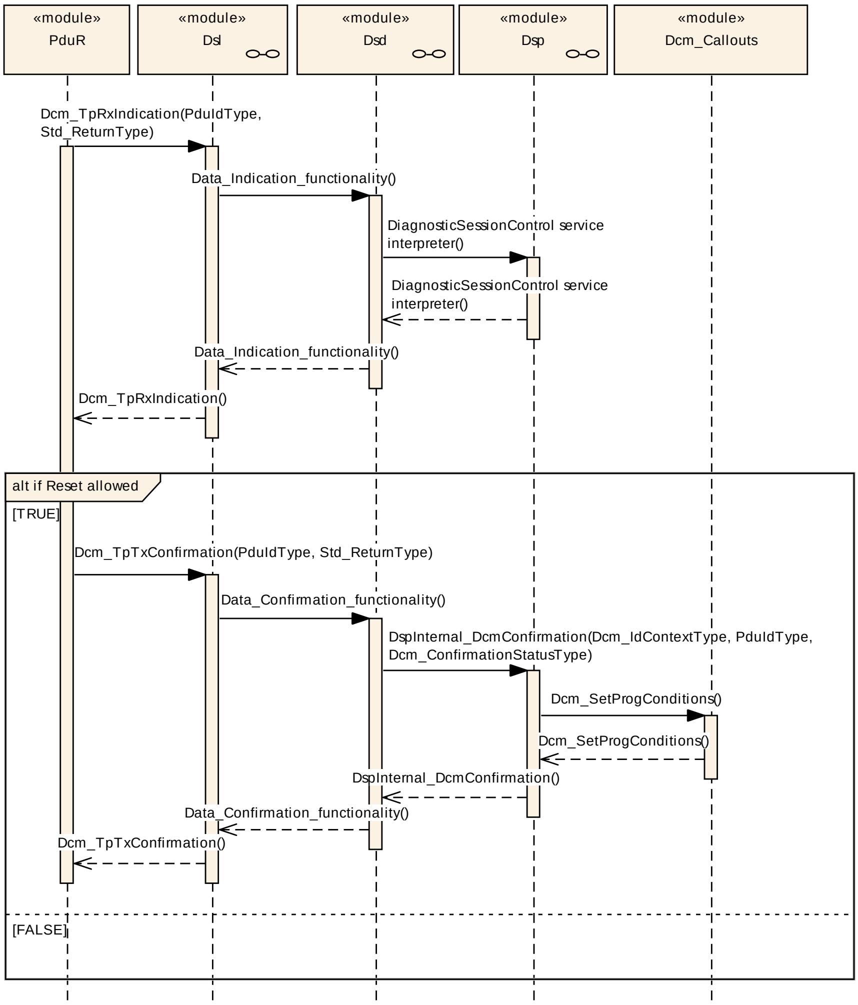

«module» PduR
«module» Dsl
«module» Dsd
«module» Dsp
«module» Dcm_Callouts
Dcm_TpRxIndication(PduIdType, Std_ReturnType)
Data_Indication_functionality()
DiagnosticSessionControl service interpreter()
DiagnosticSessionControl service interpreter()
Data_Indication_functionality()
Dcm_TpRxIndication()
alt if Reset allowed
[TRUE]
Dcm_TpTxConfirmation(PduIdType, Std_ReturnType)
Data_Confirmation_functionality()
DspInternal_DcmConfirmation(Dcm_IdContextType, PduIdType, Dcm_ConfirmationStatusType)
Dcm_SetProgConditions()
Dcm_SetProgConditions()
DspInternal_DcmConfirmation()
Data_Confirmation_functionality()
Dcm_TpTxConfirmation()
[FALSE]

Figure 9.22

Above sequence diagram shows processing of a jump to bootloader on reception of DiagnosticSessionControl. On reception of DiagnosticSessionControl, the Dcm checks if the requested session is configured to trigger a jump to bootloader. In positive case, the Dcm start the jump to bootloader process:

- Transmission of NRC 0x78 (ResponsePending)
- On confirmation of transmission of NRC 0x78, the Dcm calls the callout DcmSetProgConditions to store all information needed for the bootloader

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals.

It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.3 specifies the structure (containers) and the parameters of the module <MODULE_ABBREVIATION>.

Chapter 10.5 specifies published information of the module <MODULE_ABBREVIATION>.

10.1 How to read this chapter

For details refer to the chapter 10.1 "Introduction to configuration specification" in SWS_BSWGeneral [7].

10.2 Configuration constraints

[SWS_Dcm_CONSTR_6000] Harmonize the naming between interfaces and modes ⌈ The shortname of `DcmDspSessionRow` shall match names of Dcm_SesCtrlType and of the mode declarations of DcmDiagnosticSessionControl. The "DCM_" prefix is mandatory for all shortnames. ⌋*()*

[SWS_Dcm_CONSTR_6001] Provide standardized names for ISO standardized diagnostic sessions ⌈The following values of `DcmDspSessionLevel` which represent ISO defined diagnostic sessions shall be used for the shortname of `DcmDspSessionRow`:
1 DCM_DEFAULT_SESSION
2 DCM_PROGRAMMING_SESSION
3 DCM_EXTENDED_DIAGNOSTIC_SESSION
4 DCM_SAFETY_SYSTEM_DIAGNOSTIC_SESSION
⌋*()*

[SWS_Dcm_CONSTR_6048] Composite sub elements accessible only by read ⌈ Composite sub elements can only be referred from Read `DID` i.e. Write and Control `DID` are not supported. ⌋*()*

[SWS_Dcm_CONSTR_6053] ⌈ The aggregation of `DcmDspTextTableMapping` at `DcmDspAlternativeDataType` is only valid if the category of the CompuMethod of the DataType referenced by DcmDspAlternativeDataType.DcmApplicationDataType has category set to TEXTTABLE or SCALE_LINEAR_AND_TEXTTABLE. ⌋*()*

[SWS_Dcm_CONSTR_6051] ⌈ The configuration parameter `DcmDspDidControlMaskSize` shall be only present if `DcmDspDidControlMask` is equal to `DCM_CONTROLMASK_EXTERNAL` or `DCM_CONTROLMASK_INTERNAL`. ⌋*()*

10.3 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

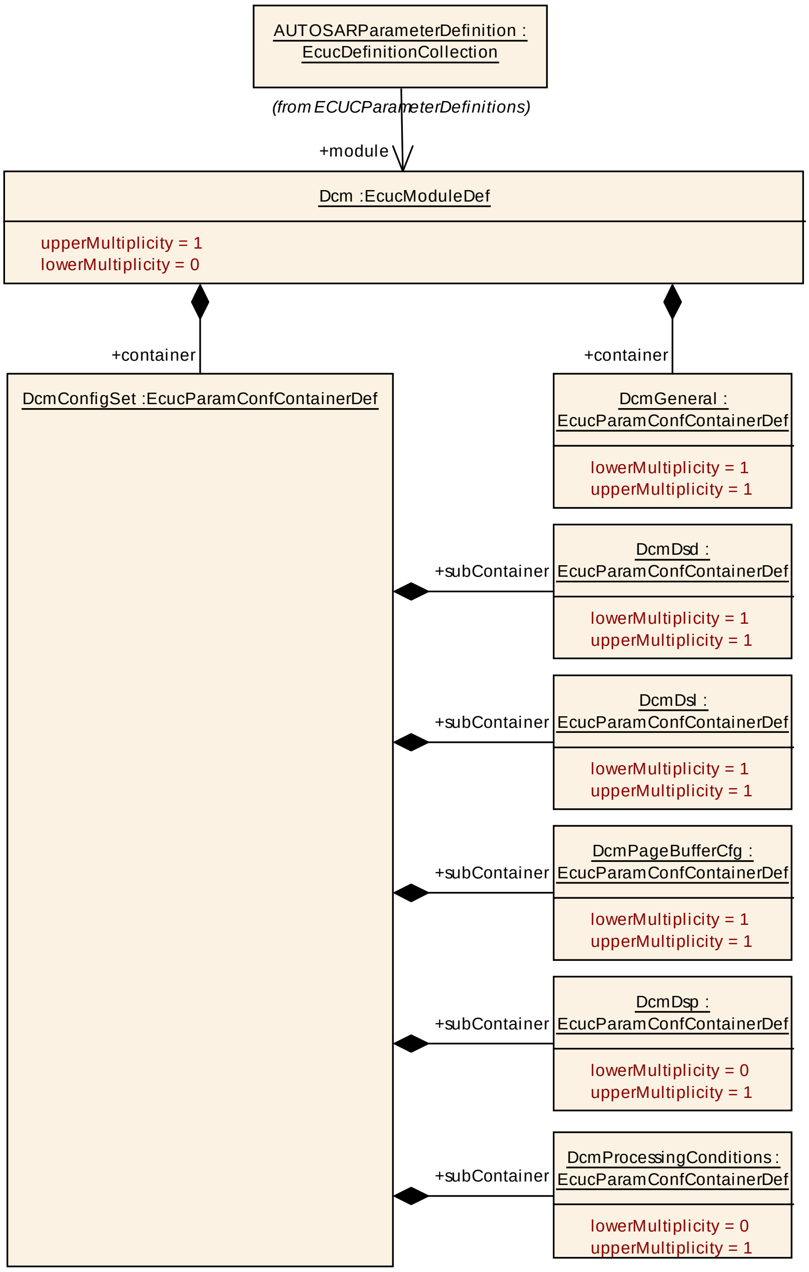


Figure 10.1: Configuration overview

10.3.1 Dcm

Module SWS Item	ECUC_Dcm_01082	
Module Name	Dcm	
Module Description	Configuration of the Dcm (Diagnostic Communications Manager) module.	
Post-Build Variant Support	true	
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE	
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.
DcmGeneral	1	Contains general configuration parameters valid for the entire Dcm module.

10.3.2 DcmConfigSet

SWS Item	[ECUC_Dcm_00819]
Container Name	DcmConfigSet
Description	This container contains the configuration parameters and sub containers of the DCM module supporting multiple configuration sets.
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
DcmDsd	1	These parameters configure the Diagnostic Service Dispatcher submodule.
DcmDsl	1	These parameters configure the Diagnostic Session Layer submodule.
DcmDsp	0..1	These parameters apply to Diagnostic Service Processing. There will always be one set of these parameters per Dcm. Please note: Although the multiplicity is set to 0..1. It can be expected that this container exists in any valid DCM configuration.
DcmPageBufferCfg	1	This container contains the configuration (parameters) for Page Buffer handling
DcmProcessingConditions	0..1	This container contains the configuration for mode arbitration functionality of the Dcm

10.3.2.1 DcmPageBufferCfg

SWS Item	[ECUC_Dcm_00775]
Container Name	DcmPageBufferCfg
Description	This container contains the configuration (parameters) for Page Buffer handling
Configuration Parameters	

Name	DcmPagedBufferEnabled [ECUC_Dcm_00776]		
Parent Container	DcmPageBufferCfg		
Description	Allow to enable or disable the Paged buffer mechanism. true = Paged buffer handling enabled false = Paged Buffer handling disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

10.3.2.2 DcmProcessingConditions

SWS Item	[ECUC_Dcm_00932]
Container Name	DcmProcessingConditions
Description	This container contains the configuration 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 or an environmental conditions which can be used as argument in DcmModeRules. One DcmModeCondition shall contain either one DcmSwcModeRef or one DcmBswModeRef or one DcmSwcSRDataElementRef. 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. In case of a configured DcmSwcModeRef or DcmBswModeRef only the DcmConditionType DCM_EQUALS or DCM_EQUALS_NOT are applicable. In case of DcmSwcSRDataElementRef all literals of DcmConditionType are possible.

DcmModeRule	1..*	This container contains the configuration of a mode rule which represents a logical expression with DcmModeConditions 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

10.3.3 DcmDsd

10.3.3.1 DcmDsd

SWS Item	[ECUC_Dcm_00688]
Container Name	DcmDsd
Description	These parameters configure the Diagnostic Service Dispatcher submodule.
Configuration Parameters	

Name	DcmDsdRequestManufacturerNotificationEnabled [ECUC_Dcm_00783] (Obsolete)		
Parent Container	DcmDsd		
Description	Allows to enable or disable the requested notification mechanism for the Manufacturer. **Tags:** atp.Status=obsolete		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDsdRequestSupplierNotificationEnabled [ECUC_Dcm_00868] (Obsolete)
Parent Container	DcmDsd
Description	Allows to enable or disable the requested notification mechanism for the Supplier. **Tags:** atp.Status=obsolete
Multiplicity	1
Type	EcucBooleanParamDef
Default Value	

Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDsdServiceRequestManufacturerNotification	0..*	The name of this container is used to define the name of the R-Port through which the DCM accesses the interface ServiceRequestNotification. The R-Port is named ServiceRequestManufacturerNotification_{Name} where {Name} is the name of the container DcmDsdServiceRequestManufacturerNotification. The lowerMultiplicity is 0: If container DcmDsdServiceRequestManufacturerNotification does not exist the Indication API is not available.
DcmDsdServiceRequestSupplierNotification	0..*	The name of this container is used to define the name of the R-Port through which the DCM accesses the interface ServiceRequestNotification. The R-Port is named ServiceRequestSupplierNotification_<SWC> where <SWC> is the name of the container DcmDsdServiceRequestSupplierNotification. The lowerMultiplicity is 0: If the container DcmDsdRequestSupplierNotification does not exist the Indication API is not available.
DcmDsdServiceTable	1..256	This container contains the configuration (DSD parameters) for a Service Identifier Table. Note: It is allowed to add OBD services to a DcmDsdServiceTable related to a UDS Protocol. But it is not allowed to add UDS services to a DcmDsdServiceTable related to an OBD Protocol.

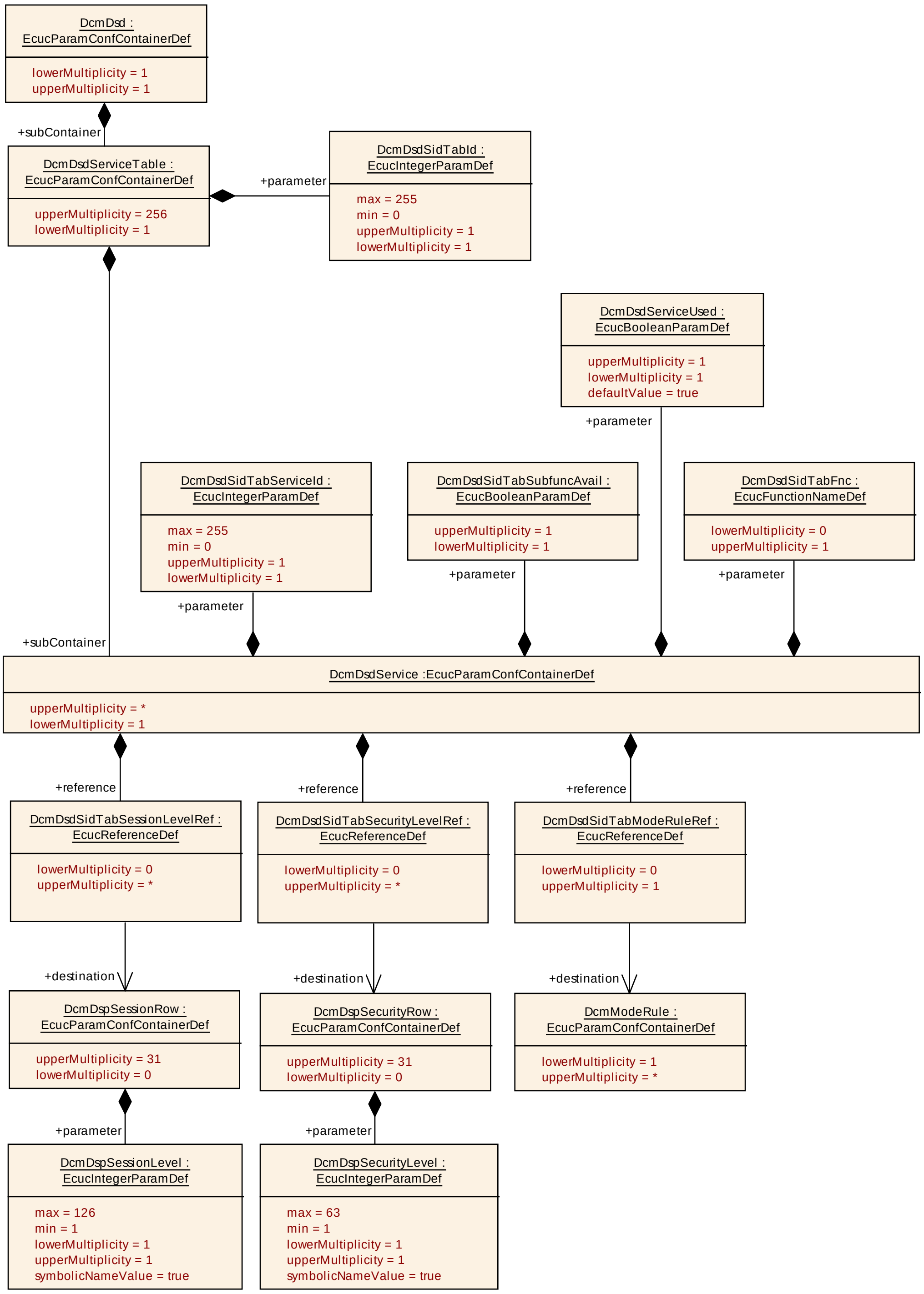


Figure 10.2: DcmDsd configuration overview

10.3.3.2 DcmDsdService

SWS Item	[ECUC_Dcm_00689]
Container Name	DcmDsdService
Description	This container contains the configuration (DSD parameters) for a Service.

Configuration Parameters

Name	DcmDsdServiceUsed [ECUC_Dcm_01044]		
Parent Container	DcmDsdService		
Description	Allows to activate or deactivate the usage of a Service. This parameter can be used for multi-purpose ECUs. true - service is available false - service is not available		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	true		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	DcmDsdSidTabFnc [ECUC_Dcm_00777]		
Parent Container	DcmDsdService		
Description	Callback function of the ECU Supplier specific component for the particular service. The function's prototype is as described for <Module>_<DiagnosticService>. If this parameter is not configured, the service is handled Dcm-internally.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDsdSidTabServiceId [ECUC_Dcm_00735]		
Parent Container	DcmDsdService		
Description	Identifier of the service. The possible service identifiers are defined in ISO 14229-1 and ISO 15031-5.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDsdSidTabSubfuncAvail [ECUC_Dcm_00737]		
Parent Container	DcmDsdService		
Description	Information about whether the service has subfunctions or not. This parameter is used for the handling of the "suppressPosRspMsgIndicationBit" as defined in ISO 14229-1, which can be used as a reference for the configuration. true - service has subfunctions, suppressPosRspMsgIndicationBit is available false - service has no subfunctions, suppressPosRspMsgIndicationBit is not available		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDsdSidTabModeRuleRef [ECUC_Dcm_00918]
Parent Container	DcmDsdService
Description	Reference to a DcmDspModeRule which controls the execution of the service. If there is no reference configured, no mode rule check shall be performed.
Multiplicity	0..1
Type	Reference to DcmModeRule
Post-Build Variant Multiplicity	false

Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDsdSidTabSecurityLevelRef [ECUC_Dcm_00733]		
Parent Container	DcmDsdService		
Description	Reference to a Security Level in which the service is allowed to be executed. Multiple references are allowed for a service. Please refer to ISO 14229-1, ISO 15031-5 and chapter "Verification of the Service Security Access levels." If there is no reference configured, no service security verification shall be performed.		
Multiplicity	0..*		
Type	Reference to DcmDspSecurityRow		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDsdSidTabSessionLevelRef [ECUC_Dcm_00734]		
Parent Container	DcmDsdService		
Description	Reference to a Session Level in which the service is allowed to be executed. Multiple references are allowed for a service. Please refer to ISO 14229-1, ISO 15031-5 and chapter "Verification of the Diagnostic Session". If there is no reference configured, no diagnostic session verification shall be performed.		
Multiplicity	0..*		
Type	Reference to DcmDspSessionRow		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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 a subservice of a service. Only those services may have subservices, which have the DcmDsdSidTabSubfuncAvail configured as TRUE.

10.3.3.3 DcmDsdServiceRequestManufacturerNotification

SWS Item	[ECUC_Dcm_00681]
Container Name	DcmDsdServiceRequestManufacturerNotification
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_{Name} where {Name} is the name of the container DcmDsdServiceRequestManufacturerNotification. The lowerMultiplicity is 0: If container DcmDsdServiceRequestManufacturerNotification does not exist the Indication API is not available.
Configuration Parameters	

No Included Containers

10.3.3.4 DcmDsdServiceRequestSupplierNotification

SWS Item	[ECUC_Dcm_00816]
Container Name	DcmDsdServiceRequestSupplierNotification
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 DcmDsdServiceRequestSupplierNotification. The lowerMultiplicity is 0: If the container DcmDsdRequestSupplierNotification does not exist the Indication API is not available.
Configuration Parameters	

No Included Containers

10.3.3.5 DcmDsdServiceTable

SWS Item	[ECUC_Dcm_00732]
Container Name	DcmDsdServiceTable
Description	This container contains the configuration (DSD parameters) for a Service Identifier Table. Note: It is allowed to add OBD services to a DcmDsdServiceTable related to a UDS Protocol. But it is not allowed to add UDS services to a DcmDsdServiceTable related to an OBD Protocol.
Configuration Parameters	

Name	DcmDsdSidTabId [ECUC_Dcm_00736]		
Parent Container	DcmDsdServiceTable		
Description	Due to using possibly more service tables, the unique DcmDsdSidTabId can be used to identify them.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**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 a Service.

Note : The Dcm internal interaction with the DSP is implementation specific and therefore not explicitly configured.

10.3.3.6 DcmDsdSubService

SWS Item	[ECUC_Dcm_00802]
Container Name	DcmDsdSubService
Description	This container contains the configuration (DSD parameters) for a subservice of a service. Only those services may have subservices, which have the DcmDsdSidTabSubfuncAvail configured as TRUE.
Configuration Parameters	

Name	DcmDsdSubServiceFnc [ECUC_Dcm_00942]		
Parent Container	DcmDsdSubService		
Description	Callback function of the ECU Supplier specific component for the particular service. The function's prototype is as described for <Module>_<DiagnosticService>_<SubService>. If this parameter is not configured, the subservice is handled Dcm-internally.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDsdSubServiceId [ECUC_Dcm_00803]		
Parent Container	DcmDsdSubService		
Description	Identifier of the subservice. The possible subservice identifiers are defined in ISO 14229-1 and ISO 15031-5.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 127		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDsdSubServiceUsed [ECUC_Dcm_01047]		
Parent Container	DcmDsdSubService		
Description	Allows to activate or deactivate the usage of a Subservice. This parameter can be used for multi-purpose ECUs. true - subservice is available false - subservice is not available.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	true		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	DcmDsdSubServiceModeRuleRef [ECUC_Dcm_00924]
Parent Container	DcmDsdSubService
Description	Reference to a DcmDspModeRule which controls the execution of the subservice. If there is no reference configured, no mode rule check shall be performed.
Multiplicity	0..1
Type	Reference to DcmModeRule
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false

Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDsdSubServiceSecurityLevelRef [ECUC_Dcm_00812]		
Parent Container	DcmDsdSubService		
Description	Reference to a Security Level in which the subservice is allowed to be executed. Multiple references are allowed for a subservice. Please refer to ISO 14229-1, ISO 15031-5 and chapter "Verification of the Service Security Access levels." If there is no reference configured, no subservice security verification shall be performed.		
Multiplicity	0..*		
Type	Reference to DcmDspSecurityRow		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDsdSubServiceSessionLevelRef [ECUC_Dcm_00804]
Parent Container	DcmDsdSubService
Description	Reference to a Session Level in which the subservice is allowed to be executed. Multiple references are allowed for a subservice. Please refer to ISO 14229-1, ISO 15031-5 and chapter "Verification of the Diagnostic Session". If there is no reference configured, no diagnostic session verification shall be performed.
Multiplicity	0..*
Type	Reference to DcmDspSessionRow
Post-Build Variant Multiplicity	false

Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.4 DcmDsl

10.3.4.1 DcmDsl

SWS Item	[ECUC_Dcm_00690]
Container Name	DcmDsl
Description	These parameters configure the Diagnostic Session Layer submodule.
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDslBuffer	1..256	This container contains the configuration of a diagnostic buffer.
DcmDslCallbackDCMRequestService	0..*	Each DcmDslCallbackDCMRequestService container defines an R-Port with the CallbackDCMRequestServices interface which the Dcm uses to ask permission for protocol changes from the application software. The R-Port has the name CallbackDCMRequestServices_<SWC> where <SWC> is the name of this container.
DcmDslDiagResp	1	This container contains the configuration of the automatic requestCorrectlyReceivedResponsePending response management in the Dcm.
DcmDslProtocol	1	This container contains the configurations of the diagnostic protocols used in Dcm.

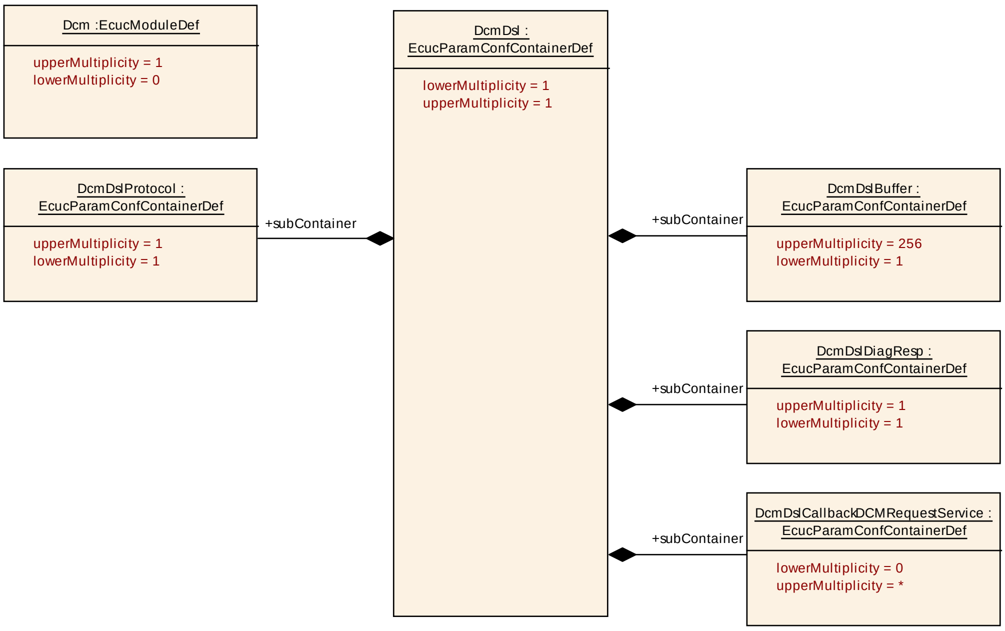


Figure 10.3: DcmDsl configuration overview

10.3.4.2 DcmDslBuffer

SWS Item	[ECUC_Dcm_00739]
Container Name	DcmDslBuffer
Description	This container contains the configuration of a diagnostic buffer.
Configuration Parameters	

Name	DcmDslBufferSize [ECUC_Dcm_00738]		
Parent Container	DcmDslBuffer		
Description	Size of the diagnostic buffer in bytes. For a linear buffer the size shall be as large as the longest diagnostic message (request or response). For a paged buffer the size has impacts on the application performance.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	8 .. 4294967294		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

10.3.4.3 DcmDslCallbackDCMRequestService

SWS Item	[ECUC_Dcm_00679]
Container Name	DcmDslCallbackDCMRequestService
Description	Each DcmDslCallbackDCMRequestService container defines an R-Port with the CallbackDCMRequestServices interface which the Dcm uses to ask permission for protocol changes from the application software. The R-Port has the name CallbackDCMRequestServices_<SWC> where <SWC> is the name of this container.
Configuration Parameters	

No Included Containers

10.3.4.4 DcmDslDiagResp

SWS Item	[ECUC_Dcm_00691]
Container Name	DcmDslDiagResp
Description	This container contains the configuration of the automatic requestCorrectlyReceivedResponsePending response management in the Dcm.
Configuration Parameters	

Name	DcmDslDiagRespMaxNumRespPend [ECUC_Dcm_00693]		
Parent Container	DcmDslDiagResp		
Description	Maximum number of negative responses with response code 0x78 (requestCorrectlyReceivedResponsePending) allowed for a request. If Dcm reaches this limit, an automatic 0x10 (generalReject) final response will be trasmitted and the service processing will be cancelled.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslDiagRespOnSecondDeclinedRequest [ECUC_Dcm_00914]		
Parent Container	DcmDslDiagResp		
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			
Post-Build Variant Value	false		
Value Configuration Class	**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

10.3.4.5 DcmDslProtocol

SWS Item	[ECUC_Dcm_00694]
Container Name	DcmDslProtocol
Description	This container contains the configurations of the diagnostic protocols used in Dcm.
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDslProtocolRow	1..*	This container contains the configuration of one particular diagnostic protocol used in Dcm.

10.3.4.6 DcmDslProtocolRow

SWS Item	[ECUC_Dcm_00695]
Container Name	DcmDslProtocolRow
Description	This container contains the configuration of one particular diagnostic protocol used in Dcm.
Configuration Parameters	

Name	DcmDslProtocolID [ECUC_Dcm_00696] (Obsolete)	
Parent Container	DcmDslProtocolRow	
Description	The diagnostic protocol type for the DCM DSL protocol that is being configured. Implementation Type: Dcm_ProtocolType **Tags:** atp.Status=obsolete	
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_IP	
	DCM_PERIODICTRANS_ON_CAN	
	DCM_PERIODICTRANS_ON_FLEXRAY	
	DCM_PERIODICTRANS_ON_IP	
	DCM_ROE_ON_CAN	
	DCM_ROE_ON_FLEXRAY	
	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	UDS on FlexRay (Manufacturer specific; ISO14229-1)
	DCM_UDS_ON_IP	
	DCM_UDS_ON_LIN	
Post-Build Variant Value	false	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolMaximumResponseSize [ECUC_Dcm_01020]		
Parent Container	DcmDslProtocolRow		
Description	This parameter is mandatory and defines the maximum length of the response message in case DcmPagedBufferEnabled == TRUE		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default Value	4095		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolPreemptTimeout [ECUC_Dcm_00698] (Obsolete)		
Parent Container	DcmDslProtocolRow		

Description	This parameter is the timeout value used in protocol preemption if this protocol preempts another diagnostic protocol. The protocol shall be started maximum DcmDslProtocolPreemptTimeout time after the first request in the new protocol.
	Tags:
	atp.Status=obsolete
	atp.StatusRevisionBegin=4.3.1		

Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 1]		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolPriority [ECUC_Dcm_00699]		
Parent Container	DcmDslProtocolRow		
Description	Protocol priority used during protocol preemption. A higher priority protocol may preempt a lower priority protocol. Lower numeric values represent higher protocol priority: 0 - Highest protocol priority 255 - Lowest protocol priority		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolRowUsed [ECUC_Dcm_01043]		
Parent Container	DcmDslProtocolRow		
Description	Allows to activate or deactivate the usage of a Protocol. This parameter can be used for multi-purpose ECUs. true - protocol is available false - protocol is not available.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	true		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	DcmDslProtocolTransType [ECUC_Dcm_00700]		
Parent Container	DcmDslProtocolRow		
Description	This parameter is used only if the protocol is of type DCM_ROE_ON_xxx. It selects the transmission type of the 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.	
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolType [ECUC_Dcm_01110]	
Parent Container	DcmDslProtocolRow	
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_IP	
	DCM_PERIODICTRANS_ON_CAN	
	DCM_PERIODICTRANS_ON_FLEXRAY	
	DCM_PERIODICTRANS_ON_IP	
	DCM_ROE_ON_CAN	
	DCM_ROE_ON_FLEXRAY	
	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	UDS on FlexRay (Manufacturer specific; ISO14229-1)	
	DCM_UDS_ON_IP		
	DCM_UDS_ON_LIN		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspProtocolEcuAddr [ECUC_Dcm_01081]		
Parent Container	DcmDslProtocolRow		
Description	Ecu source address used for diagnostic communication. This parameter is required for Generic Connections.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmSendRespPendOnRestart [ECUC_Dcm_01114]
Parent Container	DcmDslProtocolRow
Description	If set to TRUE, the Dcm will send a NRC 0x78 before a transition to bootloader or performing an ECU reset. If set to False, no 0x78 is send in this case.
Multiplicity	1
Type	EcucBooleanParamDef
Default Value	true

Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmSendRespPendOnTransToBoot [ECUC_Dcm_00910] (Obsolete)		
Parent Container	DcmDslProtocolRow		
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). **Tags:** atp.Status=obsolete		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value	true		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmTimStrP2ServerAdjust [ECUC_Dcm_00729]		
Parent Container	DcmDslProtocolRow		
Description	This parameter is used to guarantee that the diagnostic response is available on the bus before reaching P2 by adjusting the current DcmDspSessionP2ServerMax. This parameter mainly represents the software architecture dependent communication delay between the time the transmission is initiated by DCM and the time when the message is actually transmitted to the bus. The parameter value is defined in seconds and must be a multiple of DcmTaskTime.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 1]		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Scope / Dependency	scope: ECU

Name	DcmTimStrP2StarServerAdjust [ECUC_Dcm_00728]		
Parent Container	DcmDslProtocolRow		
Description	This parameter is used to guarantee that the diagnostic response is available on the bus before reaching P2Star by adjusting the current DcmDspSessionP2StarServerMax. This parameter mainly represents the software architecture dependent communication delay between the time the transmission is initiated by DCM and the time when the message is actually transmitted to the bus. The parameter value is defined in seconds and must be a multiple of DcmTaskTime.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 5]		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDemClientRef [ECUC_Dcm_01083]		
Parent Container	DcmDslProtocolRow		
Description	Reference to DemClient in Dem configuration. Used by the Dem to distinguish different client calls.		
Multiplicity	1		
Type	Reference to DemClient		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolRxBufferRef [ECUC_Dcm_00701]		
Parent Container	DcmDslProtocolRow		
Description	Reference to a configured diagnostic buffer that is used for diagnostic request reception for the protocol.		
Multiplicity	1		
Type	Reference to DcmDslBuffer		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolSIDTable [ECUC_Dcm_00702]		
Parent Container	DcmDslProtocolRow		
Description	Reference to a service table that is used for diagnostic request processing for this protocol.		
Multiplicity	1		
Type	Reference to DcmDsdServiceTable		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolTxBufferRef [ECUC_Dcm_00704]		
Parent Container	DcmDslProtocolRow		
Description	Reference to a configured diagnostic buffer that is used for diagnostic response transmission for the protocol.		
Multiplicity	1		
Type	Reference to DcmDslBuffer		
Post-Build Variant Value	false		
Value Configuration Class	**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**
DcmDslConnection	1..*	This container contains the configuration of a communication channel for one particular protocol. Note that it is allowed to communicate with multiple testers, therefore multiple connections may be configured for a protocol.

10.3.4.7 DcmDslConnection

SWS Item	[ECUC_Dcm_00705]
Container Name	DcmDslConnection
Description	This container contains the configuration of a communication channel for one particular protocol. Note that it is allowed to communicate with multiple testers, therefore multiple connections may be configured for a protocol.
Configuration Parameters	

Container Choices		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDslMainConnection	0..1	This container contains the configuration for a main connection of a diagnostic protocol. Additionally it may contain references to ROE and Periodic connections if the protocol type or protocol transmission type needs them.
DcmDslPeriodicTransmission	0..1	This container contains the configuration of a periodic transmission connection.
DcmDslResponseOnEvent	0..1	This container contains the configuration of a ResponseOnEvent connection. The PDU referenced by this transmission channel can produce meta data items of type TARGET_ADDRESS_16 and SOURCE_ADDRESS_16.

10.3.4.8 DcmDslMainConnection

SWS Item	[ECUC_Dcm_00706]
Container Name	DcmDslMainConnection
Description	This container contains the configuration for a main connection of a diagnostic protocol. Additionally it may contain references to ROE and Periodic connections if the protocol type or protocol transmission type needs them.
Configuration Parameters	

Name	DcmDslProtocolRxConnectionId [ECUC_Dcm_00826]		
Parent Container	DcmDslMainConnection		
Description	Unique identifier of the tester which uses this connection for diagnostic communication.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolRxTesterSourceAddr [ECUC_Dcm_01115]		
Parent Container	DcmDslMainConnection		
Description	Tester source address uniquely describes a client and will be used e.g within the jump to Bootloader interfaces. This parameter is not required for generic connections (DcmPdus with MetaDataLength >= 1).		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslPeriodicTransmissionConRef [ECUC_Dcm_00707]		
Parent Container	DcmDslMainConnection		
Description	Reference to a periodic transmission connection which is used for the processing of periodic transmission events.		
Multiplicity	0..1		
Type	Reference to DcmDslPeriodicTransmission		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolComMChannelRef [ECUC_Dcm_00952]		
Parent Container	DcmDslMainConnection		
Description	Reference to the ComMChannel on which the DcmDslProtocolRxPdu is received and the DcmDslProtocolTxPdu is transmitted.		
Multiplicity	1		
Type	Symbolic name reference to ComMChannel		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslROEConnectionRef [ECUC_Dcm_00708]		
Parent Container	DcmDslMainConnection		
Description	Reference to a ResponseOnEvent connection which is used for the processing of ResponseOnEvent events.		
Multiplicity	0..1		
Type	Reference to DcmDslResponseOnEvent		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	**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**
DcmDslProtocolRx	1..*	This container contains the configuration parameters of a reception channel in a diagnostic connection. The PDU referenced by this reception channel can consume meta data items of type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16.
DcmDslProtocolTx	0..1	This container contains the configuration parameters of a transmission channel in a diagnostic connection. The PDU referenced by this transmission channel can produce meta data items of type TARGET_ADDRESS_16 and SOURCE_ADDRESS_16.

10.3.4.9 DcmDslProtocolRx

SWS Item	[ECUC_Dcm_00709]
Container Name	DcmDslProtocolRx
Description	This container contains the configuration parameters of a reception channel in a diagnostic connection. The PDU referenced by this reception channel can consume meta data items of type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16.
Configuration Parameters	

Name	DcmDslProtocolRxAddrType [ECUC_Dcm_00710]	
Parent Container	DcmDslProtocolRx	
Description	Selects the addressing type of the reception channel. Physical addressing is used for 1:1 communication, functional addressing is used for 1:N communication. For details refer to ISO 14229-1.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DCM_FUNCTIONAL_TYPE	FUNCTIONAL = 1 to n communication
	DCM_PHYSICAL_TYPE	PHYSICAL = 1 to 1 communications using physical addressing
Post-Build Variant Value	false	

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolRxPduId [ECUC_Dcm_00687]		
Parent Container	DcmDslProtocolRx		
Description	Identifier of the PDU that is used for this reception channel.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolRxPduRef [ECUC_Dcm_00770]		
Parent Container	DcmDslProtocolRx		
Description	Reference to a Pdu in EcuC that is used for this reception channel.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	false		
Value Configuration Class	**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

10.3.4.10 DcmDslProtocolTx

SWS Item	[ECUC_Dcm_00711]
Container Name	DcmDslProtocolTx

Description	This container contains the configuration parameters of a transmission channel in a diagnostic connection. The PDU referenced by this transmission channel can produce meta data items of type TARGET_ADDRESS_16 and SOURCE_ADDRESS_16.
Configuration Parameters	

Name	DcmDslTxConfirmationPduId [ECUC_Dcm_00864]		
Parent Container	DcmDslProtocolTx		
Description	Identifier of the PDU that is used by the lower level module for transmission confirmation of responses on this channel.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslProtocolTxPduRef [ECUC_Dcm_00772]		
Parent Container	DcmDslProtocolTx		
Description	Reference to a Pdu in EcuC that is used for this transmission channel.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	false		
Value Configuration Class	**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

10.3.4.11 DcmDslPeriodicTransmission

SWS Item	[ECUC_Dcm_00741]
Container Name	DcmDslPeriodicTransmission
Description	This container contains the configuration of a periodic transmission connection.

Configuration Parameters

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDslPeriodicConnection	0..*	This container contains the configuration of a transmission channel for a periodic transmission connection. The PDU referenced by this transmission channel can produce meta data items of type TARGET_ADDRESS_16 and SOURCE_ADDRESS_16.

10.3.4.12 DcmDslPeriodicConnection

SWS Item	[ECUC_Dcm_00897]
Container Name	DcmDslPeriodicConnection
Description	This container contains the configuration of a transmission channel for a periodic transmission connection. The PDU referenced by this transmission channel can produce meta data items of type TARGET_ADDRESS_16 and SOURCE_ADDRESS_16.
Configuration Parameters	

Name	DcmDslPeriodicTxConfirmationPduId [ECUC_Dcm_00862]		
Parent Container	DcmDslPeriodicConnection		
Description	Identifier of the PDU that is used by the lower level module for transmission confirmation of responses on this channel.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslPeriodicTxPduRef [ECUC_Dcm_00742]		
Parent Container	DcmDslPeriodicConnection		
Description	Reference to a Pdu in EcuC that is used for this periodic transmission channel.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	false		
Value Configuration Class	**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

10.3.4.13 DcmDslResponseOnEvent

SWS Item	[ECUC_Dcm_00744]
Container Name	DcmDslResponseOnEvent
Description	This container contains the configuration of a ResponseOnEvent connection. The PDU referenced by this transmission channel can produce meta data items of type TARGET_ADDRESS_16 and SOURCE_ADDRESS_16.
Configuration Parameters	

Name	DcmDslRoeTxConfirmationPduId [ECUC_Dcm_00863]		
Parent Container	DcmDslResponseOnEvent		
Description	Identifier of the PDU that is used by the lower level module for transmission confirmation of responses on this connection.		
Multiplicity	0..1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDslRoeTxPduRef [ECUC_Dcm_00743]		
Parent Container	DcmDslResponseOnEvent		
Description	Reference to a Pdu in EcuC that is used for this ResponseOnEvent transmission connection.		
Multiplicity	0..1		
Type	Reference to Pdu		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**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

10.3.5 DcmDsp

SWS Item	[ECUC_Dcm_00712]
Container Name	DcmDsp
Description	These parameters apply to Diagnostic Service Processing. There will always be one set of these parameters per Dcm. Please note: Although the multiplicity is set to 0..1. It can be expected that this container exists in any valid DCM configuration.
Configuration Parameters	

Name	DcmDspDataDefaultEndianness [ECUC_Dcm_00987]	
Parent Container	DcmDsp	
Description	Defines the default endianness belonging to a DID, RID or PID if the corresponding data does not define an endianness.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	BIG_ENDIAN	Most significant byte shall be stored at the lowest address.

	LITTLE_ENDIAN	Most significant byte shall be stored at the highest address	
	OPAQUE	Opaque data endianness	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspDDDIDcheckPerSourceDID [ECUC_Dcm_00966]		
Parent Container	DcmDsp		
Description	Defines the check for session, security and mode dependencies per source DIDs with a ReadDataByIdentifier (0x22). true: Dcm module shall check the session, security and mode dependencies per source DIDs with a ReadDataByIdentifier (0x22) with DID in the range 0xF200 to 0xF3FF false: Dcm module shall not check the session, security and mode dependencies per source DIDs with a ReadDataByIdentifier (0x22) with DID in the range 0xF200 to 0xF3FF		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspEnableObdMirror [ECUC_Dcm_01061]		
Parent Container	DcmDsp		
Description	DcmDspEnableObdMirror defines whether a DID inside the OBD range (F400-F4FF) and the OBD InfoType range (F800-F8FF) shall get the DID value as defined for OBD on reception of the UDS Service ReadDataByIdentifier (0x22), or not. It also defines whether a RID inside the OBD range (E000-E0FF) shall handle the RID as defined for OBD on reception of the UDS Service RoutineControl (0x31), or not.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspMaxDidToRead [ECUC_Dcm_00638]		
Parent Container	DcmDsp		
Description	Indicates the maximum allowed DIDs in a single "ReadDataByIdentifier" request.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspMaxPeriodicDidToRead [ECUC_Dcm_00956]	
Parent Container	DcmDsp	
Description	Indicates the maximum allowed periodicDIDs which can be read in a single "ReadDataByPeriodicIdentifier" request.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default Value		

Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspPowerDownTime [ECUC_Dcm_00818]		
Parent Container	DcmDsp		
Description	This parameter indicates to the client the minimum time of the stand-by sequence the server will remain in the power-down sequence. The resolution of this parameter is one second per count. The following values are valid: 00 - FE hex: 0 - 254 s powerDownTime; FF hex: indicates a failure or time not available. This value needs to be defined by the integrator according to the ECU capabilities. This parameter has to be available if the service EcuReset, sub-service enableRapidPowerShutDown is configured.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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**
DcmDspClearDTC	0..1	This container contains the configuration for the Clear DTC service.

DcmDspComControl	0..1	Provides the configuration of the CommunicationControl mechanism.
DcmDspCommon Authorization	0..*	This container contains the configuration (parameters) for the common Authorization being equal for several services / sub-services.
DcmDspControlDTC Setting	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 one 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
DcmDspEcuReset	0..1	This container contains the configuration for DcmDspEcuReset service
DcmDspMemory	0..1	This container contains the configuration of the memory access.
DcmDspPeriodic Transmission	0..1	This container contains the configuration (parameters) for Periodic Transmission Scheduler.
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.
DcmDspRequestFile Transfer	0..1	This container contains the configuration for RequestFileTransfer. This container only exists if RequestFileTransfer is configured.
DcmDspRoe	0..1	Provide the configuration of the ResponseOnEvent mechanism.
DcmDspRoutine	0..*	This container contains the configuration (parameters) for Routines
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	Parent container holding single rows to configure particular sessions
DcmDspVehInfo	0..*	This container contains the configuration (parameters) for one single VehicleInfoType of service $09

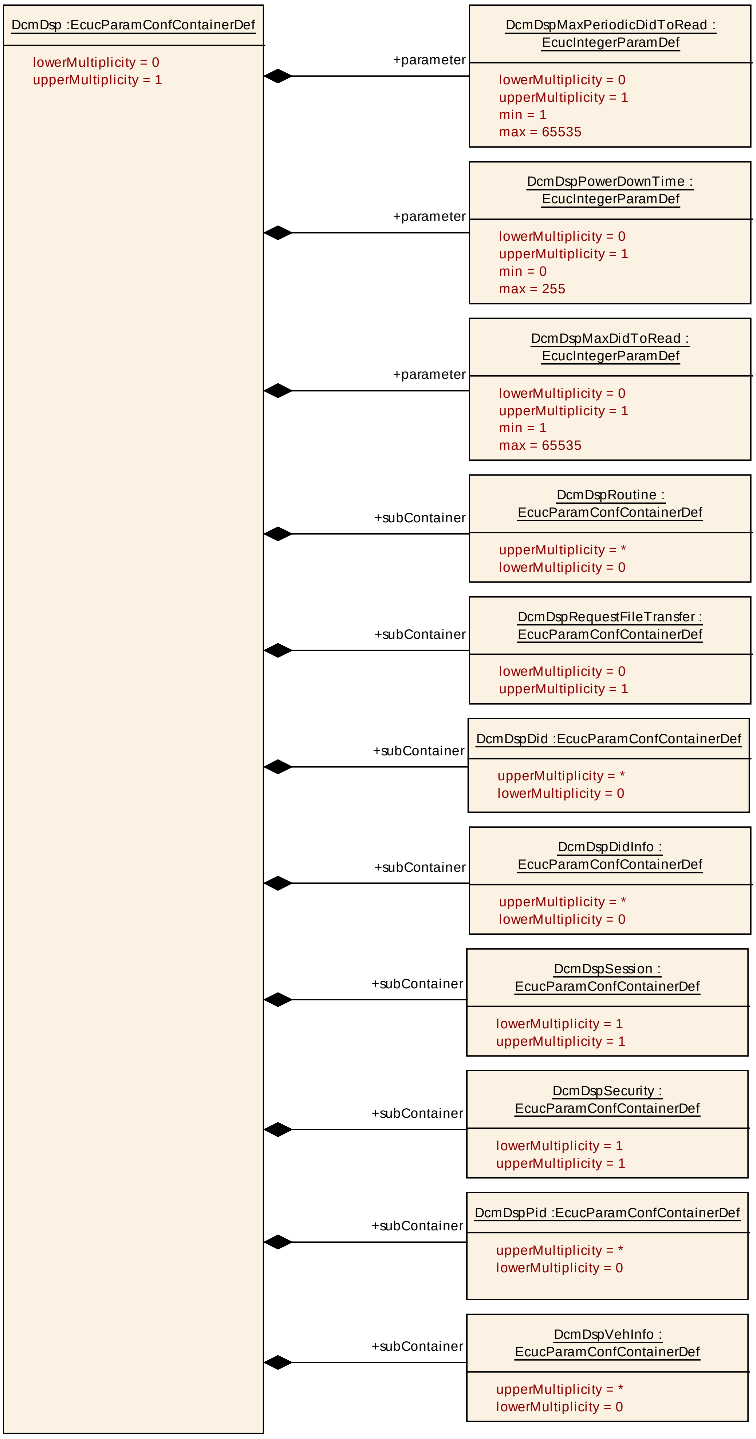


Figure 10.4: DcmDsp configuration overview Part1

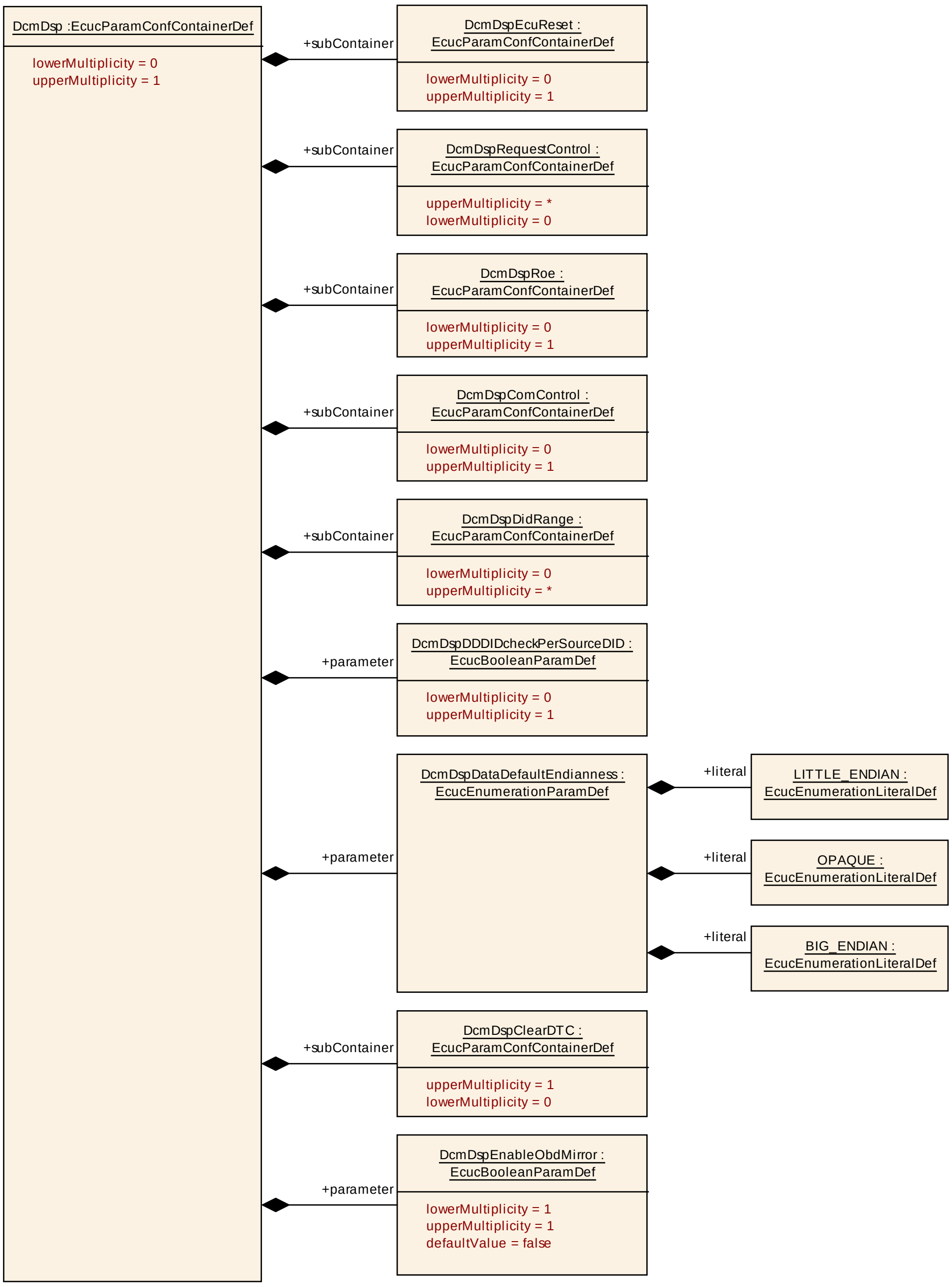

DcmDsp :EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
DcmDspEcuReset : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
DcmDspRequestControl : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 0
+subContainer
DcmDspRoe : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
DcmDspComControl : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
DcmDspDidRange : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+parameter
DcmDspDDDIDcheckPerSourceDID : EcucBooleanParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
DcmDspDataDefaultEndianness : EcucEnumerationParamDef
+literal
LITTLE_ENDIAN : EcucEnumerationLiteralDef
+literal
OPAQUE : EcucEnumerationLiteralDef
+literal
BIG_ENDIAN : EcucEnumerationLiteralDef
+subContainer
DcmDspClearDTC : EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 0
+parameter
DcmDspEnableObdMirror : EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
defaultValue = false

Figure 10.5: DcmDsp configuration overview Part2

10.3.5.1 Communication Control

10.3.5.1.1 DcmDspComControl

SWS Item	[ECUC_Dcm_00900]
Container Name	DcmDspComControl
Description	Provides the configuration of the CommunicationControl mechanism.
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.
DcmDspComControlSubNode	0..65535	This container gives information about the node identification number and the ComM channel used to address a sub-network.

10.3.5.1.2 DcmDspComControlAllChannel

SWS Item	[ECUC_Dcm_00901]
Container Name	DcmDspComControlAllChannel
Description	Collection of ComM channels which shall be controlled if all networks are addressed.
Configuration Parameters	

Name	DcmDspComControlAllChannelUsed [ECUC_Dcm_01045]		
Parent Container	DcmDspComControlAllChannel		
Description	Allow to activate or deactivate the usage of a ComM channel collection to be controlled, for multi purpose ECUs true = ComM channel collection used false = ComM channel collection not used		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	true		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	DcmDspAllComMChannelRef [ECUC_Dcm_00902]
Parent Container	DcmDspComControlAllChannel
Description	Reference to ComM channel.
Multiplicity	1
Type	Symbolic name reference to ComMChannel
Post-Build Variant Value	true

Value Configuration Class	**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

10.3.5.1.3 DcmDspComControlSetting

SWS Item	[ECUC_Dcm_00943]
Container Name	DcmDspComControlSetting
Description	Provide the configuration of the Communication control.
Configuration Parameters	

Name	DcmDspComControlCommunicationReEnableModeRuleRef [ECUC_Dcm_00944]		
Parent Container	DcmDspComControlSetting		
Description	Reference to DcmModeRule Mode rule which controls re-enabling of communication by DCM. [ref. SWS_Dcm_00753]		
Multiplicity	0..1		
Type	Reference to DcmModeRule		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.5.1.4 DcmDspComControlSpecificChannel

SWS Item	[ECUC_Dcm_00903]
Container Name	DcmDspComControlSpecificChannel
Description	Assigns subnet number to ComM channel which will be controlled.

Configuration Parameters

Name	DcmDspComControlSpecificChannelUsed [ECUC_Dcm_01046]		
Parent Container	DcmDspComControlSpecificChannel		
Description	Allow to activate or deactivate the usage of a Subnet assigned to the ComM channel which will be controlled, for multi purpose ECUs. true = Subnet used false = Subnet not used		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	true		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	DcmDspSubnetNumber [ECUC_Dcm_00905]		
Parent Container	DcmDspComControlSpecificChannel		
Description	Subnet Number which controls the specific ComMChannel.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 14		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspSpecificComMChannelRef [ECUC_Dcm_00904]		
Parent Container	DcmDspComControlSpecificChannel		
Description	Reference to ComM channel.		
Multiplicity	1		
Type	Symbolic name reference to ComMChannel		
Post-Build Variant Value	true		
Value Configuration Class	**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

10.3.5.1.5 DcmDspComControlSubNode

SWS Item	[ECUC_Dcm_01033]
Container Name	DcmDspComControlSubNode
Description	This container gives information about the node identification number and the ComM channel used to address a sub-network.
Configuration Parameters	

Name	DcmDspComControlSubNodeId [ECUC_Dcm_01031]		
Parent Container	DcmDspComControlSubNode		
Description	The node identification number DcmDspComControlSubNodeId is addressed by the CommunicationControl (0x28) request.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspComControlSubNodeUsed [ECUC_Dcm_01032]		
Parent Container	DcmDspComControlSubNode		
Description	This parameter determines if a node control function is available or not.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	true		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	DcmDspComControlSubNodeComMChannelRef [ECUC_Dcm_01030]		
Parent Container	DcmDspComControlSubNode		
Description	This parameter references a ComM channel where this node is connected to.		
Multiplicity	1		
Type	Symbolic name reference to ComMChannel		
Post-Build Variant Value	true		
Value Configuration Class	**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

10.3.5.2 DcmDspCommonAuthorization

SWS Item	[ECUC_Dcm_01025]
Container Name	DcmDspCommonAuthorization
Description	This container contains the configuration (parameters) for the common Authorization being equal for several services / sub-services.
Configuration Parameters	

Name	DcmDspCommonAuthorizationModeRuleRef [ECUC_Dcm_01028]		
Parent Container	DcmDspCommonAuthorization		
Description	Reference to DcmModeRule Mode rule which controls this service/ sub-service. If there is no reference, no check of the mode rule shall be done.		
Multiplicity	0..1		
Type	Reference to DcmModeRule		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspCommonAuthorizationSecurityLevelRef [ECUC_Dcm_01026]		
Parent Container	DcmDspCommonAuthorization		
Description	Reference to DcmDspSecurityRow Security levels allowed to control this service/ sub-service. If there is no reference, no check of security level shall be done.		
Multiplicity	0..*		
Type	Reference to DcmDspSecurityRow		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspCommonAuthorizationSessionRef [ECUC_Dcm_01027]		
Parent Container	DcmDspCommonAuthorization		
Description	Reference to DcmDspSessionRow Sessions allowed to control this service/ sub-service. If there is no reference, no check of session level shall be done.		
Multiplicity	0..*		
Type	Reference to DcmDspSessionRow		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.5.3 DIDs

10.3.5.3.1 DcmDspDid

SWS Item	[ECUC_Dcm_00601]
Container Name	DcmDspDid
Description	This container contains the configuration (parameters) of the DID.
Configuration Parameters	

Name	DcmDspDidIdentifier [ECUC_Dcm_00602]		
Parent Container	DcmDspDid		
Description	2 byte Identifier of the DID Within each DcmConfigSet all DcmDspDidIdentifier values shall be unique.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidSize [ECUC_Dcm_01099]		
Parent Container	DcmDspDid		
Description	Length of a DID in byte(s).		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidUsed [ECUC_Dcm_00805]
Parent Container	DcmDspDid
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	

Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	DcmDspDidInfoRef [ECUC_Dcm_00604]		
Parent Container	DcmDspDid		
Description	Reference to DcmDspDidInfo containing information on this DID.		
Multiplicity	1		
Type	Reference to DcmDspDidInfo		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidRef [ECUC_Dcm_00606]		
Parent Container	DcmDspDid		
Description	Reference to DcmDspDid in case this DID refer to one or serveral other DID's **Attributes:** requiresIndex=true		
Multiplicity	0..*		
Type	Reference to DcmDspDid		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspDidSignal	0..*	This container defines the reference to 1 DcmDspData container and position relevant for this DID.

DcmDspDidSupportInfo	0..1	This container defines the support information to declare the usability of the data bytes within the DIDs

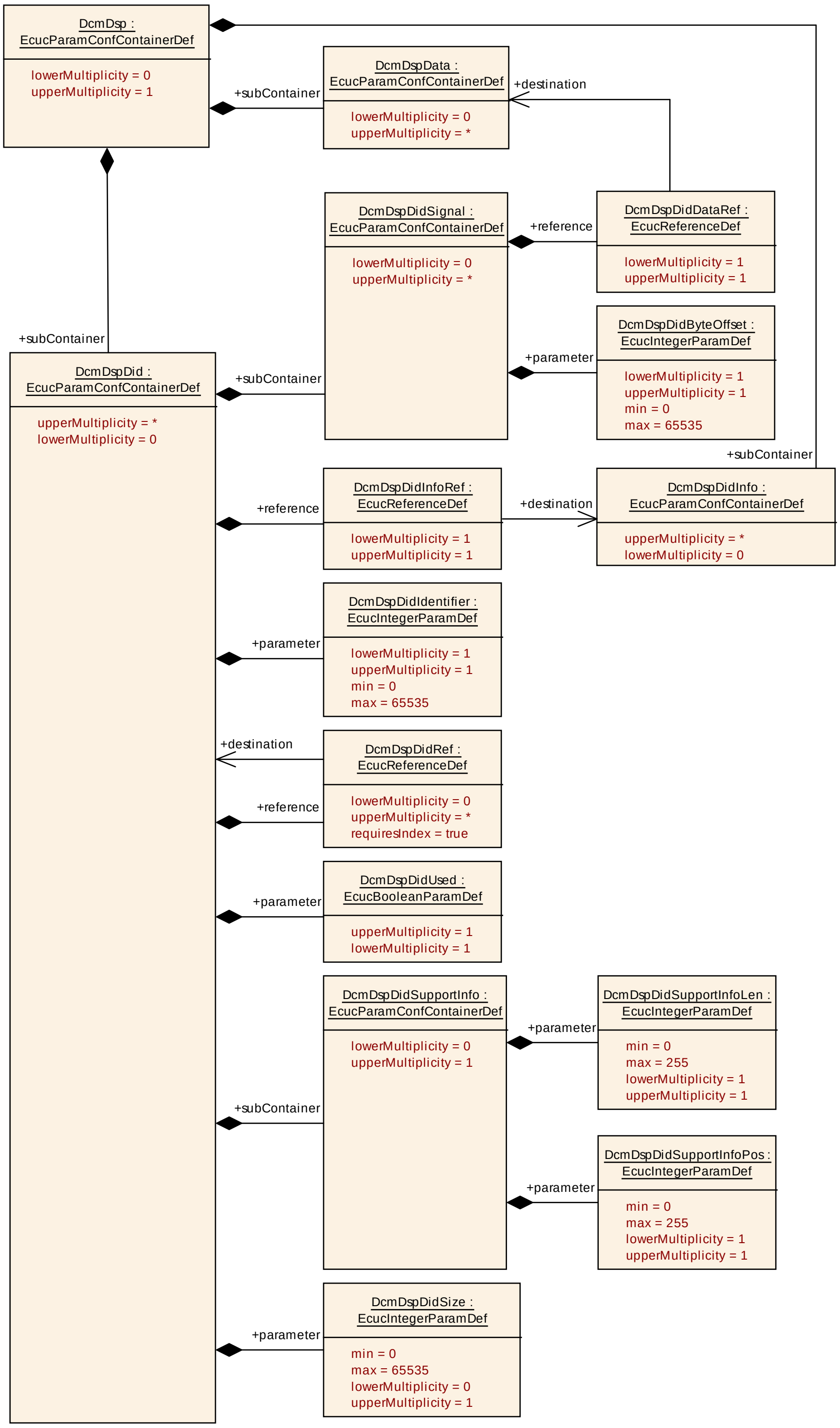


Figure 10.6: DcmDspDid configuration overview

10.3.5.3.2 DcmDspDidInfo

SWS Item	[ECUC_Dcm_00607]
Container Name	DcmDspDidInfo
Description	This container contains the configuration (parameters) of the DID's Info
Configuration Parameters	

Name	DcmDspDDDIDMaxElements [ECUC_Dcm_00970]		
Parent Container	DcmDspDidInfo		
Description	Maximum number of source elements of a DDDID.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspDidDynamicallyDefined [ECUC_Dcm_00612]		
Parent Container	DcmDspDidInfo		
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			
Post-Build Variant Value	false		
Value Configuration Class	**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**
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.

10.3.5.3.3 DcmDspDidRead

SWS Item	[ECUC_Dcm_00613]
Container Name	DcmDspDidRead
Description	This container contains the configuration (parameters) of the DID read.
Configuration Parameters	

Name	DcmDspDidReadModeRuleRef [ECUC_Dcm_00917]		
Parent Container	DcmDspDidRead		
Description	Reference to DcmModeRule 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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidReadSecurityLevelRef [ECUC_Dcm_00614]
Parent Container	DcmDspDidRead
Description	Reference to DcmDspSecurityRow Referenced security levels are allowed to read this DID. If there is no reference, no check of security level shall be done.
Multiplicity	0..*
Type	Reference to DcmDspSecurityRow
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false

Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidReadSessionRef [ECUC_Dcm_00615]		
Parent Container	DcmDspDidRead		
Description	Reference to DcmDspSessionRow Referenced sessions are allowed to read this DID. If there is no reference, no check of session level shall be done.		
Multiplicity	0..*		
Type	Reference to DcmDspSessionRow		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.5.3.4 DcmDspDidSignal

SWS Item	[ECUC_Dcm_00813]
Container Name	DcmDspDidSignal
Description	This container defines the reference to 1 DcmDspData container and position relevant for this DID.
Configuration Parameters	

Name	DcmDspDidByteOffset [ECUC_Dcm_01105]		
Parent Container	DcmDspDidSignal		
Description	Defines the absolute byte offset of the data defined by DcmDspDidDataRef reference to DcmDspData container in the DID.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidDataRef [ECUC_Dcm_00808]		
Parent Container	DcmDspDidSignal		
Description	Reference to 1 DcmDspData container relevant for this DID.		
Multiplicity	1		
Type	Reference to DcmDspData		
Post-Build Variant Value	false		
Value Configuration Class	**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

10.3.5.3.5 DcmDspDidSupportInfo

SWS Item	[ECUC_Dcm_01102]
Container Name	DcmDspDidSupportInfo
Description	This container defines the support information to declare the usability of the data bytes within the DIDs
Configuration Parameters	

Name	DcmDspDidSupportInfoLen [ECUC_Dcm_01103]	
Parent Container	DcmDspDidSupportInfo	
Description	Length of the support information in bytes.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	

Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidSupportInfoPos [ECUC_Dcm_01100]		
Parent Container	DcmDspDidSupportInfo		
Description	Length of the support information in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**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

10.3.5.3.6 DcmDspDidRange

SWS Item	[ECUC_Dcm_00937]
Container Name	DcmDspDidRange
Description	This container defines the DID Range
Configuration Parameters	

Name	DcmDspDidRangeHasGaps [ECUC_Dcm_00941]		
Parent Container	DcmDspDidRange		
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			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Scope / Dependency	scope: ECU

Name	DcmDspDidRangeIdentifierLowerLimit [ECUC_Dcm_00938]		
Parent Container	DcmDspDidRange		
Description	Lower limit of DID range.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidRangeIdentifierUpperLimit [ECUC_Dcm_00939]		
Parent Container	DcmDspDidRange		
Description	Upper limit of DID range.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidRangeIsDidAvailableFnc [ECUC_Dcm_00946]
Parent Container	DcmDspDidRange
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	
Regular Expression	
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidRangeMaxDataLength [ECUC_Dcm_00940]		
Parent Container	DcmDspDidRange		
Description	Maximum data length in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidRangeReadDataLengthFnc [ECUC_Dcm_01067]		
Parent Container	DcmDspDidRange		
Description	Function name to request from application the length of the data of a range DID. Only relevant if DcmDspDidRangeUsePort is set to false. This parameter is related to the interface Xxx_ReadDidRangeDataLength.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidRangeReadDidFnc [ECUC_Dcm_00947]		
Parent Container	DcmDspDidRange		
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			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidRangeUsePort [ECUC_Dcm_00945]		
Parent Container	DcmDspDidRange		
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			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidRangeWriteDidFnc [ECUC_Dcm_00948]		
Parent Container	DcmDspDidRange		
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			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidRangeInfoRef [ECUC_Dcm_00950]		
Parent Container	DcmDspDidRange		
Description	Reference to DcmDspDidInfo containing information on this DID Range.		
Multiplicity	1		
Type	Reference to DcmDspDidInfo		
Post-Build Variant Value	false		
Value Configuration Class	**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

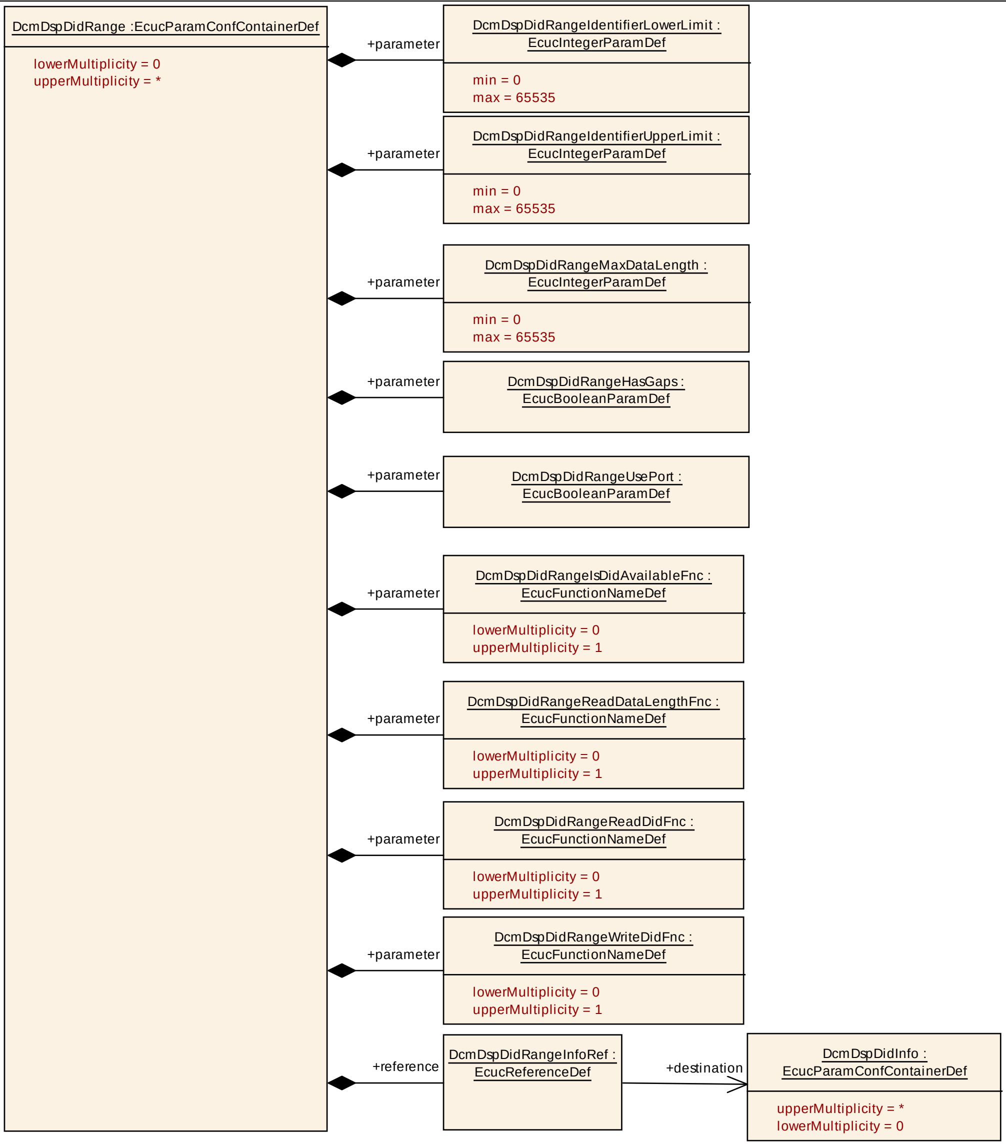

DcmDspDidRange :EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+parameter
DcmDspDidRangeIdentifierLowerLimit : EcucIntegerParamDef
min = 0
max = 65535
+parameter
DcmDspDidRangeIdentifierUpperLimit : EcucIntegerParamDef
min = 0
max = 65535
+parameter
DcmDspDidRangeMaxDataLength : EcucIntegerParamDef
min = 0
max = 65535
+parameter
DcmDspDidRangeHasGaps : EcucBooleanParamDef
+parameter
DcmDspDidRangeUsePort : EcucBooleanParamDef
+parameter
DcmDspDidRangeIsDidAvailableFnc : EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
DcmDspDidRangeReadDataLengthFnc : EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
DcmDspDidRangeReadDidFnc : EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
DcmDspDidRangeWriteDidFnc : EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
+reference
DcmDspDidRangeInfoRef : EcucReferenceDef
+destination
DcmDspDidInfo : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 0

Figure 10.7: DcmDspDidRange configuration overview

10.3.5.3.7 DcmDspDidWrite

SWS Item	[ECUC_Dcm_00616]
Container Name	DcmDspDidWrite
Description	This container contains the configuration (parameters) of the DID write.
Configuration Parameters	

Name	DcmDspDidWriteModeRuleRef [ECUC_Dcm_00922]		
Parent Container	DcmDspDidWrite		
Description	Reference to DcmModeRule 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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidWriteSecurityLevelRef [ECUC_Dcm_00617]		
Parent Container	DcmDspDidWrite		
Description	Reference to DcmDspSecurityRow Referenced security levels are allowed to wirte this DID. If there is no reference, no check of security level shall be done.		
Multiplicity	0..*		
Type	Reference to DcmDspSecurityRow		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidWriteSessionRef [ECUC_Dcm_00618]		
Parent Container	DcmDspDidWrite		
Description	Reference to DcmDspSessionRow Referenced sessions are allowed to write this DID. If there is no reference, no check of session level shall be done.		
Multiplicity	0..*		
Type	Reference to DcmDspSessionRow		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.5.4 DcmDspControlDTCSetting

SWS Item	[ECUC_Dcm_00935]
Container Name	DcmDspControlDTCSetting
Description	Provide the configuration of the ControlDTCSetting mechanism.
Configuration Parameters	

Name	DcmSupportDTCSettingControlOptionRecord [ECUC_Dcm_00965]		
Parent Container	DcmDspControlDTCSetting		
Description	This configuration switch defines if the DTCSettingControlOptionRecord is in general supported in the request message or not.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspControlDTCSettingReEnableModeRuleRef [ECUC_Dcm_00936]		
Parent Container	DcmDspControlDTCSetting		
Description	Reference to DcmModeRule 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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.5.5 Data elements

10.3.5.5.1 DcmDspData

SWS Item	[ECUC_Dcm_00869]
Container Name	DcmDspData
Description	This container contains the configuration (parameters) of a Data belonging to a DID
Configuration Parameters	

Name	DcmDspDataByteSize [ECUC_Dcm_01106]		
Parent Container	DcmDspData		
Description	Defines the array length in bytes or the the maximum array length for variable datalengths.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataConditionCheckReadFnc [ECUC_Dcm_00677]		
Parent Container	DcmDspData		
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. This parameter is related to the interface Xxx_ConditionCheckRead.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataConditionCheckReadFncUsed [ECUC_Dcm_00955]		
Parent Container	DcmDspData		
Description	This parameter determines if a condition check function is available or not.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspDataEcuSignal [ECUC_Dcm_00825]		
Parent Container	DcmDspData		
Description	Function name to control the access to a certain ECU Signal by the DCM. (IoHwAb_Dcm_<symbolic name of ECU signal>-function).		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataEndianness [ECUC_Dcm_00986]		
Parent Container	DcmDspData		
Description	Defines the endianness of the data belonging to a DID in a diagnostic request or response message.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN	Most significant byte shall be stored at the lowest address.	
	LITTLE_ENDIAN	Most significant byte shall be stored at the highest address.	
	OPAQUE	Opaque data endianness	
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspDataFreezeCurrentStateFnc [ECUC_Dcm_00674]		
Parent Container	DcmDspData		
Description	Function name to request to application to freeze the current state of an IOControl. (FreezeCurrentState-function). This parameter is related to the interface Xxx_FreezeCurrentState.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataGetScalingInfoFnc [ECUC_Dcm_00676]		
Parent Container	DcmDspData		
Description	Function name to request to application the scaling information of the DID. (GetScalingInformation-function). This parameter is related to the interface Xxxx_GetScalingInformation.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataReadDataLengthFnc [ECUC_Dcm_00671]		
Parent Container	DcmDspData		
Description	Function name to request from application the data length of a DID. (ReadDataLength-function). This parameter is related to the interface Xxx_ReadDataLength.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataReadEcuSignal [ECUC_Dcm_00824]		
Parent Container	DcmDspData		
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			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataReadFnc [ECUC_Dcm_00669]		
Parent Container	DcmDspData		
Description	Function name to request from application the data value of a DID. (ReadData-function). This parameter is related to the interface Xxx_ReadData.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspDataResetToDefaultFnc [ECUC_Dcm_00673]		
Parent Container	DcmDspData		
Description	Function name to request to application to reset an IOControl to default value. (ResetToDefault-function). This parameter is related to the interface Xxx_ResetToDefault.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataReturnControlToEcuFnc [ECUC_Dcm_00672]		
Parent Container	DcmDspData		
Description	Function name to request to application to return control to ECU of an IOControl. (ReturnControlToECU-function). This parameter is related to the interface Xxx_ReturnControlToECU.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataShortTermAdjustmentFnc [ECUC_Dcm_00675]		
Parent Container	DcmDspData		
Description	Function name to request to application to adjust the IO signal. (ShortTermAdjustment-function). This parameter is related to the interface Xxx_ShortTermAdjustment.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataType [ECUC_Dcm_00985]		
Parent Container	DcmDspData		
Description	Provide the implementation data type of data belonging to a DID.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BOOLEAN	Type of the data is boolean.	
	SINT16	Type of the data is sint16.	
	SINT16_N	Type of the data is sint16 array.	
	SINT32	Type of the data is sint32.	
	SINT32_N	Type of the data is sint32 array.	
	SINT8	Type of the data is sint8.	
	SINT8_N	Type of the data is sint8 array.	
	UINT16	Type of the data is uint16.	
	UINT16_N	Type of the data is uint16 array.	
	UINT32	Type of the data is uint32.	
	UINT32_N	Type of the data is uint32 array.	
	UINT8	Type of the data is uint8.	
	UINT8_DYN	Type of the data is uint8 array with dynamic length.	
	UINT8_N	Type of the data is uint8 array.	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataUsePort [ECUC_Dcm_00713]	
Parent Container	DcmDspData	
Description	Defines which interface shall be used to access the data.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	USE_BLOCK_ID	The DCM will access the Data using the NVRAM Apis with the BlockId defined in DcmDspDataBlockId
	USE_DATA_ASYNCH_CLIENT_SERVER	The DCM will access the Data using an R-Port requiring a asynchronous ClientServertInterface DataServices_{Data}. The R-Port is named DataServices_{Data} where {Data} is the name of the container DcmDspData.
	USE_DATA_ASYNCH_CLIENT_SERVER_ERROR	The Dcm will access the Data using an R-Port requiring a asynchronous ClientServertInterface DataServices_{Data}. The parameter ErrorCode can be returned to allow the application to trigger a negative response during the operation. The R-Port is named DataServices_{Data} where {Data} is the name of the container DcmDspData.
	USE_DATA_ASYNCH_FNC	The DCM will access the Data using the functions that are defined in parameters of type EcucFunctionNameDef (but without DcmDspDataReadDataLengthFnc) in the DcmDspData container. DCM_E_PENDING return is allowed. OpStatus is existing as IN parameter.
	USE_DATA_ASYNCH_FNC_ERROR	The DCM will access the Data using the functions that are defined in parameters of type EcucFunctionNameDef (but without DcmDspDataReadDataLengthFnc) in the DcmDspData container. DCM_E_PENDING return is allowed. OpStatus is existing as IN parameter. The parameter ErrorCode can be returned to allow the application to trigger a negative response during the operation.
	USE_DATA_SENDER_RECEIVER	The DCM will access the Data using an Port requiring a SenderReceiverInteface (with isService=false) DataServices_{Data}. The Port is namedDataServices_{Data} where {Data} is the name of the container DcmDspData.

	USE_DATA_SENDER_RECEIVER_AS_SERVICE	The DCM will access the Data using an service Port requiring a SenderReceiverInteface (with isService=true) DataServices_{Data} . The Port is namedDataServices_{Data} where {Data} is the name of the container DcmDspData.	
	USE_DATA_SYNCH_CLIENT_SERVER	The DCM will access the Data using an R-Port requiring a synchronous ClientServertInterface DataServices_{Data}. The R-Port is named DataServices_{Data} where {Data} is the name of the container DcmDspData.	
	USE_DATA_SYNCH_FNC	The DCM will access the Data using the functions that are defined in parameters of type EcucFunctionNameDef (but without DcmDspDataReadDataLengthFnc) in the DcmDspData container. DCM_E_PENDING return value is not allowed and OpStatus parameter is not existing in the prototype.	
	USE_ECU_SIGNAL	The DCM will access the Data using a direct access to IoHwAb	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataWriteFnc [ECUC_Dcm_00670]		
Parent Container	DcmDspData		
Description	Function name to request application to write the data value of a DID. (WriteData-function). This parameter is related to the interface Xxx_WriteData.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspOdxDataDescription [ECUC_Dcm_00988]		
Parent Container	DcmDspData		
Description	Defines additional description for ODX documentation		
Multiplicity	0..1		
Type	EcucAddInfoParamDef		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspDataBlockIdRef [ECUC_Dcm_00809]		
Parent Container	DcmDspData		
Description	NRAM blockId to access the data. Only relevant if DcmDspDataUsePort==USE_BLOCK_ID.		
Multiplicity	0..1		
Type	Symbolic name reference to NvMBlockDescriptor		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDataInfoRef [ECUC_Dcm_00811]		
Parent Container	DcmDspData		
Description	Reference to 1 DcmDspDataInfo		
Multiplicity	0..1		
Type	Reference to DcmDspDataInfo		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspDiagnosisScaling	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.
DcmDspDidDataSupportInfo	0..1	This container defines the supported information.
DcmDspExternalSRDataElementClass	0..1	This container defines the source of data in a provided port which shall be read respectively the target of data in a required port which shall be written. This container shall contain either one DcmSubElementInDataElementInstance OR DcmDataElementInstance OR DcmSubElementInImplDataElementInstance reference.

10.3.5.5.2 DcmDspDiagnosisScaling

SWS Item	[ECUC_Dcm_00993]
Container Name	DcmDspDiagnosisScaling
Description	This container contains the configuration (parameters) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.
Configuration Parameters	

Container Choices		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspAlternativeDataInterface	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of a VariableDataPrototoype in a DataInterface. Additionally a reference to PortInterfaceMapping can be defined which provide already the mapping rules between the VariableDataPrototoype in a DataInterface used by the software component (DcmDspExternalSRDataElementClass) and the intended Diagnosis Representation defined by DcmDataElement.
DcmDspAlternativeDataType	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of an ApplicationDataType. Additionally the definition of a text table mapping can be a defined for ApplicationDataTypes with a CompuMethod of category TEXTTABLE and SCALE_LINEAR_AND_TEXTTABLE.
DcmDspAlternativeDiagnosticDataElement	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of Diagnostic Extract.

10.3.5.5.3 DcmDspArgumentScaling

SWS Item	[ECUC_Dcm_01062]
Container Name	DcmDspArgumentScaling
Description	This container contains the configuration (arguments) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.
Configuration Parameters	

Container Choices		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspAlternativeArgumentData	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of a ArgumentDataPrototype.
DcmDspAlternativeDataType	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of an ApplicationDataType. Additionally the definition of a text table mapping can be a defined for ApplicationDataTypes with a CompuMethod of category TEXTTABLE and SCALE_LINEAR_AND_TEXTTABLE.
DcmDspAlternativeDiagnosticDataElement	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of Diagnostic Extract.

10.3.5.5.4 DcmDspAlternativeArgumentData

SWS Item	[ECUC_Dcm_01055]
Container Name	DcmDspAlternativeArgumentData
Description	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of a ArgumentDataPrototype.
Configuration Parameters	

Name	DcmDataElement [ECUC_Dcm_01056]		
Parent Container	DcmDspAlternativeArgumentData		
Description	Alternative Diagnosis Representation for the data defined by the means of a ArgumentDataPrototype. The CompuMethod of the data type of the referenced ArgumentDataPrototype will be applied to the data type of the ArgumentDataPrototype in the interface used by the Dcm.		
Multiplicity	1		
Type	Foreign reference to ARGUMENT-DATA-PROTOTYPE		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

No Included Containers

10.3.5.5.5 DcmDspTextTableMapping

SWS Item	[ECUC_Dcm_00999]
Container Name	DcmDspTextTableMapping
Description	The purpose of the DcmDspTextTableMapping is to associate a texttable value defined in the context of the Dcm to a texttable value defined in the context of a CompuMethod referenced by a DataType that shall be taken to create a dataElement in a SenderReceiverInterface. By this means it is possible to create a primitive version of a TexttableMapping (which can only be applied if a dataElement already exists). In other words, the DcmDspTextTableMapping provides a similar mechanism to the TexttableMapping in a situation where the TexttableMapping cannot be applied since the SenderReceiverInterface for the PortPrototype on the Dcm ServiceComponent does not yet exist.
Configuration Parameters	

Name	DcmDspDiagnosisRepresentationDataValue [ECUC_Dcm_01001]		
Parent Container	DcmDspTextTableMapping		
Description	The data value in the diagnosis representation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspInternalDataValue [ECUC_Dcm_01000]		
Parent Container	DcmDspTextTableMapping		
Description	The ECU internal data value.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

10.3.5.5.6 DcmDspAlternativeDataInterface

SWS Item	[ECUC_Dcm_00994]
Container Name	DcmDspAlternativeDataInterface
Description	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of a VariableDataPrototoype in a DataInterface. Additionally a reference to PortInterfaceMapping can be defined which provide already the mapping rules between the VariableDataPrototoype in a DataInterface used by the software component (DcmDspExternalSRDataElementClass) and the intended Diagnosis Representation defined by DcmDataElement.
Configuration Parameters	

Name	DcmDataElement [ECUC_Dcm_00995]		
Parent Container	DcmDspAlternativeDataInterface		
Description	Alternative Diagnosis Representation for the data defined by the means of a VariableDataPrototoype in a DataInterface. The CompuMethod of the data type of the referenced VariableDataPrototype will be applied to the data type of the VariableDataPrototype in the interface used by the Dcm.		
Multiplicity	1		
Type	Foreign reference to VARIABLE-DATA-PROTOTYPE		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

Name	DcmPortInterfaceMapping [ECUC_Dcm_00996]		
Parent Container	DcmDspAlternativeDataInterface		
Description	Optional reference to PortInterfaceMapping which defines the mapping rules. The PortInterfaceMapping is used to get the DataPrototypeMapping that describes a conversion between the data prototype referenced by DcmDataElement and the data prototype referenced from DcmDspExternalSRDataElementClass.		
Multiplicity	0..1		
Type	Foreign reference to PORT-INTERFACE-MAPPING		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

No Included Containers

10.3.5.5.7 DcmDspAlternativeDataType

SWS Item	[ECUC_Dcm_00997]
Container Name	DcmDspAlternativeDataType
Description	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of an ApplicationDataType. Additionally the definition of a text table mapping can be a defined for ApplicationDataTypes with a CompuMethod of category TEXTTABLE and SCALE_LINEAR_AND_TEXTTABLE.
Configuration Parameters	

Name	DcmApplicationDataType [ECUC_Dcm_00998]		
Parent Container	DcmDspAlternativeDataType		
Description	Alternative Diagnosis Representation for the data defined by the means of a ApplicationDataType of category VALUE, BOOLEAN or ARRAY. The CompuMethod that applies to the referenced ApplicationDataType in case of category VALUE or BOOLEAN will be applied to the data type of the VariableDataPrototype in the interface used by the Dcm.		
Multiplicity	1		
Type	Foreign reference to APPLICATION-DATA-TYPE		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspTextTableMapping	0..*	The purpose of the DcmDspTextTableMapping is to associate a texttable value defined in the context of the Dcm to a texttable value defined in the context of a CompuMethod referenced by a DataType that shall be taken to create a dataElement in a SenderReceiverInterface. By this means it is possible to create a primitive version of a TexttableMapping (which can only be applied if a dataElement already exists). In other words, the DcmDspTextTableMapping provides a similar mechanism to the TexttableMapping in a situation where the TexttableMapping cannot be applied since the SenderReceiverInterface for the PortPrototype on the Dcm ServiceComponent does not yet exist.

10.3.5.5.8 DcmDspAlternativeDiagnosticDataElement

SWS Item	[ECUC_Dcm_01084]
Container Name	DcmDspAlternativeDiagnosticDataElement

Description	This container contains the configuration (parameters) of an alternative Diagnosis Representation by the means of Diagnostic Extract.
Configuration Parameters	

Name	DcmDspDiagnosticDataElementRef [ECUC_Dcm_01085]		
Parent Container	DcmDspAlternativeDiagnosticDataElement		
Description	Alternative Diagnosis Representation for the data defined by the means of a DiagnosticDataElement in the Diagnostic Extract. This EcucForeignReference enables the access to all SwDataDefProps, in particular BaseType, CompuMethod and DataConstr The CompuMethod and DataConstr that applies to the referenced DiagnosticDataElement will be applied to the data type of the VariableDataPrototype in the interface used by the Dcm. The mapped ImplementationDataType needs to match the given BaseType.		
Multiplicity	1		
Type	Foreign reference to DIAGNOSTIC-DATA-ELEMENT		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3.5.5.9 DcmDataElementInstance

SWS Item	[ECUC_Dcm_01010]
Container Name	DcmDataElementInstance
Description	Instance Reference to the primitive data in a port where the data element is typed with an ApplicationPrimitveDataType or an ImplementationDataType.
Configuration Parameters	

Name	DcmDataElementInstanceRef [ECUC_Dcm_00991]		
Parent Container	DcmDataElementInstance		
Description	Instance Reference to the primitive or array data which shall be read or written. Supported are VariableDataPrototypes in SenderReceiverInterfaces and NvDataInterfaces and ParameterDataPrototypes in ParameterInterfaces (read only). This reference is applicable if the AutosarDataPrototype is typed with a ApplicationPrimitiveDataType of category VALUE or BOOLEAN or ApplicationArrayDataType or if the AutosarDataPrototype is typed with a ImplementationDataType of category VALUE, ARRAY or TYPE_REFERENCE that in turn boils down to VALUE or ARRAY		
Multiplicity	1		
Type	Instance reference to AUTOSAR-DATA-PROTOTYPE context: ROOT-SW-COMPOSITION-PROTOTYPE SW-COMPONENT-PROTOTYPE PORT-PROTOTYPE		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

No Included Containers

10.3.5.5.10 DcmSubElementInDataElementInstance

SWS Item	[ECUC_Dcm_01009]
Container Name	DcmSubElementInDataElementInstance
Description	Instance Reference to the primitve sub-element (at any level) of composite data in a port where the data element is typed with an ApplicationCompositeDataType.
Configuration Parameters	

Name	DcmSubElementInDataElementInstanceRef [ECUC_Dcm_00990]		
Parent Container	DcmSubElementInDataElementInstance		
Description	Instance Reference to the primitve sub-element (at any level) of composite data in a port which shall be read. Supported are VariableDataPrototypes in SenderReceiverInterfaces and NvDataInterfaces and ParameterDataPrototypes in ParameterInterfaces (read only). This reference is applicable if the AutosarDataPrototype is typed with a ApplicationCompositeDataType.		
Multiplicity	1		
Type	Instance reference to APPLICATION-COMPOSITE-ELEMENT-DATA-PROTOTYPE context: ROOT-SW-COMPOSITION-PROTOTYPE SW-COMPONENT-PROTOTYPE PORT-PROTOTYPE AUTOSAR-DATA-PROTOTYPE APPLICATION-COMPOSITE-ELEMENT-DATA-PROTOTYPE*		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

No Included Containers

10.3.5.5.11 DcmSubElementInImplDataElementInstance

SWS Item	[ECUC_Dcm_01011]
Container Name	DcmSubElementInImplDataElementInstance
Description	Instance Reference to the primitve sub-element (at any level) of composite data in a port where the data element is typed with an ImplementationDataType.
Configuration Parameters	

Name	DcmSubElementInImplDataElementInstanceRef [ECUC_Dcm_00992]		
Parent Container	DcmSubElementInImplDataElementInstance		
Description	Instance Reference to the primitve sub-element (at any level) of composite data in a port which shall be read. Supported are VariableDataPrototypes in SenderReceiverInterfaces and NvDataInterfaces and ParameterDataPrototypes in ParameterInterfaces (read only). This reference is applicable if the AutosarDataPrototype is typed with a ImplementationDataType of category STRUCTURE or ARRAY. Please note that in case of ARRAY the index attribute in the target reference has to be set to select a single array element.		
Multiplicity	1		
Type	Instance reference to IMPLEMENTATION-DATA-TYPE-ELEMENT context: ROOT-SW-COMPOSITION-PROTOTYPE SW-COMPONENT-PROTOTYPE PORT-PROTOTYPE AUTOSAR-DATA-PROTOTYPE I MPLEMENTATION-DATA-TYPE-ELEMENT*		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

No Included Containers

10.3.5.5.12 DcmDspDidDataSupportInfo

SWS Item	[ECUC_Dcm_01104]
Container Name	DcmDspDidDataSupportInfo
Description	This container defines the supported information.
Configuration Parameters	

Name	DcmDspDidDataSupportInfoBit [ECUC_Dcm_01097]		
Parent Container	DcmDspDidDataSupportInfo		
Description	Referenced Bit of the SupportInfo		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidDataSupportInfoRef [ECUC_Dcm_01098]		
Parent Container	DcmDspDidDataSupportInfo		
Description	Reference to DcmDspDidSupportInfo		
Multiplicity	1		
Type	Reference to DcmDspDidSupportInfo		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.5.5.13 DcmDspDataInfo

SWS Item	[ECUC_Dcm_00810]
Container Name	DcmDspDataInfo
Description	This container contains the configuration (parameters) of one Data.
Configuration Parameters	

Name	DcmDspDataScalingInfoSize [ECUC_Dcm_00611]		
Parent Container	DcmDspDataInfo		
Description	If Scaling information service is available for this Data, it provides the size in bytes of the scaling information.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Value Configuration Class	**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

10.3.5.6 DcmDspDidControl

SWS Item	[ECUC_Dcm_00619]
Container Name	DcmDspDidControl
Description	This container contains the configuration (parameters) of the DID control.
Configuration Parameters	

Name	DcmDspDidControlMask [ECUC_Dcm_01059]		
Parent Container	DcmDspDidControl		
Description	This indicates the presence of "controlEnableMask" in SWC service interfaces and defines how the Dcm treats a service request.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DCM_CONTROLMASK_EXTERNAL	The control enable mask record shall be forwarded within each interface and is handled externally.	
	DCM_CONTROLMASK_INTERNAL	The control enable mask record is handled internally and Dcm controls only the included signals.	
	DCM_CONTROLMASK_NO	No control enable mask handling.	
Default Value	DCM_CONTROLMASK_INTERNAL		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidControlMaskSize [ECUC_Dcm_01060]	
Parent Container	DcmDspDidControl	
Description	The value defines the size of the controlEnableMaskRecord in bytes.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 4	
Default Value		

Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidFreezeCurrentState [ECUC_Dcm_00624]		
Parent Container	DcmDspDidControl		
Description	This indicates the presence of "FreezeCurrentState".		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidResetToDefault [ECUC_Dcm_00623]		
Parent Container	DcmDspDidControl		
Description	This indicates the presence of "ResetToDefault".		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidShortTermAdjustment [ECUC_Dcm_00625]		
Parent Container	DcmDspDidControl		
Description	This indicates the presence of "ShortTermAdjustment".		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Scope / Dependency	scope: ECU

Name	DcmDspDidControlModeRuleRef [ECUC_Dcm_00923]		
Parent Container	DcmDspDidControl		
Description	Reference to DcmModeRule 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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidControlSecurityLevelRef [ECUC_Dcm_00620]		
Parent Container	DcmDspDidControl		
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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspDidControlSessionRef [ECUC_Dcm_00621]		
Parent Container	DcmDspDidControl		
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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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**
DcmDspDidControlEnableMask	0..32	The shortname of the container value defines the symbol of the controlMask.

10.3.5.7 DcmDspDidControlEnableMask

SWS Item	[ECUC_Dcm_01057]
Container Name	DcmDspDidControlEnableMask
Description	The shortname of the container value defines the symbol of the controlMask.
Configuration Parameters	

Name	DcmDspDidControlMaskBitPosition [ECUC_Dcm_01058]	
Parent Container	DcmDspDidControlEnableMask	
Description	Defines the position of the bit in the controlMask starting from most significant bit (MSB first) to least significant bit. This Bit endianness is identical to the controlMask in UDS. The DcmDspDidControlMaskSize should be considered for most significant bit.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 31	
Default Value		
Post-Build Variant Multiplicity	false	
Post-Build Variant Value	false	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

10.3.5.8 Ecu Reset

10.3.5.8.1 DcmDspEcuReset

SWS Item	[ECUC_Dcm_01111]
Container Name	DcmDspEcuReset
Description	This container contains the configuration for DcmDspEcuReset service
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspEcuResetRow	1..*	This container contains the configuration for each DcmDspEcuReset subservice.

10.3.5.8.2 DcmDspEcuResetRow

SWS Item	[ECUC_Dcm_01112]
Container Name	DcmDspEcuResetRow
Description	This container contains the configuration for each DcmDspEcuReset subservice.
Configuration Parameters	

Name	DcmDspEcuResetId [ECUC_Dcm_01113]		
Parent Container	DcmDspEcuResetRow		
Description	Defines the subfunction ID		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 127		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmResponseToEcuReset [ECUC_Dcm_01039]		
Parent Container	DcmDspEcuResetRow		
Description	Defines the answer to EcuReset service should come: Before or after the reset.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	AFTER_RESET	Answer to EcuReset service should come after the reset.	
	BEFORE_RESET	Answer to EcuReset service should come before the reset.	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

10.3.5.9 Memory

10.3.5.9.1 DcmDspMemory

SWS Item	[ECUC_Dcm_00784]
Container Name	DcmDspMemory
Description	This container contains the configuration of the memory access.
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspAddressAndLengthFormatIdentifier	0..1	This container contains the configuration of the supported AddressAndLengthFormatIdentifiers for memory access.
DcmDspMemoryIdInfo	1..*	Provides the value of memory identifier used to select the desired memory device This container contains the configuration of the memory access requested through diagnostic services : ReadMemoryByAddress, WriteMemoryByAddress, RequestDownload, RequestUpload

10.3.5.9.2 DcmDspAddressAndLengthFormatIdentifier

SWS Item	[ECUC_Dcm_00963]
Container Name	DcmDspAddressAndLengthFormatIdentifier
Description	This container contains the configuration of the supported AddressAndLengthFormatIdentifiers for memory access.
Configuration Parameters	

Name	DcmDspSupportedAddressAndLengthFormatIdentifier [ECUC_Dcm_00964]		
Parent Container	DcmDspAddressAndLengthFormatIdentifier		
Description	This parameter defines the supported AddressAndLengthFormatIdentifier of the request message.		
Multiplicity	1..*		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.5.9.3 DcmDspMemoryIdInfo

SWS Item	[ECUC_Dcm_00911]
Container Name	DcmDspMemoryIdInfo
Description	Provides the value of memory identifier used to select the desired memory device This container contains the configuration of the memory access requested through diagnostic services : ReadMemoryByAddress, WriteMemoryByAddress, RequestDownload, RequestUpload
Configuration Parameters	

Name	DcmDspMemoryIdValue [ECUC_Dcm_00913]		
Parent Container	DcmDspMemoryIdInfo		
Description	Value of the memory device identifier used. Each DcmDspMemoryIdInfo should have a unique ID. The MemoryIdValue is retrieved from the request messages (RMBA,WMBA,RD,RU,DDDI) according to ISO-14229-1.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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**
DcmDspReadMemoryRangeByLabelInfo	0..*	Provides a memory range allowed for reading via labels (lower and higher address configured as strings).
DcmDspReadMemoryRangeInfo	0..*	Provides the range of memory address allowed for reading
DcmDspWriteMemoryRangeByLabelInfo	0..*	Provides a memory range allowed for writing via labels (lower and higher address configured as strings).
DcmDspWriteMemoryRangeInfo	0..*	Provides the range of memory address allowed for writting.

10.3.5.9.4 DcmDspReadMemoryRangeByLabelInfo

SWS Item	[ECUC_Dcm_01068]
Container Name	DcmDspReadMemoryRangeByLabelInfo
Description	Provides a memory range allowed for reading via labels (lower and higher address configured as strings).
Configuration Parameters	

Name	DcmDspReadMemoryRangeByLabelHigh [ECUC_Dcm_01070]		
Parent Container	DcmDspReadMemoryRangeByLabelInfo		
Description	High memory address as label (string) of a range allowed for reading.		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression	[a-zA-Z0-9_]([a-zA-Z0-9\._])*		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspReadMemoryRangeByLabelLow [ECUC_Dcm_01069]		
Parent Container	DcmDspReadMemoryRangeByLabelInfo		
Description	Low memory address as label (string) of a range allowed for reading.		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression	[a-zA-Z0-9_]([a-zA-Z0-9\._])*		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspReadMemoryRangeModeRuleRef [ECUC_Dcm_01072]		
Parent Container	DcmDspReadMemoryRangeByLabelInfo		
Description	Reference to DcmModeRule 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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Scope / Dependency	scope: ECU		

Name	DcmDspReadMemoryRangeSecurityLevelRef [ECUC_Dcm_01071]		
Parent Container	DcmDspReadMemoryRangeByLabelInfo		
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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspReadMemoryRangeSessionLevelRef [ECUC_Dcm_01088]		
Parent Container	DcmDspReadMemoryRangeByLabelInfo		
Description	Link to the session level needed for access to this memory address range. If there is no reference, no check of session level shall be done.		
Multiplicity	0..*		
Type	Reference to DcmDspSessionRow		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.5.9.5 DcmDspReadMemoryRangeInfo

SWS Item	[ECUC_Dcm_00785]
Container Name	DcmDspReadMemoryRangeInfo
Description	Provides the range of memory address allowed for reading
Configuration Parameters	

Name	DcmDspReadMemoryRangeHigh [ECUC_Dcm_00787]		
Parent Container	DcmDspReadMemoryRangeInfo		
Description	High memory address of a range allowed for reading		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967294		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspReadMemoryRangeLow [ECUC_Dcm_00786]		
Parent Container	DcmDspReadMemoryRangeInfo		
Description	Low memory address of a range allowed for reading		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967294		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspReadMemoryRangeModeRuleRef [ECUC_Dcm_00920]
Parent Container	DcmDspReadMemoryRangeInfo
Description	Reference to DcmModeRule 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
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false

Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspReadMemoryRangeSecurityLevelRef [ECUC_Dcm_00788]		
Parent Container	DcmDspReadMemoryRangeInfo		
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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspReadMemoryRangeSessionLevelRef [ECUC_Dcm_01086]		
Parent Container	DcmDspReadMemoryRangeInfo		
Description	Link to the session level needed for access to this memory address range. If there is no reference, no check of session level shall be done.		
Multiplicity	0..*		
Type	Reference to DcmDspSessionRow		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Value Configuration Class	**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

10.3.5.9.6 DcmDspWriteMemoryRangeByLabelInfo

SWS Item	[ECUC_Dcm_01073]
Container Name	DcmDspWriteMemoryRangeByLabelInfo
Description	Provides a memory range allowed for writing via labels (lower and higher address configured as strings).
Configuration Parameters	

Name	DcmDspWriteMemoryRangeByLabelHigh [ECUC_Dcm_01075]		
Parent Container	DcmDspWriteMemoryRangeByLabelInfo		
Description	High memory address as label (string) of a range allowed for writing.		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression	[a-zA-Z0-9_]([a-zA-Z0-9\._])*		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspWriteMemoryRangeByLabelLow [ECUC_Dcm_01074]		
Parent Container	DcmDspWriteMemoryRangeByLabelInfo		
Description	Low memory address as label (string) of a range allowed for writing.		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression	[a-zA-Z0-9_]([a-zA-Z0-9\._])*		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspWriteMemoryRangeModeRuleRef [ECUC_Dcm_01077]		
Parent Container	DcmDspWriteMemoryRangeByLabelInfo		
Description	Reference to DcmModeRule 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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspWriteMemoryRangeSecurityLevelRef [ECUC_Dcm_01076]		
Parent Container	DcmDspWriteMemoryRangeByLabelInfo		
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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspWriteMemoryRangeSessionLevelRef [ECUC_Dcm_01089]		
Parent Container	DcmDspWriteMemoryRangeByLabelInfo		
Description	Link to the session level needed for access to this memory address range. If there is no reference, no check of session level shall be done.		
Multiplicity	0..*		
Type	Reference to DcmDspSessionRow		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.5.9.7 DcmDspWriteMemoryRangeInfo

SWS Item	[ECUC_Dcm_00789]
Container Name	DcmDspWriteMemoryRangeInfo
Description	Provides the range of memory address allowed for writting.
Configuration Parameters	

Name	DcmDspWriteMemoryRangeHigh [ECUC_Dcm_00791]		
Parent Container	DcmDspWriteMemoryRangeInfo		
Description	High memory address of a range allowed for writing.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967294		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspWriteMemoryRangeLow [ECUC_Dcm_00790]		
Parent Container	DcmDspWriteMemoryRangeInfo		
Description	Low memory address of a range allowed for writing		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967294		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspWriteMemoryRangeModeRuleRef [ECUC_Dcm_00916]		
Parent Container	DcmDspWriteMemoryRangeInfo		
Description	Reference to DcmModeRule 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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspWriteMemoryRangeSecurityLevelRef [ECUC_Dcm_00793]
Parent Container	DcmDspWriteMemoryRangeInfo
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
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false

Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspWriteMemoryRangeSessionLevelRef [ECUC_Dcm_01087]		
Parent Container	DcmDspWriteMemoryRangeInfo		
Description	Link to the session level needed for access to this memory address range. If there is no reference, no check of session level shall be done.		
Multiplicity	0..*		
Type	Reference to DcmDspSessionRow		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.5.10 PIDs

10.3.5.10.1 DcmDspPid

SWS Item	[ECUC_Dcm_00626]
Container Name	DcmDspPid
Description	This container defines the availability of a PID to the DCM.
Configuration Parameters	

Name	DcmDspPidIdentifier [ECUC_Dcm_00627]		
Parent Container	DcmDspPid		
Description	1 byte Identifier of the PID Within each DcmConfigSet all DcmDspPidIdentifier values shall be unique.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspPidService [ECUC_Dcm_00893]		
Parent Container	DcmDspPid		
Description	Indicates if a PID is used with service $01 and/or $02		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DCM_SERVICE_01	A PID is used with service $01 only.	
	DCM_SERVICE_01_02	A PID is used with service $01 and $02. Allowed with a PID configuration containing data elements on byte basis.	
	DCM_SERVICE_02	A PID is used with service $02 only. Allowed with a PID configuration containing data elements on byte basis.	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspPidSize [ECUC_Dcm_00870]	
Parent Container	DcmDspPid	
Description	Length of a PID in byte(s).	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default Value		
Post-Build Variant Value	false	

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspPidUsed [ECUC_Dcm_00806]		
Parent Container	DcmDspPid		
Description	Allow to activate or deactivate the usage of a PID, for multi purpose ECUs true = PID is available false = PID is not available		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**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 packeted PIDs (e.g. PID$68).

10.3.5.10.2 DcmDspPidSupportInfo

SWS Item	[ECUC_Dcm_00871]
Container Name	DcmDspPidSupportInfo
Description	This container defines the support information (typically byte A) to declare the usability of the data bytes within the so-called packeted PIDs (e.g. PID$68).
Configuration Parameters	

Name	DcmDspPidSupportInfoLen [ECUC_Dcm_00873]	
Parent Container	DcmDspPidSupportInfo	
Description	Length of the support information in bytes.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default Value		

Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspPidSupportInfoPos [ECUC_Dcm_00872]		
Parent Container	DcmDspPidSupportInfo		
Description	Position of the support information in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**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

10.3.5.10.3 DcmDspPidData

SWS Item	[ECUC_Dcm_00865]
Container Name	DcmDspPidData
Description	This container defines the parameter for a Signal in the PID.
Configuration Parameters	

Name	DcmDspPidByteOffset [ECUC_Dcm_01107]	
Parent Container	DcmDspPidData	
Description	This is the position in bytes of the PID structure and will not start at position 0 in case a support information is available (for packeted PIDs).	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default Value		
Post-Build Variant Value	false	

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspPidDataByteSize [ECUC_Dcm_01108]		
Parent Container	DcmDspPidData		
Description	Defines the array length in bytes or the the maximum array length for variable datalengths.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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.

10.3.5.10.4 DcmDspPidService01

SWS Item	[ECUC_Dcm_00894]
Container Name	DcmDspPidService01
Description	Contains specific configuration parameter of PID for service $01. This container exists only if DcmDspPidService is set to DCM_SERVICE_01 or DCM_SERVICE_01_02.
Configuration Parameters	

Name	DcmDspPidDataEndianness [ECUC_Dcm_01012]		
Parent Container	DcmDspPidService01		
Description	Defines the endianness of the data belonging to a PID in a diagnostic response message. If no DcmDspPidDataEndianness is defined the value of DcmDspDataDefaultEndianness is applicable.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN	Most significant byte shall be stored at the lowest address.	
	LITTLE_ENDIAN	Most significant byte shall be stored at the highest address	
	OPAQUE	Opaque data endianness	
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspPidDataReadFnc [ECUC_Dcm_00629]		
Parent Container	DcmDspPidService01		
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			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspPidDataType [ECUC_Dcm_01018]		
Parent Container	DcmDspPidService01		
Description	Provide the implementation data type of data belonging to a PID.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BOOLEAN	Type of the data is boolean.	
	SINT16	Type of the data is sint16.	
	SINT16_N	Type of the data is sint16 array.	
	SINT32	Type of the data is sint32.	
	SINT32_N	Type of the data is sint32 array.	
	SINT8	Type of the data is sint8.	
	SINT8_N	Type of the data is sint8 array.	
	UINT16	Type of the data is uint16.	
	UINT16_N	Type of the data is uint16 array.	
	UINT32	Type of the data is uint32.	
	UINT32_N	Type of the data is uint32 array.	
	UINT8	Type of the data is uint8.	

	UINT8_DYN	Type of the data is uint8 array with dynamic length.
		Tags:
		atp.Status=obsolete
		atp.StatusRevisionBegin=4.3.1	

	UINT8_N	Type of the data is uint8 array.	
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspPidDataUsePort [ECUC_Dcm_00720]		
Parent Container	DcmDspPidService01		
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 or USE_DATA_SENDER_RECEIVER_AS_SERVICE, 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_SENDER_RECEIVER_AS_SERVICE		
	USE_DATA_SYNCH_CLIENT_SERVER		
	USE_DATA_SYNCH_FNC		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspDiagnosisScaling	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.
DcmDspPidService01ExternalSRDataElementClass	0..1	This container defines the source of data in a provided port which shall be read respectively the target of data in a required port which shall be written. This container shall contain either one DcmSubElementInDataElementInstance OR DcmDataElementInstance OR DcmSubElementInImplDataElementInstance reference.

10.3.5.10.5 DcmDspPidService02

SWS Item	[ECUC_Dcm_00895]
Container Name	DcmDspPidService02

Description	Contains specific configuration parameter of PID for service $02. This container exists only if DcmDspPidService is set to DCM_SERVICE_02 or DCM_SERVICE_01_02.
Configuration Parameters	

Name	DcmDspPidDataDemRef [ECUC_Dcm_00887]		
Parent Container	DcmDspPidService02		
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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.5.10.6 DcmDspPidDataSupportInfo

SWS Item	[ECUC_Dcm_00874]
Container Name	DcmDspPidDataSupportInfo
Description	This container defines the supported information.
Configuration Parameters	

Name	DcmDspPidDataSupportInfoBit [ECUC_Dcm_00876]	
Parent Container	DcmDspPidDataSupportInfo	
Description	Referenced Bit of the SupportInfo	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default Value		
Post-Build Variant Value	false	

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspPidDataSupportInfoRef [ECUC_Dcm_00875]		
Parent Container	DcmDspPidDataSupportInfo		
Description	Reference to DcmDspPidSupportInfo		
Multiplicity	1		
Type	Reference to DcmDspPidSupportInfo		
Post-Build Variant Value	false		
Value Configuration Class	**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

10.3.5.11 DcmDspRequestControl

SWS Item	[ECUC_Dcm_00637]
Container Name	DcmDspRequestControl
Description	This container contains the configuration (parameters) of the "Request control of on-board system, test or component" service (Service $08). The DCM will request the control using an R-Port requiring a PortInteface RequestControlServices_{Tid}. The R-Port is named RequestControlServices_{Tid} where {Tid} is the name of the container DcmDspRequestControl.
Configuration Parameters	

Name	DcmDspRequestControlInBufferSize [ECUC_Dcm_00722]	
Parent Container	DcmDspRequestControl	
Description	Number of bytes to be provided in the input buffer of the interface RequestControlServices_{Tid} for OBD Service $08	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default Value		
Post-Build Variant Value	false	

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRequestControlInfoByte [ECUC_Dcm_01078]		
Parent Container	DcmDspRequestControl		
Description	Manufacturer specific value reported to the tester for the record identifiers 0xE000 to OxE1FF. (WWH-OBD use cases)		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRequestControlOutBufferSize [ECUC_Dcm_00723]		
Parent Container	DcmDspRequestControl		
Description	Number of bytes to be provided in the output buffer of the interface RequestControlServices_{Tid} for OBD Service $08		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRequestControlTestId [ECUC_Dcm_00656]	
Parent Container	DcmDspRequestControl	
Description	Test Id for Service $08	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default Value		
Post-Build Variant Value	false	

Value Configuration Class	**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

10.3.5.12 DcmDspRequestFileTransfer

SWS Item	[ECUC_Dcm_01034]
Container Name	DcmDspRequestFileTransfer
Description	This container contains the configuration for RequestFileTransfer. This container only exists if RequestFileTransfer is configured.
Configuration Parameters	

Name	DcmRequestFileTransferFileSizeParameterLength [ECUC_Dcm_01035]		
Parent Container	DcmDspRequestFileTransfer		
Description	Length of the fileSizeCompressed and fileSizeUncompressedOrDirInfoLength in the Dcm_ProcessRequestFileTransfer operation and response message.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmRequestFileTransferLengthFormatIdentifier [ECUC_Dcm_01036]	
Parent Container	DcmDspRequestFileTransfer	
Description	Defines the length (number of bytes) of the maxNumberOfBlockLength parameter.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default Value		
Post-Build Variant Value	false	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

10.3.5.13 Response on Event

10.3.5.13.1 DcmDspRoe

SWS Item	[ECUC_Dcm_00858]
Container Name	DcmDspRoe
Description	Provide the configuration of the ResponseOnEvent mechanism.
Configuration Parameters	

Name	DcmDspRoeInterMessageTime [ECUC_Dcm_00856]		
Parent Container	DcmDspRoe		
Description	Provide the minimum time in seconds between two transmissions of ROE event. It is used for the delay between two different consecutive Roe transmissions.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 5]		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspRoeEvent	1..255	This container contains a list of all supported Roe eventTypeRecords which are accepted by this ECU. At most one DcmDspRoeEvent container is allowed to define a DcmDspRoeEventProperties container with the choice DcmDspRoeOnDTCStatusChange.
DcmDspRoeEventWindowTime	1..*	This container configures the available EventWindowTime in this Ecu. This container contains a sub-set of EventWindowTimes supported by the Dcm, to limit the Ecu resources.

10.3.5.13.2 DcmDspRoeEvent

SWS Item	[ECUC_Dcm_00973]
Container Name	DcmDspRoeEvent
Description	This container contains a list of all supported Roe eventTypeRecords which are accepted by this ECU. At most one DcmDspRoeEvent container is allowed to define a DcmDspRoeEventProperties container with the choice DcmDspRoeOnDTCStatusChange.
Configuration Parameters	

Name	DcmDspRoeEventId [ECUC_Dcm_00976]		
Parent Container	DcmDspRoeEvent		
Description	EventId for a global identification of this ROE event it is used within APIs Dcm_TriggerOnEvent() and the ModeDeclarationGroup. The ratio Ids should be sequentially ordered beginning with 0 and no gaps in between.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 254		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspRoeInitialEventStatus [ECUC_Dcm_00980]		
Parent Container	DcmDspRoeEvent		
Description	Initial Roe status of this RoeEvent		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DCM_ROE_CLEARED		
	DCM_ROE_STOPPED		
Default Value	DCM_ROE_CLEARED		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspRoeEventProperties	1	This container contains the properties of Roe eventTypeRecords. In one DcmDspRoeEventProperties container one DcmDspRoeOnDTCStatusChange or DcmDspRoeOnChangeOfDataIdentifier container shall be defined.

10.3.5.13.3 DcmDspRoeEventProperties

SWS Item	[ECUC_Dcm_00978]
Container Name	DcmDspRoeEventProperties
Description	This container contains the properties of Roe eventTypeRecords. In one DcmDspRoeEventProperties container one DcmDspRoeOnDTCStatusChange or DcmDspRoeOnChangeOfDataIdentifier container shall be defined.
Configuration Parameters	

Container Choices		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspRoeOnChangeOfDataIdentifier	0..1	This container contains the eventTypeRecord supported for onChangeOfDataIdentifier eventType.
DcmDspRoeOnDTCStatusChange	0..1	This container contains the eventTypeRecord supported for onDTCStatusChange eventType.

10.3.5.13.4 DcmDspRoeOnChangeOfDataIdentifier

SWS Item	[ECUC_Dcm_00975]
Container Name	DcmDspRoeOnChangeOfDataIdentifier
Description	This container contains the eventTypeRecord supported for onChangeOfDataIdentifier eventType.
Configuration Parameters	

Name	DcmDspRoeDidRef [ECUC_Dcm_00979]		
Parent Container	DcmDspRoeOnChangeOfDataIdentifier		
Description	Reference to a Did which is watched.		
Multiplicity	1		
Type	Reference to DcmDspDid		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3.5.13.5 DcmDspRoeOnDTCStatusChange

SWS Item	[ECUC_Dcm_00974]
Container Name	DcmDspRoeOnDTCStatusChange
Description	This container contains the eventTypeRecord supported for onDTCStatusChange eventType.
Configuration Parameters	

Name	DcmDspRoeDTCStatusMask [ECUC_Dcm_01109]		
Parent Container	DcmDspRoeOnDTCStatusChange		
Description	Value of the relevant DTCStatusMask		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3.5.13.6 DcmDspRoeEventWindowTime

SWS Item	[ECUC_Dcm_00981]
Container Name	DcmDspRoeEventWindowTime
Description	This container configures the available EventWindowTime in this Ecu. This container contains a sub-set of EventWindowTimes supported by the Dcm, to limit the Ecu resources.
Configuration Parameters	

Name	DcmDspRoeEventWindowTime [ECUC_Dcm_00982]		
Parent Container	DcmDspRoeEventWindowTime		
Description	Value of the EventWindowTime		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DCM_ROE_EVENT_WINDOW_CURRENT_AND_FOLLOWING_CYCLE		
	DCM_ROE_EVENT_WINDOW_CURRENT_CYCLE		
	DCM_ROE_EVENT_WINDOW_INFINITE		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspRoeStorageState [ECUC_Dcm_00983] (Obsolete)		
Parent Container	DcmDspRoeEventWindowTime		

Description	If this parameter is set to TRUE the StorageStateBit will be evaluated if this EventWindowTime is requested.
	Tags:
	atp.Status=obsolete
	atp.StatusRevisionBegin=4.3.1		

Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3.5.14 Routines

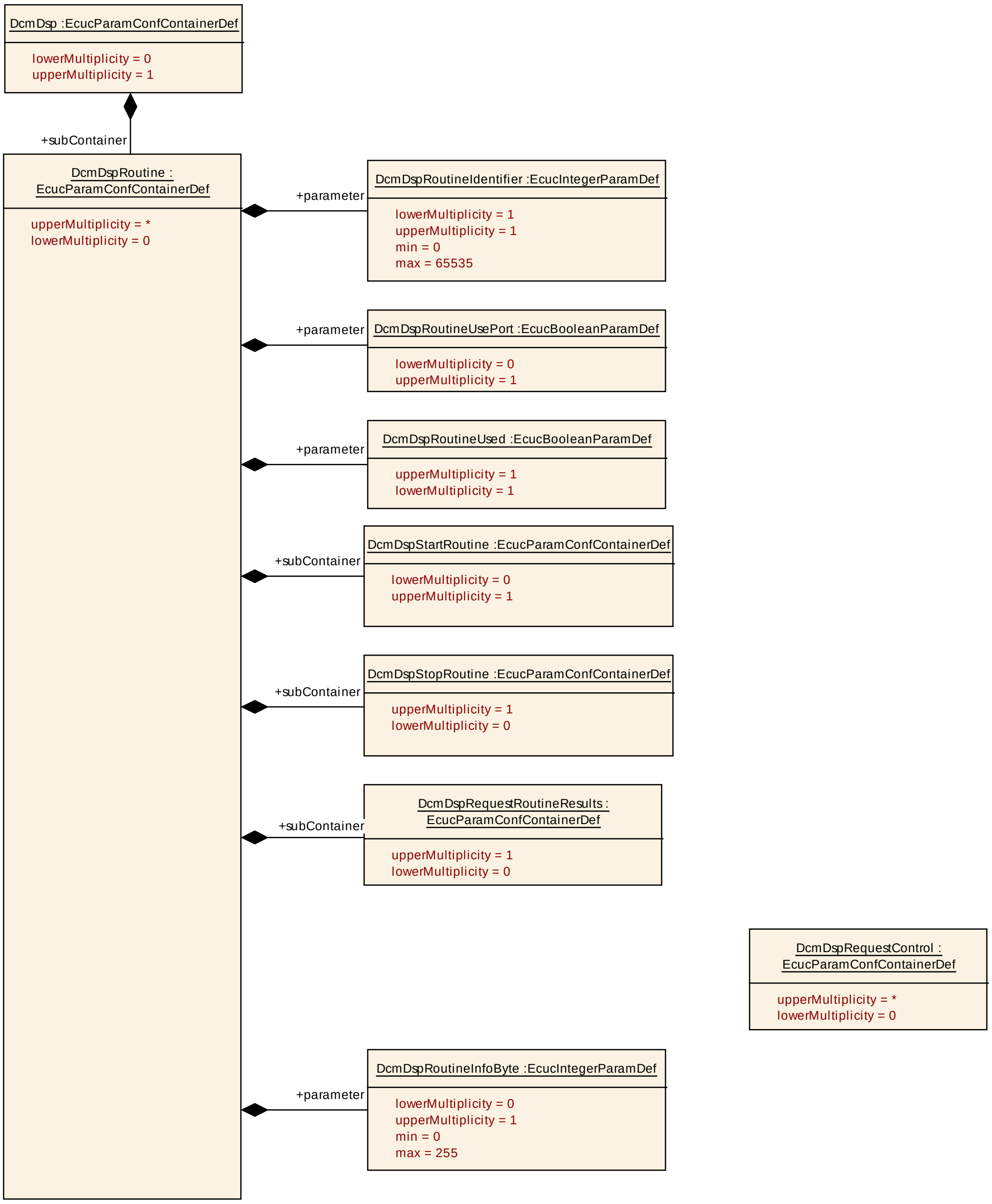


Figure 10.8

10.3.5.14.1 DcmDspRoutine

SWS Item	[ECUC_Dcm_00640]
Container Name	DcmDspRoutine
Description	This container contains the configuration (parameters) for Routines
Configuration Parameters	

Name	DcmDspRoutineIdentifier [ECUC_Dcm_00641]		
Parent Container	DcmDspRoutine		
Description	2 bytes Identifier of the RID Within each DcmConfigSet all DcmDspRoutineIdentifier values shall be unique.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineInfoByte [ECUC_Dcm_01063]		
Parent Container	DcmDspRoutine		
Description	Manufacturer specific value reported to the tester for the record identifiers 0xE000 to OxE1FF. (OBD use cases)		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineUsed [ECUC_Dcm_00807]		
Parent Container	DcmDspRoutine		
Description	Allow to activate or deactivate the usage of a Routine, for multi purpose ECUs True = Routine is available False = Routine is not available		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineUsePort [ECUC_Dcm_00724]		
Parent Container	DcmDspRoutine		
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	0..1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspRequestRoutineResults	0..1	Provides the configuration of RequestResult subservice for RoutineControl service. Existence indicates that the RequestRoutineResults in the RoutineControl is supported.
DcmDspStartRoutine	0..1	Provides the configuration of Start subservice for RoutineControl service.
DcmDspStopRoutine	0..1	Provides the configuration of Stop subservice for RoutineControl service. Existence indicates that the StopRoutine in the RoutineControl is supported.

10.3.5.14.2 DcmDspRequestRoutineResults

SWS Item	[ECUC_Dcm_01023]
Container Name	DcmDspRequestRoutineResults
Description	Provides the configuration of RequestResult subservice for RoutineControl service. Existence indicates that the RequestRoutineResults in the RoutineControl is supported.
Configuration Parameters	

Name	DcmDspRequestRoutineResultsConfirmationEnabled [ECUC_Dcm_01091]		
Parent Container	DcmDspRequestRoutineResults		
Description	Allows to enable/disable the confirmation function to indicate the transmission of a response to a RequestRoutineResults request		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRequestRoutineResultsConfirmationFnc [ECUC_Dcm_01090]		
Parent Container	DcmDspRequestRoutineResults		
Description	C-function to call if a transmission confirmation is needed by the issuer (BSW module)		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRequestRoutineResultsFnc [ECUC_Dcm_00753]		
Parent Container	DcmDspRequestRoutineResults		
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			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRequestRoutineResultsCommonAuthorizationRef [ECUC_Dcm_01054]		
Parent Container	DcmDspRequestRoutineResults		
Description	Reference to DcmDspCommonAuthorization Common authorization configuration taken from the referenced DcmDspRequestRoutineResultsCommonAuthorizationRef. If there is no reference, no check on the commonly defined authorization conditions shall be done to get the routine result.		
Multiplicity	0..1		
Type	Reference to DcmDspCommonAuthorization		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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**
DcmDspRequestRoutineResultsOut	0..1	Provide description of output parameter of RequestResult subservice for RoutineControl service.

10.3.5.14.3 DcmDspRequestRoutineResultsOut

SWS Item	[ECUC_Dcm_00831]
Container Name	DcmDspRequestRoutineResultsOut
Description	Provide description of output parameter of RequestResult subservice for RoutineControl service.
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspRequestRoutineResultsOutSignal	1..*	Provides description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataOutN elements in the XXX_RequestResult function call.

10.3.5.14.4 DcmDspRequestRoutineResultsOutSignal

SWS Item	[ECUC_Dcm_00836]
Container Name	DcmDspRequestRoutineResultsOutSignal
Description	Provides description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataOutN elements in the XXX_RequestResult function call. **Attributes:** requiresIndex=true
Configuration Parameters	

Name	DcmDspRoutineParameterSize [ECUC_Dcm_00838]	
Parent Container	DcmDspRequestRoutineResultsOutSignal	
Description	Provide the size of a RoutineControl parameter in bytes	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default Value		
Post-Build Variant Multiplicity	false	
Post-Build Variant Value	false	

Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineSignalEndianness [ECUC_Dcm_01013]		
Parent Container	DcmDspRequestRoutineResultsOutSignal		
Description	Defines the endianness of the data belonging to a Routine Out Signal for RequestResult subfunction.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN	Most significant byte shall be stored at the lowest address.	
	LITTLE_ENDIAN	Most significant byte shall be stored at the highest address	
	OPAQUE	Opaque data endianness	
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineSignalPos [ECUC_Dcm_00837]		
Parent Container	DcmDspRequestRoutineResultsOutSignal		
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			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Scope / Dependency	scope: ECU

Name	DcmDspRoutineSignalType [ECUC_Dcm_00881]		
Parent Container	DcmDspRequestRoutineResultsOutSignal		
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.	
	SINT16_N	Type of the signal is sint16 array.	
	SINT32	Type of the signal is sint32.	
	SINT32_N	Type of the signal is sint32 array.	
	SINT8	Type of the signal is sint8.	
	SINT8_N	Type of the signal is sint8 array.	
	UINT16	Type of the signal is uint16.	
	UINT16_N	Type of the signal is uint16 array.	
	UINT32	Type of the signal is uint32.	
	UINT32_N	Type of the signal is uint32 array.	
	UINT8	Type of the signal is uint8.	
	UINT8_N	Type of the signal is uint8 array.	
	VARIABLE_LENGTH	Type of the signal is uint8[DcmDspRoutineParameterSize]. This is only valid for the last signal and when DcmDspRoutineSignalType is set to VARIABLE_LENGTH.	
Post-Build Variant Value	false		
Value Configuration Class	**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**
DcmDspArgumentScaling	0..1	This container contains the configuration (arguments) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.

10.3.5.14.5 DcmDspStartRoutine

SWS Item	[ECUC_Dcm_01021]
Container Name	DcmDspStartRoutine
Description	Provides the configuration of Start subservice for RoutineControl service.
Configuration Parameters	

Name	DcmDspStartRoutineConfirmationEnabled [ECUC_Dcm_01093]		
Parent Container	DcmDspStartRoutine		
Description	Allows to enable/disable the confirmation function to indicate the transmission of a response to a StartRoutine request		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspStartRoutineConfirmationFnc [ECUC_Dcm_01094]		
Parent Container	DcmDspStartRoutine		
Description	C-function to call if a transmission confirmation is needed by the issuer (BSW module)		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspStartRoutineFnc [ECUC_Dcm_00664]		
Parent Container	DcmDspStartRoutine		
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			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspStartRoutineCommonAuthorizationRef [ECUC_Dcm_01052]		
Parent Container	DcmDspStartRoutine		
Description	Reference to DcmDspCommonAuthorization Common authorization configuration taken from the referenced DcmDspStartRoutineCommonAuthorizationRef. If there is no reference, no check on the commonly defined authorization conditions shall be done to start the routine.		
Multiplicity	0..1		
Type	Reference to DcmDspCommonAuthorization		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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**
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.

10.3.5.14.6 DcmDspStartRoutineIn

SWS Item	[ECUC_Dcm_00834]
Container Name	DcmDspStartRoutineIn
Description	Provide description of input parameter of Start subservice for RoutineControl service
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspStartRoutineIn Signal	1..*	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataInN elements in the XXX_Start function call.

10.3.5.14.7 DcmDspStartRoutineInSignal

SWS Item	[ECUC_Dcm_00845]
Container Name	DcmDspStartRoutineInSignal
Description	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataInN elements in the XXX_Start function call. **Attributes:** requiresIndex=true
Configuration Parameters	

Name	DcmDspRoutineParameterSize [ECUC_Dcm_00847]	
Parent Container	DcmDspStartRoutineInSignal	
Description	Provide the size of a RoutineControl parameter in bytes	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default Value		
Post-Build Variant Multiplicity	false	
Post-Build Variant Value	false	

Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineSignalEndianness [ECUC_Dcm_01016]		
Parent Container	DcmDspStartRoutineInSignal		
Description	Defines the endianness of the data belonging to a Routine In Signal for Start subfunction.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN	Most significant byte shall be stored at the lowest address.	
	LITTLE_ENDIAN	Most significant byte shall be stored at the highest address	
	OPAQUE	Opaque data endianness	
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineSignalPos [ECUC_Dcm_00846]		
Parent Container	DcmDspStartRoutineInSignal		
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			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Scope / Dependency	scope: ECU

Name	DcmDspRoutineSignalType [ECUC_Dcm_00884]		
Parent Container	DcmDspStartRoutineInSignal		
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.	
	SINT16_N	Type of the signal is sint16 array.	
	SINT32	Type of the signal is sint32.	
	SINT32_N	Type of the signal is sint32 array.	
	SINT8	Type of the signal is sint8.	
	SINT8_N	Type of the signal is sint8 array.	
	UINT16	Type of the signal is uint16.	
	UINT16_N	Type of the signal is uint16 array.	
	UINT32	Type of the signal is uint32.	
	UINT32_N	Type of the signal is uint32 array.	
	UINT8	Type of the signal is uint8.	
	UINT8_N	Type of the signal is uint8 array.	
	VARIABLE_LENGTH	Type of the signal is uint8[DcmDspRoutineParameterSize]. This is only valid for the last signal and when DcmDspRoutineSignalType is set to VARIABLE_LENGTH.	
Post-Build Variant Value	false		
Value Configuration Class	**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**
DcmDspArgumentScaling	0..1	This container contains the configuration (arguments) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.

10.3.5.14.8 DcmDspStartRoutineOut

SWS Item	[ECUC_Dcm_00835]
Container Name	DcmDspStartRoutineOut
Description	Provide description of output parameter of Start subservice for RoutineControl service.
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspStartRoutineOutSignal	1..*	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataOutN elements in the XXX_Start function call.

10.3.5.14.9 DcmDspStartRoutineOutSignal

SWS Item	[ECUC_Dcm_00848]
Container Name	DcmDspStartRoutineOutSignal
Description	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataOutN elements in the XXX_Start function call. **Attributes:** requiresIndex=true
Configuration Parameters	

Name	DcmDspRoutineParameterSize [ECUC_Dcm_00850]		
Parent Container	DcmDspStartRoutineOutSignal		
Description	Provide the size of a RoutineControl parameter in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineSignalEndianness [ECUC_Dcm_01017]		
Parent Container	DcmDspStartRoutineOutSignal		
Description	Defines the endianness of the data belonging to a Routine Out Signal for Start subfunction.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN	Most significant byte shall be stored at the lowest address.	
	LITTLE_ENDIAN	Most significant byte shall be stored at the highest address	
	OPAQUE	Opaque data endianness	
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineSignalPos [ECUC_Dcm_00867]		
Parent Container	DcmDspStartRoutineOutSignal		
Description	Provide the position of the signal in the RoutineControl request/response. The position is defined in bits.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineSignalType [ECUC_Dcm_00885]		
Parent Container	DcmDspStartRoutineOutSignal		
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.	
	SINT16_N	Type of the signal is sint16 array.	
	SINT32	Type of the signal is sint32.	
	SINT32_N	Type of the signal is sint32 array.	
	SINT8	Type of the signal is sint8.	
	SINT8_N	Type of the signal is sint8 array.	
	UINT16	Type of the signal is uint16.	
	UINT16_N	Type of the signal is uint16 array.	
	UINT32	Type of the signal is uint32.	
	UINT32_N	Type of the signal is uint32 array.	
	UINT8	Type of the signal is uint8.	
	UINT8_N	Type of the signal is uint8 array.	
	VARIABLE_LENGTH	Type of the signal is uint8[DcmDspRoutineParameterSize]. This is only valid for the last signal and when DcmDspRoutineSignalType is set to VARIABLE_LENGTH.	
Post-Build Variant Value	false		
Value Configuration Class	**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**
DcmDspArgumentScaling	0..1	This container contains the configuration (arguments) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.

10.3.5.14.10 DcmDspStopRoutine

SWS Item	[ECUC_Dcm_01022]
Container Name	DcmDspStopRoutine
Description	Provides the configuration of Stop subservice for RoutineControl service. Existence indicates that the StopRoutine in the RoutineControl is supported.
Configuration Parameters	

Name	DcmDspStopRoutineConfirmationEnabled [ECUC_Dcm_01095]		
Parent Container	DcmDspStopRoutine		
Description	Allows to enable/disable the confirmation function to indicate the transmission of a response to a StopRoutine request		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspStopRoutineConfirmationFnc [ECUC_Dcm_01096]		
Parent Container	DcmDspStopRoutine		
Description	C-function to call if a transmission confirmation is needed by the issuer (BSW module)		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspStopRoutineFnc [ECUC_Dcm_00752]		
Parent Container	DcmDspStopRoutine		
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			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspStopRoutineCommonAuthorizationRef [ECUC_Dcm_01053]		
Parent Container	DcmDspStopRoutine		
Description	Reference to DcmDspCommonAuthorization Common authorization configuration taken from the referenced DcmDspStopRoutineCommonAuthorizationRef. If there is no reference, no check on the commonly defined authorization conditions shall be done to stop the routine.		
Multiplicity	0..1		
Type	Reference to DcmDspCommonAuthorization		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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**
DcmDspStopRoutineIn	0..1	Provide description of input parameter of Stop subservice for RoutineControl service.
DcmDspStopRoutineOut	0..1	Provide description of output parameter of Stop subservice for RoutineControl service.

10.3.5.14.11 DcmDspStopRoutineIn

SWS Item	[ECUC_Dcm_00832]
Container Name	DcmDspStopRoutineIn
Description	Provide description of input parameter of Stop subservice for RoutineControl service.
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspStopRoutineIn Signal	1..*	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataInN elements in the XXX_Stop function call.

10.3.5.14.12 DcmDspStopRoutineInSignal

SWS Item	[ECUC_Dcm_00839]
Container Name	DcmDspStopRoutineInSignal
Description	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataInN elements in the XXX_Stop function call. **Attributes:** requiresIndex=true
Configuration Parameters	

Name	DcmDspRoutineParameterSize [ECUC_Dcm_00841]
Parent Container	DcmDspStopRoutineInSignal
Description	Provide the size of a RoutineControl parameter in bytes
Multiplicity	0..1
Type	EcucIntegerParamDef
Range	0 .. 65535
Default Value	
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false

Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineSignalEndianness [ECUC_Dcm_01014]		
Parent Container	DcmDspStopRoutineInSignal		
Description	Defines the endianness of the data belonging to a Routine In Signal for Stop subfunction.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN	Most significant byte shall be stored at the lowest address.	
	LITTLE_ENDIAN	Most significant byte shall be stored at the highest address	
	OPAQUE	Opaque data endianness	
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineSignalPos [ECUC_Dcm_00840]		
Parent Container	DcmDspStopRoutineInSignal		
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			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Scope / Dependency	scope: ECU

Name	DcmDspRoutineSignalType [ECUC_Dcm_00882]		
Parent Container	DcmDspStopRoutineInSignal		
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.	
	SINT16_N	Type of the signal is sint16 array.	
	SINT32	Type of the signal is sint32.	
	SINT32_N	Type of the signal is sint32 array.	
	SINT8	Type of the signal is sint8.	
	SINT8_N	Type of the signal is sint8 array.	
	UINT16	Type of the signal is uint16.	
	UINT16_N	Type of the signal is uint16 array.	
	UINT32	Type of the signal is uint32.	
	UINT32_N	Type of the signal is uint32 array.	
	UINT8	Type of the signal is uint8.	
	UINT8_N	Type of the signal is uint8 array.	
	VARIABLE_LENGTH	Type of the signal is uint8[DcmDspRoutineParameterSize]. This is only valid for the last signal and when DcmDspRoutineSignalType is set to VARIABLE_LENGTH.	
Post-Build Variant Value	false		
Value Configuration Class	**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**
DcmDspArgumentScaling	0..1	This container contains the configuration (arguments) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.

10.3.5.14.13 DcmDspStopRoutineOut

SWS Item	[ECUC_Dcm_00833]
Container Name	DcmDspStopRoutineOut
Description	Provide description of output parameter of Stop subservice for RoutineControl service.
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspStopRoutineOutSignal	1..*	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataOutN elements in the XXX_Stop function call.

10.3.5.14.14 DcmDspStopRoutineOutSignal

SWS Item	[ECUC_Dcm_00842]
Container Name	DcmDspStopRoutineOutSignal
Description	Provide description of a routine signal used in RoutineControl service. The ordering defined via the index attribute of the subcontainers in this list represents the order of the dataOutN elements in the XXX_Stop function call. **Attributes:** requiresIndex=true
Configuration Parameters	

Name	DcmDspRoutineParameterSize [ECUC_Dcm_00844]		
Parent Container	DcmDspStopRoutineOutSignal		
Description	Provide the size of a RoutineControl parameter in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineSignalEndianness [ECUC_Dcm_01015]		
Parent Container	DcmDspStopRoutineOutSignal		
Description	Defines the endianness of the data belonging to a Routine Out Signal for Stop subfunction.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN	Most significant byte shall be stored at the lowest address.	
	LITTLE_ENDIAN	Most significant byte shall be stored at the highest address	
	OPAQUE	Opaque data endianness	
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineSignalPos [ECUC_Dcm_00843]		
Parent Container	DcmDspStopRoutineOutSignal		
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			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspRoutineSignalType [ECUC_Dcm_00883]	
Parent Container	DcmDspStopRoutineOutSignal	
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.
	SINT16_N	Type of the signal is sint16 array.
	SINT32	Type of the signal is sint32.

	SINT32_N	Type of the signal is sint32 array.	
	SINT8	Type of the signal is sint8.	
	SINT8_N	Type of the signal is sint8 array.	
	UINT16	Type of the signal is uint16.	
	UINT16_N	Type of the signal is uint16 array.	
	UINT32	Type of the signal is uint32.	
	UINT32_N	Type of the signal is uint32 array.	
	UINT8	Type of the signal is uint8.	
	UINT8_N	Type of the signal is uint8 array.	
	VARIABLE_LENGTH	Type of the signal is uint8[DcmDspRoutineParameterSize]. This is only valid for the last signal and when DcmDspRoutineSignalType is set to VARIABLE_LENGTH.	
Post-Build Variant Value	false		
Value Configuration Class	**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**
DcmDspArgumentScaling	0..1	This container contains the configuration (arguments) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.

10.3.5.15 Session Security and Modes

10.3.5.15.1 DcmDspSecurity

SWS Item	[ECUC_Dcm_00764]
Container Name	DcmDspSecurity
Description	This container contains the configuration (DSP parameter) for security level configuration (per security level) Description This container contains Rows of DcmDspSecurityRow
Configuration Parameters	

Name	DcmDspSecurityMaxAttemptCounterReadoutTime [ECUC_Dcm_01101]		
Parent Container	DcmDspSecurity		
Description	Delay, in seconds, from startup (measured from the first call of the Dcm_MainFunction()), allowed for all AttemptCounter values to be obtained from the Application. Must be a multiple of the DcmTaskTime. min: A value equal to the DcmTaskTime		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. 65535[		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspSecurityRow	0..31	Definition of a single Row of configuration for security level configuration (per security level) The name of this container is used to define the name of the R-Port through which the DCM accesses the interface SecurityAccess_{SecurityLevel}. The R-Port is named SecurityAccess_{SecurityLevel} where {SecurityLevel} is the name of the container DcmDspSecurityRow. If there is no reference, no check of security level shall be done.

10.3.5.15.2 DcmDspSecurityRow

SWS Item	[ECUC_Dcm_00759]
Container Name	DcmDspSecurityRow
Description	Definition of a single Row of configuration for security level configuration (per security level) The name of this container is used to define the name of the R-Port through which the DCM accesses the interface SecurityAccess_{SecurityLevel}. The R-Port is named SecurityAccess_{SecurityLevel} where {SecurityLevel} is the name of the container DcmDspSecurityRow. If there is no reference, no check of security level shall be done.
Configuration Parameters	

Name	DcmDspSecurityADRSize [ECUC_Dcm_00725]	
Parent Container	DcmDspSecurityRow	
Description	Size in bytes of the AccessDataRecord used in GetSeed	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	

Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSecurityAttemptCounterEnabled [ECUC_Dcm_01050]		
Parent Container	DcmDspSecurityRow		
Description	Allows to enable the external handling of the security attempt counter (e.g. to survive a reset of the ECU).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspSecurityCompareKeyFnc [ECUC_Dcm_00969]		
Parent Container	DcmDspSecurityRow		
Description	Function name to request the result of a key comparison. Parameter is only relevant if DcmDspSecurityUsePort=="USE_ASYNCH_FNC". This parameter is related to the interface Xxx_CompareKey.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSecurityDelayTime [ECUC_Dcm_00757]		
Parent Container	DcmDspSecurityRow		
Description	Delay time after failed security access in seconds. This is started after DcmDspSecurityNumAttDelay number of failed security accesses. min: A negative value is not allowed.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 65535]		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSecurityDelayTimeOnBoot [ECUC_Dcm_00726]		
Parent Container	DcmDspSecurityRow		
Description	Value of the delay timer in case of 'power on' in seconds. This delay indicates the time at ECU boot power-on time during which the Dcm does not accept a security access. min: A negative value is not allowed.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 65535]		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSecurityGetAttemptCounterFnc [ECUC_Dcm_01048]		
Parent Container	DcmDspSecurityRow		
Description	Function name to request the value of an attempt counter. This parameter is related to the interface Xxx_ GetSecurityAttemptCounter.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSecurityGetSeedFnc [ECUC_Dcm_00968]		
Parent Container	DcmDspSecurityRow		
Description	Callout function name used to request a seed. This parameter is related to the interface Xxx_GetSeed.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSecurityKeySize [ECUC_Dcm_00760]	
Parent Container	DcmDspSecurityRow	
Description	size of the security key (in Bytes).	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	

Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSecurityLevel [ECUC_Dcm_00754]		
Parent Container	DcmDspSecurityRow		
Description	Value of Security level. The locked state cannot be configured explicitly. 1,2,3...63: configuration dependent - Conversion formula to calculate SecurityLevel out of tester requested SecurityAccessType parameter: SecurityLevel = (SecurityAccessType (requestSeed) + 1) / 2 Type: Dcm_SecLevelType		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	1 .. 63		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSecurityNumAttDelay [ECUC_Dcm_00762]		
Parent Container	DcmDspSecurityRow		
Description	Number of failed security accesses after which the delay time is activated		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSecuritySeedSize [ECUC_Dcm_00755]		
Parent Container	DcmDspSecurityRow		
Description	size of the security seed (in Bytes).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSecuritySetAttemptCounterFnc [ECUC_Dcm_01049]		
Parent Container	DcmDspSecurityRow		
Description	Function name to set the value of an attempt counter. This parameter is related to the interface Xxx_ SetSecurityAttemptCounter.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSecurityUsePort [ECUC_Dcm_00967]	
Parent Container	DcmDspSecurityRow	
Description	Defines which kind of interface shall be used for security access.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	USE_ASYNCH_CLIENT_SERVER	The DCM will access the data using an R-Port requiring a asynchronous ClientServertInterface SecurityAccess_{SecurityLevel}. The R-Port is described in DcmDspSecurityRow description.

	USE_ASYNCH_FNC	The DCM will access the data using the functions that are defined in the parameters DcmDspSecurityGetSeedFnc and DcmDspSecurityCompareKeyFnc as well as the functions defined in DcmDspSecurityGetAttemptCounterFnc and DcmDspSecuritySetAttemptCounterFnc, if enabled by the parameter DcmDspSecurityAttemptCounterEnabled. DCM_E_PENDING return is allowed and OpStatus is existing as IN parameter.	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3.5.15.3 DcmDspSession

SWS Item	[ECUC_Dcm_00769]
Container Name	DcmDspSession
Description	Parent container holding single rows to configure particular sessions
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspSessionRow	0..31	This container holds all parameters needed to configure a single session

10.3.5.15.4 DcmDspSessionRow

SWS Item	[ECUC_Dcm_00767]
Container Name	DcmDspSessionRow
Description	This container holds all parameters needed to configure a single session
Configuration Parameters	

Name	DcmDspSessionForBoot [ECUC_Dcm_00815]		
Parent Container	DcmDspSessionRow		
Description	This parameter defines whether this diagnostic session allows to jump to Bootloader (OEM Bootloader or System Supplier Bootloader) and determines, from which unit the final response will be sent. 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 and bootloader sends final response.	
	DCM_OEM_BOOT_RESP APP	This diagnostic session allows to jump to OEM Bootloader and application sends final response.	
	DCM_SYS_BOOT	This diagnostic session allows to jump to System Supplier Bootloader and bootloader sends final response.	
	DCM_SYS_BOOT_RESP APP	This diagnostic session allows to jump to System Supplier Bootloader and application sends final response.	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSessionLevel [ECUC_Dcm_00765]		
Parent Container	DcmDspSessionRow		
Description	subFunction value of the DiagnosticSession. 0, 127 and all values above 127 are reserved by ISO		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	1 .. 126		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSessionP2ServerMax [ECUC_Dcm_00766]		
Parent Container	DcmDspSessionRow		
Description	This is the session value for P2ServerMax in seconds (per Session). 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. This value is reported to the tester within the response to the 'Session Control' service.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 1]		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspSessionP2StarServerMax [ECUC_Dcm_00768]		
Parent Container	DcmDspSessionRow		
Description	This is the session value for P2*ServerMax in seconds (per Session). 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. This value is reported to the tester within the response to the 'Session Control' service.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 100]		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3.5.15.5 DcmModeCondition

SWS Item	[ECUC_Dcm_00928]
Container Name	DcmModeCondition

Description	This container contains the configuration of a mode condition or an environmental conditions which can be used as argument in DcmModeRules. One DcmModeCondition shall contain either one DcmSwcModeRef or one DcmBswModeRef or one DcmSwcSRDataElementRef. 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. In case of a configured DcmSwcModeRef or DcmBswModeRef only the DcmConditionType DCM_EQUALS or DCM_EQUALS_NOT are applicable. In case of DcmSwcSRDataElementRef all literals of DcmConditionType are possible.
Configuration Parameters	

Name	DcmConditionType [ECUC_Dcm_00929]		
Parent Container	DcmModeCondition		
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		
	DCM_GREATER_OR_EQUAL		
	DCM_GREATER_THAN		
	DCM_LESS_OR_EQUAL		
	DCM_LESS_THAN		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmBswModeRef [ECUC_Dcm_00931]		
Parent Container	DcmModeCondition		
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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmSwcModeRef [ECUC_Dcm_00930]		
Parent Container	DcmModeCondition		
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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmSwcSRDataElementRef [ECUC_Dcm_001037]		
Parent Container	DcmModeCondition		
Description	Reference to environmental conditions. It is possible to reference a S/R Receiver-Port to read physical values and compare (equal, greater, less,...) them with a configured value that is defined by DcmSwcSRDataElementValueRef.		
Multiplicity	0..1		
Type	Choice reference to [DcmDspExternalSRDataElementClass,DcmDspPidService01ExternalSRDataElementClass]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmSwcSRDataElementValueRef [ECUC_Dcm_001038]		
Parent Container	DcmModeCondition		
Description	Reference to a constant specification defining the compare value for environmental condition.		
Multiplicity	0..1		
Type	Foreign reference to CONSTANT-SPECIFICATION		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3.5.15.6 DcmModeRule

SWS Item	[ECUC_Dcm_00925]

Container Name	DcmModeRule
Description	This container contains the configuration of a mode rule which represents a logical expression with DcmModeConditions 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	

Name	DcmLogicalOperator [ECUC_Dcm_00926]		
Parent Container	DcmModeRule		
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		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmModeRuleNrcValue [ECUC_Dcm_00949]		
Parent Container	DcmModeRule		
Description	Optional parameter which defines the NRC to be sent in case the mode rule condition is not valid.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmArgumentRef [ECUC_Dcm_00927]		
Parent Container	DcmModeRule		
Description	This is a choice reference either to a mode condition or a an other mode rule serving as sub-expression. **Attributes:** requiresIndex=true		
Multiplicity	1..*		
Type	Choice reference to [DcmModeCondition,DcmModeRule]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3.5.16 DcmDspVehInfo

SWS Item	[ECUC_Dcm_00630]
Container Name	DcmDspVehInfo
Description	This container contains the configuration (parameters) for one single VehicleInfoType of service $09
Configuration Parameters	

Name	DcmDspVehInfoInfoType [ECUC_Dcm_00631]		
Parent Container	DcmDspVehInfo		
Description	value of InfoType. Within each DcmConfigSet all DcmDspVehInfoInfoType values shall be unique.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspVehInfoNODIProvResp [ECUC_Dcm_01051]		
Parent Container	DcmDspVehInfo		
Description	Indicate the Dcm, which side is responsible to fill the number of data items (NODI), Dcm or the provider of the InfoType data. In case the responsibility is on provider side, only one DcmDspVehInfoData container is allowed. • true: Provider is responsible for providing the number of data items parameter • false or not existing: Dcm is responsible for providing the number of data items parameter		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
DcmDspVehInfoData	1..*	Data Item of an InfoType; ShortName is post-fix of the port interface name.

10.3.5.17 DcmDspVehInfoData

SWS Item	[ECUC_Dcm_00888]
Container Name	DcmDspVehInfoData

Description	Data Item of an InfoType; ShortName is post-fix of the port interface name.
Configuration Parameters	

Name	DcmDspVehInfoDataOrder [ECUC_Dcm_00891]		
Parent Container	DcmDspVehInfoData		
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			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspVehInfoDataReadFnc [ECUC_Dcm_00889]		
Parent Container	DcmDspVehInfoData		
Description	Callout function name for reading InfoType data item. Only required in case parameter 'DcmDspVehInfoDataUsePort' is set to 'false'		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspVehInfoDataSize [ECUC_Dcm_00890]		
Parent Container	DcmDspVehInfoData		
Description	Size in bytes of the InfoType data item.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspVehInfoDataUsePort [ECUC_Dcm_00727]		
Parent Container	DcmDspVehInfoData		
Description	When this parameter is set to true the DCM will access the Data using an R-Port requiring a PortInterface IInfotypeServices_{VehInfoData}. The R-Port is named InfotypeServices_{VehInfoData} where {VEHINFODATA} is the name of the container DcmDspVehInfoData. In that case, the DcmDspVehInfoDataReadFnc is ignored and the RTE APIs are used. When this parameter is set to false, the DCM calls the function defined in DcmDspVehInfoDataReadFnc.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

10.3.5.18 DcmDspPeriodicTransmission

SWS Item	[ECUC_Dcm_00957]
Container Name	DcmDspPeriodicTransmission
Description	This container contains the configuration (parameters) for Periodic Transmission Scheduler.
Configuration Parameters	

Name	DcmDspMaxPeriodicDidScheduler [ECUC_Dcm_00962]		
Parent Container	DcmDspPeriodicTransmission		
Description	Defines the maximum number of periodicDataIdentifiers that can be scheduled concurrently.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspPeriodicTransmissionFastRate [ECUC_Dcm_00960]		
Parent Container	DcmDspPeriodicTransmission		
Description	This parameter give the transmission rate of the requested periodicDataIdentifiers to be used if the parameter transmissionMode given in the ReadDataByPeriodicID request is equal to 0x03 ("sendAtFastRate"). This parameter value in seconds have to be configured as a multiple of DcmTaskTime. min: A negative value and zero is not allowed.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[1E-4 .. 0.255]		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspPeriodicTransmissionMediumRate [ECUC_Dcm_00959]		
Parent Container	DcmDspPeriodicTransmission		
Description	This parameter give the transmission rate of the requested periodicDataIdentifiers to be used if the parameter transmissionMode given in the ReadDataByPeriodicID request is equal to 0x02 ("sendAtMediumRate"). This parameter value in seconds have to be configured as a multiple of DcmTaskTime. min: A negative value and zero is not allowed.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[1E-4 .. 0.255]		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDspPeriodicTransmissionSlowRate [ECUC_Dcm_00958]		
Parent Container	DcmDspPeriodicTransmission		
Description	This parameter give the transmission rate of the requested periodicDataIdentifiers to be used if the parameter transmissionMode given in the ReadDataByPeriodicID request is equal to 0x01 ("sendAtSlowRate"). This parameter value in seconds have to be configured as a multiple of DcmTaskTime. min: A negative value and zero is not allowed.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[1E-4 .. 0.255]		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3.5.19 DcmDspClearDTC

SWS Item	[ECUC_Dcm_01064]
Container Name	DcmDspClearDTC
Description	This container contains the configuration for the Clear DTC service.
Configuration Parameters	

Name	DcmDspClearDTCCheckFnc [ECUC_Dcm_01066]		
Parent Container	DcmDspClearDTC		
Description	Callback function for condition check, manufacturer / supplier specific checks on the groupOfDTC, which is requested to clear. This parameter is related to the interface : Xxx_ClearDTCCheckFnc.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmDspClearDTCModeRuleRef [ECUC_Dcm_01065]		
Parent Container	DcmDspClearDTC		
Description	Reference to DcmModeRule Mode rule which controls to clear the DTCs. If there is no reference, no check of the mode rule shall be done.		
Multiplicity	0..1		
Type	Reference to DcmModeRule		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**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

10.3.6 DcmGeneral

SWS Item	[ECUC_Dcm_00822]
Container Name	DcmGeneral
Description	Contains general configuration parameters valid for the entire Dcm module.
Configuration Parameters	

Name	DcmDDDIDStorage [ECUC_Dcm_00971]
Parent Container	DcmGeneral
Description	This configuration switch defines, whether DDDID definition is stored non-volatile or not. true: DDDID are stored non-volatile false: DDDID are only maintained volatile
Multiplicity	0..1
Type	EcucBooleanParamDef
Default Value	false
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmDevErrorDetect [ECUC_Dcm_00823]		
Parent Container	DcmGeneral		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmHeaderFileInclusion [ECUC_Dcm_01019]		
Parent Container	DcmGeneral		
Description	Name of the header file(s) to be included by the Dcm module containing the used C-callback declarations.		
Multiplicity	0..*		
Type	EcucStringParamDef		
Default Value			
Regular Expression	[a-zA-Z0-9_]([a-zA-Z0-9\._])*		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmRespondAllRequest [ECUC_Dcm_00600]		
Parent Container	DcmGeneral		
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			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	DcmTaskTime [ECUC_Dcm_00820]		
Parent Container	DcmGeneral		
Description	Allow to configure the time for the periodic cyclic task. Please note: This configuration value shall be equal to the value in the RTE module. The AUTOSAR configuration standard is to use SI units, so this parameter is defined as float value in seconds. Dcm configuration tools must convert this float value to the appropriate value format for the use in the software implementation of Dcm. min: A negative value and zero is not allowed. upperMultiplicity: Exactly one TaskTime must be specified per configuration. lowerMultiplicity: Exactly one TaskTime must be specified per configuration.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmVersionInfoApi [ECUC_Dcm_00821]		
Parent Container	DcmGeneral		
Description	Preprocessor switch to enable or disable the output Version info of the functionality.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	DcmVinRef [ECUC_Dcm_00984]		
Parent Container	DcmGeneral		
Description	Reference to the Did containing the VIN Information. This parameter is needed for function Dcm_GetVin		
Multiplicity	0..1		
Type	Reference to DcmDspDid		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.4 Protocol Configuration Example

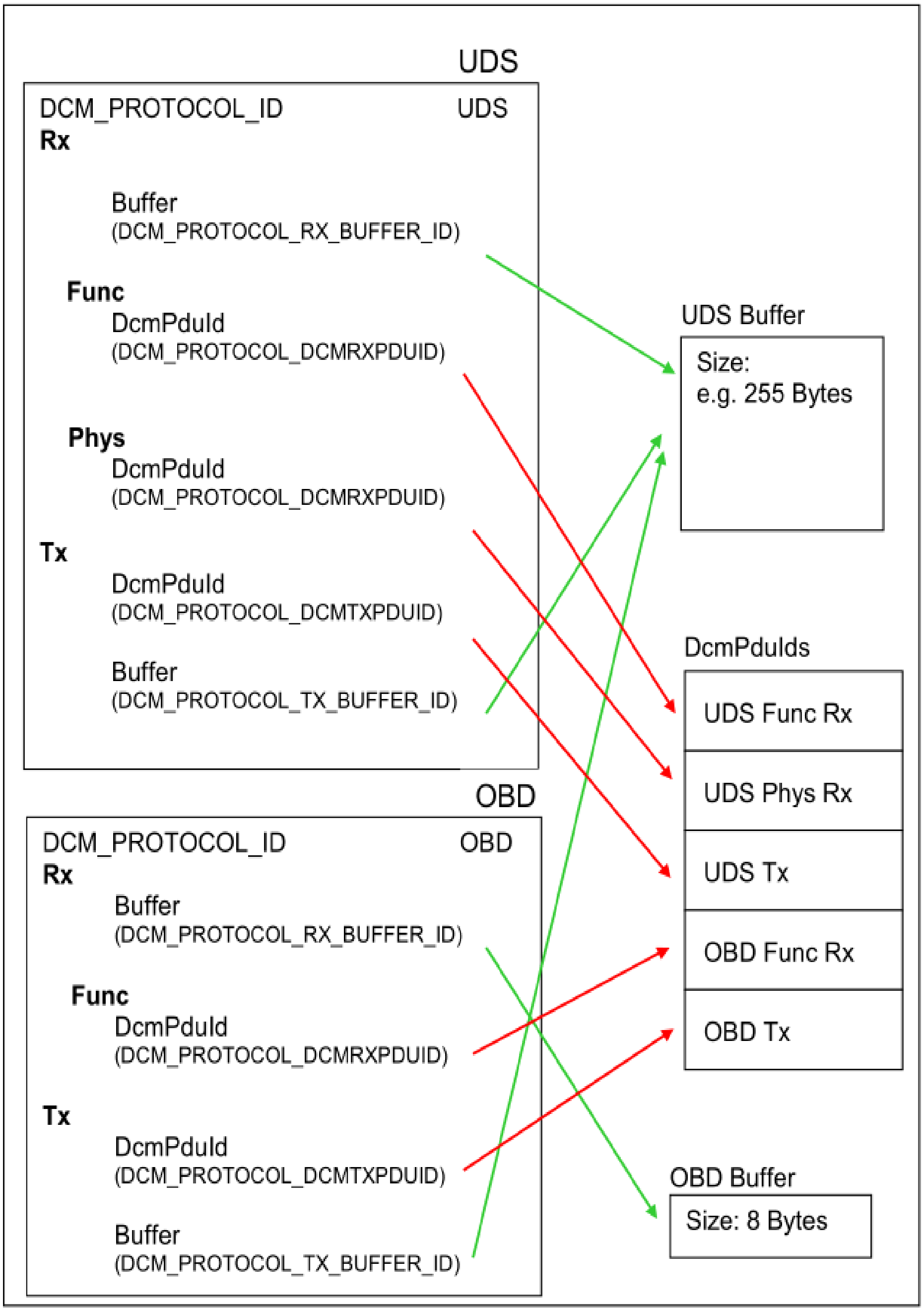


Figure 10.9: Examples of protocol configuration with focus on buffer / DcmPduId settings

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"

10.5 Published Information

For details refer to the chapter 10.3 "Published Information" in SWS_BSWGeneral [7].

A Not applicable requirements

[SWS_Dcm_NA_00999] ⌈ These requirements are not applicable to this specification. ⌋*(SRS_BSW_00005, SRS_BSW_00006, SRS_BSW_00007, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00158, SRS_BSW_00159, SRS_BSW_00160, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00164, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00171, SRS_BSW_00172, SRS_BSW_00300, SRS_BSW_00301, SRS_BSW_00304, SRS_BSW_00305, SRS_BSW_00306, SRS_BSW_00307, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00310, SRS_BSW_00312, SRS_BSW_00314, SRS_BSW_00318, SRS_BSW_00321, SRS_BSW_00323, SRS_BSW_00325, SRS_BSW_00327, SRS_BSW_00328, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00333, SRS_BSW_00334, SRS_BSW_00335, SRS_BSW_00336, SRS_BSW_00339, SRS_BSW_00341, SRS_BSW_00342, SRS_BSW_00343, SRS_BSW_00344, SRS_BSW_00345, SRS_BSW_00346, SRS_BSW_00347, SRS_BSW_00350, SRS_BSW_00351, SRS_BSW_00353, SRS_BSW_00357, SRS_BSW_00358, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00361, SRS_BSW_00371, SRS_BSW_00374, SRS_BSW_00375, SRS_BSW_00377, SRS_BSW_00378, SRS_BSW_00379, SRS_BSW_00380, SRS_BSW_00383, SRS_BSW_00384,* SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00394, SRS_BSW_00395, SRS_BSW_00396, SRS_BSW_00397, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00401, SRS_BSW_00402, SRS_BSW_00403, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00406, SRS_BSW_00408, SRS_BSW_00409, SRS_BSW_00410, SRS_BSW_00411, SRS_BSW_00413, SRS_BSW_00414, SRS_BSW_00415, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00437, SRS_BSW_00439, SRS_BSW_00440, SRS_BSW_00441, SRS_BSW_00447, SRS_BSW_00448, SRS_BSW_00449, SRS_BSW_00450, SRS_BSW_00451, SRS_BSW_00452, SRS_BSW_00453, SRS_BSW_00454, SRS_BSW_00456, SRS_BSW_00457, SRS_BSW_00458, SRS_BSW_00459, SRS_BSW_00460, SRS_BSW_00461, SRS_BSW_00462, SRS_BSW_00463, SRS_BSW_00464, SRS_BSW_00465, SRS_BSW_00466, SRS_BSW_00467, SRS_BSW_00469, SRS_BSW_00470, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00473, SRS_BSW_00477, SRS_BSW_00478, SRS_BSW_00479, SRS_BSW_00480, SRS_BSW_00481, SRS_Diag_04002, SRS_Diag_04007, SRS_Diag_04019, SRS_Diag_04024, SRS_Diag_04031, SRS_Diag_04032, SRS_Diag_04057, SRS_Diag_04059, SRS_Diag_04063, SRS_Diag_04064, SRS_Diag_04068, SRS_Diag_04071, SRS_Diag_04077, SRS_Diag_04085, SRS_Diag_04086, SRS_Diag_04087, SRS_Diag_04089, SRS_Diag_04090, SRS_Diag_04091, SRS_Diag_04093, SRS_Diag_04097, SRS_Diag_04100, SRS_Diag_04101, SRS_Diag_04105, SRS_Diag_04107, SRS_Diag_04109, SRS_Diag_04110, SRS_Diag_04111, SRS_Diag_04112, SRS_Diag_04113, SRS_Diag_04115, SRS_Diag_04117, SRS_Diag_04118, SRS_Diag_04119, SRS_Diag_04120, SRS_Diag_04121, SRS_Diag_04123, SRS_Diag_04124, SRS_Diag_04125, SRS_Diag_04126, SRS_Diag_04127, SRS_Diag_04129, SRS_Diag_04131, SRS_Diag_04133, SRS_Diag_04135, SRS_Diag_04136, SRS_Diag_04137, SRS_Diag_04139, SRS_Diag_04140, SRS_Diag_04143, SRS_Diag_04144, SRS_Diag_04145, SRS_Diag_04146, SRS_Diag_04148, SRS_Diag_04150, SRS_Diag_04151)